International Network Optimization Conference 2022

Aachen, 7-10 June 2022

Book of Abstracts

All times are Central European Summer Time(CEST)

Contents

Influence maximization in (social) networks

Markus Leitner

VU Amsterdam, The Netherlands

Abstract

Online social networks have become crucial communication channels. Millions of people participate in such networks including entities with commercial interests such as companies. The latter increasingly incorporate campaigns promoted via social networks into their marketing mix. Fundamental problems that arise in quantitative social network analysis in the context of (viral) marketing include the identification of influential network nodes that may trigger a large information propagation cascade and the identification of (homogeneous) communities. In this talk we focus on the former problem which is typically referred to as the influence maximization problem. We first provide an overview over considered problem variants and recently proposed exact algorithms for this problem. In the second part of the talk, we motivate several extensions that have been recently introduced. These include the (explicit) consideration of individuals who view content but do not forward it and alternative objective functions. We discuss a generic mixed-integer non-linear programming formulation that incorporates these aspects and an exact linearization based on generalized Benders decomposition. Finally, we report results from a computational study showing that the new problem variants can lead to more effective marketing campaigns and discuss topics for future research.

Date: Wednesday 8 June 2022
Time: 9:00-10:15
Lecture Hall I

The Current and Future Waves of Interdiction Models and Algorithms

J. Cole Smith

Syracuse University, USA

Abstract

This presentation will consists of three parts. The first part will introduce concepts related to network interdiction and robust optimization over networks, with an eye toward describing the truly important work performed in this field in the 1960s and 1970s that could become increasingly important in future studies. The second part will discuss a selection of contemporary interdiction studies, especially those related to assumptions on information asymmetry, stochasticity, players' risk preferences, and learning. The third part will cover a sampling of some fascinating new applications and problem variations that are currently being studied, with an eye toward forecasting what the next wave of research in this field might address.

Date: Thursday 9 June 2022
Time: 9:00-10:00
Lecture Hall I

Robust optimization: Concepts, results and applications in public transportation networks

Anita Schöbel

Fraunhofer ITWM, TU Kaiserslautern, Germany

Abstract

Most real-world problems contain parameters which are not known at the time a decision is to be made. Data may not be measurable in the precision needed or may depend on future developments. An optimal solution which does not take such an uncertainty into account often becomes bad or even infeasible for the scenario which is finally realized.
This is, e.g., true for optimal routes or for timetables in public transport which easily become infeasible if delays occur. In robust optimization one specifies the uncertainty as a scenario set. Classical robust optimization aims at finding a solution which is feasible for all scenarios. Such a solution comes with a high price: A robust route or a robust timetable would have much too long nominal traveling times. The concept is hence not suitable for many applications. There exist less conservative robustness concepts. In the first part of this talk, several such definitions will be shown. Two of them will be discussed in more detail: Light robustness and a scenario-based approach to recovery robustness.
The second part of the talk goes one step further: How to handle uncertain optimization problems in which more than one objective function is to be considered? This yields a robust multi-objective optimization problem, a class of problems only recently introduced.
Concepts on how to define robust Pareto solutions will be developed. Mathematical properties will be derived as well as approaches on how to compute robust efficient solutions. All concepts will be illustrated at routing and planning problems in public transportation.

Date: Friday 10 June 2022
Time: 11:40-12:50
Lecture Hall I

Session Session 1A: network flows

Wednesday 8 June 2022, 10:45-12:25

Lecture Hall I

On the complexity of robust transshipment under consistent flow constraints

Christina Büsing*
RWTH Aachen University,
Combinatorial Optimization
Aachen, Germany
buesing@combi.rwth-aachen.de

Arie M.C.A. Koster
RWTH Aachen University,
Lehrstuhl II für Mathematik
Aachen, Germany
koster@math2.rwth-aachen.de

Sabrina Schmitz†
RWTH Aachen University,
Combinatorial Optimization
Aachen, Germany
schmitz@combi.rwth-aachen.de

ABSTRACT

In this paper, we study the complexity of robust transshipment under consistent flow constraints. We consider demand uncertainty represented by a finite set of scenarios and characterize a subset of arcs as so-called fixed arcs. In each scenario, we require a flow that satisfies the respective balance constraints. In addition, we require on each fixed arc equal flow for all scenarios. The objective is to minimize the maximum cost occurring among all scenarios.

We show that the problem is strongly $\mathcal{NP}$-hard on acyclic digraphs by a reduction from the $(3, B2)$-Sat problem. Further, we prove that the problem is weakly $\mathcal{NP}$-hard on series-parallel digraphs by a reduction from Partition. If in addition the number of scenarios is constant, we suggest a pseudo-polynomial algorithm based on dynamic programming. Finally, we present a special case solvable in polynomial time for series-parallel digraphs.

KEYWORDS

Transshipment Problem, Minimum Cost Flow, Equal Flow Problem, Robust Flows, Demand Uncertainty, Series-Parallel Digraphs

1 INTRODUCTION

In this paper, we consider the *robust transshipment problem under consistent flow constraints* (RobT≡). The problem is motivated by long-term decisions on transshipment that have to be made despite uncertainties in demand. For instance, in logistic applications the transshipment is often agreed in advance by long-term contracts with subcontractors. A solution to the RobT≡ problem facilitates cost-efficient decision-making which is robust against demand uncertainty.

The RobT≡ problem is the uncapacitated version of the robust minimum cost flow problem under consistent flow constraints (RobMCF≡), introduced in our previous work [4]. The complexity results for the RobMCF≡ problem rely on the existence of (tight) capacities. Hence, the question raises whether the problem is solvable in polynomial time if the capacity restrictions are neglected. This is the case for instance for the integral multi-commodity flow problem, which is $\mathcal{NP}$-hard but becomes polynomially solvable for uncapacitated networks.

*Supported by the Freigeist-Fellowship of the Volkswagen Stiftung and by the German research council (DFG) Research Training Group 2236 UnRAVeL.
†Supported by the Freigeist-Fellowship of the Volkswagen Stiftung.

As in the transshipment problem [8], we consider an uncapacitated network in the RobT≡ problem. To represent demand uncertainty, we consider vertex balances for a discrete number of demand scenarios. Furthermore, we characterize a subset of arcs as so-called fixed arcs. In each scenario, we require a flow that satisfies the respective balance constraints. In addition, we require on each fixed arc equal flow for all scenarios. The objective is to minimize the maximum cost that may occur among all demand scenarios.

The main contribution of this paper can be summarized as follows. We prove that finding a feasible solution to the RobT≡ problem is strongly $\mathcal{NP}$-complete on acyclic digraphs, even if only two demand scenarios are considered. On series-parallel (SP) digraphs, we show that the decision version of the RobT≡ problem is weakly $\mathcal{NP}$-complete. We identify the pseudo-polynomial time solvability in the special case of a constant number of scenarios. If all demand scenarios have the same single source and sink in SP digraphs, we propose a polynomial time algorithm.

The outline of this paper is as follows. In Section 2, we provide an overview of related work. In Section 3, we define the problem and introduce the notations of this paper. In Sections 4 and 5, we analyze the complexity of the RobT≡ problem on acyclic and SP digraphs, respectively. In Section 6, we conclude our results.

2 RELATED WORK

In the literature, there are several extensions to the maximum flow (MF) and minimum cost flow (MCF) problem that consider equal flow requirements on specified arc sets. Sahni [11] introduces the integral flow with homologous arcs problem (homIF). In addition to the set-up of the MF problem, the flow value has to be equal on specified arcs. Sahni proves the $\mathcal{NP}$-hardness of the problem by a reduction from the Non-Tautology problem. The MCF version of the homIF problem is known as the (integer) equal flow problem (EF). Using standard techniques, the complexity results can be transformed from the homIF to the EF problem [1].

Meyers and Schulz [10] discuss the transshipment version of the EF problem. For instance, they prove the strong $\mathcal{NP}$-hardness of the problem by a reduction from the Exact Cover by 3-sets problem, even in the case of a single source and sink. Furthermore, they prove the strong $\mathcal{NP}$-hardness for the special case where all sets have cardinality two, which was first investigated for the capacitated version by Ali et al. [2].

Unlike the research referenced above, we do not consider only one demand scenario in the RobT≡ problem. Like in the RobMCF≡ problem, we consider several demand scenarios. We stress that the equal flow requirements are only of importance across more than two different demand scenarios. The flow value of a specified arc

INOC 2022, June 7-10, 2022, Aachen, Germany

has to be equal among all scenarios. In turn, the flow value of two specified arcs may differ in one scenario.

To the best of our knowledge, equal flow requirements and demand uncertainty are combined in our previous study [4] for the first time. Demand uncertainty is frequently studied in the context of (uncapacitated) network design. Three examples are as follows. Gutiérrez et al. [5] present a robustness approach to uncapacitated network design problems. Lien et al. [9] provide an efficient and robust design for transshipment networks by chain configurations. Holmberg and Hellstrand [6] concentrate on finding an optimal solution to the uncapacitated network design problem for commodities with a single source and sink by a Lagrangean heuristic within a branch-and-bound framework.

3 DEFINITION & NOTATIONS

The RobT≡ problem is the uncapacitated version of the RobMCF≡ problem. We define the problem on the basis of our previous work [4]. Let *digraph* $G = (V, A)$ with vertex set V and arc set A be given. The set of arcs A is divided into two disjoint sets A^{fix} and A^{free}, termed *fixed* and *free arcs*, respectively. If not explicitly defined, we specify the sets of vertices, arcs, fixed arcs, and free arcs of a digraph G by $V(G)$, $A(G)$, $A^{\text{fix}}(G)$, and $A^{\text{free}}(G)$, respectively. Let arc *cost* $c : A \to \mathbb{Z}_{\geq 0}$ be given. The demand uncertainty is represented by the finite set of discrete scenarios Λ. For every scenario $\lambda \in \Lambda$, vertex *balances* $b^\lambda : V \to \mathbb{Z}$ with $\sum_{v \in V} b^\lambda(v) = 0$ that define the supply and demand realizations are given, denoted by $\boldsymbol{b} = (b^1, \ldots, b^{|\Lambda|})$. A vertex with a positive or negative balance is termed *source* or *sink*, respectively. In general, the source (sink) vertices do not necessarily have to be the same in every scenario. If all scenarios have only one vertex with a positive (negative) balance and if it is the same vertex in all scenarios, we say that the problem has a *unique source (sink)*. In sum, we obtain the *network* $(G = (V, A = A^{\text{fix}} \cup A^{\text{free}}), c, \boldsymbol{b})$.

For a single scenario $\lambda \in \Lambda$, a b^λ*-flow* in digraph G is defined by a function $f^\lambda : A \to \mathbb{Z}_{\geq 0}$ that satisfies the *flow balance constraints*

$$\sum_{a=(v,w)\in A} f^\lambda(a) - \sum_{a=(w,v)\in A} f^\lambda(a) = b^\lambda(v)$$

at every vertex $v \in V$. The cost of a b^λ-flow f^λ is defined by

$$c(f^\lambda) = \sum_{a \in A} c(a) \cdot f^\lambda(a).$$

For the entire set of scenarios Λ, a *robust* $\boldsymbol{b}$*-flow* $\boldsymbol{f} = (f^1, \ldots, f^{|\Lambda|})$ is defined by a $|\Lambda|$-tuple of b^λ-flows $f^\lambda : A \to \mathbb{Z}_{\geq 0}$ that satisfy the *consistent flow constraints* $f^\lambda(a) = f^{\lambda'}(a)$ on all fixed arcs $a \in A^{\text{fix}}$ for all scenarios $\lambda, \lambda' \in \Lambda$. The cost of a robust $\boldsymbol{b}$-flow $\boldsymbol{f}$ is defined by

$$c(\boldsymbol{f}) = \max_{\lambda \in \Lambda} c(f^\lambda).$$

Finally, the RobT≡ problem is defined as follows.

Definition 3.1 (RobT≡). Given a network $(G = (V, A = A^{\text{fix}} \cup A^{\text{free}}), c, \boldsymbol{b})$, the *robust transshipment problem under consistent flow constraints* aims at computing a robust $\boldsymbol{b}$-flow $\boldsymbol{f} = (f^1, \ldots, f^{|\Lambda|})$ of minimum cost.

We note that in the case of a single scenario, i.e., $|\Lambda| = 1$, the RobT≡ problem corresponds to the transshipment problem [8]. Analogously to the RobMCF≡ problem, we stress that the integral flow property of the transshipment problem (uncapacitated MCF problem) does not hold for the RobT≡ problem. In general, the solution of the continuous relaxation of the RobT≡ problem is not integral.

4 COMPLEXITY FOR ACYCLIC DIGRAPHS

In this section, we investigate the complexity of the RobT≡ problem for networks based on acyclic digraphs. The reduction is performed from the strongly $\mathcal{NP}$-complete $(3, B2)$-Sat problem, introduced by Berman et al. [3]. The $(3, B2)$-Sat problem is a special case of the 3-Sat problem where every literal occurs exactly twice. We use the notation $[n] := \{1, \ldots, n\}$.

Theorem 4.1. *Deciding whether or not a feasible solution exists to the RobT≡ problem for networks based on acyclic digraphs is strongly $\mathcal{NP}$-complete, even if a unique source and a unique sink are given and only two scenarios are considered.*

Proof. The RobT≡ problem is contained in $\mathcal{NP}$ as we can check in polynomial time whether the flow balance and consistent flow constraints are satisfied for every scenario. Let $\{x_1, \ldots, x_n\}$ be the set of variables and $C_1, \ldots, C_m$ be the clauses of the $(3, B2)$-Sat instance $\mathcal{I}$. For a set of two scenarios $\Lambda = \{1, 2\}$, we construct a RobT≡ instance $\widetilde{\mathcal{I}} = (G, c, \boldsymbol{b})$. An example of a RobT≡ instance corresponding to a $(3, B2)$-Sat instance with four clauses and three variables is visualized in Figure 1. In general, the instance is based on a digraph $G = (V, A)$ defined as follows. The vertex set V consists of one vertex v_i per variable x_i, $i \in [n]$, one dummy vertex v_{n+1}, and one vertex u_j per clause C_j, $j \in [m]$. For every literal x_i ($\overline{x}_i$), $i \in [n]$, four auxiliary vertices w_i^ℓ ($\overline{w}_i^\ell$), $\ell \in [4]$ are included. Furthermore, set V consists of one auxiliary vertex t and vertices r_ℓ for $\ell \in [2(2n+2)]$. Arc set A contains arcs that connect two successive variable vertices v_i, v_{i+1}, $i \in [n]$ by two parallel paths p_i and $\overline{p}_i$ defined along the auxiliary vertices, i.e., $p_i = v_i w_i^1 w_i^2 w_i^3 w_i^4 v_{i+1}$ and $\overline{p}_i = v_i \overline{w}_i^1 \overline{w}_i^2 \overline{w}_i^3 \overline{w}_i^4 v_{i+1}$ for $i \in [n]$. Path p_i represents the positive literal x_i and path $\overline{p}_i$ the negative literal $\overline{x}_i$ of instance $\mathcal{I}$. As each literal occurs exactly twice, we identify two arcs of paths p_i and $\overline{p}_i$ each with the literals. More precisely, let x_i^k ($\overline{x}_i^k$) denote literal x_i ($\overline{x}_i$) which occurs the k-th time, $k \in [2]$ in the formula. *Literal arc* (w_i^{2k-1}, w_i^{2k}) $((\overline{w}_i^{2k-1}, \overline{w}_i^{2k}))$ corresponds to literal x_i^k ($\overline{x}_i^k$). Using this correspondence, we add arc (w_i^{2k}, u_j) $((\overline{w}_i^{2k}, u_j))$ for every literal x_i^k ($\overline{x}_i^k$), $i \in [n]$, $k \in [2]$ included in clause C_j, $j \in [m]$. In the next step, we create a path $\widetilde{p}$ from vertex $v_{n+1} =: r_0$ along vertices r_ℓ, $\ell \in [2(2n+2)-1]$ to vertex $z := r_{2(2n+2)}$. Before introducing the last arcs, we identify all literal arcs and every second arc of path $\widetilde{p}$ as the only fixed arcs in the network, i.e.,

$$A^{\text{fix}} = \left\{(w_i^\ell, w_i^{\ell+1}), (\overline{w}_i^\ell, \overline{w}_i^{\ell+1}) \mid \ell \in \{1, 3\},\ i \in [n]\right\} \cup \left\{(r_\ell, r_{\ell+1}) \mid \ell \in \{1, 3, \ldots, 2(2n+2)-1\}\right\}.$$

We add arcs that connect vertex v_1 with every literal arc and every literal arc with auxiliary vertex t, i.e., (v_1, w_i^ℓ), $(v_1, \overline{w}_i^\ell)$ for $\ell \in \{1, 3\}$ and (w_i^ℓ, t), $(\overline{w}_i^\ell, t)$ for $\ell \in \{2, 4\}$. The clause vertices are connected with the first m fixed arcs of path $\widetilde{p}$, i.e., (u_j, r_{2j-1}) for all $j \in [m]$. The auxiliary vertex t is connected with the successive $2n - m$ fixed arcs of path $\widetilde{p}$ by (t, r_ℓ), $\ell \in \{2m+1, 2m+3, \ldots, 4n-1\}$. We

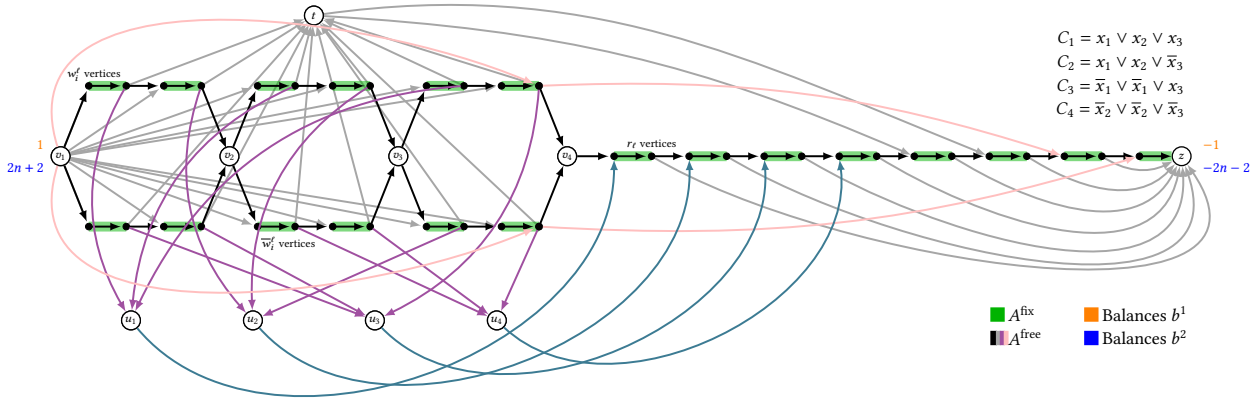


Figure 1: Construction of RobT≡ instance $\widetilde{\mathcal{I}}$.

add arcs (v_1, w_n^4), $(v_1, \overline{w}_n^4)$, (w_n^4, r_{4n+1}), and $(\overline{w}_n^4, r_{4n+3})$. Finally, we connect all $2n+2$ fixed arcs of path $\widetilde{p}$ with vertex z, i.e., (r_ℓ, z) for all $\ell \in \{2, 4, \ldots, 4n+2\}$. We set the cost $c \equiv 0$ and define the balances $\boldsymbol{b} = (b^1, b^2)$ by

$$b^1(v) = \begin{cases} 1 & \text{if } v = v_1, \\ -1 & \text{if } v = z, \\ 0 & \text{otherwise,} \end{cases} \qquad b^2(v) = \begin{cases} 2n+2 & \text{if } v = v_1, \\ -(2n+2) & \text{if } v = z, \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a feasible RobT≡ instance $\widetilde{\mathcal{I}} = (G, c, \boldsymbol{b})$ that is constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a feasible robust $\boldsymbol{b}$-flow exists.

Let $x_1, \ldots, x_n$ be a satisfying truth assignment for instance $\mathcal{I}$. We define the first scenario flow f^1 of instance $\widetilde{\mathcal{I}}$ as follows

$$f^1(a) = \begin{cases} 1 & \text{for all } a \in A(p_i) \text{ if } x_i = \textsc{True}, \\ 1 & \text{for all } a \in A(\overline{p}_i) \text{ if } x_i = \textsc{False}, \\ 1 & \text{for all } a \in A(\widetilde{p}), \\ 0 & \text{otherwise.} \end{cases}$$

Flow f^1 uses either path p_i or $\overline{p}_i$, $i \in [n]$ to send one unit from source v_1 to vertex v_{n+1}. The unit is forwarded from vertex v_{n+1} to sink z along path $\widetilde{p}$. As $x_1, \ldots, x_n$ is a satisfying truth assignment, there exists one designated verifying literal x_i^k or $\overline{x}_i^k$, $k \in [2]$, $i \in [n]$ for each clause C_j, $j \in [m]$. Using this, we define the first part of the second scenario flow f^2 as follows

$$f^2(a) = \begin{cases} 1 & \text{for all } a \in A(q_i^k) \text{ with } q_i^k = v_1 w_i^{2k-1} w_i^{2k} u_j r_{2j-1} r_{2j} z \\ & \text{if } x_i^k \in C_j \text{ is verifying,} \\ 1 & \text{for all } a \in A(q_i^k) \text{ with } q_i^k = v_1 \overline{w}_i^{2k-1} \overline{w}_i^{2k} u_j r_{2j-1} r_{2j} z \\ & \text{if } \overline{x}_i^k \in C_j \text{ is verifying,} \\ 0 & \text{otherwise.} \end{cases}$$

Flow f^2 sends m units from the source v_1 to the clause vertices $u_1, \ldots, u_m$ along the literal arcs corresponding to the verifying literals. The m units are forwarded via the subsequent fixed arc to sink z. Further, we define the second part of the second scenario flow f^2 that sends $2n - m$ units along the remaining literal arcs to vertex t. The flow is forwarded to sink z, i.e., $f^2(a) = 1$ for all

$$\begin{cases} a \in A(p_i^k) & \text{with } p_i^k = v_1 w_i^{2k-1} w_i^{2k} t \text{ if } x_i^k = \textsc{True} \\ & \text{and } x_i^k \text{ is not chosen as a clause-verifying literal,} \\ a \in A(p_i^k) & \text{with } p_i^k = v_1 \overline{w}_i^{2k-1} \overline{w}_i^{2k} t \text{ if } x_i^k = \textsc{False} \\ & \text{and } x_i^k \text{ is not chosen as a clause-verifying literal,} \\ a \in A(p_\ell) & \text{with } p_\ell = t r_\ell r_{\ell+1} z, \ell \in \{2m+1, 2m+3, \ldots, 4n-1\}. \end{cases}$$

Finally, one unit is sent along path $v_1 w_n^4 r_{4n+1} r_{4n+2} z$ and one along path $v_1 \overline{w}_n^4 r_{4n+3} z$. We have constructed a feasible robust $\boldsymbol{b}$-flow $\boldsymbol{f} = (f^1, f^2)$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a feasible robust $\boldsymbol{b}$-flow. Flows f^1 and f^2 send one and $2n+2$ units from source v_1 to sink z, respectively. By construction of the network, the only option to reach the sink requires the usage of at least two fixed arcs, namely one literal arc and one fixed arc of path $\widetilde{p}$ (except for the two paths $v_1 w_n^4 r_{4n+1} r_{4n+2} z$ and $v_1 \overline{w}_n^4 r_{4n+3} z$ which include only one fixed arc of path $\widetilde{p}$). Due to the integral flow f^1 sending one unit within the acyclic digraph, it holds $f^1(a) = f^2(a) \in \{0, 1\}$ for all fixed arcs $a \in A^{\text{fix}}$. Consequently, flows f^1 and f^2 use at least $4n+2$ fixed arcs in order to meet the demand of flow f^2. To use sufficient fixed arcs in the first scenario, flow f^1 sends the unit along either path p_i or $\overline{p}_i$ (but due to the integrality requirement and the acyclic construction never both simultaneously) for all $i \in [n]$ and subsequently along path $\widetilde{p}$. If flow f^1 sends flow along path p_i, $i \in [n]$, we set $x_i = \textsc{True}$. If flow f^1 sends flow along path $\overline{p}_i$, $i \in [n]$, our choice is $x_i = \textsc{False}$. To use sufficient fixed arcs in the second scenario, flow f^2 sends one unit via each clause vertex and $2n - m$ units via vertex t. Accordingly, flow f^2 sends one unit along either path $v_1 w_i^\ell w_i^{\ell+1} u_j r_{2j-1} r_{2j} z$ or $v_1 \overline{w}_i^\ell \overline{w}_i^{\ell+1} u_j r_{2j-1} r_{2j} z$, $\ell \in \{1, 3\}$, $i \in [n]$ for all $j \in [m]$ but never both simultaneously due to the consistent flow constraints. In the former case, clause C_j is verified due to the previous assignment $x_i = \textsc{True}$ induced by flow f^1 and the fact that $x_i \in C_j$ holds. In the latter case, clause C_j is verified due to the previous assignment $x_i = \textsc{False}$ induced by flow f^1 and the fact that $\overline{x}_i \in C_j$ holds. Two extra units are sent along paths $v_1 w_n^4 r_{4n+1} r_{4n+2} z$ and $v_1 \overline{w}_n^4 r_{4n+3} z$ using the last two fixed arcs on path $\widetilde{p}$. The two extra units sent are needed as otherwise, there

might exist a feasible robust flow whose second scenario flow sends a unit along path $v_1 w_n^3 w_n^4 v_{n+1} r_1 r_2 z$ or $v_1 \overline{w}_n^3 \overline{w}_n^4 v_{n+1} r_1 r_2 z$ which in turn allows one unsatisfied clause. Overall, $x_1, \ldots, x_n$ is a satisfying truth assignment for instance $\mathcal{I}$. □

We note that the construction of our previous work's reduction [4] exploits (tight) capacities. For this reason, the adjusted construction containing path $\widetilde{p}$ is essential for the proof of Theorem 4.1. Otherwise, without capacities, we cannot control that one flow unit is sent via every clause vertex.

5 ROBT≡ PROBLEM ON SP DIGRAPHS

In this section, we consider the RobT≡ problem on SP digraphs. In Section 5.1, we show the weak $\mathcal{NP}$-completeness of the problem. In Section 5.2, we provide two algorithms running in polynomial time for the special case of networks with a unique source and a unique sink.

We consider SP digraphs based on the edge SP multi-graphs definition of Valdes et al. [12]. In short, SP digraphs are recursively composed serially or in parallel by SP digraphs, where a single arc itself is defined as an SP digraph. The corresponding SP tree represents its individual arcs (L-vertices) and the order of its series (S-vertices) and parallel (P-vertices) compositions by a binary decomposition computable in polynomial time [12].

5.1 Multiple Sources & Sinks Networks

Before discussing the case of networks based on SP digraphs with multiple sources and multiple sinks, we concentrate on the complexity of the RobT≡ problem in case of a unique source. We perform a reduction from the Partition problem, which is known to be weakly $\mathcal{NP}$-complete [7].

Theorem 5.1. *The decision version of the RobT≡ problem on networks based on SP digraphs with a unique source and multiple sinks is weakly $\mathcal{NP}$-complete, even if only two scenarios are considered.*

Proof. Let $\mathcal{I}$ be a Partition instance with positive integers s_i, $i \in [n]$ such that $\sum_{i=1}^{n} s_i = 2w$ holds. For a set of two scenarios $\Lambda = \{1, 2\}$, we construct a RobT≡ instance $\widetilde{\mathcal{I}} = (G, c, \boldsymbol{b})$ as visualized in Figure 2. The network is based on an SP digraph $G = (V, A)$. Vertex set V consists of two auxiliary vertices v_0, t and two vertices t_i, v_i per integer s_i, $i \in [n]$. Arc set A consists of two parallel arcs a_i^1, a_i^2 that connect vertices v_{i-1}, t_i, $i \in [n]$. Furthermore, arcs $a_i^3 = (t_i, v_i)$ and $a_i^4 = (v_{i-1}, v_i)$ for $i \in [n]$ and arc $a_{n+1} = (v_n, t)$ are included. The fixed arcs of set A are defined by arcs $a_i^2 = (v_{i-1}, t_i)$, i.e., $A^{\text{fix}} = \{a_i^2 \mid i \in [n]\}$. All other arcs are included in set A^{free}. The subgraph induced by vertices v_{i-1}, t_i, v_i represents integer s_i, $i \in [n]$. The cost c is given such that the use of arcs a_i^1 and a_i^2 cost two and one times the integer value s_i, $i \in [n]$ per flow unit, respectively. The use of arc a_{n+1} costs $2w$ per flow unit. The use of all other arcs causes zero cost. We define balances $\boldsymbol{b} = (b^1, b^2)$ by

$$b^1(v) = \begin{cases} 1 & \text{if } v = v_0, \\ -1 & \text{if } v = t, \\ 0 & \text{otherwise,} \end{cases} \qquad b^2(v) = \begin{cases} n & \text{if } v = v_0, \\ -1 & \text{if } v = t_i, i \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a RobT≡ instance $\widetilde{\mathcal{I}} = (G, c, \boldsymbol{b})$ that is constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a robust $\boldsymbol{b}$-flow exists with cost of at most $3w$.

Let S_1 and S_2 be a feasible partition for instance $\mathcal{I}$. We define the first scenario flow f^1 for instance $\widetilde{\mathcal{I}}$ by

$$f^1(a) = \begin{cases} 1 & \text{for arcs } a = a_i^4 \in A,\ i \in [n] \text{ if } s_i \in S_1, \\ 1 & \text{for arcs } a \in \{a_i^2, a_i^3\} \subseteq A,\ i \in [n] \text{ if } s_i \in S_2, \\ 1 & \text{for arc } a = a_{n+1} \in A, \\ 0 & \text{otherwise.} \end{cases}$$

The cost is

$$\begin{aligned} c(f^1) &= \sum_{a \in A^{\text{fix}}} c(a) f^1(a) + \sum_{a \in A^{\text{free}} \setminus \{a_{n+1}\}} c(a) f^1(a) + c(a_{n+1}) f^1(a_{n+1}) \\ &= w + 0 + 2w = 3w. \end{aligned}$$

According to flow f^1 and the partition, we define the second scenario flow f^2 by

$$f^2(a) = \begin{cases} 1 & \text{for arcs } a = a_i^1 \in A,\ i \in [n] \text{ if } s_i \in S_1, \\ 1 & \text{for arcs } a = a_i^2 \in A,\ i \in [n] \text{ if } s_i \in S_2, \\ n - i & \text{for arcs } a = a_i^4 \in A,\ i \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

The cost is

$$c(f^2) = \sum_{a \in A^{\text{fix}}} c(a) f^2(a) + \sum_{a \in A^{\text{free}}} c(a) f^2(a) = w + 2w = 3w.$$

Consequently, we have constructed a robust $\boldsymbol{b}$-flow $\boldsymbol{f} = (f^1, f^2)$ with cost $c(\boldsymbol{f}) = 3w$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a robust $\boldsymbol{b}$-flow with cost $c(\boldsymbol{f}) = \max\{c(f^1), c(f^2)\} \leq 3w$. The first scenario flow f^1 sends one unit from source v_0 to sink t. To reach sink t, arc $a_{n+1} = (v_n, t)$ with cost of $2w$ is used. If all arcs a_i^4, $i \in [n]$ of cost zero were used to reach vertex v_n, no fixed arc could be used in the second scenario due to the consistent flow constraints. This in turn means that flow f^2 would have to use all arcs a_i^1, $i \in [n]$ to send one unit from source v_0 to each of the sinks $t_1, \ldots, t_n$. However, the cost would be

$$\sum_{i=1}^{n} c(a_i^1) = \sum_{i=1}^{n} 2s_i = 4w > 3w \geq \max\{c(f^1), c(f^2)\}.$$

Thus, flow f^2 uses at least as many fixed arcs a_i^2, $i \in [n]$ as cost of w is saved. In return, flow f^1 uses as many fixed arcs a_i^2, $i \in [n]$ as cost of at most w is caused. Consequently, to reach sink t, flow f^1 uses either arc a_i^4 or the two successive arcs a_i^2, a_i^3, $i \in [n]$ of each subgraph but due to the integrality requirement and the acyclic construction never simultaneously. As a result, the sets

$$\begin{aligned} S_1 &:= \{s_i \mid f^1(a_i^4) = 1 \text{ for } i \in [n]\}, \\ S_2 &:= \{s_i \mid f^1(a_i^2) = 1 \text{ and } f^1(a_i^3) = 1 \text{ for } i \in [n]\} \end{aligned}$$

form a feasible partition for instance $\mathcal{I}$. □

Corollary 5.2. *The decision version of the RobT≡ problem on networks based on SP digraphs with multiple sources and multiple sinks is weakly $\mathcal{NP}$-complete, even if only two scenarios are considered.*

In the special case of a constant number of scenarios, we can solve the RobT≡ problem on networks based on SP digraphs with multiple sources and multiple sinks by the pseudo-polynomial algorithm presented in our previous work [4]. As the algorithm is based on dynamic programming, it needs arc capacities as input to limit

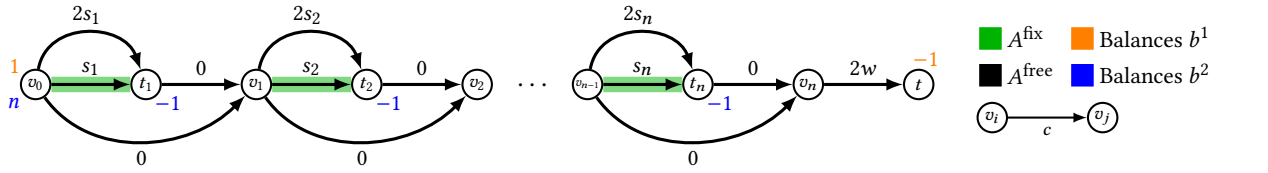


Figure 2: Construction of RobT≡ instance $\widetilde{\mathcal{I}}$.

the number of occurring labels. We can simply set the capacity for every arc to the maximum total demand among all scenarios.

5.2 Unique Source & Unique Sink Networks

For the special case of networks based on SP digraphs with a unique source and a unique sink, we can solve the RobT≡ problem by the polynomial time algorithm presented in our previous work [4]. We set the capacities required for the input to the maximum source's balance among all scenarios. The algorithm reduces to the computation of two shortest paths – one in digraph G and one in digraph $G - A^{\text{fix}}$.

In the following, we discuss an alternative polynomial algorithm which provides further insight into the RobT≡ problem on SP digraphs. Exploiting the SP structure of the digraph, we introduce two shrinking procedures, the parallel- and the series-shrinking procedure. Applying these procedures, we only need to solve the RobT≡ problem on a resulting digraph consisting of one multi-arc.

Without loss of generality, we assume a digraph G with multi-arcs of the form $a = (a^1, \ldots, a^{r(a)}, a^{r(a)+1}, \ldots, a^{k(a)})$ with $r(a) \in \mathbb{Z}_{\geq 0}$ fixed and $k(a) - r(a) \in \mathbb{Z}_{\geq 0}$ free arcs, i.e., $a^1, \ldots, a^{r(a)} \in A^{\text{fix}}(G)$ and $a^{r(a)+1}, \ldots, a^{k(a)} \in A^{\text{free}}(G)$, ordered by their costs $c(a^1) \leq \ldots \leq c(a^{r(a)})$ and $c(a^{r(a)+1}) \leq \ldots \leq c(a^{k(a)})$. We define the *parallel-shrinking procedure* as visualized in Case 1 of Figure 3. The procedure shrinks a multi-arc to a multi-arc consisting of at most one fixed and one free arc.

Require: SP digraph $G = (V, A)$, cost c, multi-arc a
Ensure: Reduced SP digraph $\widetilde{G} = (V, A \setminus \{a\} \cup \{\tilde{a}\})$
procedure Parallel-Shrinking(G, c, a)
  if $k(a) = r(a)$ **then**
    Set $\tilde{a} := (a^1)$
  else if $r(a) = 0$ or $c(a^{r(a)+1}) \leq c(a^1)$ **then**
    Set $\tilde{a} := (a^{r(a)+1})$
  else
    Set $\tilde{a} := (a^1, a^{r(a)+1})$
return $\widetilde{G} := (V, A \setminus \{a\} \cup \{\tilde{a}\})$

Applying the parallel-shrinking procedure, we obtain the following result.

Lemma 5.3 (Parallel-Shrinking). *Let $\mathcal{I} = (G, c, \boldsymbol{b})$ be a RobT≡ instance where G is an SP digraph. Further, let $\widetilde{\mathcal{I}} = (\widetilde{G}, c, \boldsymbol{b})$ be the RobT≡ instance where digraph $\widetilde{G}$ results from applying the parallel-shrinking procedure on arc $a \in A(G)$. The problem of finding an optimal robust $\boldsymbol{b}$-flow for instance $\mathcal{I}$ can be reduced to the problem of finding an optimal robust $\boldsymbol{b}$-flow for instance $\widetilde{\mathcal{I}}$.*

Proof. Let $\tilde{\boldsymbol{f}}$ be an optimal robust $\boldsymbol{b}$-flow for instance $\widetilde{\mathcal{I}}$. Assume flow $\tilde{\boldsymbol{f}}$ is not an optimal robust $\boldsymbol{b}$-flow for instance $\mathcal{I}$. There exists a $\boldsymbol{b}$-flow $\boldsymbol{f}$ with less cost, i.e., $c(\boldsymbol{f}) < c(\tilde{\boldsymbol{f}})$. Flow $\boldsymbol{f}$ uses at least

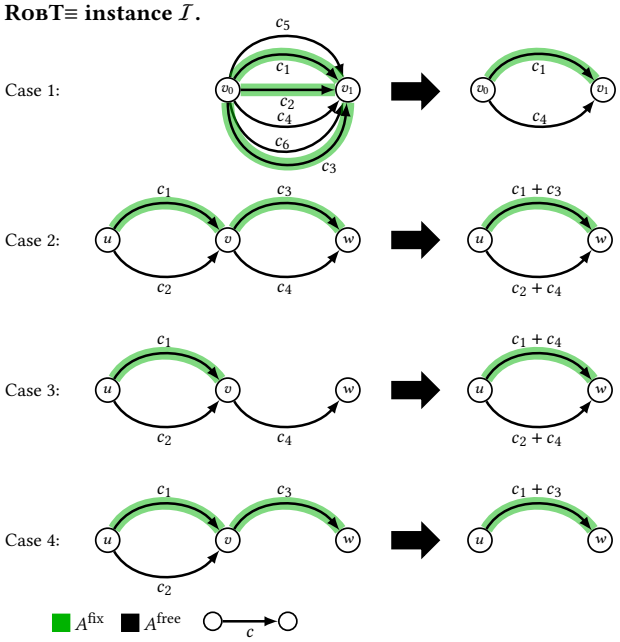


Figure 3: Parallel- and series-shrinking.

one arc of set $\hat{A} := A(G) \setminus A(\widetilde{G})$. If $f^\lambda(a) > 0$ holds for a fixed or free arc $a \in \hat{A} = \{a^2, \ldots, a^{r(a)}, a^{r(a)+2}, \ldots, a^{k(a)}\}$ in one scenario $\lambda \in \Lambda$, we shift the flow to arc a^1 or arc $a^{r(a)+1}$ (provided they exist), respectively. This results in a feasible robust $\boldsymbol{b}$-flow $\hat{\boldsymbol{f}}$ for instance $\widetilde{\mathcal{I}}$ with cost $c(\hat{\boldsymbol{f}}) \leq c(\boldsymbol{f}) < c(\tilde{\boldsymbol{f}})$, which contradicts the assumption. □

In the next step, we define the *series-shrinking procedure*. Let $a_{uv} \in A(G)$ denote a multi-arc directed from vertex u to vertex v with $u, v \in V(G)$. Further, let a_{uv} and a_{vw} be multi-arcs that consist of at most one fixed and one free arc, indicated by the labels 'fix' and 'free'. By the parallel-shrinking procedure, we assume without loss of generality that multi-arcs are of the form $a_{uv} = (a_{uv}^{\text{fix}}, a_{uv}^{\text{free}})$ with $c(a_{uv}^{\text{fix}}) < c(a_{uv}^{\text{free}})$. The series-shrinking procedure reduces a series composition of multi-arcs a_{uv} and a_{vw} associated with an S-vertex in the corresponding SP tree to a single multi-arc $\tilde{a}_{uw}$. Depending on whether multi-arc a_{vw} consists of a fixed and/or a free arc, the series-shrinking procedure is visualized in Cases 2 – 4 of Figure 3.

Require: SP digraph $G = (V, A)$, cost c, SP tree T, S-vertex $s \in V(T)$ with associated subgraph $G_s = (\{u, v, w\}, \{a_{uv}, a_{vw}\})$
Ensure: Reduced SP digraph $\widetilde{G} = (V \setminus \{v\}, A \setminus \{a_{uv}, a_{vw}\} \cup \{\tilde{a}_{uw}\})$
procedure Series-Shrinking(G, c, a_{uv}, a_{vw})
  if $a_{vw} = (a_{vw}^{\text{fix}}, a_{vw}^{\text{free}})$ **then**

Set $\tilde{a}_{uw} := (\tilde{a}^{\text{fix}}_{uw}, \tilde{a}^{\text{free}}_{uw})$
Set $c(\tilde{a}^{\text{fix}}_{uw}) = c(a^{\text{fix}}_{uv}) + c(a^{\text{fix}}_{vw})$
and $c(\tilde{a}^{\text{free}}_{uw}) = c(a^{\text{free}}_{uv}) + c(a^{\text{free}}_{vw})$
else if $a_{vw} = (a^{\text{free}}_{vw})$ **then**
Set $\tilde{a}_{uw} := (\tilde{a}^{\text{fix}}_{uw}, \tilde{a}^{\text{free}}_{uw})$
Set $c(\tilde{a}^{\text{fix}}_{uw}) = c(a^{\text{fix}}_{uv}) + c(a^{\text{free}}_{vw})$
and $c(\tilde{a}^{\text{free}}_{uw}) = c(a^{\text{free}}_{uv}) + c(a^{\text{free}}_{vw})$
else
Set $\tilde{a}_{uw} := (\tilde{a}^{\text{fix}}_{uw})$
Set $c(\tilde{a}^{\text{fix}}_{uw}) = c(a^{\text{fix}}_{uv}) + c(a^{\text{fix}}_{vw})$
return $\widetilde{G} := (V \setminus \{v\}, A \setminus \{a_{uv}, a_{vw}\} \cup \{\tilde{a}_{uw}\})$

Lemma 5.4 (Series-Shrinking). *Let $\mathcal{I} = (G, c, \boldsymbol{b})$ be a RobT≡ instance where G is an SP digraph. Let $a_{uv}, a_{vw} \in A(G)$ be multi-arcs consisting of at most one fixed and one free arc. Further, let $a_{uv}, a_{vw} \in A(G)$ be arcs whose series composition is associated with an S-vertex in the SP tree of digraph G. Let $\widetilde{\mathcal{I}} = (\widetilde{G}, c, \boldsymbol{b})$ be a RobT≡ instance where digraph $\widetilde{G}$ results from applying the series-shrinking procedure on arcs $a_{uv}, a_{vw} \in A(G)$. The problem of finding an optimal robust $\boldsymbol{b}$-flow for instance $\mathcal{I}$ can be reduced to the problem of finding an optimal robust $\boldsymbol{b}$-flow for instance $\widetilde{\mathcal{I}}$.*

Proof. Let $\tilde{a}_{uw} \in A(\widetilde{G})$ denote the shrunk arc. Let $\tilde{\boldsymbol{f}}$ be an optimal robust $\boldsymbol{b}$-flow for instance $\widetilde{\mathcal{I}}$. We define a corresponding robust $\boldsymbol{b}$-flow $\boldsymbol{f}$ for instance $\mathcal{I}$ as follows. For all arcs $a \in A(G) \setminus \{a_{uv}, a_{vw}\}$, we set $\boldsymbol{f}(a) = \tilde{\boldsymbol{f}}(a)$. For multi-arcs $a_{uv}, a_{vw} \in A(G)$, we distinguish between the following three cases, assuming $a_{uv} = (a^{\text{fix}}_{uv}, a^{\text{free}}_{uv})$ without loss of generality.

Case 1: $a_{vw} = (a^{\text{fix}}_{vw}, a^{\text{free}}_{vw})$

$$\boldsymbol{f}(a) = \begin{cases} \tilde{\boldsymbol{f}}(\tilde{a}^{\text{fix}}_{uw}) & \text{for arcs } a \in \{a^{\text{fix}}_{uv}, a^{\text{fix}}_{vw}\}, \\ \tilde{\boldsymbol{f}}(\tilde{a}^{\text{free}}_{uw}) & \text{for arcs } a \in \{a^{\text{free}}_{uv}, a^{\text{free}}_{vw}\}. \end{cases}$$

Case 2: $a_{vw} = (a^{\text{free}}_{vw})$

$$\boldsymbol{f}(a) = \begin{cases} \tilde{\boldsymbol{f}}(\tilde{a}^{\text{fix}}_{uw}) & \text{for arc } a = a^{\text{fix}}_{uv}, \\ \tilde{\boldsymbol{f}}(\tilde{a}^{\text{free}}_{uw}) & \text{for arc } a = a^{\text{free}}_{uv}, \\ \tilde{\boldsymbol{f}}(\tilde{a}^{\text{fix}}_{uw}) + \tilde{\boldsymbol{f}}(\tilde{a}^{\text{free}}_{uw}) & \text{for arc } a = a^{\text{free}}_{vw}. \end{cases}$$

Case 3: $a_{vw} = (a^{\text{fix}}_{vw})$

$$\boldsymbol{f}(a) = \begin{cases} \tilde{\boldsymbol{f}}(\tilde{a}^{\text{fix}}_{uw}) & \text{for arcs } a \in \{a^{\text{fix}}_{uv}, a^{\text{fix}}_{vw}\}, \\ 0 & \text{for arc } a = a^{\text{free}}_{uv}. \end{cases}$$

As the flow balance and consistent flow constraints are still satisfied, flow $\boldsymbol{f}$ is a feasible robust $\boldsymbol{b}$-flow for instance $\mathcal{I}$. Furthermore, flow $\boldsymbol{f}$ causes in every scenario $\lambda \in \Lambda$ the same cost as flow $\tilde{\boldsymbol{f}}$. Assume flow $\boldsymbol{f}$ is not an optimal $\boldsymbol{b}$-flow for instance $\mathcal{I}$. There exists a robust $\boldsymbol{b}$-flow $\hat{\boldsymbol{f}}$ with less cost, i.e., $c(\hat{\boldsymbol{f}}) < c(\boldsymbol{f}) = c(\tilde{\boldsymbol{f}})$. We can transform flow $\hat{\boldsymbol{f}}$ to a feasible flow $\boldsymbol{f'}$ for instance $\widetilde{\mathcal{I}}$. By definition of the shrinking procedure, flow $\boldsymbol{f'}$ causes the same cost as flow $\hat{\boldsymbol{f}}$, i.e., $c(\boldsymbol{f'}) = c(\hat{\boldsymbol{f}}) < c(\boldsymbol{f}) = c(\tilde{\boldsymbol{f}})$, which contradicts to the assumption that $\tilde{\boldsymbol{f}}$ is an optimal flow for instance $\widetilde{\mathcal{I}}$. □

Using the shrinking procedures, we obtain the following result.

Theorem 5.5. *The RobT≡ problem can be solved in polynomial time on networks based on SP digraphs with a unique source and a unique sink.*

Proof. We construct SP digraph G with its unique source $o \in V(G)$ by instructions from the SP tree. If we apply the series- and the parallel-shrinking procedure after each parallel and each series composition of two subgraphs, we obtain a reduced digraph $\widetilde{G} = (\{\tilde{u}, \tilde{v}\}, \{\tilde{a}\})$. We consider the RobT≡ problem on digraph $\widetilde{G}$. For determining a robust $\boldsymbol{b}$-flow $\tilde{\boldsymbol{f}}$, we distinguish between three cases.

If $\tilde{a} = (\tilde{a}^{\text{fix}})$, there exists no feasible solution to the RobT≡ problem as the consistent flow constraints cannot be satisfied (unless the demand of all scenarios is equal). If $\tilde{a} = (\tilde{a}^{\text{free}})$, we set $\tilde{f}^{\lambda}(\tilde{a}) = b^{\lambda}(o)$ for all scenarios $\lambda \in \Lambda$. If $\tilde{a} = (\tilde{a}^{\text{fix}}, \tilde{a}^{\text{free}})$, we set

$$\tilde{f}^{\lambda}(a) = \begin{cases} \min_{\lambda \in \Lambda} b^{\lambda}(o) & \text{for arc } a = \tilde{a}^{\text{fix}}, \\ b^{\lambda}(o) - \min_{\lambda \in \Lambda} b^{\lambda}(o) & \text{for arc } a = \tilde{a}^{\text{free}}, \end{cases}$$

for scenario $\lambda \in \Lambda$. The retransformation of the reduced digraph $\widetilde{G}$ to digraph G and the analog transformation of flow $\tilde{\boldsymbol{f}}$ (cf. proof of Lemmas 5.3 and 5.4) result in a robust minimum cost $\boldsymbol{b}$-flow $\boldsymbol{f}$ for digraph G. Considering the runtime, we can construct an SP tree in $O(|A(G)|)$ time. The series- and parallel-shrinking procedures as well as the (re-)transformation are done in $O(1)$ time for all vertices of the SP tree. In total, the algorithm runs in $O(|A(G)|)$ time. □

6 CONCLUSION

In this paper, we considered the RobT≡ problem. On acyclic networks, we proved that finding a feasible solution is strongly $\mathcal{NP}$-complete, even if a unique source and sink are given and only two scenarios are considered. On networks based on SP digraphs, we proved the weak $\mathcal{NP}$-completeness, even if a unique source is given and only two scenarios are considered. For the special case of a constant number of scenarios, we showed how to solve the problem in pseudo-polynomial time. For the special case of a unique source and sink, we presented two algorithms running in polynomial time.

For the future work, we will study the complexity of the RobT≡ problem on networks based on SP digraphs with a unique source and one sink per scenario (but not a unique sink). Furthermore, we will consider the case if the number of scenarios is part of the input.

REFERENCES

[1] R K Ahuja, T L Magnanti, and J B Orlin. 1988. Network flows. (1988).
[2] A I Ali, J Kennington, and B Shetty. 1988. The equal flow problem. *European Journal of Operational Research* 36, 1 (1988), 107–115.
[3] P Berman, M Karpinski, and A D Scott. 2004. *Approximation hardness of short symmetric instances of MAX-3SAT.* Technical Report.
[4] C Büsing, A MCA Koster, and S Schmitz. 2021. Robust minimum cost flow problem under consistent flow constraints. *Annals of Operations Research* (2021), 1–32.
[5] G J Gutiérrez, P Kouvelis, and A A Kurawarwala. 1996. A robustness approach to uncapacitated network design problems. *European Journal of Operational Research* 94, 2 (1996), 362–376.
[6] K Holmberg and J Hellstrand. 1998. Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound. *Operations research* 46, 2 (1998), 247–259.
[7] D S Johnson and M R Garey. 1979. *Computers and intractability: A guide to the theory of NP-completeness.* WH Freeman.
[8] B H Korte and J Vygen. 2011. *Combinatorial optimization.* Vol. 1. Springer.
[9] R W Lien, S MR Iravani, K Smilowitz, and M Tzur. 2011. An efficient and robust design for transshipment networks. *Production and Operations Management* 20, 5 (2011), 699–713.
[10] C A Meyers and A S Schulz. 2009. Integer equal flows. *Operations Research Letters* 37, 4 (2009), 245–249.
[11] S Sahni. 1974. Computationally related problems. *SIAM J. Comput.* 3, 4 (1974), 262–279.
[12] J Valdes, R E Tarjan, and E L Lawler. 1982. The recognition of series parallel digraphs. *SIAM J. Comput.* 11, 2 (1982), 298–313.

Branch-and-Benders-Cut Algorithm for the Weighted Coflow Completion Time Minimization Problem

Youcef Magnouche, Sebastien Martin, Jeremie Leguay
Huawei Technologies, France Research Center
Boulogne-Billancourt, France
{firstname.name}@huawei.com

Francesco De-Pellegrini, Rachid El-Azouzi, Cedric Richier
University of Avignon
Avignon, France
{firstname.name}@univ-avignon.fr

ABSTRACT

An ever increasing amount of data is being processed by parallel computation frameworks such as MapReduce [9] in data centers. In this context, the coflow scheduling problem aims at accelerating the completion of data processing tasks. In particular, it consists in scheduling individual flows according to their relationship at application level, i.e. their membership to *coflows*, such that the total average weighted coflow completion time is minimized. In this paper, we propose a compact mixed integer linear formulation for the problem and apply a Branch-and-Benders-Cut algorithm to efficiently solve it. On a diverse set of instances, we show that our algorithm significantly improves the CPU time computation compared to the solution of the compact model.

1 INTRODUCTION

Most cloud providers nowadays feature the provisioning of cluster computing as a service. Customers can launch their compute-intensive tasks on big data frameworks such as MapReduce [9] or Spark [21]. Such software frameworks rely on the so called *dataflow* computing model for large-scale data processing. It consists in a distributed computing paradigm where each intermediate computation stage is distributed over a set of nodes and its output is transferred to nodes hosting the next stage. In between two computation stages, these dataflows are producing a set of flows, called a *coflow* [5], that are bound together by the same application task. Coflows represent a standard traffic pattern abstraction in datacenters. In MapReduce for instance, a coflow is a set of concurrent flows sent from *mapper* nodes, i.e., senders, to a set of *reducer* nodes, i.e., receivers. Such flows are launched after mappers have completed their computing tasks. The data transfer phase between mappers and reducers is called the *s*huffle phase and completes only when all constituent flows are over.

The research line on datacenter coflow scheduling has been initiated by the the seminal work of Chowdhury and Stoica [5, 8]. They observed that traffic management policies accounting for the coflow structure significantly improve application-level performance. Since then coflow scheduling is a mainstream topic in network traffic engineering. The main challenge lies in the fact that computing frameworks can generate simultaneously thousands of flows per job [8]. When many jobs run in parallel, network congestion occurs due to concurrent coflows.

In general, the coflow scheduling problem to minimize the Coflow Completion Time (CCT) is strongly NP-hard and exact solutions based on time-indexed MILPs (Mixed Integer Linear Programs) suffer obvious scalability issues. Furthermore, it has been proved recently that, for a related open-shop scheduling problem, approximating the optimal solution is only possible by a factor of $2-\epsilon$, for any $\epsilon > 0$ [4]. In other words, this renders not viable to provide tight approximations of the optimal solution in a short amount of time. In fact, to date, the best deterministic approximation ratio equals to 4 (and 5 if coflows have release times) [1, 19, 20]. Even schedulers relying on relaxation of time-indexed programs face issues related to the number of variables [4, 15, 19]. Several types of schedulers have been proposed [4] in the literature. Clairvoyant schedulers work under perfect information on traffic sources, i.e., engaged ports, flow volumes and flow release times. Semi-clairvoyant schedulers [23] are robust to the lack of information, in particular on exact flow volumes. Non-clairvoyant methods have been studied as well in [8, 10, 11, 22] when prior knowledge is not available, e.g., flow volumes, coflow release times or even the coflow structure per flow is unknown [22].

One popular idea appearing in many research works suggests to equalize flow transfer times per coflow [6] to let all flows of a coflow finish at the same time. In fact, finishing some flows before the bottlenecked one is irrelevant w.r.t. the CCT of a coflow. In standard flow scheduling, shortest-flow-first heuristics grant average flow service time minimization [17]. Varys [7] is a baseline reference for clairvoyant heuristics. Even though recent scheduling algorithms like Sincronia [1] have better performance, Varys has introduced several key concepts at once. First, it combines shortest-flow-first, with coflow equalization. Furthermore, it works based on the notion of *bottleneck* link of a coflow, that is the link of the fabric which experiences the maximal data transfer time. The schedule is performed using a *priority order*: the priority of coflows is assigned *dynamically* and given to the coflow that would end the soonest in isolation (i.e, if alone in the network). Hence, its traffic on the bottleneck link is served in priority, possibly pre-empting lower-priority coflows.

Sincronia [1, 20] resorts to a related primal-dual problem and provides a greedy algorithm which is a $(2 - \frac{2}{n+1})$ approximation for the primal problem. It relies on the results for concurrent open shop scheduling on parallel machines described in [16]. At each iteration, it computes the total load

of each link and finds the link with the heaviest load. It then chooses the coflow using the Smith rule, i.e., the minimum ratio of weight and load on the bottleneck. It iterates on the unsorted coflows until a sorting of all the coflows is obtained. Once the order it decided by Sincronia, it has been shown that any work-conserving rate allocation mechanism achieves a weighted coflow completion time within 4 of the optimal as long as coflows are prioritized respecting the order.

While a number of good heuristics and approximation algorithms have already been proposed for the Weighted Coflow Completion Time Minimization (WCCTM) problem, little has been done on exact methods to attempt to solve optimally the problem at larger scale. In this context, we model the problem as an integer linear program to capture the case of a general network topology where routes can be selected, i.e., multiple routes are possible for flows of a coflow having the same source-destination pairs. Furthermore, we derive a Branch-and-Benders-Cut algorithm (BBC) to decouple the decisions for completion times (Master problem) and individual flow rates (Sub-problem). Based on completion time decisions, we show how the sub-problems can be pre-processed, by removing redundant constraints and variables. Our extensive numerical exploration indicates that the proposed Branch-and-Benders-Cut algorithm accelerates the solving time compared to the compact model.

The paper is organized as follows. Sec. 2 introduces some definitions and notations. Sec 3 presents a compact model for the WCCTM problem. Sec. 4 describes the Branch-and-Benders-Cut algorithm. Sec. 5 reports numerical results and Sec. 6 concludes the paper.

2 NOTATIONS

In this section we give some definitions and notations used throughout this paper. Consider a directed graph $G = (V, \mathcal{L})$ where V is a set of nodes and $\mathcal{L}$ is a set of arcs with capacity $b_l \in \mathbb{R}^+$, for all $l \in \mathcal{L}$. We consider that K coflows are running in parallel $\mathcal{C} = \{C_1, C_2, ..., C_K\}$. Each coflow C_k is composed of n^k flows with $F_k = \{f^{k1}, f^{k2}, .., f^{kn^k}\}$. Each constituent flow f^{kj} is defined by a 4-tuple $(s^{kj}, d^{kj}, \mathcal{P}^{kj}, v^{kj})$ where $s^{kj}, d^{kj} \in V$ are source and destination nodes, respectively. We denote $\mathcal{P}^{kj}$ the set of paths between s^{kj} to d^{kj}, and v^{kj} the flow volume, i.e., the total amount of data to be transferred by flow f^{kj}. A coflow is considered completed only when all its flows are over and the last time when the coflow is active is called *the Coflow Completion Time (CCT) for coflow* C_k. We use w_k to denote the weight, i.e., the importance of each coflow $C_k \in \mathcal{C}$, that is typically given by the application scheduler.

3 COMPACT MODEL

We now present a time-indexed compact model for the WCCTM problem (based on the discretization of a time horizon).

Let T be a time-horizon that we partition into T_s disjoint slots of duration Δ units of time, denoted by u. Let $\mathcal{T} = \{1, \dots, T_s\}$. The model computes the fraction of the total volume to be transferred by each flow at each time slot together with the completion time of each coflow. We suppose that b_l represents the link capacity associated with $l \in \mathcal{L}$, in Mb per time slot. For this model, three types of variables are required: $x_p^{kj}(t) \in [0, 1]$, which represents the fraction of the total volume of flow f^{kj}, associated with coflow $C_k \in \mathcal{C}$ sent during time-slot $t \in \mathcal{T}$ on path $p \in \mathcal{P}^{kj}$; $y^k(t) \in \{0, 1\}$, which equals 1 if time-slot $t \in \mathcal{T}$ is the final time-slot used by coflow $C_k \in \mathcal{C}$, 0 otherwise; $\gamma^k(t) \in [0, 1]$, which represents the unused percentage of the final time-slot $t \in \mathcal{T}$ (the last where coflow $C_k \in \mathcal{C}$ is active). Note that for a coflow $C_k \in \mathcal{C}$, the completion time CT_k equals $\sum_{t \in \mathcal{T}} \Delta(t y^k(t) - \gamma^k(t))$. Hence the WCCTM problem writes as the following MILP:

$$\min \sum_{C_k \in \mathcal{C}} w_k \sum_{t \in \mathcal{T}} \Delta(t y^k(t) - \gamma^k(t)) \tag{1}$$

$$\sum_{t \in \mathcal{T}} y^k(t) = 1 \qquad \forall C_k \in \mathcal{C}, \tag{2}$$

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}^{kj}} x_p^{kj}(t) = 1 \qquad \forall C_k \in \mathcal{C}, \forall f^{kj} \in F_k, \tag{3}$$

$$\gamma^k(t) \leq y^k(t) \qquad \forall C_k \in \mathcal{C}, \forall t \in \mathcal{T} \tag{4}$$

$$\sum_{C_k \in \mathcal{C}} \sum_{f^{kj} \in F_k} v^{kj} \sum_{p \in \mathcal{P}^{kj} : l \in p} x^{kj}(t) \leq b_l \qquad \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{5}$$

$$\sum_{t'=t}^{T_s} \Big(\sum_{p \in \mathcal{P}^{kj}} x_p^{kj}(t') - y^k(t') \Big) \leq 0 \qquad \forall C_k \in \mathcal{C}, \forall f^{kj} \in F_k, \forall t \in \mathcal{T}, \tag{6}$$

$$\sum_{f^{kj} \in F_k} v^{kj} \sum_{p \in \mathcal{P}^{kj} : l \in p} x^{kj}(t) \leq (1 - \gamma^k(t)) b_l \qquad \forall C_k \in \mathcal{C}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{7}$$

$$0 \leq x^{kj}(t) \leq 1 \qquad \forall C_k \in \mathcal{C}, \forall f^{kj} \in F_k, \forall t \in \mathcal{T},$$

$$0 \leq \gamma^k(t) \leq 1 \qquad \forall C_k \in \mathcal{C}, \forall t \in \mathcal{T},$$

$$y^k(t) \in \{0, 1\} \qquad \forall C_k \in \mathcal{C}, \forall t \in \mathcal{T}.$$

Constraints (2) select exactly one final time-slot for each coflow. Constraints (3) guarantee that all flows are served. Constraints (4) link y and γ variables. Constraints (5) represent the port capacity constraints. Constraints (6) ensure that, for every coflow, all flows are sent before the final time-slot. Finally, Constraints (7) decreases the port capacity during the final time-slot. This allows to compute the unused part of the final time-slot of each coflow.

Model (1)-(7) is composed of one family of integer variables and two families of continuous variables. The number of continuous variables is huge ($O(|\mathcal{C}| \times n^{\max} \times p^{\max} \times T_s)$ where $n^{\max} = \max_{C_k \in \mathcal{C}} \{n^k\}$ and $p^{\max} = \max_{C_k \in \mathcal{C}} \max_{f^{kj} \in F_k} \{\mathcal{P}^{kj}\}$).

(a) Coflow 0, Time Slot 0 **(b) Coflow 1, Time Slot 0**

(c) Coflow 0, Time Slot 1 **(d) Coflow 1, Time Slot 1**

Figure 1: Example of two coflows where each row represents 1 time slot of duration $1u$.

In time-indexed coflow scheduling models, setting the time slot to a smaller duration Δ provides finer rate control variables. But, since the number of controlled variables used in the MILP increases it would be preferable to set it to a larger value. An example is reported in Fig. 1 with two coflows of weight 1, each one having 1 flow of volume $1Mb$. Their traffic is sent over a single link of capacity $1Mb/u$ and the length of time horizon is 2 units of time ($2u$). Coflow 1 needs to wait until coflow 0 finishes to start sending traffic, due to capacity constraints. Therefore, coflow 0 finishes after $1u$, and coflow 1 finishes after $2u$. Then, the weighted average CCT is 1.5, which is the optimal solution. One may argue that the model should not be necessarily time-indexed discretized and only 1 time slot of duration $2u$ allows to solve the problem optimally. In this case, the total traffic of $2Mb$ (1Mb per coflow) can be sent in $1u$ as the capacity constraints are for the single time slot, leading to a weighted average CCT of 1, for a possibly unfeasible rate allocation. Thus, we obtain just a lower bound of the exact weighted average w.r.t. the CCT under a discretization with steps of 1 time units. In general, the model accuracy increases with the granularity of the time discretization, i.e., the smaller the duration of time slots, the more the model is accurate.

Also, we observe that variables $\gamma^k(t) \in [0,1]$ allows finer tuning on the effective time slot size: if coflow 2 has volume $0.5Mb$, standard models in literature predict again a weighted average CCT of 1.5 under steps of 1 time units, whereas our model correctly reports 0.75.

We illustrate how solutions of Model (1)-(7) look like, Fig. 2 shows a bubble plot where each line represents a flow, each color represents a coflow and each column represents a time slot. The size of each bubble represents the volume of traffic sent by one flow at the associated time slot. We notice that one coflow starts at time slot 0 and finishes at time slot 20. Also, we observe that the optimal solution is obtained in the class of pre-emptive schedulers, where some flows may send part of their volume during the initial time slots, whereas the remaining volume is sent later on thus prioritizing the flows of some other coflow. In the example this is the case of coflows red and orange, which complete at the end of the time horizon.

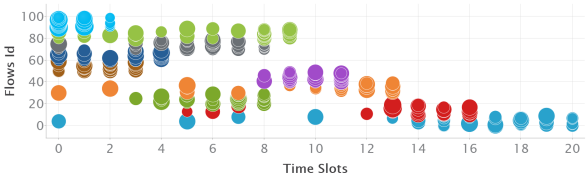


Figure 2: Example with 10 Coflows with 10 flows each. Each line represents a flow, each basic color represents a coflow.

4 BRANCH-AND-BENDERS-CUT

Benders decomposition [3] is a well-known method for solving large-scale combinatorial optimization problems [2]. It consists in decomposing the original problem into one master problem and several sub-problems. The first-stage variables are determined by solving the master problem: the sub-problems check whether they represent a feasible optimal solution. If not, new constraints, called *Benders cuts*, are added to the master problem, and the procedure is repeated. Otherwise, it stops. When sub-problems are linear programs, the approach is guaranteed to converge to an optimal solution.

The Benders decomposition [3] is, generally, effective when the number of integer variables is much smaller than the number of continuous variables. This leads to a master problem with a much smaller dimension than the original one. In the WCCTM problem, the number of coflows is typically much smaller than the number of flow per coflow, i.e., $|C| << n^k$ for all $C_k \in C$. Hence, the gap between the number of integer and continuous variables is often very high.

In this paper we develop a Branch-and-Benders-Cut (BBC) algorithm where, in contrast with the Benders decomposition that solves the master problem at every iteration, a single Branch-and-Cut tree is constructed and the Benders cuts are added during the exploration of the Branch-and-Cut tree.

Developing Branch-and-Benders-Cut algorithm for WCCTM allows to decouple the two types of variables, keeping the integer variables in the master problem, and the continuous variables in the sub-problem (1 sub-problem in our case). The master problem selects the final time-slot for every coflow and the sub-problem checks if all flows can be sent.

Consider an integer solution $y^* \in \{0,1\}^K$. The sub-problem is equivalent to the following linear program

$$\min \quad -\sum_{C_k \in C} w_j \sum_{t \in \mathcal{T}} \Delta \gamma^k(t) \tag{8}$$

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}^{kj}} x_p^{kj}(t) = 1 \qquad \forall C_k \in C, \forall f^{kj} \in F_k, \tag{9}$$

$$\gamma^k(t) \le y^{*k}(t) \qquad \forall t \in \mathcal{T}, \forall C_k \in C. \tag{10}$$

$$\sum_{C_k \in C} \sum_{f^{kj} \in F_k} \sum_{p \in \mathcal{P}^{kj} : l \in p} v^{kj} x_p^{kj}(t) \le b_l \qquad \forall l \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{11}$$

$$\sum_{t'=t}^{T_s} \Big(\sum_{p \in \mathcal{P}^{kj}} x_p^{kj}(t') - y^{*k}(t') \Big) \le 0 \qquad \forall C_k \in C, \forall f^{kj} \in F_k, \forall t \in \mathcal{T}, \tag{12}$$

$$\sum_{f^{kj} \in F_k} \sum_{p \in \mathcal{P}^{kj} : l \in p} v^{kj} x_p^{kj}(t) \le (1 - \gamma^k(t)) b_l \qquad \forall C_k \in C, \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{13}$$

$$0 \le x^{kj}(t) \le 1 \qquad \forall C_k \in C, \forall f^{kj} \in F_k, \forall t \in \mathcal{T},$$

$$0 \le \gamma^k(t) \le 1 \qquad \forall C_k \in C, \forall t \in \mathcal{T}.$$

Constraints (9)-(13) are exactly the same constraints as (3)-(7) respectively, after fixing the values of y using y^*. The sub-problem can be solved in a polynomial time since all variables are continuous. The goal of the sub-problem is to check if there exists a feasible allocation of traffic that respects the completion times of all coflows given by the master.

Let $\alpha, \zeta, \sigma, \gamma, \theta$ be the dual variable vectors of the sub-problem associated with Constraints (9), (10) (11), (12) and (13), respectively. Let $z \in \mathbb{R}$ be an additional variable representing the objective value of the sub-problem, i.e., $z = -\sum_{C_k \in C} w_j \sum_{t \in \mathcal{T}} \Delta \lambda_j^t$.

The master problem consists in minimizing:

$$\sum_{C_k \in C} w_k \Delta \sum_{t \in \mathcal{T}} t y^k(t) + z$$

under Constraints (2) and the following Benders cuts:

$$\sum_{C_k \in C} \sum_{f^{kj} \in F_k} (\alpha^{kj} - \sum_{t \in \mathcal{T}} \gamma^{kj}(t) \sum_{t'=t}^{T_s} y^k(t')) - \sum_{t \in \mathcal{T}} (\sum_{l \in \mathcal{L}} \sigma_l(t) b_l + \sum_{C_k \in C} (\sum_{l \in \mathcal{L}} \theta_l^k(t) b_l + \zeta^k(t) y^k(t))) \leq z \qquad \forall (\alpha, \gamma, \sigma, \theta, \zeta), \quad (14)$$

$$\sum_{C_k \in C} \sum_{f^{kj} \in F_k} (\hat{\alpha}^{kj} - \sum_{t \in \mathcal{T}} \hat{\gamma}^{kj}(t) \sum_{t'=t}^{T_s} y^k(t')) - \sum_{t \in \mathcal{T}} (\sum_{l \in \mathcal{L}} \hat{\sigma}_l(t) b_l + \sum_{C_k \in C} (\sum_{l \in \mathcal{L}} \hat{\theta}_l^k(t) b_l + \hat{\zeta}^k(t) y^k(t))) \leq 0 \qquad \forall (\hat{\alpha}, \hat{\gamma}, \hat{\sigma}, \hat{\theta}), \quad (15)$$

$$y^k(t) \in \{0,1\} \qquad \forall C_k \in C, \forall t \in \mathcal{T}.$$

where $(\hat{\alpha}, \hat{\gamma}, \hat{\sigma}, \hat{\theta}, \hat{\zeta})$ represents the extreme rays of the sub-problem. Note that Constraints (14) are of the form $g(y) \leq z$ where $g(y)$ is the objective function associated to the dual of the sub-problem (8)-(13) and non-negativity inequalities. Constraints (15) are used in the case where the sub-problem is unfeasible and then no Constraint (14) can be generated.

4.1 Sub-problem pre-processing

Once integer decision variables y^* are fixed for the main problem, several pre-processing operations are possible to reduce the size of the sub-problem, thus reducing its computational cost. Let t_k^* be the *completion time slot* of coflow $C_k \in C$, i.e., $y^{*k}(t_k^*) = 1$. Let $t_{max}^* = \max_{C_k \in C} \{t_k^*\}$. The following results hold.

Proposition 1. *Constraint* (11) *associated with time-slot* $t \in \{t_{max}^* + 1, \ldots, T_s\}$ *and arc* $l \in \mathcal{L}$ *are redundant.*

Proof. By multiplying every constraint (12) associated with t by v^{kj} and summing all resulting constraints, we obtain

$$\sum_{C_k \in C} \sum_{f^{kj} \in F_k} \sum_{t'=t}^{T_s} \sum_{p \in \mathcal{P}^{kj}} v^{kj} x_p^{kj}(t') \leq 0 \leq b_l$$

Every constraint (11) associated with t can be obtained by summing the above inequality with the following trivial constraints

$$-v^{kj} x_p^{kj}(t') \leq 0 \quad \forall C_k \in C, f^{kj} \in F_k, p \in \mathcal{P}^{kj}, t' \in \{t+1, \ldots, T_s\}$$

and the result follows. □

Proposition 2. *For all* $C_k \in C$, *Constraints* (12) *associated with* $f^{kj} \in F_k$ *and time-slot* $t \leq t_k^*$ *are redundant.*

Proof. The upper bound of any Constraint (12) associated with time-slot $t \leq t_k^*$ is equal to 1. It follows that any Constraint (12) associated with time-slot $t \leq t_k^*$ can be obtained by summing Constraint (9) with trivial inequalities

$$-x_p^{kj}(t') \leq 0, \qquad \forall t' \leq t, \forall p \in \mathcal{P}^{kj}$$

and the result follows. □

Proposition 3. *For all* $C_k \in C$, *Constraints* (12) *associated with* $f^{kj} \in F_k$ *and time-slot* $t \in \{t_k^* + 2, \ldots, T_s\}$ *are redundant.*

Proof. Constraint (12) associated with time-slot $t \in \{t_k^* + 2, \ldots, T_s\}$ can be obtained by summing constraint (12) associated with time-slot $t_k^* + 1$ and trivial inequalities

$$-x_p^{kj}(t') \leq 0, \qquad \forall t' \in \{t_k^* + 1, \ldots, t-1\}, \forall p \in \mathcal{P}^{kj}$$

and the result follows. □

Proposition 4. *Constraints* (13) *associated with Coflow* $C_k \in C$, *time-slot* $t \in \mathcal{T} \setminus \{t_k^*\}$ *and arc* $l \in \mathcal{L}$ *are redundant.*

Proof. By constraint (10), $\gamma^k(t) = 0$ for all $t \in \mathcal{T} \setminus \{t_k^*\}$. Then Constraint (13) associated with Coflow $C_k \in C$, time-slot $t \in \mathcal{T} \setminus \{t_k^*\}$ and arc $l \in \mathcal{L}$ can by obtained by summing Constraint (11) associated with time-slot t and arc l with the following trivial constraints

$$-v^{kj} x_p^{kj}(t) \leq 0 \quad \forall C_{k'} \in C \setminus \{C_k\}, f^{kj} \in F_{k'}, p \in \mathcal{P}^{k'j}$$

and the result follows. □

Proposition 5. *For all* $C_k \in C$, *flow* $f^{kj} \in F_k$ *and path* $p \in \mathcal{P}^{kj}$, *variables* $x_p^{kj}(t)$ *associated with* $t \in \{t_{max}^* + 2, \ldots, T_s\}$ *can be removed.*

Proof. By definition, $y^{*k}(t) = 0$ for all $t \in \{t_{max}^* + 2, \ldots, T_s\}$. Therefore, by Constraints (12), $x_p^{kj}(t) = 0$ for all $C_k \in C$, flow $f^{kj} \in F_k$, path $p \leq \mathcal{P}^{kj}$ and $t \in \{t_{max}^* + 2, \ldots, T_s\}$. □

Note that, there must exists an optimal dual solution for the sub-problem where all dual variables associated with redundant constraints are equal to 0.

5 NUMERICAL RESULTS

In this section we present the numerical results obtained from the compact model and BBC algorithm. The algorithms have been implemented in C++ using Cplex 12.6 [12] as MILP-solver on a machine with Intel(R) Xeon(R) CPU E5-4627 v2of 3.30GHz with 504GB RAM, running under Linux 64 bits. A maximum of 1 thread has been used. A time limit is set to 3 hours and a memory limit for the B&B tree is set to 5Gb.

In BBC algorithm, constraints (14) and (15) are exponential in number. They are generated dynamically thanks to the *lazy constraint* callback of Cplex. Each time an integer solution is found for the master problem, the sub-problem is solved. If it is feasible, an optimality Benders cut (14) is added. Otherwise, a feasiblity Benders cut (15) is added. All Benders cuts in the master problems are separated iteratively. In order to avoid starting the Branch-and-Benders-Cut algorithm without Benders cuts, some of them can be added to the first master problem which help the convergence of the algorithm. The procedure we propose consists in supposing that all coflows finish before a tagged time slot $t \in \mathcal{T}$. Therefore, by solving the sub-problem for each $t \in \mathcal{T}$, we may generate T_s Benders cuts for the master problem.

To evaluate our algorithm, we use two types of network instances. We first consider public instances from SNDLib [18] and the Internet Topology Zoo [13] that are a mix of real (e.g., Abilene, BtEurope, Geant) and synthetic networks [1]. We also

[1] All instances with topology and traffic information will be made public here: https://github.com/MagYou/coflow-scheduling-benders

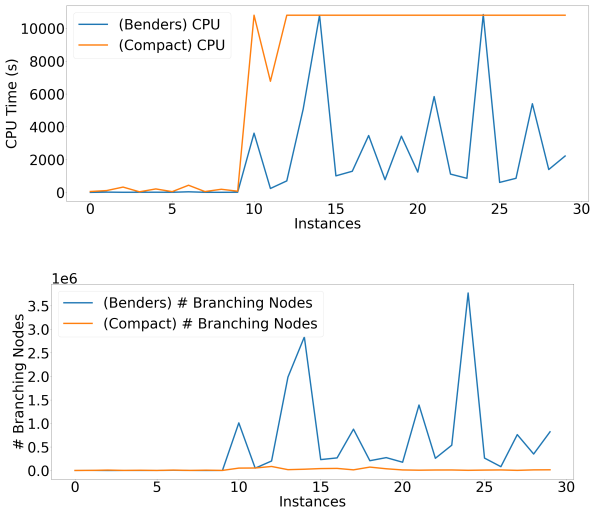


Figure 3: Numerical results on BigSwitch instances for the CPU time and the number of branching nodes.

generate instances following the standard abstraction for data center topologies, called the *Big-Switch* model [7]. This model captures the fact that congestion only occurs at Top-of-Rack (ToR) switches and that the core of the fabric is largely over-provisioned [14].

More precisely, we have considered the three following types of instances in the rest if this paper:

- Big-Switch instances, where the coflow sources are connected (by arcs) to ingress ports of a Big-Switch fabric that connects to all coflow destinations, attached to outgoing ports. We consider several instance sizes:
  - 8 ports and 64 flows : 4 coflows and 16 flows/coflow
  - 12 ports and 144 flows : 8 coflows and 18 flows/coflow
  - 14 ports and 196 flows : 7 and 28 flows/coflow
- SNDLib instances with the following traffic patterns:
  - 30 flows : 3 coflows and 10 flows/coflow
  - 50 flows : 5 coflows and 10 flows/coflow
  - 60 flows: 3 coflows and 20 flows/coflow
  - 100 flows flows: 5 coflows and 20 flows/coflow
- Internet Topology Zoo instances with 150 flows : 6 coflows and 25 flows/coflow.

The length of the horizon time is set to 30 units of times. On SNDLib and Internet Topology Zoo instances, 3 paths are computed for every flow (if they exist). All coflows have been generated randomly (sources, destinations, volumes and paths).

The following figures display the comparison of the CPU time and the number of branching nodes between the BBC algorithm and the compact model.

Fig. 3 reports on the results of Big-Switch instances. We can note that the compact model reaches the time limit on 60% of the instances while the BBC algorithm reaches it on only two instances. However the compact model generates much less branching nodes (around 19000 in average) than the BBC algorithm (around 550000 in average). This can be explained by the fact that the compact model reaches the maximum

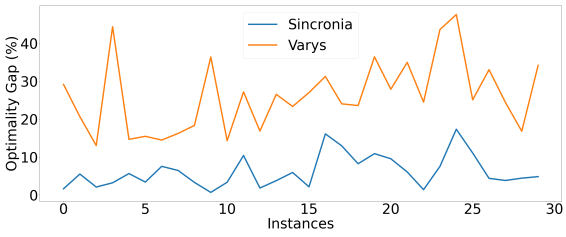


Figure 4: Optimality gaps of Varys [7] and Sincronia [1] (state of the art heuristics) on Big-Switch instances.

time limit on all instances where it has a lower number of branching nodes. Hence, the branching algorithm did not finish generating all nodes. Clearly, the Branch-and-Benders-Cut algorithm performs much better as, in practice, a small master problem is solved at each iteration (around 910 Benders cuts are generated on average at the end of the optimization).

Fig. 4 compares the optimality gaps for state of the art heuristic algorithms such as Varys [7] and Sincronia [1] on Big-Switch instances. As we can see the gap for Varys is quite high, 26.17% in average, while it is much lower for Sincronia, 6.18% in average. While the exact method based on Benders can be used in practice on some of the instances with a reasonable running time, it can also be used to evaluate heuristic algorithms for benchmarking and assess their relative performance.

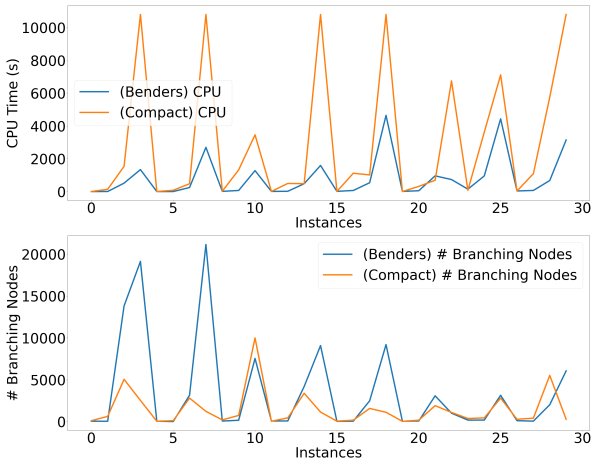


Figure 5: Numerical results on SNDLib instances for the CPU time and the number of branching nodes.

Fig. 5 shows results on SNDLib instances. We can observe that the BBC algorithm did not reach the time limit while the compact model reaches it on 5 instances. The number of branching nodes generated by the BBC algorithm is slightly higher than for the compact model (for the same reason as for Big-Switch instances) since it is around 3500 nodes on average for the BBC algorithm and around 1500 in average for the compact model. The number of Benders cuts generated is

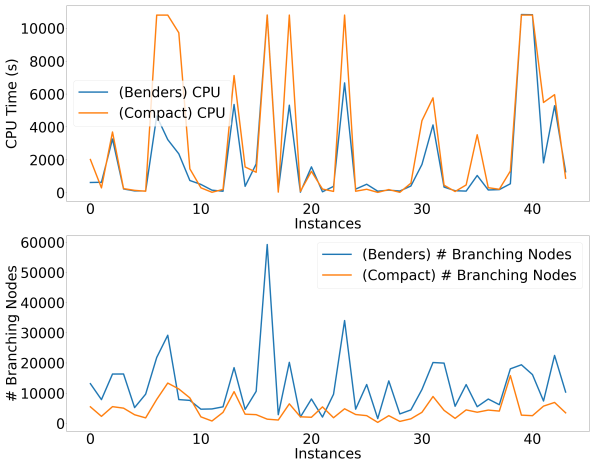


Figure 6: Numerical results on Internet Topology Zoo instances for the CPU time and the number of branching nodes.

also small over all the instances since it is around 600 cuts in average.

Finally, Fig. 6 shows the results for the Internet Topology Zoo instances with 150 flows. We can see that the BBC algorithm gives better results since the compact model reaches the time limit multiple times. Similarly to previous instances, the number of generated nodes in slightly higher for the BBC algorithm. The number of generated Benders cuts is around 2700 cuts on average.

6 CONCLUSION

In this paper, we have addressed exact solution methods for the Weighted Coflow Completion Time Minimization problem, which is NP-hard. The literature on coflows has mostly addressed heuristics and approximation algorithms, and exact solution methods are not discussed in depth to the best of the authors' knowledge. We have proposed a new compact model for the CCTM problem and a Branch-and-Benders-Cut algorithm has been developed which decouples continuous and integer variables. This decomposition permits to pre-process every sub-problem and remove a significant number of useless constraints and variables in advance. Our computational results indicate that the Branch-and-Benders-Cut algorithm for he CCTM improves significantly the CPU time compared to the compact model and thus represents an important contribution to assess the relative performance of coflow schedulers existing in the coflow literature.

REFERENCES

[1] Saksham Agarwal, Shijin Rajakrishnan, Akshay Narayan, Rachit Agarwal, David Shmoys, and Amin Vahdat. 2018. Sincronia: Near-optimal network design for coflows. In Proc. ACM SIGCOMM.

[2] Simon Belieres, Mike Hewitt, Nicolas Jozefowiez, Frédéric Semet, and Tom Van Woensel. 2020. A Benders decomposition-based approach for logistics service network design. European Journal of Operational Research 286, 2 (2020), 523–537.

[3] Jacques F Benders. 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische mathematik 4, 1 (1962), 238–252.

[4] Mosharaf Chowdhury, Samir Khuller, Manish Purohit, Sheng Yang, and Jie You. 2019. Near optimal coflow scheduling in networks. In The 31st ACM Symposium on Parallelism in Algorithms and Architectures. 123–134.

[5] Mosharaf Chowdhury and Ion Stoica. 2012. Coflow: A Networking Abstraction for Cluster Applications. In Proc. ACM HotNets workshop.

[6] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I Jordan, and Ion Stoica. 2011. Managing data transfers in computer clusters with orchestra. ACM SIGCOMM Computer Communication Review 41, 4 (2011), 98–109.

[7] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. 2014. Efficient Coflow Scheduling with Varys. In Proc. ACM SIGCOMM.

[8] NM Mosharaf Kabir Chowdhury. 2015. Coflow: A Networking Abstraction for Distributed Data-Parallel Applications. Ph.D. Dissertation. University of California, Berkeley.

[9] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. (2004).

[10] Fahad R. Dogar, Thomas Karagiannis, Hitesh Ballani, and Antony Rowstron. 2014. Decentralized Task-Aware Scheduling for Data Center Networks. In Proc. ACM SIGCOMM.

[11] Yuanxiang Gao, Hongfang Yu, Shouxi Luo, and Shui Yu. 2016. Information-agnostic coflow scheduling with optimal demotion thresholds. In 2016 IEEE International Conference on Communications (ICC).

[12] IBM. [n.d.]. ILOG CPLEX Solver. https://www.ibm.com/analytics/cplex-optimizer

[13] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. 2011. The Internet Topology Zoo. Selected Areas in Communications, IEEE Journal on 29, 9 (october 2011), 1765 –1775.

[14] Y. Liu, J. K. Muppala, M. Veeraraghavan, Dong Lin, and M. Hamdi. 2013. A Survey of Data Center Network Architectures. Springer.

[15] Ruijiu Mao, Vaneet Aggarwal, and Mung Chiang. 2018. Stochastic non-preemptive co-flow scheduling with time-indexed relaxation. In IEEE INFOCOM WKSHPS.

[16] Monaldo Mastrolilli, Maurice Queyranne, Andreas Schulz, Ola Svensson, and Nelson Uhan. 2010. Minimizing the sum of weighted completion times in a concurrent open shop. Oper. Res. Lett. 38 (09 2010), 390–395. https://doi.org/10.1016/j.orl.2010.04.011

[17] Mohammad Noormohammadpour and Cauligi S. Raghavendra. 2018. Datacenter Traffic Control: Understanding Techniques and Tradeoffs. IEEE Communications Surveys Tutorials 20, 2 (2018), 1492–1525.

[18] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. 2007. SNDlib 1.0–Survivable Network Design Library. In Proceedings of the 3rd International Network Optimization Conference (INOC 2007).

[19] Mehrnoosh Shafiee and Javad Ghaderi. 2018. An Improved Bound for Minimizing the Total Weighted Completion Time of Coflows in Datacenters. IEEE/ACM Transactions on Networking 26, 4 (2018), 1674–1687.

[20] Zhiliang Wang, Han Zhang, Xingang Shi, Xia Yin, Yahui Li, Haijun Geng, Qianhong Wu, and Jianwei Liu. 2019. Efficient Scheduling of Weighted Coflows in Data Centers. IEEE Transactions on Parallel and Distributed Systems 30, 9 (2019), 2003–2017.

[21] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica, et al. 2010. Spark: Cluster computing with working sets. HotCloud 10, 10-10 (2010), 95.

[22] Hong Zhang, Li Chen, Bairen Yi, Kai Chen, Mosharaf Chowdhury, and Yanhui Geng. 2016. CODA: Toward Automatically Identifying and Scheduling Coflows in the Dark. In Proc. ACM SIGCOMM.

[23] Tong Zhang, Fengyuan Ren, Ran Shu, and Bo Wang. 2018. Scheduling Coflows with Incomplete Information. In 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).

Coworking Scheduling with Network Flows

Mariia Anapolska*
Research Group Combinatorial Optimization
RWTH Aachen University, Germany
anapolska@combi.rwth-aachen.de

Christina Büsing†
Research Group Combinatorial Optimization
RWTH Aachen University, Germany
buesing@combi.rwth-aachen.de

Tabea Krabs‡
Research Group Combinatorial Optimization
RWTH Aachen University, Germany
krabs@combi.rwth-aachen.de

Tobias Mömke§
Department of Computer Science
University of Augsburg, Germany
moemke@informatik.uni-augsburg.de

ABSTRACT

Collaborative usage of resources is becoming increasingly popular in various fields. One common example are coworking spaces — office rooms with work places that can be rented by individuals on hourly basis. We consider the problem of assigning all booking requests for a day to equivalent office rooms with different but fixed opening times and fixed interchangeable closing times. The closing times are flexible due to daily maintenance, e.g. cleaning, which must be done in all rooms in an arbitrary order.

This problem is related to the known Interval Scheduling Problem with Machine Availabilities (ISMA), where each machine has a contiguous availability interval, and each job presents a specific time interval which has to be scheduled. According to our coworking scheduling application, we extend ISMA to Flexible Multithread ISMA (FlexMISMA) by introducing machine capacities that model the number of work places per room and by allowing to permute the end times of machines' availability periods.

In this paper, we determine a tight classification of necessary conditions for the existence of a polynomial time algorithm for FlexMISMA, assuming $P \neq NP$. More specifically, we develop a network flow model and present polynomial time algorithms for instances (i) with two machines, and (ii) with arbitrarily many machines of capacity one each. In the same time, we prove that increasing the machine capacity to two renders FlexMISMA NP-hard for arbitrarily many machines. Furthermore, we complement result (i) by showing that the problem is NP-hard already for instances with three machines as a special case of the Vertex-Disjoint Paths problem.

*Supported by the Freigeist-Fellowship of the Volkswagen Stiftung and by the German research council (DFG) Research Training Group 2236 UnRAVeL.
†Supported by the German Federal Ministry of Education and Research (grant no. 05M16PAA) within the project "HealthFaCT - Health: Facility Location, Covering and Transport', by the Freigeist-Fellowship of the Volkswagen Stiftung and by the German research council (DFG) Research Training Group 2236 UnRAVeL.
‡Supported by the Freigeist-Fellowship of the Volkswagen Stiftung and by the German research council (DFG) Research Training Group 2236 UnRAVeL.
§Partially supported by the DFG Grant 439522729 (Heisenberg-Grant).

1 INTRODUCTION

Collaborative usage of resources is becoming increasingly popular in various fields, presenting new challenges for planning and scheduling. For example, small businesses rent parts of storage facilities or computer clusters for fixed time intervals when owning a whole facility is not economical. One further example of such collaboration are coworking spaces – office spaces typically consisting of several office rooms, with work places that can be rented by individuals on an hourly basis. Consider a coworking space with several identical office rooms. The customers can book single desks for an arbitrary continuous time period during the day, and they are guaranteed not to be required to change the room during their stay. Every day at the beginning and at the end of the official opening hours, a cleaning team must visit all the rooms one by one. Cleaning every room takes the same amount of time, thus the times at which the team switches to another room are fixed (these times are equal to the start time of the cleaning phase plus multiples of the cleaning duration). The order in which the rooms are cleaned is flexible. Since the rooms must be empty during the cleaning, the order of room cleaning defines the effective availability periods of rooms to the customers. When a new booking comes in, the planner needs to decide whether it can be accepted, i.e. whether there is a cleaning sequence and an assignment of all received bookings to rooms which respects the rooms' capacity and does not interfere with the cleaning.

It is easy to recognize that this problem presents a variant of Interval Scheduling with restricted machine availabilities, where machines represent the rooms, intervals represent single bookings, and restricted availability periods of machines are caused by room cleaning in the morning and in the evening. Note that the rooms accommodate several desks, which corresponds to machines being able to process several jobs at a time.

1.1 Related work

There have been many approaches to extend Interval Scheduling by restricted machine availabilities, cf. [8]. Brucker and Nordman [3] introduce a variant of Interval Scheduling, the k-track assignment problem, in which every machine is available only for a given time interval. The authors consider both the case of identical machines and a generalization where machines can process only given subsets of jobs. Later, Kolen et al. [8] studied this problem using the name Interval Scheduling with Machine Availabilities (ISMA). They showed that the problem is NP-complete if the number of machines

is part of the input but polynomially solvable for a fixed number of machines.

There are several approaches to allow for multitasking machines in ISMA. Mertzios et al. [9] consider a variant, called Interval Scheduling with Bounded Parallelism, in which all machines can concurrently process up to a given number of jobs. The authors consider two objectives: minimizing the total active time of the machines needed to process all jobs, and maximizing the number of processed jobs given a number of machines. Another approach to allow for multitasking machines was presented by Angelelli and Filippi [1]. They introduce Interval Scheduling with a Resource Constraint (FISRC), where every machine and every job is additionally characterized by a resource supply or demand.

Generalizations of interval scheduling include the well-studied Unsplittable Flow problem on a path (UFP). In UFP, instead of machines we have a resource capacity that can be used by all scheduled jobs and jobs have individual demands. While it is easy to decide whether all jobs can be scheduled, the optimization problem where we have to select a maximum cardinality or maximum weight subset of jobs is NP-hard (generalizing Knapsack) and the currently best result is a 5/3-approximation algorithm [7].

UFP has a geometric version called the Storage Allocation Problem (SAP) [2, 10] where all scheduled jobs have to be drawn as non-overlapping axis-parallel rectangles. SAP with uniform job demands corresponds to a multithread version of ISMA, where for each pair of machines either the availability interval of one machine is contained in the interval of the other or the two intervals are disjoint.

1.2 Our Contribution

To model the coworking scheduling problem, we propose a twofold generalization of the Interval Scheduling problem with Machine Availabilities (ISMA) [8].

For one thing, we switch to *multithread* machines, i.e., we allow machines to process several jobs at a time. In order to keep the machines equivalent, we require all machines to have the same capacity. Note that in this case, we may assume that the machines' end times are interchangeable.

In the setting of coworking scheduling, we assume that all rooms are identical, and that the transition time between the rooms for the cleaning team can be neglected, so the rooms can be cleaned in any order. However, every time the cleaning crew enters a new room in the evening, the room becomes unavailable for the customers. This implies that the periods during which the rooms are available for bookings are not fixed, since the cleaning start times can be permuted; the number of available rooms, in contrast, is fixed for any point in time.

Therefore, the second adjustment to ISMA is the possibility to permute the machine end times. In other words, the start and end times mainly depict information about when and how the number of available machines changes. However, we may still assume without loss of generality that every machine is preassigned a start time. The task is to assign every machine an end time and to schedule all jobs, or to decide that there is no such end time assignment and schedule. Due to the increased flexibility, we call the problem Flexible Multithread ISMA (FlexMISMA). Interestingly, despite the added flexibility, FlexMISMA turns out to be at least as hard as ISMA.

In this paper, we determine a tight classification of conditions that are required to obtain a polynomial time algorithm for FlexMISMA assuming P ≠ NP.

We start with analyzing the hardness depending on the number of machines. In Section 3.1 we provide a network flow interpretation of the problem. With its help we show that FlexMISMA can be solved in polynomial time if the number of machines is at most two. The general idea consists in computing a schedule for only one machine at a time using a MaxFlow Algorithm to find vertex-disjoint paths in a special graph. In a second step, we then transform the solution to a feasible solution for both machines.

However, this technique does not work if the number of machines is at least three. We show in Section 3.2 that such instances are NP-hard for FlexMISMA.

We continue by considering the case of machines with fixed capacity and show in Theorem 2 that FlexMISMA is NP-hard for machine capacity equal to two. To this end, we show that FlexMISMA is at least as hard as ISMA, which is known to be NP-hard [8]. For FlexMISMA with unit machine capacities, however, we show in Theorem 3 that the problem essentially boils down to solving an interval coloring instance and is thus solvable in polynomial time.

2 PROBLEM FORMULATION

We formulate the problem of coworking scheduling in terms of classical machine scheduling.

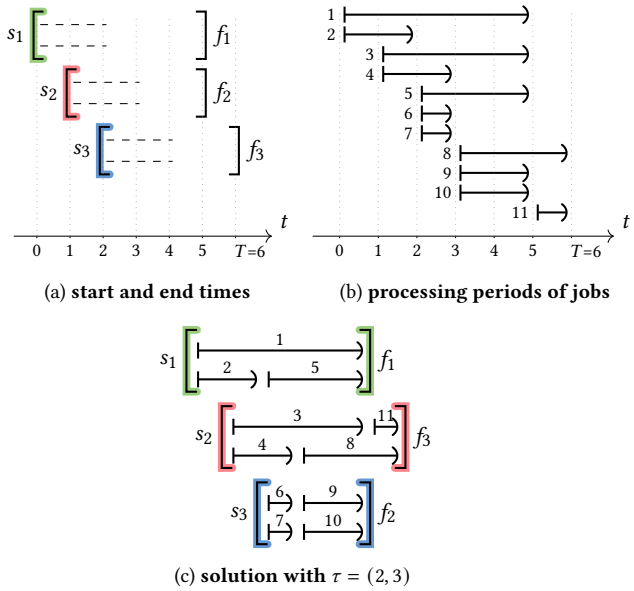
Interval Scheduling with Machine Availabilities (ISMA) is a known scheduling problem that incorporates a machine availability constraint into the Interval Scheduling problem. Specifically, ISMA assumes that every machine has a fixed availability period. Kolen et al. define ISMA as follows [8].

Definition 1 (ISMA)**.** Given m machines that are available in periods $[s_i, f_i)$ for $i \in [m]$, and n jobs that require processing in the periods $[a_j, b_j)$ for $j \in [n]$, ISMA asks for a schedule that respects the availability of each machine and schedules no two jobs with overlapping processing intervals onto the same machine.

Since ISMA assumes that every machine processes at most one job at a time, it is insufficient for modeling the entire coworking scheduling problem. We extend the formulation of ISMA to allow for *multithread* machines that can process several jobs simultaneously, which models the coworking offices hosting several desks.

Further, we integrate the interchangeability of the machines' end times (which represent the start times of the evening room cleaning). We formulate the problem in a general way by allowing arbitrary, not necessarily equidistant, start and end times. This leads to the following variant of Interval Scheduling where each machine has an assigned start time, and the end times are given but not preassigned to the machines.

Definition 2 (The Flexible Multithread ISMA problem (FlexMISMA))**.** An instance of the Flexible Multithread ISMA (FlexMISMA) problem is given by m machines, their capacity $C \in \mathbb{N}$, start times $(s_i)_{i\in[m]}$ for every machine, m end times $(f_i)_{i\in[m]}$ that still have to be assigned to a machine, n jobs, and the jobs' processing intervals $[a_j, b_j)$ for $j \in [n]$. FlexMISMA



(a) **start and end times** (b) **processing periods of jobs**

(c) **solution with** $\tau = (2, 3)$

Figure 1: An example of a FlexMISMA instance with $m = 3$ machines of capacity $C = 2$.

asks for two assignments: a bijective assignment $\tau\colon [m] \to [m]$ of machines to end times with $s_i \leqslant f_{\tau(i)}$ for all $i \in [m]$, and an assignment $\alpha\colon [n] \to [m]$ of jobs to machines such that every machine $i \in [m]$ processes at most C jobs simultaneously and only between its start and end time, i.e., $\left|\{j \in \alpha^{-1}(i) \mid t \in [a_j, b_j)\}\right| \leqslant C$ for all $t \in [s_i, f_{\tau(i)})$, and $s_i \leqslant a_j < b_j \leqslant f_{\tau(i)}$ for all $j \in \alpha^{-1}(i)$.

All input parameters are assumed to be integer. Without loss of generality, we assume that the earliest start time is 0, i.e., $0 = \min_{i\in[m]} s_i$, and we define the latest end time as $T := \max_{i\in[m]} f_i$.

Observe that we can consider every multi-thread machine of capacity C as a group of C single-thread machines that are required to have the same start and end times. Hence, FlexMISMA is an extension of ISMA in which the machines are partitioned into groups by availability and the end times must be equal within each group, but can be permuted between the groups.

Figure 1 shows an exemplary instance of FlexMISMA. This instance is given by three machines with capacity $C = 2$, and by the job set with start and end times as displayed in the figure. One feasible solution for the example instance is presented in Figure 1c.

The bijective function τ can also be interpreted as a permutation on set $[m]$. Therefore, we use in the following the permutation representation. For example, in the solution presented in Figure 1c, the end time assignment is given by the permutation $\tau = (2, 3)$.

The time scale of an instance of both ISMA and FlexMISMA can be compressed so as to contain only the *significant* time units – the time units which are start or end times of machines or jobs. The number of such time units is at most $2n + 2m$. Hence, we may assume that $T \leqslant 2n + 2m$. Rescaling an instance in this manner takes $O((n + m) \log(n + m))$ time.

Hence, we can check in polynomial time whether, at some point in time, more jobs need to be processed than there are machine threads available, as the number of available machines at each point in time can be derived from the start and end times. If that is the case, the instance is automatically infeasible. We thus assume in the following that at no point in time more jobs need to be processed than there are threads available. Note that this does not automatically imply feasibility. In the same way, we can verify in polynomial time whether all machines are utilized to full capacity. If this is not the case, we transform the instance by adding auxiliary jobs with processing intervals of length one for the respective underutilized time periods. This transformation has no influence on the feasibility of a solution and needs only a polynomial number of auxiliary jobs. Thus, we assume without loss of generality that every machine is utilized to full capacity in the available time period.

Finally, note that if there exist $i, k \in [m]$ so that $s_i = f_k$, we can simply remove them from the instance and reduce the number of machines by one. Otherwise, depending on the assignment of end times τ, we obtain either one machine with an empty availability period or two machines with adjoining availability periods. In the latter case, we can then combine those two machines to one machine with a longer availability period. Thus, we assume in the following $s_i \neq f_k$ for all $i, k \in [m]$.

The size of an instance of FlexMISMA is defined by three parameters: the number of jobs n, the number of machines m and the machine capacity C. Remarks above imply that the number of jobs is bounded from below by the total machine capacity, i.e., $n \geqslant m \cdot C$. We observe that the number of jobs is the main determinant of the size of an instance of FlexMISMA. In the following complexity study, we will focus on cases differentiated by values of parameters m and C, while the number of jobs remains unbounded.

3 COMPLEXITY FOR A CONSTANT NUMBER OF MACHINES

In this section, we start studying the complexity of FlexMISMA. We consider the case of a fixed number of machines, which corresponds to scheduling a coworking space with a constant number of rooms. The case of one machine is trivial, as it reduces to Interval Scheduling; therefore, we proceed with the case of two machines, which represents a coworking space with two equivalent rooms.

3.1 Polynomial-time algorithm for two machines

The assumption of full machine utilization implies that in any feasible solution for FlexMISMA, every machine processes exactly C jobs at any time unit of its availability period. We call a non-empty subset $\mathcal{J}$ of jobs *feasible for machine* $i \in [m]$ *and end time* $f \in [1, T]$, if for every time unit $s_i \leqslant t < f$, the set $\mathcal{J}$ contains exactly C jobs that must be processed at time t, i.e., if for all $t \in \mathbb{N}$

$$\left|\{j \in \mathcal{J} \mid t \in [a_j, b_j)\}\right| = \begin{cases} C, & \text{if } s_i \leqslant t < f, \\ 0, & \text{otherwise.} \end{cases}$$

Note that all constraints of FlexMISMA are satisfied for a machine with some assigned end time if the jobs of a corresponding feasible set are assigned to it. In general, finding a feasible job set for one machine and some end time is not sufficient to solve FlexMISMA. However, this is sufficient if an instance of FlexMISMA has only two machines.

Lemma 1. *For an instance of FlexMISMA with $m = 2$ machines, n jobs and end times (f_1, f_2), let the subset $\mathcal{J}_1 \subseteq [n]$ of jobs be feasible for machine 1 with end time $f_i \in \{f_1, f_2\}$. Denote by $f_{i'}$ the unique remaining end time, where $i' \neq i$. Then the set $\mathcal{J}_2 := [n] \setminus \mathcal{J}_1$ is feasible for machine 2 with end time $f_{i'}$.*

We omit the straightforward but technical proof of this Lemma.

As a consequence of Lemma 1, it suffices to find a feasible job set for one machine and end time in order to solve FlexMISMA for two machines. Therefore, we continue by proposing a method based on network flow for finding such a feasible job set.

First, observe that the assumption of full utilization implies that every job is immediately followed by another job, unless the former ends at some machine's end time, i.e., for a job $j \in [n]$ holds $b_j = f_i$ for some $i \in [m]$ or there exists a job j' with $a_{j'} = b_j$. If the latter holds true, we call job j' a *successor* of j.

We use this connection between jobs to construct a directed graph $G = (V, A)$ that represents the FlexMISMA instance. This graph is called the *successor graph* and contains three types of nodes: a source vertex u for every machine, a target vertex w for every end time, and a transit v vertex for every job, i.e.,

$$V := \{u_i, w_i \mid i \in [m]\} \cup \{v_j \mid j \in [n]\}.$$

The arcs of the network G reflect the succession relationship: For machine $i \in [m]$, we construct arcs between the source vertex u_i and all vertices v_j whose corresponding jobs start at the same time as i becomes available, i.e., $s_i = a_j$. For end time number $i \in [m]$, we construct arcs between the target vertex w_i and every vertex v_j whose corresponding job ends at f_i, i.e., $b_j = f_i$. For every two transit vertices v_j and $v_{j'}$, we construct an arc from v_j to $v_{j'}$ if and only if j' is a successor of j, i.e., $b_j = a_{j'}$. Therefore,

$$\begin{aligned} A :=& \{(u_i, v_j) \mid i \in [m],\ j \in [n],\ s_i = a_j\} \\ &\cup \{(v_j, w_i) \mid i \in [m],\ j \in [n],\ b_j = f_i\} \\ &\cup \{(v_j, v_{j'}) \mid j, j' \in [n],\ b_j = a_{j'}\}. \end{aligned}$$

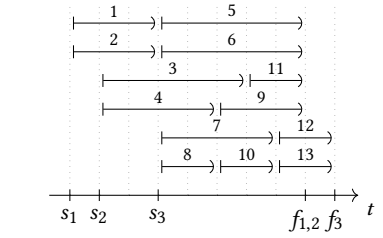
An exemplary FlexMISMA instance and its corresponding successor graph are shown in Figure 2. Remark that the successor graph is acyclic and has $|V| = 2 \cdot m + n$ vertices and $|A| = O(n^2 + m \cdot n)$ arcs. Therefore, its construction requires time polynomial in the size of the underlying FlexMISMA instance.

We use the successor graph to construct feasible job sets for the machines by computing families of vertex-disjoint u_i-$w_{i'}$-paths. Here, we use the term *disjoint* for internally vertex-disjoint paths.

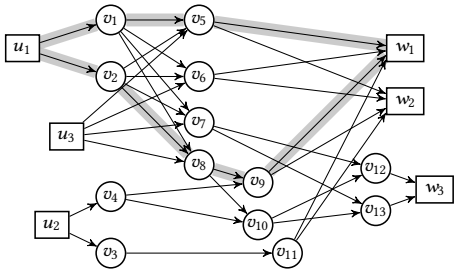
Lemma 2. *Let C vertex-disjoint u_i-$w_{i'}$-paths in the successor graph of a FlexMISMA instance be given, where u_i is a source node and $w_{i'}$ is a target node. Then the set of jobs represented by the nodes that are traversed by these paths is a feasible set for machine $i \in [m]$ and end time $f_{i'}$. Conversely, if there is a feasible job set for machine i and end time $f_{i'}$, then the successor graph contains C disjoint u_i-$w_{i'}$-paths.*

Proof. Let C u_i-$w_{i'}$-paths $\mathcal{P}_1, \ldots, \mathcal{P}_C$ be given in the successor graph, where $i, i' \in [m]$. For each path $\mathcal{P}_l$, with $l = 1, \ldots, C$, let $J_l \subseteq [n]$ be the set of corresponding jobs, i.e., $J_l = \{j \in [n] \mid v_j \in \mathcal{P}_l\}$. Let $\mathcal{J}$ denote the union of all these job sets:

$$\mathcal{J} := \bigcup_{l=1}^{C} J_l.$$



(a) **job set, start and end times**



(b) **successor graph**

Figure 2: Successor graph for the FlexMISMA instance with $m = 3$ and $C = 2$.

We show that the job set $\mathcal{J}$ satisfies the conditions of a feasible set for machine i. Since the paths are vertex-disjoint, the job sets J_l are pairwise disjoint as well. By construction of the successor graph, for each $l = 1, \ldots, C$, the jobs in J_l have disjoint processing intervals that cover in total exactly the interval $[s_i, f_{i'})$, i.e., for all $s_i \leqslant t < f_{i'}$ there exists exactly one job $j \in J_l$ with $t \in [a_j, b_j)$ and $[a_j, b_j) \subseteq [s_i, f_{i'})$ for all $j \in J_l$. As a result,

$$\left|\{j \in \mathcal{J} \mid t \in [a_j, b_j)\}\right| = \begin{cases} C, & \text{if } s_i \leqslant t < f, \\ 0, & \text{otherwise,} \end{cases}$$

holds true and $\mathcal{J}$ is a feasible set for machine i.

Conversely, let $\mathcal{J} \subseteq [n]$ be a feasible job set for machine $i \in [m]$ and end time $f_{i'}$. We explicitly construct the corresponding disjoint paths. First, we color the jobs in $\mathcal{J}$ with C colors so that intersecting jobs have different colors. Such a coloring exists by definition of a feasible job set. Next, we color the corresponding transit vertices in the successor graph accordingly. In addition, for easier notation, we assign all C colors to vertices u_i and $w_{i'}$.

Next, we show that every color class J_l, where $l \in [C]$, yields a u_i-$w_{i'}$-path. Notice that by definition of a feasible set for machine i, for every time unit t the set $\mathcal{J}$ contains C jobs spanning t. Therefore, at any time unit of the machine's availability period and for each color $l \in [C]$, there is a job colored with color l. Thus, in the successor graph, every transit vertex of color l is adjacent to exactly two other vertices of color l, to one by an outgoing and to one by an incoming arc. Additionally, the source vertex u_i and the target vertex $w_{i'}$ are adjacent each to exactly one transit vertex of color l. As a result, the vertices corresponding to job set J_l form a u_i-$w_{i'}$-path in the successor graph. Since the color classes are pairwise disjoint, so are the paths constructed from distinct color classes of $\mathcal{J}$. □

Lemma 1 and Lemma 2 imply that to solve FlexMISMA with two machines, it suffices to find a family of C disjoint paths with common source and target vertices in the successor graph, or to show that no such family exists. We suggest using a MaxFlow subroutine to search for such disjoint paths. A flow in a graph with unit edge capacities corresponds in general to a family of edge-disjoint paths. We use the commonly known transformation in order to ensure that the paths are also vertex disjoint: We split every transit vertex v_j into two vertices v_j^- and v_j^+ connected by an arc (v_j^-, v_j^+), and make all incoming arcs of the original vertex incident to v_j^- whereas the outgoing arcs of the original vertex become incident to v_j^+.

We use a subroutine that, given two vertices u and w and an integer C, finds a u-w-flow of value exactly C. It can be easily derived from MaxFlow solvers and runs in polynomial time. The procedure solving FlexMISMA with two machines works as follows after the successor graph is constructed. First, ask for a u_1-w_1-flow of value C. If there is no such flow, repeat the request for the target w_2 instead of w_1. If again no flow was found, abort — the instance is infeasible. Once a u_1-w_i-flow ϕ_1 of value C was found for some $i \in \{1, 2\}$, assign to machine 1 the end time f_i and all jobs $j \in [n]$ for which the corresponding node v_j was traversed by the flow. Next, delete the sub-graph induced by the flow ϕ_1 from the network. The remaining graph, called *reduced*, contains exactly one source node, u_2, and one target node, say w'. The remaining end time, as well as all remaining jobs, are assigned to machine 2. In this manner, the procedure not only finds two families of disjoint paths in the successor graph, but also directly constructs a solution for FlexMISMA.

The runtime of the procedure is determined by the runtime of the MaxFlow subroutine, which, for $m = 2$, is called at most two times. Hence, the procedure runs in polynomial time, and its correctness follows from Lemmata 1 and 2.

Naturally, the algorithm can be applied to instances with any constant number of machines while preserving the polynomial runtime. However, since Lemma 1 does not hold for three or more machines, the algorithm is not guaranteed to find a feasible solution. An example for which the algorithm fails is shown in Figure 2: If the highlighted flow is selected in the first iteration, then the reduced graph contains no further u-w-flow of value C. Nevertheless, due to its polynomial runtime and partial correctness, our algorithm is a promising heuristic for FlexMISMA.

3.2 FlexMISMA is NP-complete for three or more machines

In the previous section, we saw a polynomial time algorithm for FlexMISMA with two machines which loses its complete correctness for three or more machines. In fact, FlexMISMA is NP-complete for more than two machines.

Theorem 1. *FlexMISMA is NP-complete if the number of machines is fixed and greater than two.*

We prove the NP-completeness by a three-staged reduction. Due to the page limit, we present only a sketch of the proof. While the main ideas of the proof were already used by Garey et al. [6], we have to take care of some small but important differences.

First, we show that FlexMISMA is at least as hard as Multithread ISMA (MISMA) — an extension of ISMA which allows machines to have different capacities greater than one (but preserves fixed availability periods). The reduction is similar to the one presented further in Theorem 2. Next, we show that MISMA is at least as hard as the Permutation Partition problem (PPP), which is a decision problem on symmetric groups and is inspired by the *Word problem for Products of Symmetric Groups* introduced by Garey et al. [6].

PPP receives as input m numbers $C_i \in \mathbb{N}$, $i \in [m]$, a partition $\mathcal{L} := \{L_1, \dots, L_m\}$ of $[k]$, where $k = \sum_{i\in[m]} C_i$ and $|L_i| = C_i$, as well as t arbitrary subsets $P_u \subseteq [k]$ for $u \in [t]$. We further define the canonical partition $\mathcal{M} := \{M_1, \dots, M_m\}$ of $[k]$ via $M_i := \{r \in \mathbb{N} \mid \sum_{l=1}^{i-1} C_l < r \leqslant \sum_{l=1}^{i} C_l\} \subset [k]$ for $i \in [m]$ and the embedded permutation groups $G_u := \mathrm{Stab}([k] \setminus P_u) \subseteq S_k$ for $u \in [t]$. Here $\mathrm{Stab}(U)$ denotes the pointwise stabilizer of a set $U \subseteq [k]$, and S_k the symmetric group on k elements. PPP then asks whether there exists a permutation $\pi \in G_t \circ \dots \circ G_1$ such that $\pi(M_i) = L_i$ for all $i \in [m]$.

Finally, we prove the NP-completeness of PPP by a reduction from the Directed Vertex-Disjoint Paths problem.

4 COMPLEXITY FOR A CONSTANT MACHINE CAPACITY

In this section, we study FlexMISMA in the case of a fixed machine capacity C. We show that FlexMISMA can be solved in linear time for $C = 1$ using Interval Coloring, but is NP-complete for fixed capacities $C \geqslant 2$.

Theorem 2. *FlexMISMA is NP-complete if machine capacity is equal to* 2.

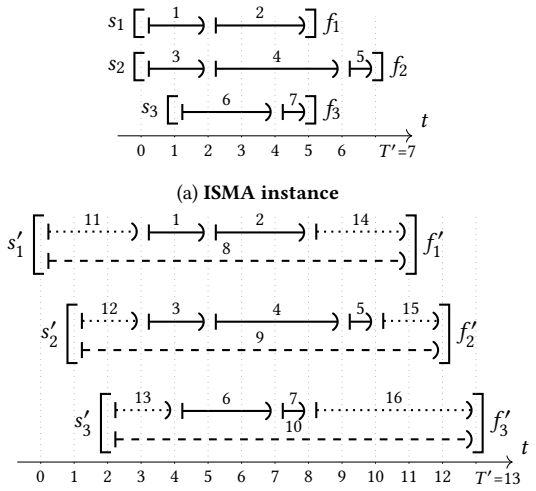
Proof. We present a reduction from ISMA, which was proven to be NP-hard [8]. Suppose that an instance of ISMA with m machines available in periods $[s_i, f_i)$ for $i \in [m]$ and n jobs with processing intervals $[a_j, b_j)$ for $j \in [n]$ is given. We denote the time horizon of this ISMA instance by $T := \max_{i\in[m]} f_i$.

Then, we construct an instance of FlexMISMA with m machines, with start times $s'_i := i$ and with end times $f'_i := T+m+i$ for $i \in [m]$, and with capacity $C' := 2$, see Figure 3. Note that, by construction, all start (end) times are pairwise different, and every machine has one thread more than its counterpart in ISMA. We further construct $n' := n + 3m$ jobs with the following processing intervals. We first shift the periods of jobs of the ISMA instance by m; then, we add m auxiliary jobs to occupy the additional threads; last, we add $2m$ jobs to pad the increased availability periods. We obtain

$$[a'_j, b'_j) := \begin{cases} [a_j + m, b_j + m), & j \in [n], \\ [s'_i, f'_i), & i \in [m],\ j = n + i, \\ [s'_i, s_i + m), & i \in [m],\ j = n + m + i, \\ [f_i + m, f'_i), & i \in [m],\ j = n + 2m + i. \end{cases}$$

It remains to prove that the constructed FlexMISMA instance is feasible if and only if the original ISMA instance is feasible.

Let $\alpha : [n] \to [m]$ represent a feasible solution to the ISMA instance. We extend this solution to a solution for FlexMISMA as follows: choose the end time assignment $\tau := \mathrm{id}_{[m]}$ and extend assignment α to $\alpha' : [n'] \to [m]$ by filling the additional threads



(a) **ISMA instance**

(b) **resulting FlexMISMA instance with capacity** $C' = 2$

Figure 3: Transformation of ISMA to FlexMISMA for an instance with three machines.

and extended availability periods with the additionally created jobs. By construction, the assignment

$$\alpha'(j) := \begin{cases} \alpha(j), & j \in [n], \\ i \in [m], & a'_j = s'_i \text{ or } b'_j = f'_i, \end{cases}$$

with time assignment $\tau = \mathrm{id}_{[m]}$ yields a feasible solution for the FlexMISMA instance.

Conversely, let $\alpha' : [n'] \to [m]$ and $\tau\colon [m] \to [m]$ represent a solution for the FlexMISMA instance. We still assume that all machines are utilized to full capacity during their entire availability period. Moreover, all start and end times of machines are distinct by construction. Therefore, all jobs j with $a'_j = s'_i$ or $b'_j = f'_{\tau(i)}$ for an $i \in [m]$ are necessarily assigned to machine i. Remark that, by construction, these are exactly the jobs $j \in [n'] \setminus [n]$. In particular, the jobs with availability periods $[s'_i, f'_i)$ guarantee that $\tau = \mathrm{id}$. Note that there exists at least one such job for every $i \in [m]$. Furthermore, these jobs occupy their assigned machine during the complete availability period. Therefore, every job $j \in [n]$ is assigned by α to a machine $i \in [m]$ during the remaining availability period $[s_i + m, f_i + m)$ with remaining capacity $C' - 1 = 1$. This corresponds one to one to the availability periods and machine capacities of the original ISMA instance. Thus, $\alpha'|_{[n]}$ is a feasible solution for the ISMA instance. □

It remains to consider FlexMISMA with machine capacity equal to one. We show an efficient algorithm for this case.

Theorem 3. *FlexMISMA with unit machine capacity is solvable in time linear in the number of jobs.*

Proof. In the case that every machine can process only one job at a time, FlexMISMA can be formulated as the well-known Interval Scheduling problem. Given an instance of FlexMISMA with n jobs and m machines, we transform it into an instance of Interval Coloring with $N := n + 2m$ intervals by representing start times s_i with intervals $(0, s_i)$ and end times f_i with intervals $(f_i, T + 1)$ for all $i \in [m]$.

The Interval Coloring problem is solved by a greedy algorithm in time linear in the number of intervals, provided that the endpoints of the intervals are sorted [4]. All interval endpoints in the instance of Interval Scheduling are non-negative integers not greater than $T+1$. Therefore, the counting sort can be applied, which has runtime in $O(2N + T + 1) = O(n)$ [5]. □

Remark that FlexMISMA with single-thread machines differs from ISMA only by the fact that permutations of the end times of machines are allowed. We saw that weakening this one constraint transforms the NP-hard ISMA into a polynomially solvable problem.

This observation completes the study of complexity of FlexMISMA with fixed machine capacity. We conclude that the coworking scheduling problem is polynomial time solvable for single-desk offices, but becomes NP-hard if offices have two or more desks each.

5 CONCLUSION

In this paper, we presented the interval scheduling extension FlexMISMA and provided a tight classification of its hardness with respect to the number of machines and their capacity. Furthermore, we provided constructive algorithms for the polynomial-time solvable cases.

There are natural optimization versions of the considered problem, which aim to find a maximum cardinality or maximum weight subset of jobs that can be scheduled. An interesting direction for future work is to find approximation algorithms for these problems. Furthermore, due to the machine capacities, it is sensible to consider jobs with individual demands, leading to a new problem closely related to the Storage Allocation Problem, see Section 1.1.

REFERENCES

[1] Enrico Angelelli and Carlo Filippi. 2011. On the complexity of interval scheduling with a resource constraint. *Theoretical Computer Science* 412, 29 (2011), 3650–3657.
[2] Reuven Bar-Yehuda, Michael Beder, and Dror Rawitz. 2017. A Constant Factor Approximation Algorithm for the Storage Allocation Problem. *Algorithmica* 77, 4 (2017), 1105–1127.
[3] P. Brucker and L. Nordmann. 1994. The k-track assignment problem. *Computing* 52, 2 (1994), 97–122.
[4] Martin C Carlisle and Errol L Lloyd. 1995. On the k-coloring of intervals. *Discrete Applied Mathematics* 59, 93 (1995), 225–235.
[5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2001. *Introduction To Algorithms* (2 ed.). MIT Press, Cambridge, Chapter 8.2.
[6] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. 1980. The Complexity of Coloring Circular Arcs and Chords. *SIAM Journal on Algebraic Discrete Methods* (1980). https://doi.org/10.1137/0601025
[7] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. 2018. A $(5/3+\epsilon)$-approximation for unsplittable flow on a path: placing small tasks into boxes. In *STOC*. ACM, 607–619.
[8] Antoon W.J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma. 2007. Interval scheduling: A survey. *Naval Research Logistics (NRL)* 54, 5 (2007), 530–543.
[9] George B. Mertzios, Mordechai Shalom, Ariella Voloshin, Prudence W.H. Wong, and Shmuel Zaks. 2015. Optimizing busy time on parallel machines. *Theoretical Computer Science* 562 (2015), 524–541. https://doi.org/10.1016/j.tcs.2014.10.033
[10] Tobias Mömke and Andreas Wiese. 2020. Breaking the Barrier of 2 for the Storage Allocation Problem. In *ICALP (LIPIcs, Vol. 168)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 86:1–86:19.

A modeling framework for designing dynamic multi-commodity network flows

Christian Puchert[1] and Tuomo Takkula[2]

[1]Ab Ovo Deutschland GmbH, Prinzenallee 9, 40549 Düsseldorf, Germany, ✉ christian.puchert@ab-ovo.com
[2]Ab Ovo Deutschland GmbH, Prinzenallee 9, 40549 Düsseldorf, Germany, ✉ tuomo.takkula@ab-ovo.com

1 Introduction

We address the problem how OR scientists may state periodic time-discrete multi-commodity network design models in a compact human-friendly way and solve them in a single application. Ab Ovo's *Network Designer* [3] comprises a modeling framework which allows to import, view, instantiate, solve such models, and inspect the solution and its graphical representation. In addition to the problem type and to the model above, the input format (Excel) also supports a number of advanced constraint types like zero-or-range, flow multipliers and convex quadratic constraints.

Dynamic networks are studied since a couple of decades and are used to model the behavior of networks and the flows in it over time—see [4] for a survey. Typical applications are material flows and transport logistics over time in networks with constrained capacity and varying speed and cost characteristics. Often facilities for storage of flow (with limited capacity) are added, and additions such as integrality constraints, parallel arcs, flow losses or minimum compulsory flow if used are not unheard of.

Network Designer came to use for instance in rail cargo empty wagon repositioning, and its versatility came to the fore when it was used to model the effects of the COVID-19 pandemic on the North-West German intensive care infrastructure [5]. Next to the applications just mentioned, demos exist also in finished vehicle logistics (intermodal, worldwide), reusable plastic Container (RPC) flow and ski resort design.

Network Designer is implemented in DELMIA Quintiq [1], a planning and optimization framework.

2 Mathematical Model

Network Designer permits to specify a multi-commodity network flow optimization problem [2] with linear and convex quadratic side constraints. Specifically, Network Designer considers a directed graph $G = (V, A)$ with nodes V and a loop-free set A of arcs, as well as a *horizon* defined as a finite set of *periods* in which the graph exists.

Hence, each node $v \in V$ and each arc $a \in A$ is defined periodically between a 'from' period a 'to' period within the horizon.

We further have a set C of commodities; each arc a is assigned a commodity group $C_a \subseteq C$, specifying the set of commodities that can be sent over it, and costs β_{ac} for each commodity $c \in C_a$.

Network Designer has the functionality to import such graphs, roll them out over time, meaning that each node and arc are copied for each period they are defined in. The optimization problem is then solved as a mixed-integer quadratic program (MIQP).

In the following, we provide the MIQP that Network Designer solves, and explain in particular the additional side constraints. For notational convenience, we omit the periods here:

$$\min \sum_{\substack{a \in A \\ c \in C}} \beta_{ac} x_{ac} + \sum_{a \in A^F} \beta_a^F y_a \tag{1}$$

$$\text{s.t.} \sum_{\substack{a=(u,v) \in A \\ c' \in \gamma_a^{-1}(c)}} m_{ac'} x_{ac'} = \sum_{a=(v,u) \in A} x_{ac} \qquad c \in C,\ v \in V \setminus (S \cup T) \tag{2}$$

$$\ell_a \le \sum_{c \in C} x_{ac} \le u_a \qquad a \in A \tag{3}$$

$$\ell_v \le \sum_{\substack{a=(u,v) \in A_v^S \\ c \in C}} m_{ac} x_{ac} + \sum_{\substack{a=(v,u) \in A_v^S \\ c \in C}} x_{ac} \le u_v \qquad v \in V \tag{4}$$

$$\ell_a^Z z_a \le \sum_{c \in C} x_{ac} \le u_a^Z z_a \qquad a \in A^Z \tag{5}$$

$$\sum_{c \in C} x_{ac} = f_a y_a \qquad a \in A^F \tag{6}$$

$$\ell_a^F \le y_a \le u_a^F \qquad a \in A^F \tag{7}$$

$$\ell_{A'}^K \le \sum_{\substack{a \in A' \\ c \in C}} x_{ac} \le u_{A'}^K \qquad A' \in \mathcal{A}^K \tag{8}$$

$$\sum_{(\alpha,a,c) \in L_j} \alpha x_{ac} + \sum_{(\alpha,a_1,c_1,a_2,c_2) \in Q_j} \alpha x_{a_1 c_1} x_{a_2 c_2} \le q_j \qquad j = 1, \dots, J \tag{9}$$

$$x_{ac} \ge 0 \qquad a \in A,\ c \in C \tag{10}$$

$$x_{ac} \in \mathbb{Z} \qquad a \in A,\ c \in C^I \tag{11}$$

$$y_a \in \mathbb{Z} \qquad a \in A^F \tag{12}$$

$$z_a \in \{0, 1\} \qquad a \in A^Z \tag{13}$$

Variables x model the flow, which in general is nonnegative (10), but must be integer for a subset $C^I \subseteq C$ (11). y and z model the multiplicity of the flow in constraints (6) and the choice in the zero-or-range constraints (5), respectively. The objective coefficients β_{ac} describe the flow costs, while β_a^F model unit costs per multiple, cf. (6).

We have flow conservation constraints (2) for each node which is neither in the set S of sources nor in the set T of sinks. Furthermore, commodities can be transformed on an arc a by a function $\gamma_a : C \to C$ and a multiplicator m_{ac}. On the left hand side, every unit of commodity c' which flows over arc a turns into $m_{ac'}$ units of commodity c when arriving at node v.

For each node v, we have a storage capacity constraint (4), limiting the amount of flow that can be sent through it on a subset $A_v^S \subseteq A$. Zero-or-range constraints (5) for an arc set $A^Z \subseteq A$ state that the flow over arc a must either be zero or within a range $\left[\ell_a^Z, u_a^Z\right]$.

Flow multiplier constraints (6) for an arc set $A^F \subseteq A$ enforce the flow over a to be an integer multiple of $f_a \in \mathbb{Z}$, where the range of multiples is given by (7). The multiplicity of is considered in the objective function (1) by a factor β_a^F .

Furthermore, we have a family $\mathcal{A}^K \subseteq \mathcal{P}(A)$ of arc subsets A', over which knapsack constraints (8) limit the total amount of flow over all arcs $a \in A'$ to be within the range $\left[\ell_{A'}^K, u_{A'}^K\right]$.

Finally, we have $J \in \mathbb{N}_0$ quadratic constraints (9) (required to be convex), containing linear terms L_j (with coefficient α, arc a and commodity c) and quadratic terms Q_j (again with coefficient α, but two arcs a_1 and a_2 and commodities c_1 and c_2). Note that these are the only non-linear constraints in the model, so if $Q_j = \emptyset$ for all j, the program is a MIP.

To conclude, we consider Network Designer to be the modeler's equivalent to the Swiss army knife.

References

[1] DELMIA Quintiq. `www.3ds.com/products-services/delmia/products/delmia-quintiq/`.

[2] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[3] Ab Ovo. Network Designer. Barbizonlaan 87, 2908 ME Capelle aan den IJssel, The Netherlands.

[4] Martin Skutella. An introduction to network flows over time. In *Research trends in combinatorial optimization*, pages 451–482. Springer, 2009.

[5] Tuomo Takkula. Modeling the SARS-CoV-2 pandemic within Ab Ovo's Network Designer framework. Technical report, Ab Ovo, 2020.

Session Session 1B: graph theory 1

Wednesday 8 June 2022, 10:45-12:25

Lecture Hall III

Max-Bisection in quadratic time for treewidth bounded graphs

Hauke Brinkop[1], Klaus Jansen[2], and Tim Weißenfels[3]

[1]Department of Computer Science, Kiel University, Kiel, Germany, ✉ hab@informatik.uni-kiel.de
[2]Department of Computer Science, Kiel University, Kiel, Germany, ✉ kj@informatik.uni-kiel.de
[3]Department of Computer Science, Kiel University, Kiel, Germany, ✉ tim.weissenfels@stu.uni-kiel.de

Unweighted Max-Cut is one of Karp's 21 NP-complete problems; given a graph $G = (V, E)$ it asks for a set $S \subseteq V$ such that the number of edges from S to $V \setminus S$ is maximal. In this talk we consider an even harder problem: (Weighted) Max-Bisection. Here we are given an undirected graph $G = (V, E)$ and a weight function $w\colon E \to \mathbb{R}_{>0}$ and the task is to find a set $S \subseteq V$ such that (i) the sum of the weights of edges from S is maximal; and (ii) S contains roughly $\frac{|V|}{2}$ vertices, that is $\big||S| - |V \setminus S|\big| \leq 1$.

Preliminaries We assume that we are given a tree decomposition of width t as part of the input; a tree decomposition is a classical decomposition of a graph into a tree structure that makes it easy to design algorithms for problems that have a specific locality, like cut problems. To design such an algorithm, the first step is usually to convert the decomposition into a very simple variant called *nice tree decomposition*, where each node has either no child (leaf nodes), has exactly one child (introduce and forget nodes), or has two childs (join nodes). When it has only one child, the bags corresponding to the nodes have to differ in exactly one vertex; if, in a bottom up manner, vertex v is added compared to the child's bag, then we say v is introduced and the node is e introduce node; forget nodes are defined analogously. Having obtained such a nice tree decomposition from a given tree decomposition in time $\mathcal{O}(nt^2)$ [7], we can obtain algorithms for cut-like problems by setting up a dynamic program.

We will use the following notation: for a node i of the tree decomposition, X_i is the corresponding bag, Y_i is the set of vertices in the subtree induced by i, that is the union of all bags associated to nodes below i (including i); and $F_i = Y_i \setminus X_i$. For the sum of the edge weights of edges between two disjunct sets A and B we write $\operatorname{size}(A, B)$.

Previous Algorithmic Results Jansen et al. [6] proposed an algorithm for Max-Bisection, which also works for Max-Cut as the same table has to be computed in both cases, which is the as follows for a node i of the decomposition:

$$\begin{aligned} &\Gamma_i\colon \{0, \dots, |F_i|\} \times 2^{X_i} \to \mathbb{R}_{>0} \\ &\Gamma_i(\ell, S) = \max_{\substack{\hat{S} \subseteq Y_i \\ |\hat{S}| = \ell \\ S \subseteq \hat{S}}} \operatorname{size}(\hat{S}, V \setminus \hat{S}). \end{aligned} \tag{1}$$

The idea is that for a node i the entry $\Gamma_i(\ell, S)$ is the size of the largest possible cut that consists of ℓ vertices from Y_i and includes the set $S \subseteq X_i$. For the root r of the tree decomposition we can compute our objective(s) by going through all entries of Γ_r and picking the best value (restricted to a subset of the in the case of Max-Bisection). A simple analysis of the algorithm yields the running time of $\mathcal{O}(2^t n^3)$; for every node of the decomposition, there might be $\mathcal{O}(2^t n)$ entries to be stored, and in join nodes results of the left and the right subtree have to be combined, which might take time $\Omega(n^2)$ in the worst case. Eiben et al. [4] improved this algorithm in its dependence on n by observing that for join node one basically has to compute the $(\max, +)$-convolution of two sequences. They modify the nice tree decomposition such that its height is bounded and use convolution (computable in $\mathcal{O}(n^2)$ for two sequences of length $\leq n$) to compute the entries in join nodes. While this might still cost $\Omega(n^2)$ for a single join node, the accumulated running time for all join nodes is significantly better; their overall running time is $\mathcal{O}(2^t t^5 n^2 \log n)$. Hanaka et al. [5] improved the analysis of the algorithm of Jansen et al. [6], yielding a running time of $\mathcal{O}(2^t (nt)^2)$. Again, the idea is that at a single join node might take time $\Omega(n^2)$ to compute its values but this cannot happen at all join nodes; the accumulated running time over all join nodes is in $\mathcal{O}(2^t (nt)^2)$.

Our Result We cautiously reformulate Equation 1

$$\Gamma_i \colon \{0, \dots, |F_i|\} \times 2^{X_i} \to \mathbb{R}_{>0}$$
$$\Gamma_i(\ell, S) = \max_{\substack{\hat{S} \subseteq F_i \\ |\hat{S}| = \ell}} \operatorname{size}(\hat{S} \cup S, V \setminus (\hat{S} \cup S)). \tag{2}$$

In comparsion to Equation 1 we now use $\Gamma_i(\ell, S)$ to store the value of the largest cut that consists of the set $S \subseteq X_i$ and ℓ further vertices that *occur in bags below i, but not in X_i*. We show that for a join node i with children j and k we only need time $\mathcal{O}(2^t |F_j \times F_k|)$ to compute Γ_i. By definition of a tree decomposition, the sets F_j and F_k do not share any vertex; this implies an overall running time of $\mathcal{O}(2^t n^2)$.

Hardness Results Lokshtanov et al. [8] proved that Max-Cut cannot be solved in time $\mathcal{O}((2 - \varepsilon)^t \operatorname{poly} n)$ for any $\epsilon > 0$, assuming SETH. Their result can be extended to our case, where a tree decomposition is given as advice, as they perform a reduction from 3-SAT to Max-Cut where the Max-Cut instances consist of graphs where such a decomposition can be computed in polynomial time. Using a standard approach (adding isolated vertices), Max-Cut can be reduced to Max-Bisection and thus the result also applies here; hence, the dependence on t is optimal in our approach. Eiben et al. [4] showed that for Max-Bisection (with a given tree decomposition), a truly subquadratic algorithm $\mathcal{O}(n^{2-\varepsilon})$ for some $\varepsilon > 0$ would imply a truly subquadratic algorithm for $(\max, +)$-convolution, which is considered unlikely [3].

Possible Generalizations and Future Outlook Our result can be extended to other cut problems, such as the minimization version of Max-Bisection, minimum edge expansion ($\min_{\emptyset \neq S \subset V} \frac{\operatorname{size}(S, V \setminus S)}{\min\{|S|, |V \setminus S|\}}$), sparsest cut ($\min_{\emptyset \neq S \subset V} \frac{\operatorname{size}(S, V \setminus S)}{|S||V \setminus S|}$), and densest cut ($\max_{\emptyset \neq S \subset V} \frac{\operatorname{size}(S, V \setminus S)}{|S||V \setminus S|}$). The "classical" dynamic programs of those are quite similiar to the one in [6]. As far as we know, the best known running time for sparsest cut was in $\mathcal{O}(2^t n^3)$ [1].

Also, allowing arbitrary weights as well as allowing edges to be directed (the decomposition then has to be a decomposition of the graph when all edges would be undirected) neither break the running time nor correctnes of our algorithm.

Eiben et al. [4] also considered a special case of Max-Bisection where all the weights are integers between 1 and W for a given $W \in \mathbb{N}$. In this case they achieve a running time of $\mathcal{O}(8^t \operatorname{poly}(tW) n^{1.864} \log n)$. The underlying idea is that $(\max, +)$-convolution can be solved in truly subquadratic time if the occuring numbers are bounded. It would be interesting to investigate if it is possible to design an algorithm with running time $\mathcal{O}(2^t n^{2-\varepsilon} \operatorname{poly}(t, W))$ or better for this special case; recall that for the general case, a truly subquadratic algorithm can exist only if there is such an algorithm for $(\max, +)$-convolution which is considered unlikely [3, 4].

References

[1] Paul S. Bonsma, Hajo Broersma, Viresh Patel, and Artem V. Pyatkin. "The complexity of finding uniform sparsest cuts in various graph classes". In: *J. Discrete Algorithms* 14 (2012), pp. 136–149. DOI: 10.1016/j.jda.2011.12.008.

[2] Hauke Brinkop, Klaus Jansen, and Tim Weißenfels. "An optimal FPT algorithm parametrized by treewidth for Weighted-Max-Bisection given a tree decomposition as advice assuming SETH and the hardness of MinConv". In: *CoRR* abs/2101.00694 (2021). arXiv: 2101.00694.

[3] Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. "On Problems Equivalent to (Min,+)-Convolution". In: *ACM Trans. Algorithms* 15.1 (Jan. 2019). DOI: 10.1145/3293465.

[4] Eduard Eiben, Daniel Lokshtanov, and Amer E. Mouawad. "Bisection of bounded treewidth graphs by convolutions". In: *J. Comput. Syst. Sci.* 119 (2021), pp. 125–132. DOI: 10.1016/j.jcss.2021.02.002.

[5] Tesshu Hanaka, Yasuaki Kobayashi, and Taiga Sone. "An Optimal Algorithm for Bisection for Bounded-Treewidth Graph". In: *Frontiers in Algorithmics*. Cham, 2020, pp. 25–36.

[6] Klaus Jansen, Marek Karpinski, Andrzej Lingas, and Eike Seidel. "Polynomial Time Approximation Schemes for MAX-BISECTION on Planar and Geometric Graphs". In: *SIAM J. Comput.* 35.1 (2005), pp. 110–119. DOI: 10.1137/S009753970139567X.

[7] Ton Kloks. *Treewidth, Computations and Approximations*. en. Vol. 842. Berlin/Heidelberg, 1994. DOI: 10.1007/BFb0045375.

[8] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal". In: *ACM Trans. Algorithms* 14.2 (2018), 13:1–13:30. DOI: 10.1145/3170442.

Weighted domination models and randomized heuristics

Lukas Dijkstra[1], Andrei Gagarin[1], and Vadim Zverovich[2]

[1]School of Mathematics, Cardiff University, Cardiff, UK, ✉ dijkstral@cardiff.ac.uk, gagarina@cardiff.ac.uk
[2]University of the West of England, Bristol, UK, ✉ vadim.zverovich@uwe.ac.uk

We consider the minimum weight and smallest weight minimum-size dominating set problems in vertex-weighted graphs. The latter is a two-objective optimization problem, which is concerned with optimizing both weight and cardinality of the dominating set. First, we reduce the two-objective optimization problem to the minimum weight dominating set problem by using Integer Linear Programming (ILP) formulations. Then, under different assumptions, we employ the probabilistic method to obtain new upper bounds on the minimum weight dominating sets in graphs. We also describe the corresponding randomized algorithms for finding small-weight dominating sets in graphs and use computational experiments to illustrate the results for two types of random graphs.

Weighted domination in graphs and networks can be used, for example, for modelling a problem of the placement of a number of transmitters in a communication network such that every site in the network either has a transmitter or is connected by a direct communication link to a site that has such a transmitter. There are usually some 'costs' associated with placing a transmitter in each particular location of the network, i.e. a vertex of the corresponding graph. The minimum weight dominating set problem usually does not place any restrictions on the size of the dominating set, i.e. the number of transmitters in this case – we only need to find a smallest weight/cost dominating set in a vertex-weighted graph. However, the total emitted radiation in the environment would be smaller with fewer transmitters installed. Similar facility location problems in road networks [2] and social networks [4] can be generalized to vertex-weighted graphs and two-criteria optimization as well. See also network problems in [5].

Given a simple graph G of order n, the weight assigned to each vertex v_i is denoted by w_i, $i = 1, ..., n$. The total weight of the graph, the minimum, maximum, and average vertex weights of G are denoted by w_G, $w_{\min}$, $w_{\max}$, and w_{ave}, respectively. The minimum vertex degree of G is denoted by $\delta = \delta(G)$. A set X of vertices of G is called a *dominating set* of G if every vertex not in X is adjacent to at least one vertex in X. The minimum cardinality of a dominating set of G is called the *domination number* of G and is denoted by $\gamma(G)$. We denote by $\gamma_w(G)$ the smallest weight of a dominating set in a graph G, and by $\gamma_w^*(G)$ the smallest weight of a minimum-cardinality dominating set X in G.

The problem of finding an exact value of $\gamma_w^*(G)$ and the corresponding dominating set X can be formulated as an ILP problem:

$$\begin{aligned} \text{minimize} \quad & z(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} \frac{w_i}{w_G} x_i \\ \text{subject to:} \quad & \sum_{v_i \in N[v_j]} x_i \geq 1, && j = 1, \ldots, n, \\ & x_i \in \{0, 1\}, && i = 1, \ldots, n, \end{aligned} \tag{1}$$

where a $(0,1)$-decision variable $x_i \in \{0,1\}$ is associated with each vertex $v_i \in G$ to indicate whether the vertex is in the solution set X or not, having $x_i = 1$ if and only if v_i is in X, $i = 1, ..., n$. Reassigning the graph vertex weights to $w_i' = 1 + \frac{w_i}{w_G}$, $i = 1, ..., n$, the ILP formulation becomes the single-objective optimization problem of finding $\gamma_{w'}(G)$ and the corresponding minimum weight dominating set in G with respect to the vertex weights w_i', $i = 1, ..., n$. At optimum, we have $\gamma_{w'}(G) = z^* = z(x_1^*, x_2^*, ..., x_n^*)$ and as $\sum_{i=1}^{n} \frac{w_i}{w_G} x_i^* < 1$, we also have $\gamma_w^*(G) = \sum_{i=1}^{n} w_i x_i^*$. In light of these results, the problems of finding $\gamma(G)$ and $\gamma_w^*(G)$ in G can be considered as particular cases of the more general problem of finding $\gamma_w(G)$ in G.

We use the probabilistic method to find several new upper bounds for $\gamma_w(G)$ in a graph G. These results are generalizations of the probabilistic method for the following classic upper bound for $\gamma(G)$:

Theorem 1. *[1, 3] For any graph G with $\delta \geq 1$,*

$$\gamma(G) \leq \left(1 - \frac{\delta}{(1+\delta)^{1+1/\delta}}\right) n.$$

The generalizations are based on the following general probabilistic construction and randomized algorithmic framework. First, given a certain probability p_i, $i = 1, ..., n$, we decide whether to include each vertex v_i of G into a set A, $i = 1, 2, ..., n$. Next, we consider the set of vertices that are not in A and do not have a neighbour in A, which is denoted by B. The set $D = A \cup B$ will then be a dominating set of G. By computing the expected total weight of the vertices in D, we obtain an upper bound for $\gamma_w(G)$. We use different ways to compute p_i to obtain the upper bounds and corresponding randomized algorithms. As the dominating set should have both small size and weight, we set $p_i = p \cdot x$, where p depends on vertex degrees in G, and x depends on vertex weights in G. By trying different expressions for x and optimizing the expected weight of D for p, we obtain the following upper bounds:

Theorem 2. *For any graph G with $\delta \geq 1$,*

$$\gamma_w(G) \leq \left(1 - \frac{\delta}{(1+\delta)^{1+1/\delta}}\right) w_G.$$

Theorem 3. *For a graph G with $\delta \geq 1$, $k = w_{\max}/w_{\text{ave}} \leq \delta + 1$, and $p = 1 - \left(\frac{k}{\delta+1}\right)^{1/\delta} \leq w_{\min}/w_{\max}$,*

$$\gamma_w(G) \leq npw_{\max} + \sum_{i=1}^{n} w_i (1-p)^{d_i+1} \leq \left(1 - \frac{\delta k^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) kw_G.$$

Theorem 4. *For a graph G with $\delta \geq 1$, $z = w_{\max}/w_{\min} \leq \delta + 1$, and $q = 1 - \left(\frac{z}{\delta+1}\right)^{1/\delta}$,*

$$\gamma_w(G) \leq qzw_G + \sum_{i=1}^{n} w_i (1-q)^{d_i+1} \leq \left(1 - \frac{\delta z^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) zw_G.$$

There are problem instances where the conditions of Theorem 3 are satisfied, but not those of Theorem 4 and vice versa. Also, Theorem 2 implicitly assumes that the ratio $w_{\max}/w_{\min}$ is reasonably close to 1.

We implemented and tested the deterministic and randomized heuristic solution methods for both problems on random graph instances of two types, one of which is the classic Erdös-Rényi random graph type, and the other is a random graph type used to prove asymptotic sharpness of the upper bounds of Theorem 1. Using the ILP formulations and a generic ILP solver (FICO® Xpress), the exact deterministic solutions to the problems of computing $\gamma_w^*(G)$ and $\gamma_w(G)$ were found in a reasonable amount of time of at most three hours for Erdös-Rényi random graphs of only at most 200 vertices, and the other type of graphs of at most 560 vertices. Then, three randomized heuristics based on Theorems 2, 3, and 4 were run on each of the random graph instances. In the case of the Erdös-Rényi random graphs, the three randomized algorithms performed similarly by the dominating set size, but the algorithm based on Theorem 2 was less successful when searching for better solutions by weight. For the other type of graphs, the algorithms corresponding to Theorems 3 and 4 performed better than that based on Theorem 2 by both parameters. Therefore, Theorems 3 and 4, whilst requiring stronger conditions, provide better randomized heuristics.

References

[1] N. Alon and J.H. Spencer. *The Probabilistic Method.* John Wiley & Sons Inc., New York, 1992.

[2] A. Gagarin and P. Corcoran. Multiple domination models for placement of electric vehicle charging stations in road networks. *Computers & Operations Research*, 96:69–79, 2018.

[3] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater. *Fundamentals of Domination in Graphs.* Marcel Dekker, New York, 1998.

[4] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan. On positive influence dominating sets in social networks. *Theoretical Computer Science*, 412(3):265–269, 2011.

[5] V. Zverovich. *Modern Applications of Graph Theory.* Oxford University Press, Oxford, 2021.

Rooted Maximum Weight Connected Subgraphs with Balancing and Capacity Constraints

Ralf Borndörfer
Zuse Institute Berlin
Berlin, Germany
borndoerfer@zib.de

Stephan Schwartz
Zuse Institute Berlin
Berlin, Germany
schwartz@zib.de

William Surau
Zuse Institute Berlin
Berlin, Germany
surau@zib.de

ABSTRACT

Finding connected subgraphs of maximum weight subject to additional constraints on the subgraphs is a common (sub)problem in many applications. In this paper, we study the Maximum Weight Connected Subgraph Problem with a given root node and a lower and upper capacity constraint on the chosen subgraph. In addition, the nodes of the input graph are colored blue and red, and the chosen subgraph is required to be balanced regarding its cumulated blue and red weight. This problem arises as an essential subproblem in district planning applications. We show that the problem is NP-hard and give an integer programming formulation. By exploiting the capacity and balancing condition, we develop a powerful reduction technique that is able to significantly shrink the problem size. In addition, we propose a method to strengthen the LP relaxation of our formulation by identifying conflict pairs, i.e., nodes that cannot be both part of a chosen subgraph. Our computational study confirms the positive impact of the new preprocessing technique and of the proposed conflict cuts.

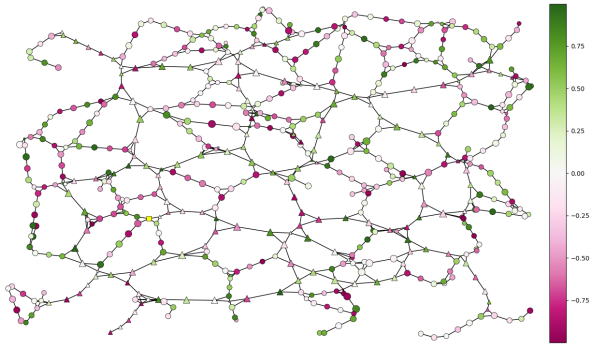


Figure 1: Exemplary BRCMWCS instance. The root node is depicted as a yellow square, circles represent nodes in V_b and triangles nodes in V_r, node sizes correspond to weights and colors to node profits.

1 INTRODUCTION

The Maximum Weight Connected Subgraph Problem (MWCS) is to find a node set of maximum cumulated weight that induces a connected subgraph in a given node-weighted graph. Popular variants of the MWCS include a given set of roots that have to be included in the chosen subgraph, and a capacitated (or budgeted) variant where additional node weights and lower and upper weight bounds for the chosen subgraph are specified. These variants occur in various applications, and have been the subject of a number of studies, see e.g. [2, 3, 11, 18].

In this paper, we study a variant of the rooted and capacitated MWCS where additional balancing constraints are imposed. Emerging from an application in district planning, the nodes are divided into blue and red nodes, and the chosen subgraph has to be balanced with respect to the color weights. The problem of balancing blue and red nodes in a connected subgraph has been studied, for instance, in [4, 20]. The combination of balanced, rooted, and capacitated MWCS, however, is new and interesting in its own right. It turns out that the combination of weight bounds and balancing constraints imposes a combinatorial structure that can be exploited to shrink the problem substantially and to derive logical implications on the shape of the subgraph.

The problem studied in this paper occurs as a subproblem when designing control districts for toll enforcement inspectors on motorways, see [6]. By a transition to the line graph of the motorway network, highway sections become nodes, whose lengths then correspond to node weights, and control districts are designed subject to lower and upper length bounds. In addition, the districts are desired to be homogeneous with respect to the traffic on its motorways, and homogeneity is measured as the cumulated difference to the median traffic of the district. Districts are generated dynamically in [6] and when fixing a root as potential median traffic node, all other nodes can be colored blue or red, representing motorways with less or, respectively, more traffic than the root node. Enforcing that the root has indeed the median traffic corresponds to the balancing condition of the BRCMWCS. The node profits stem from the duals of the restricted master problem, and are the only data that changes in each pricing round.

The contributions and the structure of this paper are as follows. In Section 2 we formalize the problem, review the literature, and present an IP formulation. In Section 3 we propose preprocessing methods that can drastically reduce the problem size. Section 4 is concerned with conflict pairs, i.e., pairs of vertices that cannot be both part of a feasible solution. Adding the resulting inequalities to the IP can considerably strengthen the LP relaxation. We conclude with a computational study in Section 5 that shows the strong effect of the preprocessing and the potential of the conflict cuts.

INOC 2022, June 7-10, 2022, Aachen, Germany

2 THE PROBLEM

For the *Balanced, Rooted, and Capacitated Maximum Weight Connected Subgraph Problem (BRCMWCS)*, we consider a graph $G = (V, E)$ with a bipartition of the nodes $V = V_b \dot\cup V_r$ into blue and red nodes, node weights $w \in \mathbb{R}^V_{\geq 0}$ and profits $p \in \mathbb{R}^V$, as well as numbers $0 \leq W_L \leq W_U$, $\Delta \geq 0$, and a root node $r \in V$. The BRCMWCS is to find a tree $T = (V_T, E_T)$ of weight $w(V_T) \in [W_L, W_U]$, such that $r \in V_T$ and $|w(V_T \cap V_b) - w(V_T \cap V_r)| \leq \Delta$, and such that $p(V_T)$ is maximized. Here and in the following, we use the short notation $w(V') := \sum_{v \in V'} w_v$ for a subset $V' \subseteq V$, and, accordingly, for $p(V')$.

Complexity. The BRCMWCS is NP-hard and the reduction can come from multiple angles. For instance, even without the balancing and capacity constraints, i.e., by setting Δ and W_U sufficiently large, a reduction from the rooted MWCS (which is NP-hard due to [2]) is possible. On the other hand, since the balanced connected subgraph problem is NP-hard (see [4]), a reduction is also possible without the root or capacity constraints. Note that if the rooted case could be solved in polynomial time, the unrooted version can be solved in polynomial time by considering every vertex as potential root node. Finally, the capacity contraints alone make the problem NP-hard. A reduction from the number partition problem is possible by considering a star graph where the leaf weights are the numbers of the partition instance and the center has weight 0. By setting $W_L = W_U$ to half of the sum of all weights, we see that the BRCMWCS is NP-hard even without the root or balancing condition.

2.1 Related Work

The MWCS is a classical optimization problem with close relations to the family of Steiner tree problems, see e.g. [12, 19, 22] for transformations and context. A number of papers from the last decade study preprocessing techniques [13, 22, 23], exact approaches [1, 14, 21], or applications [3, 8, 9] for the MWCS. The standard preprocessing approaches, however, do not carry over to the budgeted variant, since the bounds might require the inclusion of nodes with negative profit or the exclusion of nodes with positive profit in the optimal solution. Our paper adresses exactly this situation.

The budgeted MWCS with a lower bound has been considered in [15, 17, 18], but the nodes bear unit weights and $W_L = W_U = k$, i.e., the goal is a maximum weight connected subgraph with exactly k nodes. While Hochbaum and Pathria [15] propose a dynamic program that finds the optimal solution on trees and a $\frac{1}{k}$-approximation on general graphs, the authors of [18] reduce the problem to the single-rooted case which is heuristically solved in [17].

The rooted and budgeted MWCS has been studied in [2, 11], but only with an upper weight bound, i.e., $W_L = 0$. Both of these works focus on the comparison of different connectivity formulations. The results of Dilkina and Gomes [11] indicate that a single-commodity flow is best if the upper weight bound is impractically large, while in the other case, a formulation based on arc separation is preferred. Álvarez-Miranda et al. [2] propose a node separator formulation, and find that this formulation is favorable for denser graphs, compared to the arc separation. Comparisons of different connectivity formulations for the pricing problem in [6] strongly suggest that a single-commodity flow is best suited for the BRCMWCS.

For the *Balanced Connected Subgraph* Problem (BCS) we are given a graph $G = (V_b \dot\cup V_r, E)$ with nodes colored either blue or red, and seek a maximum-cardinality subgraph that contains an equal number of blue and red nodes. The problem was introduced in [4] and shown to be NP-hard, even on planar, bipartite, and chordal graph, or when a single root node is specified. When the graph is a tree, however, Bhore et al. [4] give a labelling algorithm to solve the BCS in time $O(|V|^4)$. Kobayashi et al. [16] improve the runtime to $O(|V|^2)$ by running a dynamic program on a transformed rooted binary tree with possibly additional uncolored nodes. The authors also briefly study a weighted version of BCS and give complexity results for special graph classes. Further complexity and inapproximability results as well as polynomial-time algorithms for the BCS on other special graph classes are provided in [4, 5, 10, 20]. We are not aware of any preprocessing or cutting plane approaches to the BCS.

2.2 IP Formulation for BRCMWCS

We use a flow formulation to ensure the connectivity of the chosen subgraph T. The idea is to construct a flow that emerges at the root node. Each node of the chosen subgraph consumes one unit of flow, while all other nodes satisfy flow conservation. To this end, we consider the bidirected version $D = (V, A)$ of G and introduce the variables

- $y_v \in \{0, 1\}$ for $v \in V$ indicating whether $v \in T$,
- $x_a \geq 0$ for $a \in A$ specifying the flow on arc $a \in A$.

The IP formulation of the BRCMWCS can then be stated as follows.

$$\max_{x,\, y} \quad \sum_{v \in V} p_v\, y_v \tag{1a}$$

$$\text{s.t.} \quad W_L \leq \sum_{v \in V} w_v\, y_v \leq W_U \tag{1b}$$

$$\sum_{v \in V_r} w_v\, y_v - \sum_{v \in V_b} w_v\, y_v \leq \Delta \tag{1c}$$

$$\sum_{v \in V_b} w_v\, y_v - \sum_{v \in V_r} w_v\, y_v \leq \Delta \tag{1d}$$

$$y_r = 1 \tag{1e}$$

$$\sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = y_v \qquad \forall v \in V \setminus \{r\} \tag{1f}$$

$$x_{uv} \leq M\, y_v \qquad \forall (u, v) \in A \tag{1g}$$

$$x_a \geq 0 \qquad \forall a \in A \tag{1h}$$

$$y_v \in \{0, 1\} \qquad \forall v \in V \tag{1i}$$

The objective (1a) is to maximize the profit of the chosen subgraph. The budget constraints are specified in (1b), and the balancing constraints in (1c) and (1d). The flow conservation is ensured by equalities (1f). Inequalities (1g) are necessary to activate every node that is used by the flow. The use of a big M parameter is bad for the LP relaxation, but necessary. It should be chosen as small as possible, and following [24], $M = \max_{V' \subseteq V} \{|V'| : w(V') \leq W_U\} - 1$ is a valid choice that can be computed with a greedy algorithm.

3 PREPROCESSING

Computational studies show that preprocessing methods for the MWCS generally have a huge impact on the solution time [13, 22, 23]. While the general methods for the MWCS do not carry over to the BRCMWCS, we propose an effective approach that significantly reduces the problem size and computation times for our problem.

The capacity constraints in combination with the balancing condition allow for different reduction techniques for the BRCMWCS. In particular, we can derive the following weight range for the chosen blue vertices $V_T \cap V_b$:

$$\frac{W_L - \Delta}{2} \leq w(V_T \cap V_b) \leq \min\left(\frac{W_U + \Delta}{2}, w(V_r) + \Delta\right). \quad (2)$$

The weight range for the red color class is defined accordingly, and we denote the respective upper weight bounds by W_U^b and W_U^r. A method to discard nodes based on these color weight ranges was proposed in [6]: For each color, we determine the color radius, i.e., the set of nodes that can be reached from the root on a path that satisfies the upper weight bound of the according color. The color radius can be determined with a single shortest path tree computation in the bidirected version of G using arc weights

$$\tilde{w}^b_{(u,v)} := \begin{cases} w_v & , \text{ if } v \in V_b, \\ 0 & , \text{ else,} \end{cases}$$

and $\tilde{w}^r$ defined accordingly. Every node that is outside of the color radius cannot be part of any feasible solution and can be removed.

Here, we propose a complementary preprocessing approach to discard even more nodes. The *bicolor radius* is the set of nodes that can be reached from the root on a path that satisfies the upper weight bound of both color classes. Unfortunately, the bicolor radius cannot be computed via a shortest path tree. In fact, it is essentially a constrained shortest path problem to determine if a specified node is inside the bicolor radius. We solve the problem with a Bellman-Ford-like labelling algorithm. The approach is detailed in Algorithm 1 and uses two-dimensional arc weights and node labels (for the blue and red cumulated weight, respectively). The upper weight bounds, given as a pair $\ell_{\max} = (W_U^b, W_U^r)$, are part of the input. We use the usual notion of domination, i.e., label ℓ_1 dominates label ℓ_2 if each weight in ℓ_1 is less than or equal to the corresponding weight in ℓ_2 and if at least one of the inequalities is strict. The algorithm is closely related to constrained shortest path labelling approaches and runs in pseudo-polynomial time.

The effect of the bicolor radius preprocessing can be substantial, and goes far beyond the single color radii. For the instance depicted in Figure 2, the single color radii cannot exclude a single node. The bicolor radius, on the other hand, is able to eliminate all gray nodes, essentially eliminating half of the graph.

4 CONFLICT PAIRS

In this section, we propose a method to identify and make use of conflict pairs in the BRCMWCS. A conflict pair consists of two nodes $u, v \in V$ that cannot be both part of a feasible solution, i.e., $u \in T \implies v \notin T$ for any feasible solution T. For any conflict pair (u, v) we can potentially tighten the LP relaxation of (1) with the inequality $y_u + y_v \leq 1$.

Algorithm 1: BicolorRadius

Input: $G = (V_b \dot\cup V_r, E)$, $w, r, \ell_{\max}$
Output: set of nodes within the bicolor radius

Consider arc weights ω in the bidirected version of G with $\omega_{(u,v)} \leftarrow \left(\tilde{w}^b_{(u,v)}, \tilde{w}^r_{(u,v)}\right)$;
$\ell_r \leftarrow \begin{cases} (w_r, 0) & , \text{ if } r \in V_b, \\ (0, w_r) & , \text{ else} \end{cases}$
$labels[v] \leftarrow \begin{cases} \{\ell_r\} & , \text{ if } v = r, \\ & , \text{ else} \end{cases}$;
$L \leftarrow \{(r, \ell_r)\}$;
repeat
  $L' \leftarrow$;
  for $(u, \ell_u) \in L$ **do**
    for *each neighbor* v *of* u *in* G **do**
      $\ell' \leftarrow \ell_u + \omega_{(u,v)}$;
      if ℓ' *respects* $\ell_{\max}$ *and is not dominated by any label at* v **then**
        Add ℓ' to $labels[v]$;
        Add (v, ℓ') to L' ;
        Remove dominated labels in $labels[v]$;
  $L \leftarrow L'$;
until $L =$;
return *all nodes with at least one label* ;

The idea of analyzing conflicting pairs (or even larger sets), and deriving stronger constraints has been proven to be very useful in different set packing problems, such as knapsack or matching problems. The proposed idea also translates to the rooted and budgeted MWCS, i.e., without the colors and balancing condition.

4.1 Finding Conflict Pairs

In order to identify conflict pairs, we can make use of the upper capacity bounds introduced in Section 3 again. If for two nodes u

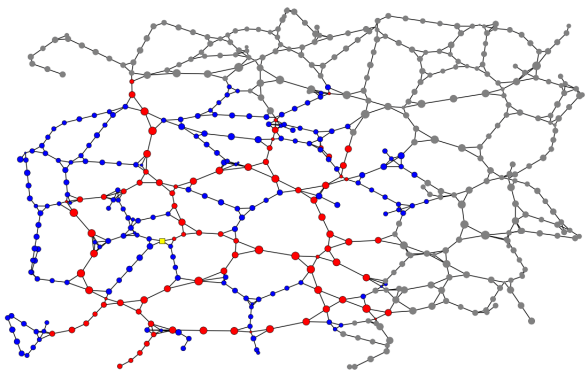


Figure 2: Effect of bicolor radius preprocessing. The root node is depicted as a yellow square, the gray nodes are outside of the bicolor radius and can be removed.

Algorithm 2: SteinerTreeConflictPairs

Input: $D = (V, A)$, r, $\tilde{w} \in \mathbb{R}^A_{\geq 0}$, $w_{\max}$
Output: set C of conflict pairs

```
1  C ← ;
2  len ← all pairs shortest path lengths in D w.r.t. w̃ ;
3  for all node pairs u, v ∈ V \ {r} do
4      for c ∈ V do
5          weight ← len[(r, c)] + len[(c, u)] + len[(c, v)] ;
6          if weight ≤ w_max then
7              // there is a feasible Steiner tree
8              Go to line 3 and check the next node pair ;
9      Add (u, v) to C ;
10 return C ;
```

and v and for every tree $T = (V_T, E_T)$ containing r, u, and v, we know that $w(V_T) > W_U$ or $w(V_T \cap V_b) > W_U^b$ or $w(V_T \cap V_r) > W_U^r$, then (u, v) is a conflict pair. Deciding if there exists a tree connecting a given set of terminal vertices at some cost, is a Steiner tree problem. Fortunately, an elegant combinatorial approach is known for the Steiner tree problem with three terminals. It is based on the insight that any Steiner tree with three terminals is a union of paths from a common center node (that is possibly a terminal itself) to the terminals.

Building on this, Algorithm 2 describes our procedure to identify conflict pairs. As input, we use the bidirected version D of G, the root node r, and one of the following three weight combinations:

- $\tilde{w}$ with $\tilde{w}_{(u,v)} = w_v$, $w_{\max} = W_U - w_r$,
- $\tilde{w}^b$, $w_{\max} = W_U^b - w_r \cdot \mathbf{1}_{V_b}(r)$,
- $\tilde{w}^r$, $w_{\max} = W_U^r - w_r \cdot \mathbf{1}_{V_r}(r)$,

where $\mathbf{1}$ is the indicator function. The union of the three respective return values of Algorithm 2 constitutes our set of conflict pairs. The proposed algorithm runs in time $O(|V|^3)$, and for all tested instances, it is able to identify a large number of conflict pairs. In fact, in most cases there are so many conflict pairs that some strategy is needed to exploit this information. We will discuss two approaches: Deriving large conflict sets and identifying essential conflicts.

4.2 The Conflict Graph

The individual conflict pair inequalities $y_u + y_v \leq 1$ can be too weak to effectively strengthen the LP relaxation. We construct the *conflict graph* on the vertex set V by introducing an edge for every conflict pair. Analogously to the set packing problem (see [7] for details), we can derive stronger inequalities from the conflict graph: Given an odd cycle C of length $2k + 1$ in the conflict graph, the inequality $\sum_{v \in C} y_v \leq k$ is known to be stronger than the ordinary conflict pair inequalities. These odd-cycle cuts, however, usually do not help the optimization process. The situation is different when considering a clique C in the conflict graph. The clique cuts $\sum_{v \in C} y_v \leq 1$ are known to be often beneficial for the LP relaxation.

We experimented with including clique cuts to formulation (1). Since the number of cliques is even much larger than the number of conflict pairs, we determined an edge covering with cliques, and only added the respective clique cuts. These additional inequalities, however, showed to have a negative effect on the solution time in our tests. A typical conflict clique together with an optimal solution for the BRCMWCS instance is depicted in Figure 3. One can see that the conflicting nodes are quite far apart and located at the boundary of the graph. Depending on the node profits of the instance, such clique cuts are rarely violated by the optimal LP relaxations. Hence, we shift our focus to identifying more meaningful conflict pairs.

4.3 Essential Conflicts

Our experiments showed that the addition of all conflict cuts results in significantly longer solution times. The same holds true if only violated cuts are added dynamically to the program. Hence, it is necessary to identify essential conflicts that actually facilitate the solution process. Such conflicts presumably involve nodes that are closer to the root node or have a high profit.

In order to specify these nodes, we define a scoring function that assigns a value from the interval $[-1, 1]$ to every vertex. A positive profit as well as a small ratio between the shortest r-v-path length and W_U increase the score of node v. Essential conflicts are then defined as all conflict pairs between nodes with a positive score. Figure 4 shows the node scores and all resulting essential conflicts for an exemplary instance. Observe that the conflict sets are now much more central. Again, we determine an edge clique cover of the essential conflict graph and only added the respective clique cuts.

5 COMPUTATIONAL RESULTS

In our computational study, we evaluate the impact of the bicolor preprocessing and the conflict pair cuts. We ran the experiments on machines equipped with Intel Xeon E3-1234 CPUs with 3.7GHz and 32GB RAM. The code is written in Python 3.6 and to solve IPs Gurobi 9.1 is used. The instances stem from a transit network generator used in [6]. These networks are edge weighted with a length and a traffic value. We transform the transit networks into node weighted graphs. The weight of a node represents the length of the corresponding edge. As a root we chose a random node and

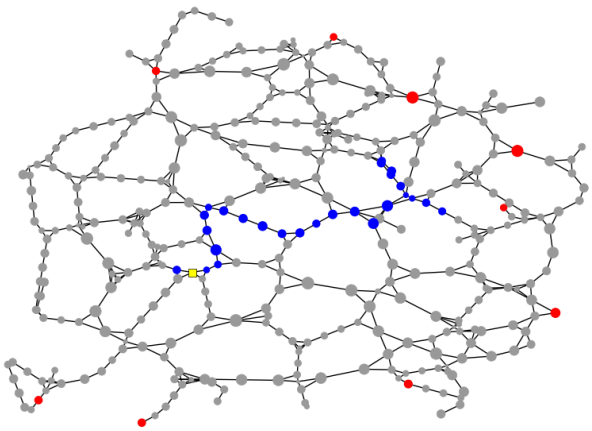


Figure 3: The nodes of an exemplary conflict clique are colored red, an optimal solution is colored blue.

Table 1: Reduction effect of the preprocessing and numbers of (essential) conflict pairs for instances grouped with respect to the four different weight bounds `W1`-`W4`.

group	*rp*	*cp*1	*ecp*1	*cp*2	*ecp*2
`W1`	227.7	13624.1	110.0	30426.5	146.6
`W2`	95.2	11132.1	103.0	46410.6	175.9
`W3`	25.3	4033.5	56.7	38420.7	146.0
`W4`	3.7	156.1	6.8	18688.0	78.2

the color of a node depends on whether the traffic value is higher or lower than the traffic at the root. Finally, Δ is set to the weight of the root node.

We constructed 5 such network instances where the number of nodes ranges from 563 to 569 and the number of edges from 845 to 885. For each network, we consider 9 different profits: 3 are actual reduced costs from the pricing problem from [6], 3 are distributed uniformly at random in the interval $[-1, 1]$, and 3 are based on the normal distribution to thin out extreme profits. For one instance per profit class, we also used a random partition into red and blue vertices. Finally, we consider 4 different upper weight bounds that lie between 40% and 75% of the graph diameter, and are motivated by our application. The lower bounds were set to half of the upper bounds but showed no effect in any computation. Altogether we have $5 \cdot 12 \cdot 4 = 240$ test instances.

The base variant of the study is the formulation without preprocessing and conflict cuts, which is denoted by `00`. To assess the impact of the proposed preprocessing, we consider the variant `10`. For a broader perspective, we consider two sets of conflict pairs generated by Algorithm 2: The set *cp*1 is the union of the return values for the input $\tilde{w}^b$ and $\tilde{w}^r$. The set *cp*2, on the other hand, is the combined output of all three weight combinations. The essential conflict sets *ecp*1 and *ecp*2 are formed by the presented scoring function. To evaluate the impact of the conflict pairs, we consider the variant `11` where, in addition to the preprocessing, conflict cuts

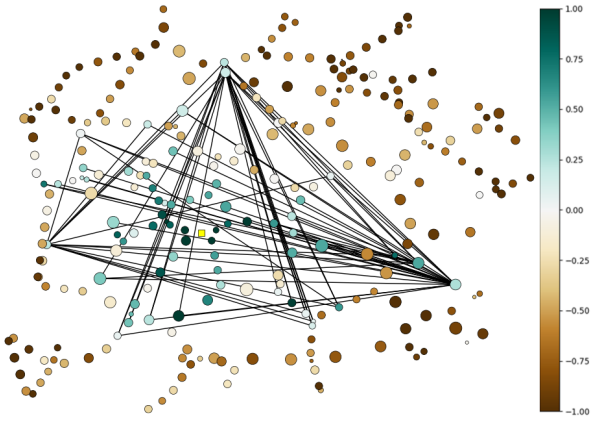


Figure 4: Essential conflicts and node scores of an exemplary instance.

Table 2: Comparison of average computation times (in seconds) for different variants and instance groups.

group	`00`	`10`	`11`	`12`	`1*`
`small`	42.3	22.7	23.8	23.9	21.6
`medium`	255.8	170.3	171.3	176.2	149.8
`large`	1996.7	1775.9	1527.6	1478.1	1188.0

to the set *ecp*1 are added, and the variant `12` defined accordingly with the set *ecp*2.

Table 1 shows the impact of the preprocessing and the number of generated (essential) conflict pairs. The instances are partitioned into four groups corresponding to the upper weight bounds. We report on the average numbers of removed nodes by the preprocessing and of (essential) conflict pairs for the two described variants. Overall, we observe that with increasing upper weight bound, the effect of the preprocessing and the number of conflict pairs decrease. Also observe that most conflict pairs are found with respect to $\tilde{w}$, and that the reduction to essential conflict pairs is significant in either case. Finding a decent subset of conflict pairs that support the solution process is a big challenge. We proposed a scoring function to determine a set of essential conflicts. While our computational results show that this approach is already beneficial in many instances, we like to point out that the potential of the conflict cuts is even larger. To this end, we consider another hypothetical variant, `1*`, that assumes for every instance the preferred cuts from *ecp*1 or *ecp*2. Consequently, the variant `1*` is attributed the minimum runtime of `11` and `12` for each instance. This variant essentially simulates a superior way to determine essential conflicts.

In order to compare the variants, we partition the 240 instances into three groups based on the magnitude of the computation time for the base variant: `small` (123 instances), `medium` (91 instances), and `large` (26 instances). All instances and instance-wise computational results are available in an online supplement[1] to this article. Table 2 contains the cumulated results. We report the geometric mean of the runtimes in seconds for the respective groups and variants. We opt for the geometric instead of the arithmetic mean to lessen the impact of outliers. In particular, there is one large instance that profits immensely from the additional cuts (speed-up factor 40).

We can observe that the preprocessing clearly helps to reduce the average computation times. While there are also instances where the preprocessing has a smaller or even a negative effect, the positive impact of the proposed method prevails in all groups. In terms of the conflict cuts, the impact is not so obvious. On average, for the small and medium instances, the cuts together with the preprocessing are slightly worse than the preprocessing alone. For the large instances, most of the positive impact admittedly stems from the single outlier instance. Nevertheless, a smaller positive effect remains when excluding this instance.

The average values, however, do not paint the whole picture. On many instances, one of the conflict cut variants is significantly better than variant `10`, on many other instances, the reverse is true. Interestingly, the variants `11` and `12` are often complementary, i.e.,

[1] https://github.com/stephanschwartz/brcmwcs

one set of essential cuts is much more favorable than the other. In order to assess the potential of the conflict cuts, variant 1⋆ serves as an oracle that always chooses the better of the two essential conflict sets. Indeed, we find that on average, this variant performs best in all groups. Furthermore, the positive impact increases for harder instances.

When considering the different variants over all instance classes, there is no clear indication on which classes some variant performs better than another. We cannot observe that a certain variant is dominant on a specific network instance. The same holds true for a specific profit class or weight bound. Even for the 15 instances that have a sibling only varying in node colors, the performance of the variants is vastly different for the pairs. Thus, it remains open in which cases the single variants are best. As one can expect, however, the computation time increases with larger (upper) weight bounds, since the number of feasible subgraphs drastically increases.

6 CONCLUSIONS AND OUTLOOK

In this paper, we studied a variant of the Maximum Weight Connected Subgraph problem that arises as a subproblem in a districting application. A distinct feature of the problem is that the chosen subgraph has to meet a given cumulated weight range and is required to have a similar weight of red and blue nodes. We use these features to develop a preprocessing method that is able to considerably reduce the average problem size and computation time.

In addition, we propose a method to identify node pairs that cannot be both part of a feasible solution. By considering the emerging conflict graph, we can strengthen the packing condition. The main problem, however, is to identify an appropriate subset of conflict pairs to add. To this end, we introduce a node scoring function that favors nodes close to the root and nodes with positive profit. Employing this scoring function, we generate two different sets of essential conflicts that are added to the IP formulation. Our computational results show that each of the sets individually has a positive impact on a number of instances. If we artificially choose the better set for each instance, the advantage of the conflict cuts becomes obvious. This suggests that the conflict cut approach is well suited for this problem, and that future work should aim to discover which conflict sets are essential for a given instance.

REFERENCES

[1] Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel. 2013. The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization*. Springer, 245–270.

[2] Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel. 2013. The rooted maximum node-weight connected subgraph problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. Springer, 300–315.

[3] Christina Backes, Alexander Rurainski, Gunnar W Klau, Oliver Müller, Daniel Stöckel, Andreas Gerasch, Jan Küntzer, Daniela Maisel, Nicole Ludwig, Matthias Hein, Andreas Keller, Helmut Burtscher, Michael Kaufmann, Eckart Meese, and Hans-Peter Lenhof. 2012. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic acids research* 40, 6 (2012), e43.

[4] Sujoy Bhore, Sourav Chakraborty, Satyabrata Jana, Joseph SB Mitchell, Supantha Pandit, and Sasanka Roy. 2019. The balanced connected subgraph problem. In *Conference on Algorithms and Discrete Applied Mathematics*. Springer, 201–215.

[5] Sujoy Bhore, Satyabrata Jana, Supantha Pandit, and Sasanka Roy. 2019. Balanced connected subgraph problem in geometric intersection graphs. In *International Conference on Combinatorial Optimization and Applications*. Springer, 56–68.

[6] Ralf Borndörfer, Stephan Schwartz, and William Surau. 2021. *Vertex Covering with Capacitated Trees*. Technical Report 21–25. Zuse Institute Berlin.

[7] Ralf Borndörfer and Robert Weismantel. 2000. Set packing relaxations of some integer programs. *Mathematical Programming* 88, 3 (2000), 425–450.

[8] Rodolfo Carvajal, Miguel Constantino, Marcos Goycoolea, Juan Pablo Vielma, and Andrés Weintraub. 2013. Imposing connectivity constraints in forest planning models. *Operations Research* 61, 4 (2013), 824–836.

[9] Chao-Yeh Chen and Kristen Grauman. 2012. Efficient activity detection with max-subgraph search. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1274–1281.

[10] Benoit Darties, Rodolphe Giroudeau, König Jean-Claude, and Valentin Pollet. 2019. The Balanced Connected Subgraph Problem: Complexity Results in Bounded-Degree and Bounded-Diameter Graphs. In *International Conference on Combinatorial Optimization and Applications*. Springer, 449–460.

[11] Bistra Dilkina and Carla P Gomes. 2010. Solving connected subgraph problems in wildlife conservation. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, 102–116.

[12] Marcus T Dittrich, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. 2008. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics* 24, 13 (2008), 223–231.

[13] Mohammed El-Kebir and Gunnar W Klau. 2014. Solving the maximum-weight connected subgraph problem to optimality. 11th DIMACS Implementation Challenge Workshop. (2014).

[14] Matteo Fischetti, Markus Leitner, Ivana Ljubić, Martin Luipersbeck, Michele Monaci, Max Resch, Domenico Salvagnin, and Markus Sinnl. 2017. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation* 9, 2 (2017), 203–229.

[15] Dorit S Hochbaum and Anu Pathria. 1994. Node-optimal connected k-subgraphs. *manuscript, UC Berkeley* (1994).

[16] Yasuaki Kobayashi, Kensuke Kojima, Norihide Matsubara, Taiga Sone, and Akihiro Yamamoto. 2019. Algorithms and Hardness Results for the Maximum Balanced Connected Subgraph Problem. In *International Conference on Combinatorial Optimization and Applications*. Springer, 303–315.

[17] Heungsoon Felix Lee and Daniel R Dooly. 1996. Algorithms for the constrained maximum-weight connected graph problem. *Naval Research Logistics (NRL)* 43, 7 (1996), 985–1008.

[18] Heungsoon Felix Lee and Daniel R Dooly. 1998. Decomposition algorithms for the maximum-weight connected graph problem. *Naval Research Logistics (NRL)* 45, 8 (1998), 817–837.

[19] Ivana Ljubić. 2020. Solving Steiner trees: Recent advances, challenges, and perspectives. *Networks* (2020), 177–204.

[20] T Martinod, V Pollet, B Darties, R Giroudeau, and J-C König. 2021. Complexity and inapproximability results for balanced connected subgraph problem. *Theoretical Computer Science* (2021).

[21] Daniel Rehfeldt, Henriette Franz, and Thorsten Koch. 2020. *Optimal Connected Subgraphs: Formulations and Algorithms*. Technical Report 20–23. Zuse Institute Berlin.

[22] Daniel Rehfeldt and Thorsten Koch. 2019. Combining NP-hard reduction techniques and strong heuristics in an exact algorithm for the maximum-weight connected subgraph problem. *SIAM Journal on Optimization* 29, 1 (2019), 369–398.

[23] Daniel Rehfeldt, Thorsten Koch, and Stephen J Maher. 2019. Reduction techniques for the prize collecting Steiner tree problem and the maximum-weight connected subgraph problem. *Networks* 73, 2 (2019), 206–233.

[24] Hamidreza Validi and Austin Buchanan. 2021. Political districting to minimize cut edges. (2021). http://www.optimization-online.org/DB_HTML/2021/04/8349.html

On the Mixed Connectivity Conjecture of Beineke and Harary

Manuel Streicher[1], Sebastian Johann[1], and Sven O. Krumke[1]

[1]Department of Mathematics, TU Kaiserslautern, Germany, ✉ streicher@mathematik.uni-kl.de

January 6, 2022

The conjecture of Beineke and Harary states that for any two vertices which can be separated by k vertices and l edges, but neither by k vertices and $l-1$ edges nor $k-1$ vertices and l edges, there are $k+l$ edge-disjoint paths connecting these two vertices of which $k+1$ are internally disjoint. We prove this conjecture for $l = 2$ and every $k \in \mathbb{N}$.

Connectivity is an extensively studied property of graphs. A well-known Theorem of Menger establishes equality between the vertex connectivity for a given pair of non-adjacent vertices and the maximum number of internally disjoint paths between this pair as well as the edge connectivity for a given pair of vertices and the maximum number of edge-disjoint paths between this pair. There are many variation and extensions of Menger's Theorem. For example Aharoni and Berger [1] proved a version of Menger's Theorem for infinite graphs and Borndörfer and Karbstein [4] interpreted and proved Menger's Theorem in hypergraphs. Here we focus on a form of connectivity in which vertices and edges may be removed at the same time. One variant of *mixed connectivity* was considered by Egawa, Kaneko and Matsumoto [5]. They prove the following mixed version of Menger's Theorem: Between two vertices v, w of a graph there are λ edge-disjoint unions of k internally disjoint paths if and only if for each set S of $0 \leq r \leq \min\{k-1, |V(G)|-2\}$ vertices the graph $G - S$ contains $\lambda(k-r)$ edge-disjoint v-w paths.

Beineke and Harary [2] proposed an alternative form of mixed connectivity between pairs of vertices. They call a pair of non-negative integers (k, l) *connectivity pair* for distinct vertices s and t if they can be separated by removing k vertices and l edges, but neither by k vertices and $l-1$ edges nor $k-1$ vertices and l edges. In the same publication Beineke and Harary claim to have proved a mixed version of Menger's Theorem: If (k, l) is a connectivity pair for s and t, then there exist $k+l$ edge-disjoint s-t paths k of which are internally disjoint. Mader [8] pointed out that the proof is erroneous.

More recently, mixed connectivity has been revisited by Erveš and Žerovnik [7], who regard transferrance of mixed connectivity along cartesian graph products and bundles, and Bonnet and Cabello [3] who regard the parametrized complexity of mixed connectivity.

The main focus here, however, is the conjecture by Beineke and Harary. The most meaningful result on the conjecture to date is due to Enomoto and Kaneko [6]. They first extended the conjecture claiming that it is possible to find $k+1$ internally disjoint paths instead of just k under the additional assumption that $l \geq 1$ and then proved their statement for certain k and l.

Theorem (Enomoto and Kaneko [6])**.** *Let q, r, k and l be integers with $k \geq 0$ and $l \geq 1$ such that $k + l = q(k+1) + r$, $1 \leq r \leq k+1$, and let s and t be distinct vertices of a graph G. If $q + r > k$ and if (k, l) is a connectivity pair for s and t, then G contains $k + l$ edge-disjoint s-t paths of which $k+1$ are internally disjoint.*

From our studies the following conjecture originally formulated by Beineke and Harary and extended by Enomoto and Kaneko may hold.

Conjecture (Beineke-Harary-Conjecture)**.** *Let G be a graph, $s, t \in V(G)$ distinct vertices and k, l non-negative integers with $l \geq 1$. If (k, l) is a connectivity pair for s and t in G, then there exist $k + l$ edge-disjoint paths, of which $k+1$ are internally disjoint.*

We prove this conjecture for $l = 2$ and any $k \in \mathbb{N}$. It is worth noting that for $l = 2$ the conjecture has *not* been proved for any $k > 1$. In particular, the result of Enomoto and Kaneko does not apply to these cases and their proof does not appear to have an easy adaption for these cases. The techniques used to prove the conjecture for $l = 2$ are novel. The main idea is to start with k internally disjoint paths

and then find the missing path by inductively moving through the graph and adjusting the k internally disjoint paths whenever necessary.

Our result is the first significant progress on the conjecture since 1994. It is possible, that the techniques used in the proof may be adjusted to play a role in proving the conjecture for further k and l, though they do not have an easy adaption to further cases. The result can also be exploited to prove the conjecture for certain graph classes. We illustrate this by showing that the conjecture holds for all k and all l if the underlying graph has treedwidth at most 3.

References

[1] Ron Aharoni and Eli Berger. Menger's Theorem for Infinite Graphs. *Inventiones mathematicae*, 176(1):1–62, 2008.

[2] Lowell W. Beineke and Frank Harary. The Connectivity Function of a Graph. *Mathematika*, 14(2):197–202, 1967.

[3] Èdouard Bonnet and Sergio Cabello. The complexity of mixed-connectivity. *Annals of Operations Research*, 307:25–35, 2021.

[4] Ralf Borndörfer and Marika Karbstein. A Note on Menger's Theorem for Hypergraphs. Technical Report 12-03, ZIB, Berlin, 2012.

[5] Yoshimi Egawa, Atsushi Kaneko, and Makoto Matsumoto. A Mixed Version of Menger's Theorem. *Combinatorica*, 11:71–74, 1991.

[6] Hikoe Enomoto and Atsushi Kaneko. The Condition of Beineke and Harary on Edge-disjoint Paths some of which are Openly Disjoint. *Tokyo Journal of Mathematics*, 17(2):355–357, 1994.

[7] Rija Erves and Janez Zerovnik. Mixed connectivity of cartesian graph products and bundles. *Ars Comb.*, 124:49–64, 2016.

[8] Wolfgang Mader. Connectivity and Edge-connectivity in Finite Graphs. In *Surveys in Combinatorics (Proceedings of the Seventh British Combinatorial Conference), London Mathematical Society Lecture Note Series*, volume 38, pages 66–95, 1979.

Session Session 1C: routing problems 1

Wednesday 8 June 2022, 10:45-12:25

Lecture Hall IV

Nash fairness solutions for balanced TSP

Minh Hieu Nguyen, Mourad Baiou, Viet Hung Nguyen*, Thi Quynh Trang Vo
INP Clermont Auvergne, Univ Clermont Auvergne, Mines Saint-Etienne, CNRS, UMR 6158 LIMOS,
1 Rue de la Chebarde, Aubiere Cedex, France
*Corresponding author : Viet Hung Nguyen
minh_hieu.nguyen@uca.fr,mourad.baiou@uca.fr,viet_hung.nguyen@uca.fr,thi_quynh_trang.vo@uca.fr

ABSTRACT

In this paper, we consider a variant of the Traveling Salesman Problem (TSP), called *Balanced Traveling Salesman Problem* (BTSP) [7]. The BTSP seeks to find a tour which has the smallest *max-min distance* : the difference between the maximum edge cost and the minimum one. We present a Mixed Integer Program (MIP) to find optimal solutions minimizing the max-min distance for BTSP. However, minimizing only the max-min distance may lead to a tour with an inefficient total cost in many situations. Hence, we propose a fair way based on Nash equilibrium [5], [11] to inject the total cost into the objective function of the BTSP. We consider a Nash equilibrium as it is defined in a context of fair competition based on proportional-fair scheduling. For BTSP, we are interested in solutions achieving a Nash equilibrium between two players: the first aims at minimizing the total cost and the second aims at minimizing the max-min distance. We call such solutions *Nash Fairness* (NF) solutions. We first show that NF solutions for BTSP exist and may be more than one. We show that NF solutions are Pareto-optimal [10] and can be found by optimizing a sequence of linear combinations of the two players objectives based on Weighted Sum Method [13]. We then focus on extreme NF solutions which are NF solutions having either the smallest value of total cost or the smallest max-min distance. Finally, we propose a Newton-based iterative algorithm which converges to extreme NF solutions in a polynomial number of iterations. Computational results on small-size instances from TSPLIB will be presented and commented.

1 INTRODUCTION

The Balanced Traveling Salesman Problem (BTSP) is a variation of the classical Traveling Salesman Problem (TSP) where instead of finding a Hamiltonian tour minimizing the total cost, we find a tour minimizing *the max-min distance*. The latter is the difference between the maximum edge cost and the minimum one in the tour. BTSP has been introduced by Larusic and Punnen (2011) [7] for finding Hamiltonian tours in several cases where the equitable distribution of edges are important, for example, the nozzle guide vane assembly problem [12] and the cyclic workforce scheduling problem [15].

BTSP can be formally defined as follows. Given an undirected graph $G = (V, E)$ where $V = [n] := \{1, \ldots, n\}$, $|E| = m$, $c_{ij} \in \mathbb{R}_+$ is a cost associated with every edge $ij \in E$ and let $\Pi(G)$ denote the set of all Hamiltonian cycles in G, BTSP can be defined as

$$\min_{H \in \Pi(G)} \left\{ \max_{ij \in H} c_{ij} - \min_{ij \in H} c_{ij} \right\}. \quad (1)$$

BTSP is NP-hard as the problem of finding a Hamiltonian cycle in G can easily be reduced to it. In [7], the authors proposed four thresholds heuristic algorithms to solve this problem. More precisely, distinct elements of c are firstly sorted in the ascending direction, i.e. $z_1 < \cdots < z_p$. The proposed algorithms find a pair (z_i, z_j) satisfying (i) a subgraph of G with the edge set $\{(i, j) \in E | z_i \leq c_{i,j} \leq z_j\}$ is Hamiltonian, and (ii) $z_i - z_j$ is as small as possible. The existence of Hamiltonian cycles in subgraphs is verified by necessary conditions for a tour and heuristic procedures instead of exact algorithms that are highly computationally expensive. As a consequence, the optimality of obtained solutions is not certified. Moreover, by minimizing only the max-min distance, the total cost of edges used in the tour is neglected and it may lead to very inefficient tours in many situations.

Another work relating to the fairness of a tour in TSP is the equitable TSP proposed by Kindable et al. [6]. In the equitable TSP, one tends to minimize the absolute difference between the total cost of two perfect matchings, which form a Hamiltonian cycle in a graph. The problem is presented with the example about the uniform distribution of distances to pedal for two people. Two integer programming formulations are proposed to solve the equitable TSP exactly. However, this problem also may not guarantee the efficiency of a tour in terms of the total cost.

In this paper, to overcome the possible inefficiency of the solutions for BTSP, we propose a fair way based on Nash equilibrium to inject the total cost into the objective function of BTSP. Nash equilibrium is the most common optimality notion for sharing resources among users [5],[11]. We consider a Nash equilibrium to be fair as it is defined in a context of fair competition based on proportional-fair scheduling that aims to provide a compromise between the utilitarian rule - which emphasizes overall system efficiency, and the egalitarian rule - which emphasizes individual fairness. For BTSP, we are interested in solutions achieving a Nash equilibrium between two players: the first aims at minimizing the total cost and the second aims at minimizing the max-min distance. We call such solutions *Nash Fairness* (NF) solutions. We first show that NF solutions for BTSP exist and may be more than one. We show that NF solutions are Pareto-optimal [10] and can be found by optimizing a sequence of linear combinations of the two players objectives based on Weighted Sum Method [13]. We then focus on extreme NF solutions which are NF solutions having either the smallest value of total cost or the smallest max-min distance. Finally, we propose a Newton-based iterative algorithm which converges to extreme NF solutions in a polynomial number of iterations. Computational

results on small size instances from TSPLIB will be presented and commented.

The paper is organized as follows. In Section 2, we present a mathematical formulation for BTSP. The notion of Nash fairness solution will be discussed in Section 3. In particular, we prove the existence of NF solutions for BTSP and show that they are optimal solutions of a weighted sum objective problem. In Section 4, Newton-based iterative algorithms for finding extreme NF solutions is given. Computational results on small size instances from TSPLIB will be presented and discussed in Section 5.

2 MIP FORMULATION FOR BTSP

Although several heuristic algorithms [7] have been developed for this problem, there is no exact formulation for the BTSP mentioned in the literature to the best of our knowledge. To formulate a MIP for BTSP, we first consider the directed version of G by replacing every edge ij by two arcs (i, j) and (j, i). The costs $c_{i,j}$ and $c_{j,i}$ associated respectively with (i, j) and (j, i) are both equal to c_{ij}. A Hamiltonian tour in the original undirected graph G correspond now to a directed tour in the directed version. We propose a MIP formulation (for complete graph G) for the BTSP as follows:

$$\min\ t \tag{2a}$$

$$\text{s.t.} \sum_{j\in[n]} x_{j,i} = 1 \qquad \forall i \in [n] \tag{2b}$$

$$\sum_{j\in[n]} x_{i,j} = 1 \qquad \forall i \in [n] \tag{2c}$$

$$\sum_{i\in Q} \sum_{j\neq i, j\in Q} x_{i,j} \leq |Q| - 1 \qquad \forall Q \subset V \tag{2d}$$

$$t \geq u - l \tag{2e}$$

$$u \geq c_{i,j} x_{i,j} \qquad \forall i, j \in [n] \tag{2f}$$

$$l \leq \sum_{j\in[n]} c_{i,j} x_{i,j} \qquad \forall i \in [n] \tag{2g}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i, j \in [n]. \tag{2h}$$

where $x_{i,j}$ is the binary variables representing the occurrence of arc (i, j) in the solution tour. The constraints (2b), (2c) are respectively the in-degree and out-degree constraints which assure that there is exactly one incoming arc and one outgoing arc incident to every vertex. The constraints (2d) are the subtour elimination constraints. These constraints represent the classical Held-Karp (linear programming) relaxation for the Asymmetric Traveling Salesman Problem. Together with the integral constraints (2h), they assure that the solution is a tour. In order to calculate the max-min distance t, we need to determine the largest and the smallest edge costs u and l in the solution tour. Constraints (2f) obviously allow to bound u from below by the largest weight arc in the solution tour. There will be exactly one non null term in the sum in the right hand side of constraints (2g) as there is exactly one arc leaving each vertex i. This non null term represents the weight of the arc leaving i in the solution tour. Hence, constraints (2g) allow to bound l from above by the smallest arc weight in the solution tour. As t is minimized, u and t will respectively take the values of the largest and the smallest edge costs.

Table 1: The BTSP results in TSPLIB instances

Instance	Heuristic algorithms [7]	Formulation (2)
att48	192	**190**
gr48	48	**46**
berlin52	151	**149**
brazil58	1125	**1097**

We have designed a special-purpose branch-and-cut algorithm based on Formulation (2). Despite of the simplicity of the latter, the algorithm is capable to find an optimal solution for instances of BTSP up to 80 vertices within 1 hour CPU time. Our experiments on instances from TSPLIB have been able to certificate the optimality of solutions found by heuristic algorithms in [7]. Furthermore, as shown in Table 1, our exact algorithm have disproved the optimality of the solution given in [7] for several instances.

However, the purpose of this paper is not to design exact solutions for BTSP and to compare with the results in [7]. Since optimal solutions for BTSP may not be very efficient in terms of the total cost, the purpose of the paper is to inject the latter into the objective function in some fair ways allowing a trade-off between the two objectives. The rest of the paper will be devoted to this question.

3 NASH FAIRNESS SOLUTIONS FOR BTSP

3.1 Characterization of NF solutions

Nash fairness (NF) solutions for maximizing the utilities of two-player problem [5] are defined by using the Nash standard of comparison. Under the latter, a transfer of utilities between the two players is considered to be fair if the percentage increase in the utility of one player is larger than the percentage decrease in utility of the other player [1].

Proportional fairness is a generalized NF solution for multiple players. In that setting, the fair allocation should be such that, if compared to any other feasible allocation of utilities, the aggregate proportional change is less than or equal to 0 [5], [1], [11].

Let U be a set of possible *states of the world* or *alternatives* and let I be a finite set, representing a collection of individuals. For each $i \in I$, $u_i : U \longrightarrow \mathbb{R}_+$ be a utility function, describing the amount of happiness an individual i derives from each possible state such that we prefer the alternative x to the alternative y if and only if $u_i(x) \geq u_i(y)$.

Definition 3.1. [1] $x^{NF} \in U$ be a NF solution in multiple players problem if and only if

$$\sum_{j=1}^{n} \frac{u_j(x) - u_j(x^{NF})}{u_j(x^{NF})} \leq 0,\ \forall x \in U, \tag{3}$$

where $u_j(x) > 0,\ \forall j \in I, \forall x \in U$.

Let P, Q represent the total cost and the max-min distance in a feasible solution tour for BTSP. We have then $P > Q \geq 0$. We first suppose that $Q > 0$. As P, Q now are two positive utility functions, we have a two-player problem. In the usual definition of NF solutions [5], [1], the alternative assigned a greater value is preferred. However, in BTSP, we prefer the alternative assigned a smaller value for two utility functions P and Q. Thus, we need to

modify the sign of each term representing the proportional change in Definition 3.1.

In the remainder of this paper, we consider (P, Q) as the solution for the total cost and the max-min distance corresponding to a feasible solution tour. Let (P^*, Q^*) be a NF solution for BTSP, condition (3) can be translated into the context of BTSP as follows

$$\frac{P^* - P}{P^*} + \frac{Q^* - Q}{Q^*} \leq 0, \ \forall (P, Q) \in S, \tag{4}$$

which is equivalent to

$$PQ^* + QP^* \geq 2P^*Q^*, \ \forall (P, Q) \in S, \tag{5}$$

where S is the set of solutions (P, Q) corresponding to all feasible solution tours for BTSP in G.

We note that in case $Q^* = 0$, the condition (5) is also satisfied. Hence, NF solution for BTSP can be generally stated as follows

Lemma 3.2. *$(P^*, Q^*) \in S$ be a NF solution for BTSP if and only if $PQ^* + QP^* \geq 2P^*Q^*, \ \forall (P, Q) \in S$.*

Remark 3.3. $(P, 0)$ is always a NF solution (i.e a solution tour with all equal edge costs).

3.2 Existence of NF solutions

In this section, we first show the existence of NF solutions for BTSP. Let us recall that in the classical multiple players problem mentioned in Section 3.1 where we prefer the alternative assigned a greater value, the NF solution can be obtained with the objective function

$$\max \sum_{j=1}^{n} \log \ u_j,$$

provided that U is convex. The necessary and sufficient first-order optimality condition for this problem is exactly the Nash standard of comparison principle for n players. Notice that the above NF solution is the one maximizing the product of the utilities over U.

On the contrary in BTSP, we prefer the alternative assigned a smaller value for two utility functions P, Q. Thus, there exists a NF solution which can be obtained by minimizing instead of maximizing the product of the utilities.

Theorem 3.4. *$(P^*, Q^*) = \text{argmin}_{(P,Q)\in S} PQ$ is a NF solution.*

Proof. Obviously, there always exists a solution $(P^*, Q^*) \in S$ such that

$$(P^*, Q^*) = \underset{(P,Q)\in S}{\text{argmin}} PQ.$$

Now $\forall (P', Q') \in S$ we have $P'Q' \geq P^*Q^*$. Then

$$P'Q^* + Q'P^* \geq 2\sqrt{P'Q'P^*Q^*} \geq 2P^*Q^*,$$

The first inequality holds by the Cauchy-Schwarz inequality. Hence, (P^*, Q^*) is a NF solution. □

Theorem 3.4 proves the existence of one NF solution for BTSP that minimizes PQ, or equivalently minimizes $(\log P + \log Q)$. However, finding such a solution may be difficult as it requires to minimize a concave function. In the following, we show that all NF solutions can be found by minimizing an appropriate linear combination of P and Q based on the Weighted Sum Method [8]. More precisely, all NF solutions can be obtained by solving the following optimization problem

$$\mathcal{P}(\alpha) = \min \ \alpha P + Q \text{ s.t } (P, Q) \in S,$$

where $\alpha \in [0, 1]$ is the coefficient to be determined. For solving $\mathcal{P}(\alpha)$, we can solve the MIP (2) in Section 2 with $\alpha P + Q$ as the objective function instead of Q.

Let $\alpha \in \mathbb{R}_+$ and (P_α, Q_α) be an optimal solution of $\mathcal{P}(\alpha)$. Denote $T_\alpha := \alpha P_\alpha - Q_\alpha$ and $C_0 := \{\alpha \in \mathbb{R}_+ | T_\alpha = 0\}$. Due to the definitions of P and $Q : P, Q$ respectively represent the total cost and the max-min distance in a solution tour, we always have $P > Q \geq 0$. Hence, if $\alpha \in C_0$ then $\alpha < 1$, if not $T_\alpha \geq P_\alpha - Q_\alpha > 0$.

Theorem 3.5. *$(P^*, Q^*) \in S$ is a NF solution if and only if there exists a coefficient $\alpha^* \in C_0$ such that (P^*, Q^*) is an optimal solution obtained by solving $\mathcal{P}(\alpha^*)$.*

Proof. Firstly, let (P^*, Q^*) be a NF solution and $\alpha^* = Q^*/P^*$. We will show that (P^*, Q^*) is an optimal solution of $\mathcal{P}(\alpha^*)$.

Since (P^*, Q^*) is a NF solution, we have

$$P'Q^* + Q'P^* \geq 2P^*Q^*, \ \forall (P', Q') \in S, \tag{6}$$

Since $\alpha^* = \frac{Q^*}{P^*}$, we have $\alpha^*P^* + Q^* = 2Q^*$.

Dividing two sides of (6) by $P^* > 0$ we obtain

$$2Q^* \leq \frac{Q^*}{P^*}P' + Q', \ \forall (P', Q') \in S, \tag{7}$$

So we deduce from (7)

$$\alpha^*P^* + Q^* \leq \alpha^*P' + Q', \ \forall (P', Q') \in S,$$

Hence, (P^*, Q^*) is an optimal solution of $\mathcal{P}(\alpha^*)$ and then $\alpha^* \in C_0$.

Now suppose $\alpha^* \in C_0$, we show that (P^*, Q^*) is a NF solution.

Since $T^* = \alpha^*P^* - Q^* = 0$, we have

$$\alpha^* = \frac{Q^*}{P^*}.$$

If (P^*, Q^*) is not a NF solution, there exists a solution $(P', Q') \in S$ such that

$$P'Q^* + Q'P^* < 2P^*Q^*,$$

We have then

$$\alpha P' + Q' = \frac{P'Q^* + Q'P^*}{P^*} < \frac{2P^*Q^*}{P^*} = \alpha^*P^* + Q^*,$$

which contradicts the optimality of (P^*, Q^*). □

Corollary 3.5.1. *NF solutions are Pareto-optimal solutions over S.*

Proof. Base on Theorem (3.5), all NF solutions can be obtained by solving $\mathcal{P}(\alpha)$. Now let (P^*, Q^*) be the corresponding NF solution of $\mathcal{P}(\alpha^*)$, we will show that (P^*, Q^*) is Pareto-optimal solution over S by contradiction.

Let us assume that there exists another solution $(P', Q') \in S$ such that $P' < P$ and $Q' < Q$. We have then

$$\alpha^*P' + Q' < \alpha^*P^* + Q^*,$$

which contradicts the optimality of (P^*, Q^*).

Hence, NF solutions are Pareto-optimal solutions over S. □

The following remark asserts that there may be more than one NF solution for BTSP.

Remark 3.6. Let (P, Q) and (P', Q') be two different feasible solutions in BTSP. The two inequalities

$$P'Q + Q'P \geq 2PQ \text{ and } P'Q + Q'P \geq 2P'Q'.$$

may be satisfied simultaneously.

The main question now is how to determine a coefficient α^* allowing to find a NF solution according to Theorem 3.5. In the next section, we present an iterative algorithm converging to α^* in a polynomial number of iterations. The value of α^* found by this algorithm corresponds to the NF solutions with the smallest total cost or the smallest max-min distance.

4 ALGORITHMS FOR FINDING EXTREME NASH FAIRNESS SOLUTIONS

As shown by Remark 3.6, there may be many NF solutions for an instance of BTSP. Among these solutions, two solutions may naturally be preferred to the others: the one with the smallest P and the one with the smallest Q. Let us call the first *Efficient Nash Fairness (ENF)* solution and the second *Balanced Nash Fairness (BNF)* solution. We call both ENF solution and BNF solution *extreme Nash Fairness* solution. In the following, we will focus first on ENF solution. As we will argue at the end of the section, all the subsequent results applied to ENF solution can be also applied to BNF solution with slight changes.

Theorem 4.1. *The ENF solution is unique.*

Proof. Suppose (P, Q) and (P', Q') are two ENF solutions. By the definition of ENF solution, we have $P \leq P'$ and $P' \leq P$ that imply $P = P'$.

Furthermore, (P, Q) and (P', Q') also are NF solutions. Hence

$$P'Q + Q'P \geq 2PQ \text{ and } P'Q + Q'P \geq 2P'Q'.$$

Since $P = P' > 0$, we have

$$Q + Q' \geq 2Q \text{ and } Q + Q' \geq 2Q'.$$

These equations lead to $Q = Q'$. □

We propose now an algorithm to find the coefficient α^* such that the optimal solution (P^*, Q^*) obtained by solving $\mathcal{P}(\alpha^*)$ is the ENF solution. This algorithm is inspired from the application of Newton method (or the Newton-Raphson method) to linear fractional programs that was first discussed by Isbell and Marlow [4] and then generalized to nonlinear fractional programs by Dinkelbach [3]. It is often called the Dinkelbach method. The algorithm can be stated as follows.

where X_i represents the solution tour correspond to (P_i, Q_i).

Denote $\{\alpha_i\}$ as the sequence constructed by Algorithm 1. We will prove that Algorithm 1 terminates in a polynomial number of iterations and the obtained solution (P_i, Q_i) is the ENF solution. Our proof will use the following lemmas.

Lemma 4.2. *Let $\alpha, \alpha' \in \mathbb{R}_+$ and (P_α, Q_α), $(P_{\alpha'}, Q_{\alpha'})$ be the optimal solutions of $\mathcal{P}(\alpha)$ and $\mathcal{P}(\alpha')$ respectively, if $\alpha \leq \alpha'$ then $P_\alpha \geq P_{\alpha'}$ and $Q_\alpha \leq Q_{\alpha'}$.*

Algorithm 1

Input: An undirected graph G with n vertices, m edges and a positive cost vector $c \in \mathbb{R}^m_+$.

Output: A Hamiltonian tour corresponding to the ENF solution.

1: $\alpha_0 \leftarrow 1$, $i \leftarrow 0$
2: **repeat**
3: solve $\mathcal{P}(\alpha_i)$ to obtain (P_i, Q_i) and X_i
4: $T_i \leftarrow \alpha_i P_i - Q_i$
5: $\alpha_{i+1} \leftarrow Q_i / P_i$
6: $i \leftarrow i + 1$
7: **until** $T_i = 0$
8: return (P_i, Q_i, X_i).

Proof. The optimality of (P_α, Q_α) and $(P_{\alpha'}, Q_{\alpha'})$ gives

$$\alpha P_\alpha + Q_\alpha \leq \alpha P_{\alpha'} + Q_{\alpha'}, \text{ and} \tag{8a}$$

$$\alpha' P_{\alpha'} + Q_{\alpha'} \leq \alpha' P_\alpha + Q_\alpha \tag{8b}$$

By adding both sides of (8a) and (8b), we obtain $(\alpha - \alpha')(P_\alpha - P_{\alpha'}) \leq 0$. Since $\alpha \leq \alpha'$, it follows that $P_\alpha \geq P_{\alpha'}$.

On the other hand, inequality (8a) implies $Q_{\alpha'} - Q_\alpha \geq \alpha(P_\alpha - P_{\alpha'}) \geq 0$ that leads to $Q_\alpha \leq Q_{\alpha'}$. □

Lemma 4.3. *During the execution of Algorithm 1, the sequence $\{\alpha_i\}$ is always non-negative and non-increasing. Moreover, $T_i \geq 0$, $\forall i \geq 0$.*

Proof. We have $\alpha_0 = 1 \geq 0$. Since $P > Q \geq 0$ $\forall (P, Q) \in S$, it follows that $\alpha_{i+1} = Q_i / P_i \geq 0$, $\forall i \geq 0$.

Our proof is given by induction on i. If $i = 0$, then $T_0 = \alpha_0 P_0 - Q_0 = P_0 - Q_0 \geq 0$ and $\alpha_0 = 1 \geq Q_0 / P_0 = \alpha_1$. Suppose that our hypothesis is true until $i = k \geq 0$, we will prove that it is also true with $i = k + 1$.

Indeed, we have

$$T_{k+1} = \alpha_{k+1} P_{k+1} - Q_{k+1} = \frac{Q_k P_{k+1} - P_k Q_{k+1}}{P_k}. \tag{9}$$

The inductive hypothesis gives $\alpha_k \geq \alpha_{k+1}$ that implies $P_{k+1} \geq P_k > 0$ and $Q_k \geq Q_{k+1} \geq 0$ according to Lemma 4.2. It leads to $Q_k P_{k+1} - P_k Q_{k+1} \geq 0$ and then $T_{k+1} \geq 0$.

On the other hand, from the definition of α_{k+2}, we get

$$\alpha_{k+1} - \alpha_{k+2} = \frac{\alpha_{k+1} P_{k+1} - Q_{k+1}}{P_{k+1}} = \frac{T_{k+1}}{P_{k+1}} \geq 0. \tag{10}$$

That concludes the proof. □

Lemma 4.4. *Algorithm 1 terminates in a polynomial number of iterations.*

Proof. By contradiction, we first show that if $T_{i+1} > 0$ then $P_i < P_{i+1}$ and $Q_i > Q_{i+1}$.

Let assume that $P_i \geq P_{i+1}$. According to Lemma 4.3, $\alpha_i \geq \alpha_{i+1}$ that implies $P_i \leq P_{i+1}$ as the result of Lemma 4.2. Thus, $P_i = P_{i+1}$. From (9), if $T_{i+1} > 0$ then $Q_i P_{i+1} > Q_{i+1} P_i$. Since $P_i = P_{i+1} > 0$, we get $Q_i > Q_{i+1}$.

On the other hand, as (P_i, Q_i) is the optimal solution $\mathcal{P}(\alpha_i)$, it shows that $\alpha_i P_i + Q_i \leq \alpha_i P_{i+1} + Q_{i+1}$. Using $P_i = P_{i+1}$, we obtain $Q_i \leq Q_{i+1}$ which leads to a contradiction.

By repeating the same argument for $Q_i \leq Q_{i+1}$, we also have a contradiction.

Consequently, while $T_k > 0$, each i^{th} $(i \leq k)$ iteration of Algorithm 1 explores a Pareto-optimal solution (P_i, Q_i) with the distinct value of Q_i.

Now let c_i^{max} and c_i^{min} be the maximum edge cost and the minimum one in the solution (P_i, Q_i). We have $Q_i = c_i^{max} - c_i^{min}$ and due to the distinctness of Q_i, we obtain $c_i^{max} - c_i^{min} \neq c_j^{max} - c_j^{min}, \ \forall i \neq j$. We have then

$$(c_i^{max}, c_i^{min}) \not\equiv (c_j^{max}, c_j^{min}), \ \forall i \neq j,$$

Thus, each Pareto-optimal solution obtained by an iteration of Algorithm 1 has distinct pair of edges corresponding to the maximum edge cost and the minimum one. For graph G with n vertices, we have at most $O(n^2)$ edges and then the maximum number of distinct pairs of edges is $O(n^4)$.

Hence, the number of iterations in worst case is also $O(n^4)$. Consequently, Algorithm 1 terminates in a polynomial number of iterations. □

Theorem 4.5. *We obtain the ENF solution by Algorithm 1.*

Proof. Let α_n be the solution obtained by Algorithm 1 and (P_n, Q_n) be the optimal solution of $\mathcal{P}(\alpha_n)$. By the stopping criteria, $T_n = 0$ and $\alpha_n \in C_0$. Hence, $T_i > 0 \ \forall i < n$ and then $\alpha_i < \alpha_{i-1} \ \forall i \leq n$. According to Theorem 3.5, (P_n, Q_n) is a NF solution. We will prove that (P_n, Q_n) is a NF solution with the smallest total cost.

Assume that (P, Q) is another NF solution such that $P < P_n$. By Theorem 3.5, there exists $\alpha \in C_0$ such that (P, Q) is the optimal solution of $\mathcal{P}(\alpha)$. Furthermore, $\alpha \in (\alpha_n, \alpha_0]$ (if not, $P_\alpha \geq P_n$). Then there exists $0 < i \leq n$ such that $\alpha \in (\alpha_i, \alpha_{i-1}]$. Since $\alpha \leq \alpha_{i-1}$, $P \geq P_{i-1}$ and $Q \leq Q_{i-1}$ due to Lemma 4.2. Thus, we get

$$\frac{Q}{P} \leq \frac{Q_{i-1}}{P_{i-1}} \tag{11}$$

By the definitions of α and α_i, inequality (11) is equivalent to $\alpha \leq \alpha_i$ which leads to a contradiction. Hence, (P_n, Q_n) is the ENF solution. □

For finding the BNF solution, we use a similar algorithm starting from $\alpha_0 = 0$ instead of 1. In this case, the sequence $\{\alpha_i\}$ is always non-negative, non-decreasing and $T_i \leq 0 \ \forall i \geq 0$. After a polynomial number of iterations, we also obtain a coefficient $\alpha_n \in C_0$ and then we can prove that the optimal solution (P_n, Q_n) obtained by solving $\mathcal{P}(\alpha_n)$ is exactly the BNF solution. More precisely, we state this algorithm as follows.

Algorithm 2

Input: An undirected graph G with n vertices, m edges and a positive cost vector $c \in \mathbb{R}_+^m$.

Output: A Hamiltonian tour corresponding to the BNF solution.

1: $\alpha_0 \leftarrow 0, \ i \leftarrow 0$
2: **repeat**
3: solve $\mathcal{P}(\alpha_i)$ to obtain (P_i, Q_i) and X_i
4: $T_i \leftarrow \alpha_i P_i - Q_i$
5: $\alpha_{i+1} \leftarrow Q_i / P_i$
6: $i \leftarrow i + 1$
7: **until** $T_i = 0$
8: return (P_i, Q_i, X_i).

Then, we also state some lemmas and theorems to prove that we obtain BNF solution by using Algorithm 2.

Lemma 4.6. *During the execution of Algorithm 2, the sequence $\{\alpha_i\}$ is always non-negative and non-decreasing. Moreover, $T_i \leq 0, \ \forall i \geq 0$.*

Lemma 4.7. *Algorithm 2 terminates in a polynomial number of iterations.*

Theorem 4.8. *We obtain the BNF solution by Algorithm 2.*

Remark 4.9. In case $Q = 0$ (i.e a solution tour with all equal edge costs), the optimal solution of BTSP is also the BNF solution.

5 NUMERICAL RESULTS

Let us denote NFBTSP for Nash Fairness Balanced TSP, the problem of finding NF extreme solutions for BTSP. In this section, we conduct several experiments aiming at solving NFBTSP with Algorithms 1 and 2 on rather small size instances from TSPLIB [14]. We also solve the classical TSP and the BTSP [7] on the same instances. The obtained solutions for three problems will be then compared and commented.

For solving the three problems TSP, BTSP and NFBTSP, we design a simple branch-and-cut algorithm devoted to minimize a linear objective function over the MIP program in Section 2 (of course, for TSP the constraints (2f) and (2g) are exluded). We use CPLEX 12.10 to implement our branch-and-cut algorithm. The constraints (2d) are set as lazy cuts which are generated only when being violated by some integer solution. For BTSP and NFBTSP, we also have some specific branching rules for variable l inspired from the threshold algorithm [9], [7]. For NFBTSP, this branch-and-cut algorithm is used in each iteration of Algorithms 1 and 2 to solve the subproblem $P(\alpha)$. All the experiments are conducted on a PC Intel Core i5-9500 3.00GHz with 6 cores and 6 threads.

Table 2 presents the results of three problems TSP, BTSP and NFBTSP in several instances of TSPLIB with a range of nodes from 14 to 29. For NFBTSP, we also provide the number of iterations for finding respectively the ENF solution and the BNF solution (column "Iters") and its corresponding final value of α. We can see by the values of P and Q in this table that the ENF and BNF solutions for NFBTSP strike a better trade-off between two objectives: the total cost and the max-min distance comparing with those for the classical TSP and the BTSP. In particular, when the solutions for classical TSP and for BTSP are too different: small P and big Q in the

Table 2: TSP, BTSP and NFBTSP results on TSPLIB problems

Instance	TSP			BTSP			ENFTSP					BNFTSP				
	P	Q	Time	P	Q	Time	P	Q	Time	Iters	alpha	P	Q	Time	Iters	alpha
burma14	3323	472	0.10	4986	134	0.15	4986	134	1.40	4	0.027	4986	134	0.70	2	0.027
ulysses16	6859	1452	0.51	14032	868	0.70	7047	1399	4.42	3	0.199	13670	868	17.56	3	0.063
gr17	2085	311	0.21	4138	119	0.35	2227	234	5.53	3	0.105	3346	139	8.63	4	0.042
gr21	2707	328	0.01	8630	115	0.68	2989	278	1.24	3	0.093	5945	120	18.60	3	0.020
ulysses22	7013	1490	141.46	19168	868	1.68	7070	1471	356.97	3	0.208	7070	1471	651.38	4	0.208
gr24	1272	83	0.07	3886	33	1.85	1282	81	1.17	3	0.063	3847	33	49.17	3	0.009
fri26	937	118	0.13	2458	21	2.72	980	82	24.39	3	0.084	2447	21	55.06	3	0.009
bays29	2020	140	0.55	6757	38	7.95	3449	59	2,033.04	5	0.017	4558	44	2471.15	3	0.010
bayg29	1610	86	0.31	4252	29	4.12	1817	63	44.92	3	0.035	3246	35	2567.73	4	0.011

solution for TSP and big P and small Q in the one of BTSP, the two extreme NF solutions for NFTSP give two interesting alternatives. More precisely, the ENF solution (respectively BNF solution) offers a better alternative than the solution of TSP (respectively BTSP) with a significant drop on the value of Q (respectively P) and a slight growth on the value of P (respectively Q). Table 2 also indicates that Algorithms 1 and 2 seem to converge very quickly after only maximum 5 iterations. One important issue is the CPU time for solving NFBTSP is quite huge comparing with the CPU time spent for solving TSP and BTSP. A deeper analysis on the iterations of Algorithms 1 and 2 tells us that the smaller is the value of α, the more difficult is $P(\alpha)$. Especially, the CPU time spent for solving $P(\alpha)$ in the last iteration occupies a very big part of the overall CPU time. Hence, a special-purpose algorithm for solving $P(\alpha)$ may be more interesting than simply optimizing a linear function over the MIP given in Section 2.

6 CONCLUSION

In this paper, we have made use of Nash fairness equilibrium to achieve a trade-off between the efficiency estimated by the total cost and the balancedness estimated by the max-min distance in solutions for balanced TSP (BTSP). We have proven first the existence of Nash Fairness (NF) equilibrium solutions for BTSP. Second, we have designed Newton-based iterative algorithms to find the two extreme NF solutions: the one with minimum total cost and the one with minimum max-min distance. Numerical results conducted on small size instances from TSPLIB have shown that comparing with the optimal solutions of the original BTSP, the NF solutions found by our algorithm have much smaller total cost with a reasonable augmentation of the max-min distance. An important notice is that the results in this paper can be also applied to various balanced combinatorial optimization [9] problems such as balanced assignment problem , balanced spanning tree problem [2],... Another future development of our work is the improvement of the CPU time for solving the problem $P(\alpha)$ especially when α is very small. One of the possible direction is to study the possibility of stopping the solving of $P(\alpha)$ once a improved solution of (P, Q) is obtained instead of stopping at optimality.

REFERENCES

[1] Dimitris Bertsimas, Vivek F.Farias, and Nikolaos Trichakis. January-February 2011. The Price of Fairness. *Operations Research* 59-1 (January-February 2011), 17–31.

[2] Paolo M. Camerini, Francesco Maffioli, Silvano Martello, and Paolo Toth. 1986. Most and least uniform spanning trees. *Discrete Applied Mathematics* 15, 2 (1986), 181–197. https://doi.org/10.1016/0166-218X(86)90041-7

[3] W. Dinkelbach. 1967. On nonlinear fractional programming. *Management Sci* 13 (1967), 492–498.

[4] J.R. Isbell and W.H. Marlow. 1956. Attrition games. *Naval Res. Logist. Quart 3* (1956), 71–94.

[5] FP Kelly, AK Maullo, and DKH Tan. November 1997. Rate control for communication networks : shadow prices, proportional fairness and stability. (November 1997).

[6] Joris Kinable, Bart Smeulders, Eline Delcour, and Frits CR Spieksma. 2017. Exact algorithms for the equitable traveling salesman problem. *European Journal of Operational Research* 261, 2 (2017), 475–485.

[7] John Larusic and Abraham P Punnen. 2011. The balanced traveling salesman-problem. *Computers & Operations Research* 38, 5 (2011), 868–875.

[8] R. Timothy Marler and Jasbir Singh Arora. 2010. The weighted sum method for multi-objective optimization: New insights. *Structural and Multidisciplinary Optimization 41(6):853-862* (2010).

[9] S Martello, W.R Pulleyblank, P Toth, and D de Werra. 1984. Balanced optimization problems. *Operations Research Letters* 3, 5 (1984), 275–278. https://doi.org/10.1016/0167-6377(84)90061-0

[10] Fiorenzo Mornati. 2013. Pareto Optimality in the work of Pareto. *European Journal of Social Sciences* 51-2 (2013), 65–82.

[11] W. Ogryczak, Hanan Luss, Michal Pioro, Dritan Nace, and Artur Tomaszewski. 2014. Fair Optimization and Networks: A survey. *Journal of Applied Mathematics* (2014), 1–26.

[12] Robert D Plante. 1988. The nozzle guide vane problem. *Operations research* 36, 1 (1988), 18–33.

[13] Jasbir Singh Arora R. Timothy Marler. June 2010. The weighted sum method for multi-objective optimization: New insights. 41(6) (June 2010), 853–862.

[14] Gerhard Reinelt. 1991. TSPLIB A traveling salesman problem library. *ORSA journal on computing* 3, 4 (1991), 376–384.

[15] Zeev Zeitlin. 1981. Minimization of maximum absolute deviation in integers. *Discrete Applied Mathematics* 3, 3 (1981), 203–220.

Drones for Relief Logistics under Demand Uncertainty

Okan Dukkanci[1], Achim Koberstein[2], and Bahar Y. Kara[3]

[1]Business Informatics & Operations Research, European University Viadrina, Frankfurt (Oder), Germany, ✉ dukkanci@europa-uni.de
[2]Business Informatics & Operations Research, European University Viadrina, Frankfurt (Oder), Germany, ✉ koberstein@europa-uni.de
[3]Department of Industrial Engineering, Bilkent University, Ankara, Turkey, ✉ bkara@bilkent.edu.tr

This study presents a post-disaster humanitarian delivery problem where critical relief items are distributed by drones to the affected people. After a disaster, in particular, an earthquake, the road connection between supply points and gathering points where the affected people gather might be disrupted due to possible debris on the road. To reduce the impact of having an incomplete road network, drones can be utilized as the main delivery option to distribute relief items.

In this study, we present a two-echelon distribution network where trucks and two different types of drones (small and big) are used to complete deliveries. At the first echelon of the network, trucks are used to carry small drones and relief items from depot(s) to areas called launch points, from which small drones are launched to make final deliveries to gathering points at the second echelon of the network. Additionally, big drones with longer range and larger capacity can be used to distribute relief items to gathering points directly from depot(s). Figure 1 represents an example of the network described above with one depot, four launch points (indicated by 'P') and several gathering points. While solid and dashed arrows represent movements of trucks and small drones, respectively, dotted arrows indicate deliveries made by big drones.

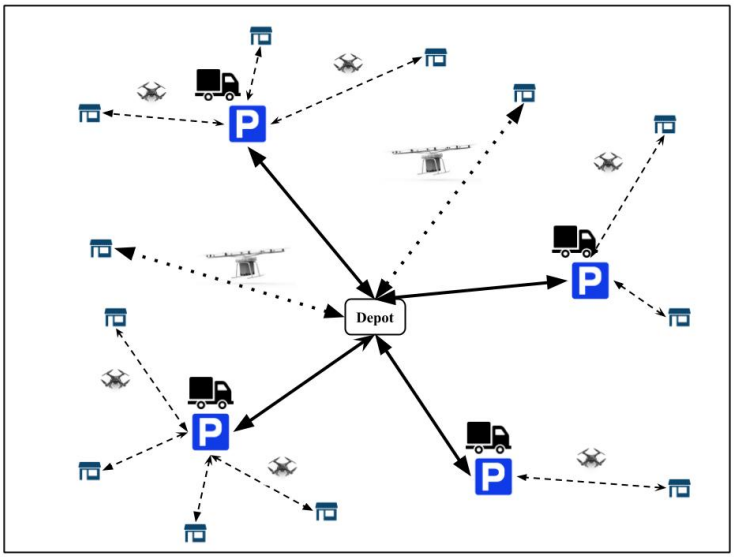


Figure 1: An illustration of the proposed drone delivery system

As time, place and magnitude of earthquakes cannot be predicted in advance, it is not easy to know how much relief items will be needed by the affected people before the earthquake occurs. For that reason, we consider uncertainty of demand. Additionally, to make sure that relief items are delivered to as many people as possible, the objective function of the problem is to minimize the expected total unsatisfied demand over all gathering points. The problem defined on the two-echelon network consists of selecting depot(s) and launch points to use from a candidate set of sites, allocating launch points to depots, allocating gathering points to launch points to be served by small drones and/or to depot(s) to be

served by big drones subject to capacity restrictions of trucks and (small and big) drones, a time bound to complete all deliveries and range of small drones.

To the best of our knowledge, this is the first study that considers a relief distribution problem under demand uncertainty on a two-echelon network where range-limited drones (with the help of trucks) are used to deliver critical items within a predetermined time bound.

As a solution approach, we, first, present a two-stage stochastic programming formulation and its deterministic equivalent problem formulation. As solving the deterministic equivalent problem formulation within reasonable times does not seem possible, we also implement an exact solution approach based on the scenario decomposition algorithm proposed by [1].

To apply this study to a real-life application, a case study is conducted based on western (European) side of Istanbul, Turkey. One of the main reasons is to choose Istanbul is that geological studies and surveys indicate in a near future, a major earthquake is expected to happen in Istanbul.

To estimate demand, different earthquake scenarios in terms of their locations and magnitudes are considered. While the magnitude of earthquakes are randomly generated, location information are based on 156 earthquakes that have happened in the last ten years near Istanbul (Figure 2). By using location and magnitude information, number of affected people are calculated by damage intensity level formulas from geology literature ([2]) and damage percentages from past earthquakes.

Figure 2: Epicenter location of earthquakes happened in Marmara Sea (last ten years) (Google Maps, 2021)

We carry out computational experiments on the performance of the scenario decomposition algorithm, value of stochasticity and expected value of perfect information under different parametric settings and also sensitivity analyses are conducted by varying key parameters of the problem such as time bound and capacity of drones.

References

[1] Shabbir Ahmed. A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters*, 41(6):565–569, 2013.

[2] Erdem Bayrak, Murat Nas, and Yusuf Bayrak. New macroseismic intensity predictive models for turkey. *Acta Geophysica*, 67(6):1483–1513, 2019.

The Traveling Salesman Problem with positional consistency constraints in healthcare scheduling

Mafalda Ponte[1], Luís Gouveia[2], and Ana Paias[3]

[1]CMAFCIO, Faculdade de Ciências - Universidade de Lisboa, Lisbon, Portugal, ✉ mafaldaponte@gmail.com
[2]CMAFCIO, Faculdade de Ciências - Universidade de Lisboa, Lisbon, Portugal, ✉ legouveia@fc.ul.pt
[3]CMAFCIO, Faculdade de Ciências - Universidade de Lisboa, Lisbon, Portugal, ✉ ampaias@fc.ul.pt

Abstract

In the Traveling Salesman Problem with positional consistency constraints we seek to generate a minimum cost set of routes, where the nodes are visited by all the routes they require service from, and consistency is verified. In this case, consistency constraints are written in terms of the relative position a node occupies in the routes it appears in, and not in terms of service time or arrival time ([4] and [5]). Information known *a priori* includes the set of nodes that must be visited by each route and the set of nodes that require consistency.
This problem is adequate to represent an application in healthcare scheduling, where each route represents a healthcare professional (doctor or nurse), the tasks (patients or groups and patients) are already assigned to one or several professionals, and each work day is divided in time slots, all in the same duration. Tasks that are performed by several healthcare professionals must be performed by all of them at the same time. Therefore, each schedule can be represented by a route, with the order of the nodes in the route representing the order of the tasks in schedule and, particularly, the relative position that the node occupies represents the allocation of the corresponding task to a time slot. Waiting time is not allowed between tasks, however, schedules with less tasks are allowed to start later and/or end earlier than larger schedules.
We compare several time-dependent formulations that were adapted from the literature ([1], [2] and [3]), since such formulations rely on decision variables indexed by time, which allows us to derive straightforward consistency constraints. In addition, a new aggregated formulation, which is not indexed by route was proposed. This new formulation can be proved to be valid for the problem.
Finally, these models were assessed and compared using a set of test instances with characteristics derived from the application.

References

[1] Kenneth R Fox, Bezalel Gavish, and Stephen C Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28(4):1018–1021, 1980.

[2] Bezalel Gavish and Stephen C Graves. The travelling salesman problem and related problems. 1978.

[3] Jean-Claude Picard and Maurice Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978.

[4] Anirudh Subramanyam and Chrysanthos E Gounaris. A branch-and-cut framework for the consistent traveling salesman problem. *European Journal of Operational Research*, 248(2):384–395, 2016.

[5] Anirudh Subramanyam and Chrysanthos E Gounaris. A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. *Transportation Science*, 52(2):386–401, 2017.

A tabu search with geometry-based sparsification methods for the AngleTSP and AngleDistanceTSP

Rossana Cavagnini[1], Alina Theiß[1], and Michael Schneider[1]

[1]Deutsche Post Chair – Optimization of distribution networks, RWTH Aachen University, Aachen, Germany, ✉ cavagnini@dpo.rwth-aachen.de

1 Introduction

The angular-metric traveling salesman problem (AngleTSP) aims to find a tour visiting a given set of vertices exactly once while minimizing the cost given by the sum of all turning angles. If the cost is obtained by combining the sum of the turning angles and traveled distance, the problem is called angular-distance-metric TSP (AngleDistanceTSP). We define both problems on a complete directed graph $G = (V, A)$ with vertex set V and arc set A.

The described problems have applications in robotics and transportation. Keeping the movements of a robot or a vehicle as straight as possible avoids energy-consuming and hard-to-control changes in driving direction.

Because both problems are NP-hard, Staněk et al. [1] proposed multiple heuristics. The best-performing methods are based either on simple construction heuristics or on matheuristics. The first ones are fast but provide an unsatisfying solution quality. The second ones lead to a superior solution quality but are expensive in terms of runtime.

We propose a granular tabu search (GTS) that considers the geometric features of the two problems in the design of the starting solutions and the sparsification methods. Compared with the best-performing heuristics developed by Staněk et al. [1], GTS improves either the solution quality or the runtime, sometimes both, and finds a large number of new best-known solutions.

Section 2 describes GTS and the proposed sparsification methods, and Section 3 presents the results and gives conclusions.

2 GTS for the AngleTSP and the AngleDistanceTSP

To generate an initial solution, we rely on two different algorithms depending on the problem under consideration. For the AngleTSP, the literature has shown that optimal tours have a snail shell shape. Similarly to Staněk et al. [1], we apply a construction heuristic based on convex hulls. For the AngleDistanceTSP, optimal tours are often simple polygons. To obtain a closed path with no intersections, we rely on a construction heuristic that scans the set V in counterclockwise direction.

Next, GTS is applied. GTS uses a composite neighborhood generated by three operators: relocate, exchange, and 2-opt. Granular neighborhoods are used to speed up the search (see below). To escape local optima, the tabu criterion forbids to reinsert an arc that has been removed by GTS for a tenure of τ iterations. In each iteration, we carry out the best non-tabu move, which can be deteriorating with respect to the best solution found so far. As aspiration criterion, we always permit a move that is tabu but improves on the best solution found. Moves deteriorating the cost are evaluated according to an extended objective function that considers the number of times the inserted arcs were accepted in previous moves. GTS terminates after η iterations without improvement.

Sparsification strategies To reduce the neighborhood size, the number of arcs used to generate the moves, i.e., the generator arc set A_g, is restricted. A_g is obtained by adding the arcs included in A_a and A_s. The list A_a contains the arcs inserted by the accepted moves in recent iterations. When the length limit $\lceil \kappa_a \mid A \mid \rceil$ of A_a is reached, the arcs added by the oldest move are removed and those added by the last move are inserted.

The set A_s contains the arcs—if not included in A_a yet—selected by one of the two alternative sparsification strategies: the random-based lens procedure (r-LENS) and the cost-based lens procedure (c-LENS). Both approaches rely on a modification of the general lens procedure (LENS) introduced by Staněk et al. [1]. In LENS, a lens of thickness γ is positioned between two vertices v_i and v_{i+1} in a tour such that the lens curvature intersects at these vertices. Only the vertices in this lens are considered for inclusion in the tour between v_i and v_{i+1}. Differently from Staněk et al. [1], who apply LENS to every arc of a tour and include all the resulting arcs in A_s, we further intensify the sparsification by limiting the cardinality of A_s to κ_s percent of $\mid A \mid$. This implies that a criterion to select the arcs on which the lens should be positioned has to be defined to guarantee that the most important arcs are included in A_s.

Starting always positioning the lens on the arc originating from vertex v_i may result in little variety of A_s due to its restricted length. To prevent this, r-LENS positions the lens on the arc of a random vertex. This ensures that different parts of the tour are targeted across iterations.

In c-LENS, the turning angles of the tour are first sorted in non-increasing order. Then, the lens is applied from the pair of arcs defining the biggest angle to the smallest one, until the cardinality limit of A_s is reached. This sparsification procedure aims to improve the most expensive parts of the tour.

For more diversification, both sparsification strategies are dynamic. In contrast to common practice in the literature, in which the dynamics is based on variations of κ_s, in r-LENS and c-LENS, the sparsification intensity is kept equal to κ_s. Instead, the lens thickness γ varies in the interval $[\gamma_{min}, 2\gamma_{min}]$. Enlarging the lens thickness when improvements are hard to be found allows to insert more diverse arcs in A_s.

3 Computational results and conclusion

GTS was implemented in C++ and compiled using clang v.12. The experiments were performed on an Intel(R) Xeon(R) computer with an CPU E5-2430 v2 processor at 2.50GHz with 64 GB RAM under CentOS GNU/Linux 7. We tested GTS on the benchmark sets proposed in [1]. Tables 1 and 2 summarize the comparison of GTS with r-LENS and c-LENS for the AngleTSP and AngleDistanceTSP, respectively, to the non-dominated algorithms of [1]. Because GTS contains randomized elements, we performed ten runs for each instance and report the best and average gap to the best-known solutions and the converted runtimes for a fair comparison. All results are reported as average over all instances. GTS with r-LENS is superior to GTS with c-LENS for both sets. For the AngleTSP instances, GTS with r-LENS shows a good quality-runtime tradeoff, and its performance lies on the Pareto frontier. For the AngleDistanceTSP instances, both GTS variants do even better: they dominate two state-of-the-art algorithms and achieve the best solution quality in competitive runtimes. Moreover, GTS with r-LENS finds new best-known solutions for 68.75% and 72.50% of the AngleTSP and the AngleDistanceTSP instances, respectively.

	$LPP^R + M^{15}$	$LPP^R + M^{20}$	$LPC_1^R + M^{15}$	$LPC_1^R + M^{20}$	GTS r-LENS	GTS c-LENS
$\overline{\Delta}_{bks}^{best}(\%)$	1.82%	1.13%	1.51%	0.91%	-0.43%	-0.20%
$\overline{\Delta}_{bks}^{avg}(\%)$	1.82%	1.13%	1.51%	0.91%	1.21%	1.22%
$\overline{t}(s)$	245.95	290.26	256.32	299.53	275.77	333.12

Table 1: AngleTSP results: comparison of the average results of the non-dominated algorithms of [1] with GTS with r-LENS and c-LENS, respectively.

	$CIF + M^{15}$	$CIF + M^{20}$	$NN_S^2 + M^{15}$	$NN_S^2 + M^{20}$	$LPC_1^R + M^{20}$	$LPC_2^R + M^{20}$	GTS r-LENS	GTS c-LENS
$\overline{\Delta}_{bks}^{best}(\%)$	1.71%	1.2%	0.63%	0.48%	0.41%	0.39%	-0.22%	-0.13%
$\overline{\Delta}_{bks}^{avg}(\%)$	1.71%	1.2%	0.63%	0.48%	0.41%	0.39%	0.11%	0.17%
$\overline{t}(s)$	25.13	42.00	90.97	106.27	180.77	190.72	188.18	165.33

Table 2: AngleDistanceTSP results: comparison of the average results of the non-dominated algorithms of [1] with GTS with r-LENS and c-LENS, respectively.

References

[1] Rostislav Staněk, Peter Greistorfer, Klaus Ladner, and Ulrich Pferschy. Geometric and LP-based heuristics for angular travelling salesman problems in the plane. *Computers & Operations Research*, 108:97–111, 2019.

Session Session 2A: network design

Wednesday 8 June 2022, 13:45-15:00

Lecture Hall I

A robust variant of the Ring Star Problem

Julien Khamphousone
julien.khamphousone@dauphine.psl.eu
Université Paris Dauphine, PSL
Paris, 75016, France

Fabián Castaño
fabian.castano@frubana.com
Frubana
Bogota, Colombia

André Rossi
andre.rossi@dauphine.psl.eu
Université Paris Dauphine, PSL
Paris, 75016, France

Sonia Toubaline
sonia.toubaline@dauphine.psl.eu
Université Paris Dauphine, PSL
Paris, 75016, France

ABSTRACT

This paper addresses a robust variant of the Ring Star Problem where we assume that at most one hub can fail, among a given subset of nodes of the network. The network should remain functional despite this failure, which means there is an edge connecting the neighbors of any hub that can fail, and every terminal is connected to two different hubs that can fail or a single hub that cannot fail. The objective is to minimize the cost of such a robust Ring Star Network structure. The problem is addressed through an integer linear programming formulation, and a Benders decomposition is proposed as an alternative solution method. Computational experiments are carried out to compare these two approaches, and the results are analyzed.

KEYWORDS

Ring star problem, Robustness, Integer linear programming, Benders decomposition.

1 INTRODUCTION

Several network design, telecommunication, transportation, and facility location problems, among many others, involve designing networks in a tributary or backbone architecture. Different designs of tributary and backbone networks have been proposed, see for instance Klincewicz[8]. In this manuscript, we consider a Ring Star network design, where a complete mixed graph with both, arcs from and to every node, as well as edges between any pair of different nodes and a particular node called the depot are given. The Ring Star Problem (RSP) introduced by Labbé et al.[10] consists in selecting a subset of nodes that includes the depot, named hubs, and link them with a cycle to form the ring. A node that is not a hub is called a terminal. Each terminal is connected to exactly one hub, forming the star topology part. The aim of RSP is to minimize the sum of three costs corresponding to (i) selecting the subsets of hubs, (ii) forming the ring, and (iii) connecting the terminals to the ring. RSP is *NP*-hard since it contains the Traveling Salsman Problem as a special case when the assignment costs are very large compared to the ring costs.

RSP has been widely studied in the literature. Labbé et al.[10] proposed a Mixed Integer Programming model, strengthened with valid inequalities resulting from a polyhedral analysis and solved with a Branch-and-Cut algorithm. Another exact approach that takes advantage of the fact that the depot must be in the ring is introduced by Kedad-Sidhoum and Nguyen[6]. Calvete et al.[1] addressed the problem using a bilevel optimization approach and proposed an evolutionary-based heuristic for solving RSP while Zang et al.[13] recently proposed an ant colony system algorithm. As a consequence of some hazardous events (failures, attacks, etc.), designing a reliable topological network might be essential. Survivable network design problems have largely been studied in the literature as a means to provide robustness to networks [4, 7].

In [11], Labbé *et al.* consider a fully connected star problem where the selected hubs form a clique. The authors investigate the polyhedral properties of the proposed model and develop a custom branch-and-cut algorithm for solving it. Fouilhoux *et al.* [3] study a 2 edge-connected star problem where the backbone structure is a 2 edge-connected subgraph (A graph is k edge-connected, for a non-negative integer k, if there are at least k edge-disjoint paths between any pair of nodes). In [5], Karaşan *et al.* consider 2 edge-connected star problems where each terminal is connected to two selected hubs. Both papers provide integer programming models and valid inequalities for the studied problems, analyze facet-defining inequalities and present exact and heuristic separation algorithms. In these works, the survivability is considered either only for the backbone network [3, 11] or for both tributary and backbone networks [5]. In all cases, the topological structure may not be preserved in the case one hub fails. Indeed, as shown in Figure 1, the backbone structure is disconnected when the hub in the central position fails.

As a consequence of some hazardous events (failures, attacks, etc.), designing a reliable topological network might be essential. Survivable network design problems have largely been studied in the literature as a means to provide robustness to networks [4, 7]. In this work, we introduce a robust version of RSP where the ring star structure should be preserved whenever a single hub fails. The notion of robustness adopted complies with the one introduced in [9].

The rest of this paper is organized as follows. Section 2 defines the robust RSP. In Section 3 an integer linear program (ILP) is proposed to model a robust version of the RSP. In Section 4, a Benders decomposition is presented as an alternative solution method. Finally, computational results are presented and discussed in Section 6.

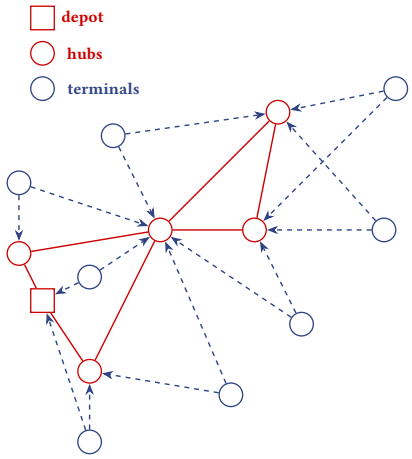


Figure 1: 2 edge-connected star (dual homed) network [5]

2 ROBUST RING STAR PROBLEM DEFINITION

In this section we first recall the RSP definition and introduce the robust version we study in this paper.

2.1 Ring Star Problem

We first recall the Ring Star Problem definition as proposed in Labbé et al.[10]. It consists of selecting a subset of nodes, called hubs, to be linked by a cycle (ring topology) and join all the terminals to the cycle (star topology). We consider a mixed graph $G = (V, E \cup A)$ with $V = \{1, 2, \ldots, n\}$ a node set where node 1 is a specific node called the depot, $E = \{ij | \quad (i, j) \in V^2, \quad i < j\}$ an edge set and $A = \{(c, h) | \quad (c, h) \in V^2\}$ an arc set.

- The **Ring** part aims to select a subset $H \subseteq V$ and link up all hubs of H with a cycle using edges of E. The cost of opening a hub $i \in V$ is $o_i \in \mathbb{R}_+$ and the cost of selecting an edge $ij \in E$ between two hubs i and j is $r_{ij} \in \mathbb{R}_+$. The depot node has to be in H.
- The **Star** requires that each terminal in $T = V \setminus H$ must be connected to exactly one hub in H. The cost of selecting arc $(t, h) \in A$ to connect terminal $t \in T$ to a hub $h \in H$ is $s_{th} \in \mathbb{R}_+$.

Finally, the RSP is to design a minimum-cost Ring Star network, composed of the sum of selecting the hubs, linking them to form the ring, and connecting the terminals.

2.2 Robust Ring Star Problem

The Robust Ring Star Problem, referred to as ρ-RSP where ρ stands for *robust*, has an additional input compared to RSP: $\widetilde{V} \subseteq V$ is a possibly empty subset of nodes that can fail if they are selected as hubs. A hub in $\widetilde{V}$ is called *uncertain* because it may fail, whereas a hub in $V \backslash \widetilde{V}$ is called *certain* as it is not supposed to fail [12].

The ρ-RSP is to build a minimal cost subgraph of G that will always contain a ring-star topology even if a hub in $\widetilde{V}$ fails. By contrast with RSP, if a hub belongs to $\widetilde{V}$, an additional edge has to join its two neighbors in the ring. This edge will be used if the hub in question fails. Furthermore, each terminal is linked to the ring by either connecting it to a single hub in $V \backslash \widetilde{V}$ (if there exist one), or by connecting it to two hubs in $\widetilde{V}$. The ρ-RSP is then to design a minimal cost robust ring-star network. Thus, it can be observed that ρ-RSP reduces to RSP when $\widetilde{V}$ is empty. Figure 2 shows an illustration of the ρ-RSP with $\widetilde{V} = V \setminus \{1, 5, 6\}$.

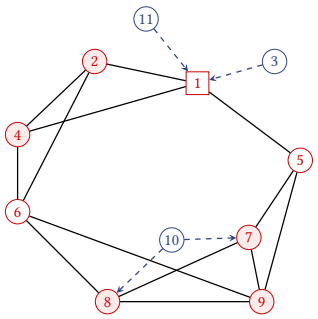


Figure 2: An instance of the ρ-RSP with $\widetilde{V} = V \backslash \{1, 5, 6\}$ and $H = V \backslash \{3, 10, 11\}$.

3 ILP FORMULATION OF ROBUST RSP

The proposed ILP formulation of ρ-RSP is based on the following decision variables: $x_{ij} \in \{0, 1\}, \ \forall (i, j) \in V^2, i < j$ is set to 1 if and only if edge $(i, j) \in E$ belongs to the ring. Note that we may sometimes refer to x_{ij} for $i > j$, in which case the actual computer implementation will simply replace x_{ij} with x_{ji}. Variable $y_{ij} \in \{0, 1\}, \ \forall (i, j) \in V^2$ is set to 1 if and only if terminal i is assigned to hub j and; y_{jj} is set to 1 if j is selected as a hub, and 0 if it is a terminal. Finally, $x'_{ik} \in \{0, 1\}$ is set to 1 if and only if i and k are hubs that have a common neighbor j that can fail in the ring (j is a hub in $\widetilde{V}$). Note that the edges for which x' is one do not need to form a cycle, for e.g. see Figure 4 and Figure 3. In Figure 3, x' and x variables equal to one are displayed. Since there are eight hubs in the ring, there are exactly eight x_{ij} nonzero variables and since there are five uncertain hubs, there are exactly five x'_{ij} nonzero variables. Finally, since the smallest ring has size 3, a robust solution must have a ring of size at least 3+1=4 in ρ-RSP unless all the hubs are in $V \backslash \widetilde{V}$. Variable σ is an integer that enforces that the ring has a size at least 4 whenever there is at least one selected hub in $\widetilde{V}$.

We will use the set of indices $\widetilde{J} = \{(i, j, k) \in V^3 : j \in \widetilde{V}, i \neq j, j \neq k, i < k\}$ and $V^{\neq} = \{(i, j) \in V^2 : i \neq j\}$ in the ILP formulation of ρ-RSP:

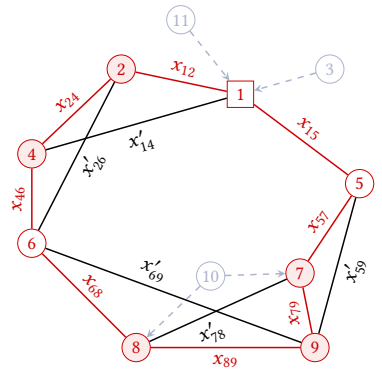


Figure 3: Same instance as in Figure 2 with x and x' variables equal to one displayed and non-hubs with lesser opacity

$$\text{Minimize} \sum_{i\in V}\sum_{\substack{j\in V\\ i<j}} r_{ij}(x_{ij}+x'_{ij}) + \sum_{i\in V} o_i y_{ii} + \sum_{i\in V}\sum_{j\in V\setminus\{i\}} s_{ij} y_{ij}$$

$$\sum_{\substack{j\in V\\ i<j}} x_{ij} + \sum_{\substack{j\in V\\ i>j}} x_{ji} = 2y_{ii} \qquad \forall i \in V \quad (1)$$

$$|S| - \frac{1}{|V\setminus S|}\sum_{i\in V\setminus S} y_{ii} \geq \sum_{i\in S}\sum_{\substack{j\in S\\ i<j}} x_{ij} \qquad \forall S \subset V : |S| \leq \tfrac{1}{2}|V| \quad (2)$$

$$\sum_{\substack{j\in V\setminus\widetilde{V}\\ i\neq j}} 2y_{ij} + \sum_{\substack{j\in\widetilde{V}\\ i\neq j}} y_{ij} = 2(1-y_{ii}) \qquad \forall i \in V \quad (3)$$

$$\sum_{i\in V}\sum_{\substack{j\in V\\ i<j}} x_{ij} \geq 3+\sigma \quad (4)$$

$$\sigma \geq y_{ii} \qquad \forall i \in \widetilde{V} \quad (5)$$

$$x_{ij} + x_{jk} \leq 1 + x'_{ik} \qquad \forall (i,j,k) \in \widetilde{J} \quad (6)$$

$$y_{ij} \leq y_{jj} \qquad \forall (i,j) \in V^{\neq} \quad (7)$$

$$y_{11} = 1 \quad (8)$$

$$\sigma \in \mathbb{N}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2, i<j$$

$$x'_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2, i<j$$

Constraints (1) correspond to connectivity constraints for the ring while (2) to subtour elimination constraints. Constraints (3) enforce that each terminal is connected to exactly two distinct hubs if these hubs are in $\widetilde{V}$, or to a single hub if it is in $V\setminus\widetilde{V}$. Constraints (4) and (5) state that the ring has size at least three if all hubs are in $V\setminus\widetilde{V}$, or four if at least one hub can fail. Constraints (6) enforce that for each hub $j \in \widetilde{V}$ having hubs i and k as neighbors in the ring, there is an edge that joins i and k, that can serve when j fails. There are $\frac{1}{2}\widetilde{n}(n-1)(n-2) = O(\widetilde{n}n^2)$ such inequalities where $n = |V|$ and $\widetilde{n} = |\widetilde{V}|$. Constraints (7) ensure that a terminal can only be linked to a hub. Finally, (8) force the depot to be part of the ring.

This model can be solved using a branch-and-cut approach, where subtour elimination constraints are added on-the-fly as Lazy Constraints.

4 BENDERS DECOMPOSITION OF ROBUST RSP

We can observe that whenever the ring is known (*i.e.*, the y_{ii} variables, the x_{ij} variables, and σ are fixed), determining all the other variables is an easy problem. Indeed, if the ring has 5 or more hubs, we can simply set x'_{ik} to one if and only if $j \in \widetilde{V}$, and $y_{jj} = x_{ij} = x_{jk} = 1$, and zero otherwise, and for each terminal i, we connect it either to the two nearest hubs in $\widetilde{V}$ or to the closest hub in $V\setminus\widetilde{V}$, the selected option being the one that incurs the minimum cost. Hence, we can devise a Benders decomposition whose master problem decides on the y_{ii}, x_{ij} and σ variables, whereas the subproblem is to set the remaining variables.

The master problem is:

$$\text{Minimize} \sum_{i\in V}\sum_{\substack{j\in V\\ i<j}} r_{ij}x_{ij} + \sum_{i\in V} o_i y_{ii} + \lambda$$

Subject to (1), (2), (4), (5), (8), and

$$\sigma \in \mathbb{N}$$

$$y_{ii} \in \{0,1\} \qquad \forall i \in V$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2, i<j$$

$$\lambda \geq 0$$

The numerical value of the x_{ij} and y_{jj} variables after solving the master problem are stored as $\hat{x}_{ij}$ and $\hat{y}_{jj}$, then passed to the subproblem, whose primal can be stated as:

$$\text{Minimize } \lambda = \sum_{i\in V}\sum_{\substack{j\in V\\ i<j}} r_{ij}x'_{ij} + \sum_{i\in V}\sum_{j\in V\setminus\{i\}} s_{ij}y_{ij}$$

$$\sum_{\substack{j\in V\setminus\widetilde{V}\\ i\neq j}} 2y_{ij} + \sum_{\substack{j\in\widetilde{V}\\ i\neq j}} y_{ij} = 2(1-\hat{y}_{ii}) \qquad \forall i \in V \quad (9)$$

$$x'_{ik} \geq \hat{x}_{ij} + \hat{x}_{jk} - 1 \qquad \forall (i,j,k) \in \widetilde{J} \quad (10)$$

$$y_{ij} \leq \hat{y}_{jj} \qquad \forall (i,j) \in V^{\neq}, \quad (11)$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^{\neq} \quad (12)$$

$$x'_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2, i<j$$

Where (9), (10), and (11) are derived from (3), (6), and (7) respectively.

This subproblem is easy to solve. Indeed, for all $(i,j,k) \in \widetilde{J}$ such that $\hat{x}_{ij} + \hat{x}_{jk} = 2$ we should set $x'_{ik} = 1$ if $\hat{y}_{jj} = 1$. If i is a terminal meaning $\hat{y}_{ii} = 0$, we compute m_i and m'_i as the closest hubs in $\widetilde{V}$.

- $m_i = \underset{j\in\widetilde{V}:\hat{y}_{jj}=1}{\arg\min}\ s_{ij}$ and
- $m'_i = \underset{j\in\widetilde{V}\setminus\{m_i\}:\hat{y}_{jj}=1}{\arg\min}\ s_{ij}$.

If no two such hubs exist, m_i and m'_i are set to zero. We also compute $m^\star_i$, as the closest hub in $V\setminus\widetilde{V}$: $m^\star_i = \underset{j\in V\setminus\widetilde{V}:\hat{y}_{jj}=1}{\arg\min}\ s_{ij}$. If no such hub exists, $m^\star_i$ is set to zero. Assuming that $s_{i,0}$ is infinite, if

$s_{i,m_i} + s_{i,m'_i} < s_{i,m^\star_i}$, then terminal i is connected to its two closest uncertain neighbors in the ring, otherwise i is connected to the closest certain hub in the ring. It can be observed that $s_{i,m_i} + s_{i,m'_i}$ and $s_{i,m^\star_i}$ cannot be simultaneously set to infinity, as the ring has at least three hubs.

The subproblem is originally an integer linear program, but it can be stated as a linear program by adding the following constraint: $y_{ij} \leq \sum_{k \in \widetilde{V}\setminus\{i,j\}: \hat{y}_{kk}=1} y_{ik}$ for all $i \in V$ such that $\hat{y}_{ii} = 0$, and for all $j \in \widetilde{V}\setminus\{i\}$ such that $s_{ij} = s_{im_i}$. If no such j exists, the constraint is not enforced. This constraint, labeled as (14) in the sequel, states that if terminal i is connected to an uncertain hub, then it should be connected to at least another one. Since (11) dominates (12), and because the objective function "pushes" the x' variables downward, integrality constraints can be dropped, and it can be shown that the linear relaxation of the subproblem (with the new constraints above) has an integral optimal solution. Let $\Omega = \{(i,j) \in V \times \widetilde{V}\setminus\{i\} : \hat{y}_{ii} = 0, s_{ij} = s_{im_i}\}$, the linear relaxation of the subproblem is:

$$\text{Minimize } \lambda = \sum_{i \in V} \sum_{\substack{j \in V \\ i<j}} r_{ij} x'_{ij} + \sum_{i \in V} \sum_{j \in V\setminus\{i\}} s_{ij} y_{ij}$$

$$\sum_{\substack{j \in V\setminus\widetilde{V} \\ i \neq j}} 2y_{ij} + \sum_{\substack{j \in \widetilde{V} \\ i \neq j}} y_{ij} = 2(1 - \hat{y}_{ii}) \qquad \forall i \in V \qquad (13)$$

$$-y_{ij} + \sum_{k \in \widetilde{V}\setminus\{i,j\}} y_{ik} \geq 0 \qquad \forall (i,j) \in \Omega \qquad (14)$$

$$x'_{ik} \geq \hat{x}_{ij} + \hat{x}_{jk} - 1 \qquad \forall (i,j,k) \in \widetilde{J} \qquad (15)$$

$$-y_{ij} \geq -\hat{y}_{jj} \qquad \forall (i,j) \in V^{\neq}, \qquad (16)$$

$$x'_{ij} \geq 0 \qquad \forall (i,j) \in V^2, i < j$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in V^{\neq}$$

5 DUAL FORMULATION OF BENDERS SUBPROBLEM RELAXATION

In this section, we state the dual of the linear relaxation of the subproblem, and the optimality cut that is used in the Benders decomposition. Then, we introduce a hybrid solution approach to solve the subproblem's dual. Rather than solving the master problem and the subproblem alternatively, the Benders decomposition is implemented as follows: whenever a feasible integer solution to the master problem is found, the subproblem is solved and an optimality cut is added to the master problem (if necessary). If the current integer solution to the master problem contains a subtour, we add a Benders feasibility cut, *i.e.*, a subtour elimination constraint (2).

5.1 Subproblem's dual and optimality cut

The decision variables of the subproblem's dual are associated with the subproblem's primal constraints as follows:

- Constraints (13) in the primal are associated with α_i $\forall i \in V$
- Constraints (14) in the primal are associated with δ_{ij} $\forall (i,j) \in \Omega$
- Constraints (15) in the primal are associated with β_{ijk} $\forall (i,j,k) \in \widetilde{J}$
- Constraints (16) in the primal are associated with γ_{ij} $\forall (i,j) \in V^{\neq}$

The dual of the relaxation of the subproblem is:

$$\text{Maximize } \lambda = \sum_{i \in V} 2(1 - \hat{y}_{ii})\alpha_i + \sum_{(i,j,k) \in \widetilde{J}} (\hat{x}_{ij} + \hat{x}_{jk} - 1)\beta_{ijk} - \sum_{(i,j) \in V^{\neq}} \hat{y}_{jj}\gamma_{ij}$$

$$2\alpha_i - \gamma_{ij} \leq s_{ij}, \qquad \forall i \in V \quad \forall j \in V\setminus(\widetilde{V} \cup \{i\}) \qquad (17)$$

$$\alpha_i - \gamma_{ij} \leq s_{ij}, \qquad \forall i \in V, \hat{y}_{ii} = 1 \quad \forall j \in \widetilde{V} \setminus \{i\} \qquad (18)$$

$$\alpha_i - \gamma_{ij} - \delta_{ij} + \sum_{\substack{k \in \widetilde{V}\setminus\{i,j\} \\ s_{ik}=s_{im_i}}} \delta_{ik} \leq s_{ij}, \qquad \forall (i,j) \in \Omega \qquad (19)$$

$$\alpha_i - \gamma_{ij} + \sum_{\substack{k \in \widetilde{V}\setminus\{i,j\} \\ s_{ik}=s_{im_i}}} \delta_{ik} \leq s_{ij}, \qquad \forall i \in V : \hat{y}_{ii} = 0 \quad \forall j \in \widetilde{V}\setminus\{i\} : s_{ij} > s_{im_i} \qquad (20)$$

$$\sum_{j \in \widetilde{V}: j \neq i, j \neq k} \beta_{ijk} \leq r_{ik}, \qquad \forall (i,k) \in V^2, i < k \qquad (21)$$

$$\alpha_i \in \mathbb{R}, \qquad \forall i \in V$$

$$\gamma_{ij} \geq 0, \qquad \forall (i,j) \in V^{\neq}$$

$$\delta_{ij} \geq 0, \qquad \forall (i,j) \in \Omega$$

$$\beta_{ijk} \geq 0, \qquad \forall (i,j,k) \in \widetilde{J}$$

The optimality cut to be added to the master problem of the Benders decomposition is then:

$$\lambda \geq \sum_{i \in V} 2(1 - y_{ii})\alpha_i + \sum_{(i,j,k) \in \widetilde{J}} (x_{ij} + x_{jk} - 1)\beta_{ijk} - \sum_{(i,j) \in V^{\neq}} y_{jj}\gamma_{ij}$$

Even if the subproblem can be stated as a linear program, it has a cubic number of β_{ijk} variables, which makes it long to solve. In order to solve it faster, we take advantage of the fact that these variables are independent of the other variables. The next section presents a quadratic time algorithm that finds a partial optimal solution to the subproblem's dual, in the sense that it sets the β_{ijk} variables. Then, the other decision variables of the subproblem's dual are set by solving the subproblem's dual where β_{ijk} variables and constraint (21) are removed. This "light" version of the subproblem dual has $O(|V|^2)$ variables and constraints, and is solved much faster.

5.2 An algorithm for solving partially the subproblem's dual

Algorithm 1 solves partially the dual of Benders subproblem's relaxation. It takes advantage of the fact that at most $|\widetilde{V}|$ of the β_{ijk} will be non-zeros and compute them in a quadratic running time. Knowing $\hat{x}_{ij}$ and $\hat{y}_{jj}$, we initially let β'_j be the adjacency list of

node $j \in \widetilde{V}$ such that $\hat{y}_{jj} = 1$: $\beta'_j[1]$ and $\beta'_j[2]$ are its two neighbors in the ring. Next we handle the special case of a four-hub ring, in which an edge joining two nonadjacent hubs may wrongly be counted twice. As an example, in Figure 4, the dashed edge between depot 1 and hub 3 must not be counted twice in the objective function as this edge preserves the ring structure when hub 2 or 4 fails. This is achieved in constant time, lines 13 to 19 in Algorithm 1: if there exist $j_1 \neq j_2$ in $\widetilde{V}$ such that $\hat{y}_{j_1 j_1} = \hat{y}_{j_2 j_2} = 1$ and $(\beta'_{j_1}[1], \beta'_{j_1}[2]) = (\beta'_{j_2}[1], \beta'_{j_2}[2])$, then we set β'_{j_2} to the empty set. Note that for all $(i, j, k) \in \widetilde{J}$, $\beta_{ijk} \neq 0$ implies that β'_j is nonempty (the converse does not hold when some ring costs are zero).

Algorithm 1: Building the robust edges in the dual of the subproblem of ρ-RSP

1 **Input:** $(\hat{y}_{ii})_{i \in V}$, $(\hat{x}_{ij})_{(i,j) \in V^2 | i<j}$ booleans
2 **Output:** $(\beta'_j)_{j \in \widetilde{V}}$
3 **foreach** $j \in \widetilde{V}$ **do**
4 $\beta'_j \leftarrow [\,]$
5 **foreach** $i \in V$ **do**
6 $\alpha_i \leftarrow 0$
7 **foreach** $j \in V : j \neq i$ **do**
8 **if** $j > i$ **and** $\hat{x}_{ij} = 1$ **and** $i \in \widetilde{V}$ **then**
9 Append j to β'_i
10 **if** $j > i$ **and** $\hat{x}_{ij} = 1$ **and** $j \in \widetilde{V}$ **then**
11 Append i to β'_j
/* Particular case of a 4-hub ring */
12 $H \leftarrow \{i \in V : \hat{y}_{ii} = 1\}$
13 **if** $|H| = 4$ **then**
14 $\widetilde{H} \leftarrow H \cap \widetilde{V}$
15 **foreach** $(j_1, j_2) \in \widetilde{H} \times \widetilde{H} : j_1 < j_2$ **do**
16 **if** $(\beta'_{j_1}[1], \beta'_{j_1}[2]) = (\beta'_{j_2}[1], \beta'_{j_2}[2])$ **then**
17 $\widetilde{H} \leftarrow \widetilde{H} \backslash \{j_2\}$
18 $\beta'_{j_2} \leftarrow \emptyset$
19 **return** $(\beta'_j)_{j \in \widetilde{V}}$

When Algorithm 1 terminates, β_{ijk} is set to r_{ik} for all $(i, j, k) \in \widetilde{J}$, and the optimality cut can be written as:

$$\lambda \geq \sum_{i \in V : \hat{y}_{ii}=0} 2(1 - y_{ii})\alpha_i - \sum_{(i,j) \in V^{\neq} : \hat{y}_{ii}=0} \gamma_{ij} y_{jj} + \sum_{j \in \widetilde{V} : \beta'_j \neq \emptyset} (x_{\beta'_j[1]j} + x_{j\beta'_j[2]} - 1) r_{\beta'_j[1]\beta'_j[2]}$$

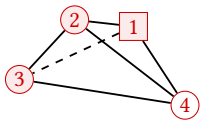


Figure 4: ρ-RSP instance with $\widetilde{V} = \{2, 3, 4\}$ with a dashed edge used when hub 2 or 4 fails.

6 COMPUTATIONAL RESULTS

The ILP formulations given in Sections 3 and 4 are addressed using a branch-and-cut approach. They have been implemented with Julia v1.5.2 and Gurobi v0.9.4 on a 16 GB RAM machine and an Intel(R) Core(TM) i7-10610U processor running at 1.80GHz.

Two types of instances have been used. First, we generated random instances with $n \in \{100, 200\}$ nodes. The nodes' coordinates are randomly drawn in $[1, n]$ for both abscissa and ordinates. The parameter $\alpha \in \{3, 5, 7, 9\}$ allows to compute the ring costs $r_{ij} = \lceil (10 - \alpha)\ell_{ij} \rceil$, for all (i, j) in E where ℓ_{ij} is the euclidean distance between $i \in V$ and $j \in V$. Opening costs o_i, for all $i \in V$ are either randomly distributed over $\mathcal{O}_i$ (where $\mathcal{O}_i$ is a random variable following the discrete distribution over the set $\mathbb{N} \cap [0.5n; 1.5n]$) or are equal to 1. Star assignment costs s_{ij} for all $(i, j) \in A$ are either randomly distributed over $\mathcal{S}_{ij}$ $\left(\text{where } \mathcal{S}_{ij} \text{ is a random variable such that } \mathcal{S}_{ij} \sim \frac{1}{\mathcal{U}([n; 3n/2])}\right)$, or defined by $s_{ij} = \lceil \alpha \ell_{ij} \rceil$, for all $(i, j) \in A$. For all random instances, we launched 5 runs and computed the average for all of them. Second, we used `eil 51` adapted from TSPLIB as in Labbé 2004 [10]. For this TSPLIB instance, we set $r_{ij} = \lceil (10 - \alpha)\ell_{ij} \rceil$, $o_i = n$. In all cases, $\widetilde{V} = V \setminus \{1\}$, and instance names are of the form `rand` $n - \alpha$ and `eil 51`$-\alpha$ in Table 1.

The formulation of ρ-RSP given in Section 3 is referred to as ILP, the one of Section 4 is referred to as Benders. Table 1 shows a comparison of ILP and Benders, with a time limit of 600 seconds per instance.

The output of our numerical results are as follows: **CPU** is the CPU Time in seconds for both methods. (TL) is indicated when the time limit is reached for at least one instance; **CPU SP** is the CPU time in seconds for Benders subproblem. Note that the master problem execution time is CPU − CPU SP; **Gap** represents the relative optimality gap of both methods computed as $\frac{|obj\_bound - obj\_val|}{|obj\_val|}$ where *obj_bound* and *obj_val* are the ILP objective bound and incumbent objective value; **n_subtour** is the number of subtour elimination constraints, *i.e.*, feasibility cuts for Benders, and Lazy Constraints for ILP; **n_cut** gives the number of optimality cuts in the Benders decomposition; and r^* corresponds to the percentage of hubs over total number of nodes in the best solution found.

We can see from Table 1 that for all random instances we generated, the Benders decomposition approach outperforms the ILP model. Those random instances are designed so that star costs are very low compared to opening and ring costs. For such instances, star costs are approximately n times smaller than the opening costs and the ring costs if opening costs are equal to 1, and n^2 smaller than the opening costs if the opening costs are randomly distributed over $\mathcal{O}_i$. In these instances, we observe that we have 4 hubs in the ring in all optimal solutions. For instances of size $n = 100$, Benders is between 1.5 to 2 times faster than ILP, and for $n = 200$, the speedup lies between 4 and 5. For `eil 51`, Benders decomposition is slower. This might be partially explained because star costs are of the same order of magnitude as opening and ring costs. For such instances, the master problem does not take into account subtour elimination constraints and the contribution of star costs. Hence, a large number of feasibility and optimality cuts are required to achieve lower bounds that are competitive with the ones provided

Table 1: Comparison of ILP and Benders

Instance n–α	Instance type		ILP				Benders Decomposition					
$n = \|V\|$	o_i	s_{ij}	CPU	Gap	r^*	n_subtour	CPU	Gap	r^*	n_subtour	CPU SP	n_cut
rand 100-3	O_i	S_{ij}	38.01	0%	0.04	4.0	16.04	0%	0.04	15.2	12.01	85.8
rand 100-5	O_i	S_{ij}	30.97	0%	0.04	46.2	13.08	0%	0.04	20	9.39	66.8
rand 100-3	1	S_{ij}	30.01	0%	0.04	36.6	19.79	0%	0.04	38.4	14.78	102.2
rand 100-5	1	S_{ij}	28.8	0%	0.04	21.6	19.78	0%	0.04	53.8	15.01	109.8
rand 200-5	1	S_{ij}	349.74	0%	0.02	59.0	71.3	0%	0.02	54	53.84	76.2
rand 200-7	1	S_{ij}	324.28	0%	0.02	81.2	76.03	0%	0.02	68.4	51.59	69.6
eil 51-3	n	$\lceil 3l_{ij} \rceil$	600.00 (TL)	26%	0.92	283.0	600.00 (TL)	47%	0.8	2492.0	58.25	754.0
eil 51-5	n	$\lceil 5l_{ij} \rceil$	600.00 (TL)	29%	0.08	566.0	600.00 (TL)	36%	0.18	3442.0	62.17	1438.0
eil 51-7	n	$\lceil 7l_{ij} \rceil$	600.00 (TL)	7%	0.08	118.0	600.00 (TL)	5%	0.08	1746.0	100.89	2332.0
eil 51-9	n	$\lceil 9l_{ij} \rceil$	11.04	0%	0.08	0.0	15.41	0%	0.08	31.0	11.17	312.0

by the solver while addressing the monolithic ILP model. This issue may be mitigated by reformulating the problem objective function to let the master problem use more knowledge about star costs. This may improve the lower bounds of the linear relaxation of the master problem.

7 CONCLUSION

Future works may be focused on accelerating the proposed Benders decomposition. A first option is to replace the hybrid method for solving the subproblem's dual by a single quadratic time algorithm to separate Benders optimality cuts faster. This would also open the way for the generation of stronger optimality cuts. Indeed, there may exist many different optimal solutions to the subproblem's dual, that may lead to different optimality cuts. It would be beneficial to return the solution that yields the strongest cut, and this aspect is currently not under control when we resort to a linear programming solver for separating Benders optimality cuts. In addition, a heuristic may be devised to address the master problem as in Costa et al. [2]. This would be useful especially for large instances, that might be beyond the range of exact approaches. Finally, lower bounds could also be proposed in an attempt to address this problem with a branch-and-bound algorithm, as this class of solution approaches does not seem have been explored to tackle ring star problem variants.

REFERENCES

[1] Herminia I Calvete, Carmen Galé, and José A Iranzo. 2013. An efficient evolutionary algorithm for the ring star problem. *European Journal of Operational Research* 231, 1 (2013), 22–33.

[2] Alysson M Costa, Jean-François Cordeau, Bernard Gendron, and Gilbert Laporte. 2012. Accelerating benders decomposition with heuristic master problem solutions. *Pesquisa Operacional* 32 (2012), 03–20.

[3] Pierre Fouilhoux, O. Ekin Karaşan, A. Ridha Mahjoub, Onur Özkök, and Hande Yaman. 2012. Survivability in hierarchical telecommunications networks. *Networks* 59, 1 (2012), 37–58.

[4] Martin Grötschel, Clyde L. Monma, and Mechthild Stoer. 1995. Design of survivable networks. In *Handbooks in operations research and management Science*, C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser (Eds.). Vol. 7. Elsevier, Amsterdam, 617–671.

[5] O. Ekin Karaşan, A. Ridha Mahjoub, Onur Özkök, and Hande Yaman. 2014. Survivability in hierarchical telecommunications networks under dual homing. *INFORMS Journal on Computing* 26, 1 (2014), 1–15.

[6] Safia Kedad-Sidhoum and Viet Hung Nguyen. 2010. An exact algorithm for solving the ring star problem. *Optimization* 59, 1 (2010), 125–140.

[7] Hervé Kerivin and A. Ridha Mahjoub. 2005. Design of survivable networks: A review. *Networks* 46 (2005), 1–21.

[8] John G. Klincewicz. 1998. Hub location in backbone/tributary network design: a review. *Location Science* 6, 1-4 (1998), 307–335.

[9] Panos Kouvelis and Gang Yu. 1997. *Robust discrete optimization and its applications*. Vol. 14. Springer Science & Business Media, Springer, Boston, MA.

[10] Martine Labbé, Gilbert Laporte, Inmaculada Rodríguez Martín, and Juan José Salazar González. 2004. The ring star problem: Polyhedral analysis and exact algorithm. *Networks: An International Journal* 43, 3 (2004), 177–189.

[11] Martine Labbé, Hande Yaman, and Éric Gourdin. 2005. A branch and cut algorithm for hub location problems with single assignment. *Mathematical Programming* 102 (2005), 371–405.

[12] André Rossi, Evgeny Gurevsky, Olga Battaïa, and Alexandre Dolgui. 2016. Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations. *Discrete Applied Mathematics* 208 (2016), 123–136.

[13] Xiaoning Zang, Li Jiang, Bin Ding, and Xiang Fang. 2020. A hybrid ant colony system algorithm for solving the ring star problem. *Applied Intelligence* 51 (2020), 3789–3800.

The r-Interdiction Network Design Problem with Commodity Outsourcing

İsmail Sevim[1], Necati Aras[2], and Mehmet Güray Güler[3]

[1]Dept. of Industrial Engineering, Boğaziçi University, İstanbul, Turkey, ✉ ismailsevi@gmail.com
[2]Dept. of Industrial Engineering, Boğaziçi University, İstanbul, Turkey, ✉ arasn@boun.edu.tr
[3]Dept. of Industrial Engineering, Yıldız Technical University, İstanbul, Turkey, ✉ mgguler@yildiz.edu.tr

In the Multi-Commodity Fixed-Charge Capacitated Network Design Problem (NDP), one deals with installing (or equivalently, designing) links between a set of given terminals and determining the flow of a set of commodities over these installed links [1]. The problem has a large number of application areas including, but not limited to, the strategic planning of airline and freight transportation, telecommunication, clean water supply, and wireless charging. A novel extension to the NDP, the NDP with Commodity Outsourcing (NPDCO), can be defined by adding the decision of outsourcing the shipment of a subset of commodities. In this study, we introduce the r-Interdiction NDPCO (RI-NDPCO) where we consider the NDPCO in an attacker-defender game (a static Stackelberg game) setting to model the strategic flight network design of a hypothetical small airline carrier (SAC). In the RI-NDPCO, the SAC targets to enter an airline market with potential threats from incumbent carriers and aims to analyze the maximum possible disruption in its flight network in the wake of an attack by a virtual attacker. A bilevel mixed-integer programming (BMIP) formulation is devised to model the problem.

We propose a Tabu Search (TS) heuristic with a data-driven neighbor sampling (DDS) procedure (TS-DDS) to solve the RI-NDPCO. A generic TS (or local search in general) with exhaustive search of the whole neighborhood of a current solution (i.e., best improvement/steepest descent) is an inefficient heuristic to solve the RI-NDPCO due to the computational complexity of the lower-level problem. To overcome this computational burden, only a fraction of the whole neighborhood of a current solution is evaluated at each iteration of TS-DDS. The DDS procedure is based on the idea of *biased* sampling of the neighboring solutions so that the probability of selecting the *best* neighbor is adequately high. The sampling relies on the predictions of a regression model trained (and re-trained periodically) using the solutions visited during the search on-the-fly where the actual objective value is the output/target variable. A substitute model that is used instead of an actual evaluation/objective function is known as a conjugate model in the data-driven metaheuristics literature and such models are proposed as operators in various heuristics [2]. To the best of our knowledge, this is the first study where a conjugate model (i) is used for biased sampling of candidate solutions, and (ii) is updated during the search. Moreover, we propose a pruning algorithm to discard a subset of neighboring solutions without evaluating them. This algorithm, called *bound pruning* (BP), is based on comparing the lower and upper bounds of all neighboring solutions where the bounds are obtained by linear relaxation and a rounding heuristic.

We work on a set of randomly generated instances Λ to analyze the ability of DDS, BP, and a hybrid method alongside with a restart diversification (RD) scheme. The test bed consists of 20 instances where the first 10 instances have the size of 9 nodes-36 arcs-72 commodities and the last 10 instances have the size of 10 nodes-45 arcs-90 commodities. We compare the following set of heuristics on these instances;

1. **TS:** TS with best improvement (BI) strategy, i.e., generic TS.
2. **TS-BP:** TS with BI strategy where some neighboring solutions are pruned by BP prior to computing the objective value of the remaining ones at each iteration.
3. **TS-DDS:** TS where a subset of all neighboring solutions are sampled by DDS prior to computing their actual objective value at each iteration.
4. **TS-BP&DDS:** TS where (i) the neighboring solutions are first pruned by BP, then (ii) a subset of remaining neighboring solutions are sampled by DDS prior to computing their objective value.

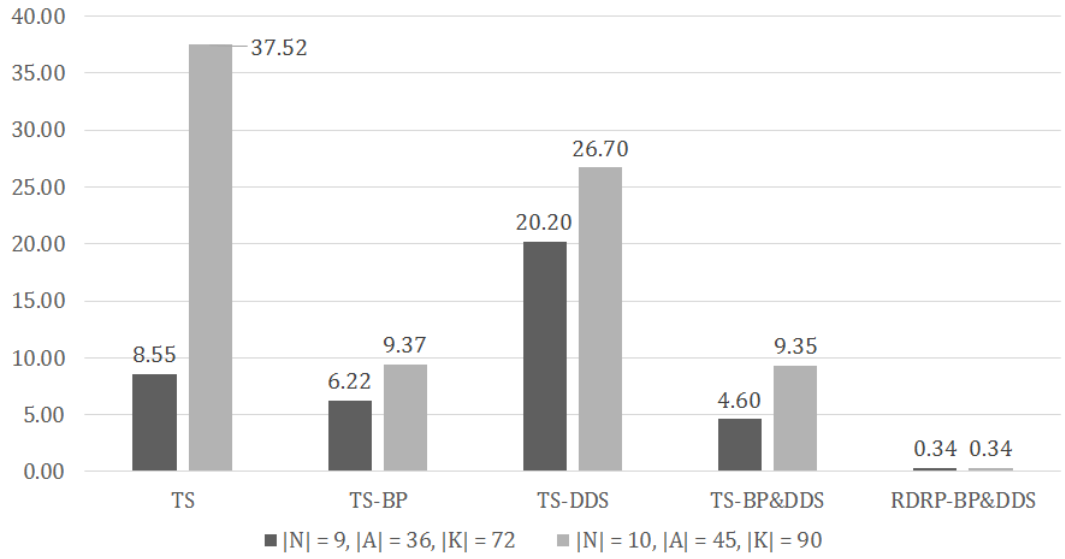
5. **RDRP-BP&DDS:** TS-BP&DDS incorporated into an RD scheme with random perturbation.

Preliminary computational results with a time limit of 3600 seconds and interdiction budget $r = 5$ are displayed in Figure 1. Due to the computational complexity of the RI-NDPCO, it is not possible to optimally solve all instances within reasonable computational time. Hence, we compare all methods using the following formula:

$$Gap(\%) = \frac{LB^i_{best} - LB^i_h}{LB^i_{best}} \times 100, \tag{1}$$

where LB^i_h is the objective function value of instance i found by heuristic h, and $LB^i_{best} = \max_h(LB^i_h)$.

Figure 1: Comparison of heuristics - $Gap(\%)$ averaged over instances of the same size.



According to the preliminary computational results, RDRP-BP&DDS returns the best average gaps for both instance sizes. Larger average gaps in TS-DDS are caused by directing the search into non-promising areas. A possible reason behind this fact is the lower prediction accuracy of the regression model at the earlier iterations of the search. However, when DDS is coupled with BP, the resulting hybrid heuristic returns better average gaps compared to TS-DDS. In future research, we plan to (i) work on RD schemes with a data-driven perturbation operator, (ii) expand the test bed, and (iii) devise a trilevel programming framework to model a defender-attacker-defender game in the context of RI-NDPCO.

References

[1] Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

[2] El-Ghazali Talbi. Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics. 2020.

Formulations for the maximum common edge subgraph problem

Etienne de Gastines[1] and Arnaud Knippel[1]

[1]LMI, INSA Rouen Normandie, Rouen, France ✉{etienne.mace_de_gastines, arnaud.knippel}@insa-rouen.fr

1 Introduction

The Maximum Common Edge Subgraph problem (MCES) was introduced by Bokhari in [5] to solve the mapping problem, where we need to assign tasks to processors while maximizing the fulfillment of communication demands. This problem also gives a measure of similarity between two given graphs and has been used as a tool for measuring the similarity of molecules.

It can be stated as follows:

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two graphs. $S = (V_S, E_S)$ is an edge subgraph of G if $V_S \subseteq V_G$ and $E_S \subseteq V_G$. If an edge subgraph of G is isomorphic to an edge subgraph of H, then we say it is a common edge subgraph of G and H. We aim to find a common subgraph with a maximal number of edges. We consider here the graphs G and H to be simple and undirected.

The problem is $\mathcal{NP}$-hard and generalizes several well-known graph problems, including subgraph isomorphism, maximum clique, maximum path and subgraph isomorphism.

Almohamad et Duffuaa proposed the first integer programming formulation in [1] (AD) to solve the more general problem of graph matching. They introduce variables $x_{u,v}\ \forall u \in V_G, v \in V_H$ that take value 1 if u is associated with v, and 0 otherwise, and variables counting the number of edges not belonging to the common edge subgraph. Another formulation was proposed independently by Marenco and Loiseau [7], [8] (ML), which uses the same x variables, and introduces variables $y_{u_1,u_2}\ \forall u_1u_2 \in E_G$ that take value 1 if edge u_1u_2 belongs to the common edge subgraph, otherwise 0. A third formulation was given by Bahiense, Manić, Piva, et de Souza [3], [4] (BMPS), which uses the same x variables, and introduces variables $c_{u_1u_2,v_1v_2}\ \forall u_1u_2 \in E_G, v_1v_2 \in E_H$, that take value 1 if edge u_1u_2 is associated with edge v_1v_2.

2 New formulations

Five new formulations are proposed : A symmetrized version of the formulation given by [3] (sym. BMPS) with a better relaxation, two reduced versions R1 and R2 which gradually reduce the number of variables. We give a reformulation of the one's proposed in [1] with fewer variables, and we show that it has the same relaxation as our more compact reduced formulation. We also give another formulation D even more compact, which introduces variables for the degree of vertices in the common edge subgraph.

3 Numerical results

We tested numerically the proposed formulations against the state of the art formulations on two datasets, one dataset with problems gathered by Marenco and Loiseau ([8]) and coming primarily from mapping problems, and a part of the ARG database ([6], [2]) composed of randomly generated graphs with diverse models. We observe that performance varies quite a bit depending on the structure and the density of the graphs. From our tests, it comes as a result that the R2 formulation is the most efficient on denser instances, and dominates the R1 formulation. BMPS and sym. BMPS formulations share similar behaviour, and are more efficient on some of the sparsest instances. While it does not solve these, the symmetrized version gets the best gaps on the hardest instances composed of meshes. On the first dataset, the ML formulation is faster on some small and medium sized instances. The formulation D is more efficient only on a few small instances. Globally, the R2 formulation is the fastest in average and solves the most instances. Moreover, it seems to have a better behaviour on the hardest instances, and often gets the best gap on unsolved instances.

Table 1: Asymptotic size of formulations and number of solved instances.

	n. variables	n. cons	solved instances
AD	$\mathcal{O}(\|V_G\|\|V_H\|)$	$\mathcal{O}(\|V_G\|\|V_H\|)$	140
ML	$\mathcal{O}(\|V_G\|\|V_H\|)$	$\mathcal{O}(\|E_G\|\|V_H\|)$	137
BMPS	$\mathcal{O}(\|E_G\|\|E_H\|)$	$\mathcal{O}(\|V_G\|\|E_H\| + \|V_H\|\|E_G\|)$	134
sym. BMPS	$\mathcal{O}(\|E_G\|\|E_H\|)$	$\mathcal{O}(\|V_G\|\|E_H\| + \|V_H\|\|E_G\|)$	136
R1	$\mathcal{O}(\|V_G\|\|E_H\| + \|E_G\|\|V_H\|)$	$\mathcal{O}(\|V_G\|\|V_H\|)$	143
R2	$\mathcal{O}(\|V_G\|\|V_H\|)$	$\mathcal{O}(\|V_G\|\|V_H\|)$	146
D	$\mathcal{O}(\|V_G\|\|V_H\|)$	$\mathcal{O}(\|V_G\|\|V_H\|)$	122

References

[1] H. A. Almohamad and S. O. Duffuaa, *A Linear Programming Approach for the Weighted Graph Matching Problem*, IEEE transactions on pattern analysis and machine intelligence, **15**, (1993), 522–525

[2] ARG Database, https://mivia.unisa.it/datasets/graph-database/arg-database/

[3] Bahiense, L. and Piva, B. and de Souza, C., *A branch&cut algorithm for the maximum common edge subgraph problem*, Electronic Notes in Discrete Mathematics **35** (2009), 47–52.

[4] Bahiense, L. and Manić, G and Piva, B. and de Souza, C., *The maximum common edge subgraph problem: A polyhedral investigation*, Discrete Applied Mathematics **160** (2012), 2523–2541.

[5] Bokhari, *On the Mapping Problem*, IEEE Transactions on Computers **3** (1981), 207–214.

[6] De Santo, M., Foggia, P., Sansone, C. and Vento, M., *A large database of graphs and its use for benchmarking graph isomorphism algorithms*, Pattern Recognition Letters **24** (2003), 1067–1079.

[7] Marenco, Javier and Loiseau, Irene, "A branch&cut algorithm for a problem arising in parallel programming environments ", Universidade de Buenos Aires, Departamento de Computación,2000.

[8] Marenco, Javier and Loiseau, Irene, "Un algoritmo branch-and-cut para el problema de mapping ", Master thesis, Universidade de Buenos Aires, Departamento de Computación,1999.

Session Session 2B: graph theory 2

Wednesday 8 June 2022, 13:45-15:00

Lecture Hall III

Optimal Vaccination Strategies for Multiple Dose Vaccinations

Jenny Segschneider[1] and Arie M.C.A. Koster[2]

[1]Lehrstuhl II für Mathematik, RWTH Aachen, Aachen, Germany, ✉ segschneider@math2.rwth-aachen.de
[2]Lehrstuhl II für Mathematik, RWTH Aachen, Aachen, Germany, ✉ koster@math2.rwth-aachen.de

1 Introduction

The ongoing COVID-19 pandemic and the shortage of vaccinations in the beginning of the vaccination campaign highlighted the importance of optimization problems considering the allocation and scheduling of vaccinations. An overview of these problems and literature has been given by Duijzer et al. [2] and in light of the ongoing pandemic by Wouters et al. in [8]. For most available vaccines, each person has to receive two doses in a specific time frame, i.e., the second dose has to be given three to six weeks after the first dose. This leads to another problem: which doses are given to new patients as a first dose and which are used as a second dose to those who already received a first dose. We will call this problem the Two-Dose Scheduling Problem. Most countries have chosen the approach of delaying the second dose as much as possible in order to achieve higher partial immunity through only one dose, for example Great Britain and Canada. Other countries, like Israel and the United States, have chosen to administer both doses as soon as possible in accordance with the phase 3 clinical trials, for example the trial [6] for the Corminaty vaccine from BioNTech and Pfizer.

Additionally, in practice, the deliveries proved to be highly uncertain. In this paper, we will study the problem without uncertainty to see how the problem can be modeled as a graph optimization problem and to determine the complexity of this and related problems. Furthermore, in the meantime, these vaccinations also require a third dose, the so-called booster shot, to be given to each person leading to the Three-Dose Scheduling Problem which will turn out to be NP-hard.

2 Related Work

The vaccination strategy has been subject of extensive research in the recent year. First, Jurgens and Lackner [3] examined the effect of delaying the second dose to 6, 9 or 12 weeks after the first dose on the population immunity. Silva et al. [7] proposed different SEIR models, that is a compartmental model using the compartments Susceptible, Exposed, Infectious and Recovered to minimize the number of expected deaths due to the pandemic and include different subpopulations. Parino et al. [5] extended an SIR model by adding different stages and applied it to the COVID-19 pandemic in Italy. Lastly, Moghadas et al. [4] proposed an agent-based model using data from the United States. However to the best of our knowledge, there have been no publications regarding the complexity of the combinatorial optimization problem addressed in this talk.

3 Summary of Results

To model the Two-Dose Scheduling Problem, we assume a finite time span in which all vaccinations take place and discrete time steps $\{1, \dots, n\}$. For example, these can be weeks or days. We first introduce a mixed integer programming formulation (MIP) for the base problem of finding a schedule for the first and second doses that obeys the necessary time gap between both shots for each patient and the delivered doses. The time gap constraint describes the fact that the second dose has to be given in a fixed time interval, i.e., 4 to 6 weeks, after the first shot and the delivery constraint ensures that only existing doses are used. We then extend this model by adding a constraint on the storage capacity, which means that

only a certain number of doses can be stored each week, and a vaccination speed constraint limiting the number of vaccination in each time step to including, for example, the limited amount of medical personal that can administer a dose of vaccine. We then solve these MIPs using Gurobi in order to understand some basic properties of the solutions for different target functions. It turns out that for both maximizing the expected immunity gained through vaccinations and minimizing the number of expected deaths through COVID leads to a "first doses first" strategy, where as many people as possible receive a first shot and second doses are delayed as much as possible and only given when necessary to satisfy the constraints.

Since this approach does not lead to a polynomial time algorithm, we next formulate the Two-Dose Scheduling Problem as a weighted b-matching problem, where each node represents a time step and each arc between two nodes represents all patients being fully vaccinated using doses delivered in these two time steps. The value of the nodes b is given by the number of doses delivered in the respective time slot. Note, that this does not mean the doses are given in the time steps belonging to the nodes. One of them may need to be delayed in order to satisfy the time gap constraint. The weight of each arc then symbolizes the value of that vaccination, which may include the immunity gained by or the number of expected deaths prevented through the vaccination. By finding a maximum weighted b-matching, the Two-Dose Scheduling Problem is solved. Hence, we can solve the base problem in polynomial time (e.g., [1]).

Next, by using an alternative interpretation we include the storage capacity and the vaccination speed constraints. This is achieved by connecting the nodes through a construction implementing the storing of doses in order to use them later. The two added constraints then become lower bounds on certain arcs in these constructions and can be assumed fixed for each matching. Thus, we can include the additional constraints and still solve the problem in polynomial time.

Lastly, we examined the Three-Dose Problem and showed its NP-hardness by reducing the three-dimensional maximum matching problem. For a given 3-D matching problem (X, Y, Z) and $T \subseteq (X \times Y \times Z)$, the idea is to define one time step for each element from $X \cup Y \cup Z$ such that each triple $(x, y, z) \in X \times Y \times Z$ is represented by three time steps satisfying the time gap constraint and vice versa. This result also implies that the Multiple-Dose Scheduling Problem is NP-hard for three or more doses.

References

[1] Richard P Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

[2] Lotty Evertje Duijzer, Willem Van Jaarsveld, and Rommert Dekker. Literature review: The vaccine supply chain. *European Journal of Operational Research*, 268(1):174–192, 2018.

[3] Graham Jurgens and Kyle Lackner. Modelled optimization of sars-cov-2 vaccine distribution: an evaluation of second dose deferral spacing of 6, 12, and 24 weeks. *medRxiv*, 2021.

[4] Seyed M Moghadas, Thomas N Vilches, Kevin Zhang, Shokoofeh Nourbakhsh, Pratha Sah, Meagan C Fitzpatrick, and Alison P Galvani. Evaluation of covid-19 vaccination strategies with a delayed second dose. *PLoS biology*, 19(4):e3001211, 2021.

[5] Francesco Parino, Lorenzo Zino, Giuseppe C Calafiore, and Alessandro Rizzo. A model predictive control approach to optimally devise a two-dose vaccination rollout: A case study on covid-19 in italy. *International Journal of Robust and Nonlinear Control*, 2021.

[6] Fernando P Polack, Stephen J Thomas, Nicholas Kitchin, Judith Absalon, Alejandra Gurtman, Stephen Lockhart, John L Perez, Gonzalo Pérez Marc, Edson D Moreira, Cristiano Zerbini, et al. Safety and efficacy of the bnt162b2 mrna covid-19 vaccine. *New England Journal of Medicine*, 383(27):2603–2615, 2020. PMID: 33301246.

[7] Paulo JS Silva, Claudia Sagastizábal, Luís Gustavo Nonato, Claudio José Struchiner, and Tiago Pereira. Optimized delay of the second covid-19 vaccine dose reduces icu admissions. *Proceedings of the National Academy of Sciences*, 118(35), 2021.

[8] Olivier J Wouters, Kenneth C Shadlen, Maximilian Salcher-Konrad, Andrew J Pollard, Heidi J Larson, Yot Teerawattananon, and Mark Jit. Challenges in ensuring global access to covid-19 vaccines: production, affordability, allocation, and deployment. *The Lancet*, 397(10278):1023–1034, 2021.

Local Certification of Reachability

Oliver Bachtler
TU Kaiserslautern
Kaiserslautern, Germany
bachtler@mathematik.uni-kl.de

Tim Bergner
TU Kaiserslautern
Kaiserslautern, Germany
bergner@mathematik.uni-kl.de

Sven O. Krumke
TU Kaiserslautern
Kaiserslautern, Germany
krumke@mathematik.uni-kl.de

ABSTRACT

Motivated by self-stabilising algorithms, the objective of local certification is to verify whether a (global) property holds while being restricted to a local view of the graph. Roughly speaking, this works as follows: a prover assigns a certificate to every vertex of a graph. Subsequently a verifier checks, for every vertex, whether its local view of the graph is consistent with the property we wish to verify. If this is the case at all vertices, then it accepts and it rejects otherwise. For a prover-verifier pair to locally certify a graph property, the verifier must accept any graph with this property that received certificates from the prover and it must reject any proof on a graph that does not have the desired property. The quality of such a pair is measured by the prover's size which is the length of the longest certificate it uses.

One common model for local certification is that of proof labelling schemes. This is a prover-verifier pair in which the verifier at a vertex v has access to the information at v as well as the certificates of all its neighbours. Determining whether an undirected graph has an st-path (for specified nodes s and t) can be done with a prover of size 1 and in the directed case it is known that $O(\log \Delta(G))$ bits suffice, where $\Delta(G)$ denotes the maximum degree of G. We prove a matching lower bound, in particular, showing that a constant amount of bits does not suffice.

KEYWORDS

Local Certification, Proof Labelling Schemes, Reachability

1 INTRODUCTION

The objective of local certification is to locally verify (global) properties in a distributed system. It is motivated by self-stabilising algorithms [4]. These algorithms are used in distributed systems which are subject to faults and have the property that they converge to a solution for a given problem. A possible way of designing such an algorithm is to first move to a solution and then to maintain it for as long as it remains correct. This so called local detection paradigm was introduced in [1]. Local certification describes this last step, where the algorithm needs to detect whether the solution is correct or not.

In local certification, a global prover has to convince a verifier that a graph has a specific property. To do so, the prover first presents a proof by assigning certificates to the vertices. Afterwards, the verifier decides at every vertex whether to accept or reject the proof provided. The decision at a vertex v is solely based on the local view of the graph around v, including certificates. If a graph has the designated property, the prover must be able to choose certificates in a way that makes the verifier accept at every vertex. Otherwise, the verifier must reject at some vertex of the graph, regardless of which certificates are given. What exactly the term local view means differs amongst the various local certification concepts.

To illustrate the concept, we sketch how local certification of bipartiteness works [10]. As local view, we assume that every vertex has access to its own certificate as well as those of its neighbours. In the case of a bipartite graph, the prover could specify a bipartition using the certificates 0 and 1. The verifier then only needs to check that the certificates of its neighbours differ from its own. If this is the case everywhere, then the graph is bipartite, so the verifier never accepts a non-bipartite graph. Furthermore, the bipartition specified by the prover makes the verifier accept. Hence, this prover-verifier pair locally certifies bipartiteness using a single bit.

Local certification does not restrict the computational power of the prover or verifier. Instead the quality is measured by the certificate lengths needed. The prover's size is the length of the longest certificate it assigns to a vertex. Given some property, a natural question to consider is what size a prover needs to have and what size suffices to locally certify this property.

The answer to this question may depend on the precise concept used since a "larger" local view of the verifier may result in smaller certificate lengths being sufficient. In this paper, we are interested mainly in two such concepts: proof labelling schemes and locally checkable proofs. These were introduced in [12] and [10], respectively. A formal definition is given in Section 2, but here it suffices to know that the verifiers in locally checkable proofs are more powerful that the ones in proof labelling schemes. Thus, ideally, one determines lower bounds on the prover's size using locally checkable proofs, and upper bounds using proof labelling schemes (since then they hold for both concepts).

Many results of this type are known, for example, certifying acyclicity [12] and planarity [6] requires $\Theta(\log n)$ bits, whilst minimum spanning trees need $\Theta(\log n \log W)$ bits [11], where n is the order of the graph and W is the largest weight of an edge. These bounds are for proof labelling schemes, though the planarity result also works for locally checkable proofs. That $\Theta(\log n)$ bits are required for acyclicity is also true in the locally checkable proof setting is was open until recently and is, in fact, also shown in [6]. For more results we refer to [5], a recent survey of local certification.

A very basic problem is that of certifying whether an st-path exists, for two specified vertices s and t, which we refer to as the st-reachability problem. In [10] it is shown how to solve this for undirected graphs using a single bit. In the directed case, a prover using $O(\log \Delta(G))$ bits, where $\Delta(G)$ in the maximum degree of the graph G, is described. Whether a constant size proof exists is posed

INOC 2022, June 7-10, 2022, Aachen, Germany

as an open question. We recapitulate how these locally checkable proofs work at the start of Section 3 and why the directed case cannot be solved analogously.

Another paper that looks at locally certifying st-reachability is [7]. Here the authors use a significantly more restrictive model in which the verifier sees less information, making it weaker. To obtain a logarithmic lower bound for st-reachability, they additionally restrict to a one-way communication model in which vertices only see their predecessors. The proof heavily relies on this restriction and a lower bound for the standard two-way communication is left open.

This suggests that the directed version is harder than the undirected one, which is a common occurrence: In the k disjoint paths problem the task to find vertex-disjoint paths between k pairs of specified vertices. It can be solved in polynomial time on undirected graphs for any fixed k [13] but it is NP-complete in the directed case, even for $k = 2$ [8]. Similarly, the feedback arc set problem is NP-complete [9] while its undirected counterpart is solved by computing a spanning tree. Another result [2] demonstrates that undirected st-reachability can be expressed by existential monadic second order logic and directed reachability cannot.

We show that locally certifying st-reachability (in the proof labelling scheme setting) is another such example. More precisely, we prove that the upper bound of $O(\log \Delta(G))$ is tight. In particular, this covers the missing lower bound in [7] and shows that no constant amount of bits suffices. Hence, this yields a new very basic problem where we now have tight bounds for proof labelling schemes, but whose lower bounds do not extend to locally checkable proofs.

Outline. In Section 2 we introduce the notation we need and formally define proof labelling schemes and locally checkable proofs. The proof of the lower bound for the directed st-reachability problem is presented in Section 3 and in Section 4 we discuss the difficulties that occur when extending our result to locally checkable proofs.

2 PRELIMINARIES

Basic notation. The notation for this paper is based on [3], but we briefly summarise what we need here. All graphs are non-empty and finite. We write uv for an edge from u to v and $E(v)$ denotes the set of edges incident to the vertex v. The set of neighbours of v in G is $N_G(v)$ or $N(v)$. We denote the maximum degree of a graph G by $\Delta(G)$ and its diameter by $\mathrm{diam}(G)$.

A *path* is a graph P with vertex set $V(P) = \{v_1, \dots, v_n\}$ and edge set $E(P) = \{v_1v_2, \dots, v_{n-1}v_n\}$, for which we write $P = v_1 \dots v_n$. If u and v are distinct vertices of a path P, then uPv is the unique path in P joining u and v. If $|uPv| \geq 3$, then we set $\mathring{u}P\mathring{v} \coloneqq uPv - \{u, v\}$. Similarly, for a uv-path P, we write $\mathring{P}$ for $P - \{u, v\}$.

Local certification. Let $\mathcal{G}$ be a class of (directed or undirected) graphs, for example, $\mathcal{G}$ could be the class of all connected undirected graphs. Further, let $\mathcal{F} \subseteq \mathcal{G}$ be the graphs in $\mathcal{G}$ that satisfy a certain property (acyclicity, for example). In this context we specify that graphs $G \in \mathcal{G}$ have an *identity* for every vertex $v \in V(G)$, which is denoted by $\mathrm{id}(v)$ and can be encoded in $O(\log |V(G)|)$ bits. All identities in a graph are distinct. Furthermore, vertices also have *labels*, $\mathrm{L}(v)$, that can contain further information (but may be empty). For example, these can be used to indicate colours of vertices or to select a certain subset of the edges.

A *proof* for a graph G is a function $\mathbf{P}\colon V(G) \to \{0,1\}^*$ that assigns a binary *certificate* (a bit string) to each vertex of G. The *size* $|\mathbf{P}|$ of a proof is the maximum number of bits in any of its certificates and the set of all proofs for a graph G is denoted by $\mathbb{P}(G)$. For the empty certificate and the empty label we write ε. Note that the label $L(v)$ of a vertex v is part of the graph G whereas the certificate $\mathbf{P}(v)$ at v is provided by the prover.

A *verifier* for a class $\mathcal{G}$ is a function $\mathcal{V}$ defined for all triples $(G, \mathbf{P}, v)$ with $G \in \mathcal{G}$, $\mathbf{P} \in \mathbb{P}(G)$, and $v \in V(G)$. Its output is a single bit, that is,

$$\mathcal{V}\colon \bigcup_{G \in \mathcal{G}} (\{G\} \times \mathbb{P}(G) \times V(G)) \to \{0,1\}.$$

A verifier $\mathcal{V}$ *accepts* (a proof $\mathbf{P}$) at a vertex v if $\mathcal{V}(G, \mathbf{P}, v) = 1$. It accepts (a proof $\mathbf{P}$ for) a graph G if it accepts at all $v \in V(G)$ and it *rejects* the proof otherwise. The verifier is *sound* for a subset $\mathcal{F}$ of $\mathcal{G}$ if it rejects any graph not in $\mathcal{F}$, regardless of the proof provided, that is, for all $G \in \mathcal{G} \setminus \mathcal{F}$ and all proofs $\mathbf{P} \in \mathbb{P}(G)$ there exists a vertex $v \in V(G)$ such that $\mathcal{V}(G, \mathbf{P}, v) = 0$.

A *prover* is a function $\mathcal{P}$ that maps every $G \in \mathcal{F}$ to a proof $\mathcal{P}(G) \in \mathbb{P}(G)$. The *size* $s(\mathcal{P})$ of a prover is the maximum size of any proof it assigns to a graph in $\mathcal{F}$, often expressed as a function of $|V(G)|$. A prover $\mathcal{P}$ is *complete* for a verifier $\mathcal{V}$ if $\mathcal{V}$ accepts $\mathcal{P}(G)$ for all $G \in \mathcal{F}$.

A pair $\pi = (\mathcal{P}, \mathcal{V})$, consisting of a prover and verifier, locally certifies $\mathcal{F} \subseteq \mathcal{G}$ if it is both *complete* and *sound*. The pair is complete if $\mathcal{P}$ is complete for $\mathcal{V}$ and it is sound if the verifier is sound for $\mathcal{F}$. The *size* $s(\pi)$ of π is just the size of its prover.

Restricting the verifier. So far the definition does not include locality and this is where the various concepts that are used in the literature differ. We now describe restrictions on the verifier that make its computation local and lead to the definition of proof labelling schemes and locally checkable proofs.

Let G be a graph and $v \in V(G)$. The set of vertices in G with distance at most r (with respect to number of edges) to v is denoted by $B^r_G(v)$ or just $B^r(v)$. For directed graphs, we always use the underlying undirected graph to determine balls. A verifier is *r-local* if it satisfies that

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^r_v, \mathbf{P}^r_v, v) \text{ for all } G, \mathbf{P}, v$$

where $G^r_v = G[B^r(v)]$ and $\mathbf{P}^r_v$ is the restriction of $\mathbf{P}$ to $B^r(v)$. We say a verifier is *neighbourhood-local* if

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^N_v, \mathbf{P}^1_v, v) \text{ for all } G, \mathbf{P}, v$$

where $G^N_v = (B^1(v), E(v))$. If, additionally, the verifier does not depend on the labels or identities of any vertex other than v when faced with $(G, \mathbf{P}, v)$, then we call the verifier *label-* or *identity-restricted*, respectively. We say it is *restricted*, if it is both label- and identity-restricted and satisfies that

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^-_v, \mathbf{P}^-_v, v) \text{ for all } G, \mathbf{P}, v$$

where G^-_v is the star with v at its centre and an edge vx_e (respectively x_ev) for each edge $e = vu$ ($e = uv$) in G. The proof $\mathbf{P}^-_v$ maps v to $\mathbf{P}(v)$ and x_e to the certificate $\mathbf{P}(u)$ if $e = vu$ or $e = uv$.

With this notation we can now define proof labelling schemes and locally checkable proofs.

Definition 2.1. A *proof labelling scheme* is a prover-verifier pair $\pi = (\mathcal{P}, \mathcal{V})$ in which $\mathcal{V}$ is restricted.

Definition 2.2. A *locally checkable proof* is a prover-verifier pair $\pi = (\mathcal{P}, \mathcal{V})$ in which $\mathcal{V}$ is r-local for some constant $r \geq 1$.

Note that, in a proof labelling scheme, the verifier at a vertex v may only use v's identity and label, as well as the certificates assigned to v and its neighbours. In contrast, the verifier in a locally checkable proof has access to much more information. Even if we restrict ourselves to a 1-local verifier, it can still use the identities and labels of the neighbours as well as the graph structure of the neighbourhood, meaning it knows which of its neighbours are the same and whether or not they are adjacent.

We introduced the term neighbourhood-local, which comes between the two. In contrast to proof labelling schemes, the verifier here can detect parallels and anti-parallels, but, unlike locally checkable proofs, it is still limited to direct neighbours and does not know which of these are adjacent. We actually prove that a pair $(\mathcal{P}, \mathcal{V})$ that locally certifies st-reachability needs $\Omega(\Delta(G))$ bits if the verifier is neighbourhood-local and identity-restricted in the next section. This yields the results for proof labelling schemes as a corollary.

An example. To wrap up the preliminaries, we illustrate these concepts on an example. We have already seen how to certify bipartiteness and we now sketch how to verify acyclicity using a proof labelling scheme as a slightly more involved example. As mentioned in the introduction, $\Theta(\log n)$ bits are needed for this and the lower bound does not extend to locally checkable proofs. We illustrate why and refer to [12] for a more detailed proof.

The upper bound is obtained by rooting a tree: An arbitrary vertex is chosen as the root and the certificate at each vertex is its distance to this root. To locally verify this, a vertex with certificate d need only check that it has exactly one neighbour with distance $d-1$ and all other neighbours have distance $d+1$ (unless $d = 0$ in which case all neighbours must have distance 1). This is a proof labelling scheme since the graph is a tree if the verifier accepts: in any cycle in the graph, the vertex with largest certificate has at least two neighbours whose certificates are not larger, which the verifier can detect.

To obtain the lower bound, assume that a proof labelling scheme exists that uses $o(\log n)$ bits. Then, for large enough n, we can assume it has fewer than $c \log n$ bits for some appropriate constant c, chosen such that any path of length n contains two disjoint pairs that are labelled identically. By connecting the second vertex of the first pair to the first vertex of the second pair, a cycle is obtained in which all local behaviour is identical. Since the path must be accepted, the verifier also accepts the cycle, contradicting soundness.

The reason this no longer works for locally checkable proofs is because, in this concept, the verifier can see the identities of its neighbours even if the size of the prover is too small to simply add these to the certificates. As a result, the verifier would notice that its neighbour has changed in the transition to the cycle, since the ends of the new edge now have a neighbour whose identity differs from the path.

3 VERIFYING REACHABILITY

The st-reachability problem. The *(directed) st-reachability problem* starts with a graph class $\mathcal{G}$ of (directed) graphs in which there is a unique vertex labelled s and one labelled t (and all other vertices have empty labels). The subclass $\mathcal{F}$ that we wish to verify contains those graphs in which t is reachable from s, that is, those graphs in $\mathcal{G}$ that have an st-path. For the remainder of this section, $\mathcal{G}$ and $\mathcal{F}$ will be used to denote these graph classes.

In the undirected case, a single bit is sufficient to verify reachability (as described in [10]): by fixing some shortest st-path P and setting $\mathbf{P}(v) = 1$ if $v \in P$ (and leaving the certificate empty otherwise), a vertex v can check that either $\mathbf{P}(v) = \varepsilon$ or exactly two of its neighbours are also assigned a non-empty certificate. The vertices s and t are exceptions, they must receive certificate 1 and require exactly one neighbour with this property.

This breaks down in the directed case, since the analogous requirement of asking for exactly one predecessor and one successor is not true. Even on shortest paths, the existence of back-edges may lead to larger quantities of both. This can be fixed by additionally specifying the distance from s (as in the verification of acyclicity), letting us detect back-edges. Alternatively, one can specify which of the edges incident to a vertex leads to the successor on the path. These yield upper bounds of $O(\log \operatorname{diam}(G))$ and $O(\log \Delta(G))$, both of which are potentially $O(\log n)$.

We now show that the second approach is best possible in the sense that $o(\Delta(G))$ bits are insufficient to obtain a proof labelling scheme in the directed case. To achieve this, we assume that a proof labelling scheme $(\mathcal{P}, \mathcal{V})$ exists that only uses x bits and therefore uses at most $c := 2^{x+1} - 1$ distinct certificates. We then construct a graph $G \in \mathcal{G}$ that the verifier would falsely accept and check that its maximum degree is polynomial in c. This yields that $\log \Delta(G)$ is some multiple of x, and $x \in \Omega(\log \Delta(G))$.

As we already mentioned at the end of the last section, our construction works even if we can detect parallels and anti-parallels and can see the labels of neighbouring nodes. It also does not use the relation between c and x. Therefore, we suppose that $(\mathcal{P}, \mathcal{V})$ is a prover-verifier pair with neighbourhood-local and identity-restricted verifier that locally verifies directed st-reachability using c distinct certificates. To facilitate our argumentation, we now provide some notation.

Split paths and their properties. Let $\overleftarrow{P}$ be an st-path with back-edges, that is, $P = sv_1 \dots v_k t$, $\overleftarrow{A} = \{v_i v_j : i > j\}$ is the set of back-edges of P, and $P \subseteq \overleftarrow{P} \subseteq P + \overleftarrow{A}$ (for some k). Note that $\overleftarrow{P} \in \mathcal{F}$. Let uv be an edge of P and ba be a back-edge in $\overleftarrow{P}$ with $uv \in \mathring{a}P\mathring{b}$. The *split-path* (of $\overleftarrow{P}$) at uv using ba is the graph $\overleftarrow{P} - \{uv, ba\} + \{bv, ua\}$, which is in $\mathcal{G} \setminus \mathcal{F}$. This operation is illustrated in Figure 1.

We now show that assigning certain certificates to the vertices of a path with back-edges would make $\mathcal{V}$ accept a split-path, and hence these assignments may not occur. For simplicity, we write $\mathbf{P}(xy)$ for $(\mathbf{P}(x), \mathbf{P}(y))$, where $\mathbf{P}$ is some proof and xy is an edge.

Lemma 3.1. *Let $\overleftarrow{P}$ be an st-path with back-edges, $\mathbf{P} = \mathcal{P}(\overleftarrow{P})$, uv be an edge of P, and ba be a back-edge in $\overleftarrow{P}$ with $uv \in \mathring{a}P\mathring{b}$. If $v \notin N(b)$, $u \notin N(a)$, and $vu \notin \overleftarrow{P}$, then $\mathbf{P}(uv) \neq \mathbf{P}(ba)$.*

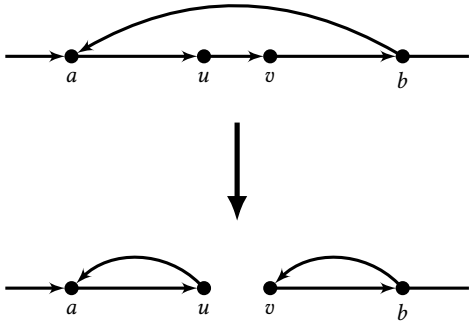


Figure 1: An illustration of the split-path operation.

Proof. Suppose $\mathbf{P}(uv) = \mathbf{P}(ba)$. Let S be the split path of $\overleftarrow{P}$ at uv using ba. We show that $\mathcal{V}$ accepts $\mathbf{P}$ for S, which contradicts $S \in \mathcal{G} \setminus \mathcal{F}$.

Notice that for all vertices $x \notin \{a, b, u, v\}$ the ball $B^1(x)$ remains unchanged and thus $\mathcal{V}$ accepts at all these vertices. Also, in the remaining four balls we only exchange the vertices a and v or b and u. Since these are assigned the same certificate by assumption, the verifier is faced with the same certificates. Moreover, none of these vertices can be s or t, so they are all unlabelled. Finally note that the remaining assumptions of the lemma ensure that none of the edges uv, ba, bv, or ua have a parallel or an anti-parallel in $\overleftarrow{P}$ or S, hence the graphs $\overleftarrow{P}^N_x$ and S^N_x also coincide for $x \in \{a, b, u, v\}$. □

Constructing a counterexample. We are now ready to construct a path with back-edges $\overleftarrow{P}$ in which any proof that the verifier accepts leads to a split path which is accepted as well. Since the verifier accepts $\mathcal{P}(\overleftarrow{P})$, this is a contradiction.

We let $r = \binom{c}{2} + c$ and define a graph G_k for $0 \le k \le r$, each of which is an st-path with back-edges. For a copy H of a graph G_k, we write $P(H)$ for the copy of the st-path of G_k in H and write $s(H), t(H)$ for its start and end, respectively. The graph G_0 is simply the path su_0v_0t. For $k \ge 1$, the graph G_k is the disjoint union of k copies $H_1, \dots, H_k$ of G_{k-1} which are combined to an st-path with back-edges as follows: copies H_i and H_{i+1} are connected by a path $t(H_i)u_iv_is(H_{i+1})$ introducing new vertices u_i and v_i. Additionally, we prepend the path $su_0v_0s(H_1)$ and append the path $t(H_k)u_kv_kt$. Finally, all possible back-edges between vertices $\{u_i, v_i\}$ and $\{u_j, v_j\}$ for $i > j$ are added.

This construction is visualised in Figure 2 and a formal definition of the edge and vertex set of the graph G_k is given below:

$$V(G_k) = \bigcup_{i=1}^{k} V(H_i) \cup \{u_i, v_i : 0 \le i \le k\} \cup \{s, t\},$$

$$E(G_k) = \bigcup_{i=1}^{k} E(H_i) \cup \{u_iv_i : 0 \le i \le k\} \cup \{v_{i-1}s(H_i), t(H_i)u_i : 1 \le i \le k\} \cup \{su_0, v_kt\}.$$

We say that a pair of certificates (c_1, c_2) is *missing* on a path with back-edges if no edge on the path has certificate (c_1, c_2). We extend this definition to sets $\{c_1, c_2\}$, where c_1 and c_2 need not be distinct, and call such a set missing if the tuples (c_1, c_2) and (c_2, c_1) are. Note that r is exactly the number of such sets. With these definitions at hand we can now prove the following lemma.

Lemma 3.2. *For every $k \le r$, the graph G_r contains a copy H of G_k in which at least $r - k$ sets are missing on the path $P(\mathring{H})$.*

Proof. We prove this by induction on k, where in the case $k = r$ there is nothing to show. For $k < r$ let H' be the copy of G_{k+1} given by the induction hypothesis and let $P' = P(\mathring{H'})$. Then $r - (k + 1)$ sets are missing on P' and every vertex on this path is assigned a certificate whose corresponding set is amongst the remaining $k + 1$ many. Since G_{k+1} has the $k + 2$ edges $u_0v_0, \dots, u_{k+1}v_{k+1}$, at least two of the corresponding edges in P' are assigned certificates that give rise to the same set $\{c_1, c_2\}$. For simplicity, we assume these edges are u_0v_0 and u_1v_1. This set is not missing in P', but we show that it is missing on the path $P = P(\mathring{H})$ where H is the copy of G_k between u_0v_0 and u_1v_1 in H'.

To see that this is indeed the case, we note that for any edge uv on the path P we can apply Lemma 3.1 to the edges uv and $ba = u_1u_0$ in G_r: since the only edges with an end in $V(H') \setminus V(H)$ and the other in $V(H)$ are $v_0s(H)$ and $t(H)u_1$, we get that $v \notin N(u_1)$ and $u \notin N(u_0)$. Moreover, $vu \notin G_r$ since G_r has no anti-parallels by construction. Therefore, the assumptions of Lemma 3.1 are satisfied, and $\mathbf{P}(uv) \ne \mathbf{P}(u_1u_0)$. The same holds for the back-edges u_1v_0, v_1u_0, and v_1u_1.

Since we assumed that both u_0v_0 and u_1v_1 are assigned a certificate corresponding to the set $\{c_1, c_2\}$, one of the four back-edges has certificate (c_1, c_2) and another has (c_2, c_1). Thus, we have ensured that both of these are missing, giving us the new missing set $\{c_1, c_2\}$ (in addition to the $r - (k + 1)$ many provided by the induction hypothesis), which completes the proof. □

By Lemma 3.2 for $k = 0$, G_r has a copy H of G_0 in which r pairs are missing on the path $P = P(\mathring{H})$. But since these are all possible pairs, the single edge in P has no valid assignment, which is a contradiction. To complete this section, we only need to see how c relates to $\Delta(G_r)$.

Observation 3.3. The maximum degree of G_r is $2r + 2 = c^2 + c + 2$.

Proof. We prove by induction that the maximum degree of G_k is $2k + 2$. Since G_0 is a path, this holds initially. For $k > 0$ let v be a vertex in G_k. If v is in some copy H of G_{k-1}, then its degree is at most $2k$: in H only the vertices $s(H)$ and $t(H)$ have incident edges to vertices outside of this copy of H, and these have degree 2 in G_k.

Any other vertex of G_k is s, t, or in $\{u_0, v_0, \dots, u_k, v_k\}$. The first two have degree 1 and any u_i or v_i has two neighbours on the path and $2k$ further neighbours in G_k, namely all u_j and v_j for $j \in \{0, \dots, k\} \setminus \{i\}$. Hence, the maximum degree of G_k is $2k + 2$.

The missing equality follows from the definition of r. □

By Observation 3.3, $\Delta(G_r)$ is indeed polynomial in c. Simple computations yield that at least $\log_2(c + 2) - 1$ bits are required to obtain $c + 1$ distinct certificates and $\log_2(\Delta(G_r)) \le 2\log_2(c) + 2$. Consequently, since we need at least $c + 1$ certificates to correctly verify G_r, at least $\frac{1}{2}\log_2(\Delta(G_r)) - 2$ bits are necessary. We have thus arrived at the desired result.

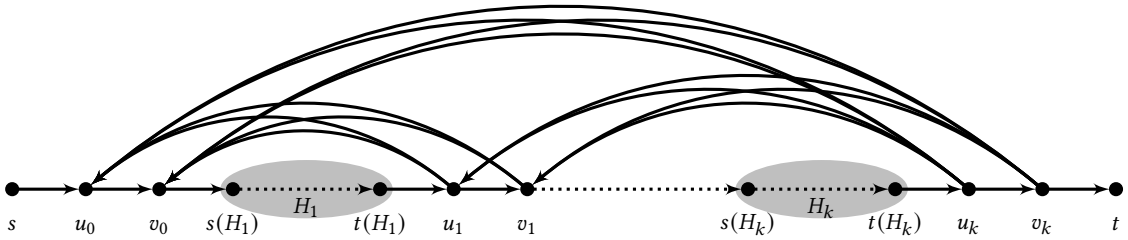


Figure 2: A visualisation of the graph G_k constructed for the proof of Theorem 3.4.

Theorem 3.4. *The st-reachability problem cannot be locally certified by a prover-verifier pair $(\mathcal{P}, \mathcal{V})$ of size $o(\Delta(G))$ if the verifier is neighbourhood-local and identity-restricted.*

As a corollary, we get the same result for proof labelling schemes.

Corollary 3.5. *There exists no proof labelling scheme of size $o(\log \Delta(G))$ for directed st-reachability.*

4 CONCLUSION AND FURTHER RESEARCH

We have shown that $\Omega(\log \Delta(G))$ bits are necessary in any proof labelling scheme for directed *st*-connectivity. This implies the only missing bound in [7], but does not answer the open question in [10], since locally checkable proofs are not covered. It would be desirable to obtain an example that fools the stronger verifiers allowed in locally checkable proofs. We now describe why this additional verification power is problematic for the example we constructed.

Note that we are able to deal with parallels and anti-parallels, but already had to exclude being able to see which neighbours are adjacent and which ones are not. The reason for this is that the proof of Lemma 3.1 no longer works in this case: if we look at u's local view of the graph in Figure 1, then it replaces v by a. But if v was adjacent to other neighbours of u and a is not, then the verifier can now detect this. Indeed, this happens in the graph G_r. In the graph G_k, the vertices $u_0, \ldots, u_k, v_0, \ldots, v_k$ form a complete graph (in the undirected sense). But if we transition to a split path at one of the edges $u_i v_i$, using a back-edge ba from outside this G_k, then we replace the neighbour v_i that is part of this complete graph by some other vertex a that neighbours none of these vertices.

Similarly, larger radii are problematic. It is not implausible to generalise Lemma 3.1 to paths (by replacing back-edges by "back-paths" and forbidding paths with the same certificate sequence below them). However, this is not helpful. The reason is that it does not allow us to generate new forbidden paths, which is easiest exemplified by assigning all intermediate vertices on back-paths a unique certificate, distinguishing them from the rest.

The last obstacle introduced by locally checkable proofs is that the identities of the neighbours are now visible, for which tools similar to the ones used in [10] seem to be required: multiple graphs in $\mathcal{F}$ with disjoint identities need to pieced together to obtain one that is not in $\mathcal{F}$, but locally appears to be.

REFERENCES

[1] Yehuda Afek, Shay Kutten, and Moti Yung. 1997. The local detection paradigm and its applications to self-stabilization. *Theoretical Computer Science* 186, 1 (1997), 199–229. https://www.sciencedirect.com/science/article/pii/S0304397596002861

[2] Miklos Ajtai and Ronald Fagin. 1990. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic* 55, 1 (1990), 113–150. https://doi.org/10.2307/2274958

[3] Reinhard Diestel. 2010. *Graph Theory* (fourth ed.). Graduate Texts in Mathematics, Vol. 173. Springer, Heidelberg; New York. https://doi.org/10.1007/978-3-662-53622-3

[4] Shlomi Dolev. 2000. *Self-stabilization.* MIT press.

[5] Laurent Feuilloley. 2021. Introduction to local certification. *Discrete Mathematics & Theoretical Computer Science* 23, 3 (Sep 2021). https://doi.org/10.46298/dmtcs.6280

[6] Laurent Feuilloley, Pierre Fraigniaud, Ivan Rapaport, Éric Rémila, Pedro Montealegre, and Ioan Todinca. 2020. Compact Distributed Certification of Planar Graphs. arXiv:2005.05863 [cs.DC]

[7] Klaus-Tycho Foerster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. 2018. Local checkability, no strings attached: (A)cyclicity, reachability, loop free updates in SDNs. *Theoretical Computer Science* 709 (2018), 48–63. https://doi.org/10.1016/j.tcs.2016.11.018 Special Issue of ICDCN 2016 (Distributed Computing Track).

[8] Steven Fortune, John Hopcroft, and James Wyllie. 1980. The directed subgraph homeomorphism problem. *Theoretical Computer Science* 10, 2 (1980), 111 – 121. https://doi.org/10.1016/0304-3975(80)90009-2

[9] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA.

[10] Mika Göös and Jukka Suomela. 2016. Locally Checkable Proofs in Distributed Computing. *Theory of Computing* 12, 19 (2016), 1–33. https://doi.org/10.4086/toc.2016.v012a019

[11] Amos Korman and Shay Kutten. 2006. Distributed Verification of Minimum Spanning Trees. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing* (Denver, Colorado, USA) *(PODC '06)*. Association for Computing Machinery, New York, NY, USA, 26–34. https://doi.org/10.1145/1146381.1146389

[12] A. Korman, S. Kutten, and D. Peleg. 2010. Proof labeling schemes. *Distributed Computing* 22 (2010), 215–233. https://doi.org/10.1007/s00446-010-0095-3

[13] N. Robertson and P.D. Seymour. 1995. Graph Minors .XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 65 – 110. https://doi.org/10.1006/jctb.1995.1006

Session Session 2C: routing problems 2

Wednesday 8 June 2022, 13:45-15:00

Lecture Hall IV

A granular local search for the demand-scenario-based district cutting problem for postal deliveries

Alina Theiß[1], Rossana Cavagnini[1], and Michael Schneider[1]

[1]Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany, ✉ theiss@dpo.rwth-aachen.de

1 Introduction and problem description

Cost-efficient routing of postmen is of major importance for companies like our industry partner Deutsche Post DHL Group (DPDHL). Currently, DPDHL pursues the strategy of clustering street segments of a city into districts and determining a tour within each district. This problem can be defined on a directed graph $G = (V, A)$. The vertex set $V = \{0, 1, 2, \ldots, n\}$ is composed of a depot 0 and a set of street segments $I = \{1, 2, \ldots, n\}$, and A is the arc set. Each street segment is characterized by a service time and each arc by a travel time. The goal is to cluster the street segments into a given number of districts and to determine the tours within the districts under the following constraints: (1) each street segment is visited exactly once, (2) each tour starts and ends at the depot, and (3) the total service time within each district stays between a given lower and upper bound. The objective is to minimize the total travel time required by the postmen for completing their tours. Because the number of letters varies on different days of the week and different seasons, we have different demand scenarios, characterized by the number of districts to form. DPDHL already solved the problem based only on the most likely demand scenario, the so-called standard demand scenario. The resulting problem is called the standard demand district cutting problem (SDDCP).

Before starting the tour each postman has to pick up the letters at the depot. The letters are already sorted on so-called preparation tables, where each table consists of shelves divided into sections representing the street segments. Because each shelf section is labeled with the names and numbers of the delivery addresses within the corresponding street segment, the order according to which the letters are sorted is fixed with respect to the SDDCP solution, i.e., it cannot be changed on a daily basis.

Motivated by the need of optimizing the use of postmen and of keeping the postmen workload balanced, we study the problem of modifying the district composition and the associated tours for demand scenarios different from the standard one, which require either a smaller or a larger number of postmen, i.e., districts. We refer to this problem as the demand-scenario-based district cutting problem (DSBDCP). The objective is to minimize the total travel time. Because the letters at the depot are sorted according to the SDDCP solution, in addition to the SDDCP constraints, the following ones must be satisfied:

1. If two street segments belong to the same district in the SDDCP solution, and if they are still in the same district in the DSBDCP solution (which is possibly different from the district of the SDDCP), then the precedence constraint referring to the visiting sequence of these two street segments becomes fixed (precedence constraints).
2. The number of changes in the predecessor and the successor of each street segment with respect to the SDDCP solution is bounded because the letters are already sorted according to that visiting sequence (adjacency constraints).
3. Because only one person can stand at each preparation table at the same time, the number of postmen visiting the same table is bounded (preparation table constraints).
4. Because each preparation table is located at a certain distance from the others and a postman can pick up the letters at a table only if the preparation work at that table has been completed, the number of tables visited by a postman is bounded (postman constraints).
5. To prevent that every postman visits as many tables as the the ones imposed by the upper bound of Point 4, an upper bound on the total number of table changes is also introduced (table change constraint).

Similarities of the DSBDCP to problems in the literature can only be found to a limited extent (see, for example, [1]). Moreover, standard solvers cannot even find a feasible solution to realistically sized instances in reasonable runtimes.

We propose a local search (LS) to address the DSBDCP and derive a preprocessing technique and sparsification methods which allow us to discard the evaluation of unpromising solutions. Preliminary results show that, on small-sized test instances, the standard solver outperforms the LS in terms of solution quality. However, on bigger test instances, finding feasible solutions is computationally expensive for the standard solver, while the LS returns feasible solutions within fractions of a second.

Section 2 describes the LS design, and Section 3 presents preliminary results and a conclusion.

2 Granular local search

We construct a feasible initial solution using two different algorithms, depending on whether the demand scenario asks for more or fewer districts than in the SDDCP solution. Then, we apply the LS. It uses a randomized variable neighborhood descent with a first-improvement strategy to explore the neighborhoods generated by three different relocate operators. Each neighborhood is searched completely before moving on to the next one. If an improving move is found, we apply the move and restart the search using the current neighborhood. An operator and an arc $(i, j) \in A$, called generator arc, uniquely identify a move which generates a neighboring solution. To reduce the neighborhood sizes and speed up the search, the number of considered generator arcs is restricted. First, we apply a preprocessing step in which all arcs violating the precedence constraints are removed. Then, for each street segment, we consider only a subset of the shortest remaining arcs. We enrich the generator arc set with the arcs of the given SDDCP solution because they always fulfill the additional DSBDCP constraints. Furthermore, we add the arcs connecting two street segments assigned to the same preparation table and not violating the precedence constraints. These arcs are always beneficial to limit the number of preparation tables visited by a postman. During the search, only feasible improving moves are carried out. Because we use a first-improvement strategy, the search trajectory depends on the order in which we evaluate the moves. Consequently, we execute the LS for ten runs and, at the beginning of each run, we randomly shuffle the order of the neighborhoods and that of the moves within each neighborhood.

3 Preliminary results and conclusion

We implemented the LS in C++ and used the clang v.12 compiler. For comparison purposes, we solved the DSBDCP model using Python 3.7.3 and Gurobi 9.0.0. The experiments were performed on an Intel(R) Xeon(R) computer with a CPU E5-2430 v2 processor at 2.50GHz with 64 GB RAM under CentOS GNU/Linux 7. We tested the LS on a small set of test instances obtained from the real-world data provided by DPDHL. The test set contains instances of three different sizes, with 30, 50, and 100 street segments, respectively. The LS is able to find feasible solutions for all instance sizes within 0.02 seconds. Gurobi solves the instances with 30 street segments to optimality within an average runtime of 1434 seconds. The average gap between the solutions found by the LS and the optimal solution is 16.49%. For instances with 50 street segments, Gurobi finds feasible but no optimal solution within three hours with an average optimality gap of 35.95%. Finding the first feasible solution for Gurobi is time-consuming: on average, it takes 149.75 seconds. The solutions found by LS are on average 34.29% better than Gurobi's first-found feasible solution and 10.67% worse than the solution found by Gurobi after three hours. For instances with 100 street segments, Gurobi cannot find a feasible solution within three hours for most of the instances. Because real-world instances contain approximately 500 street segments, and the problem needs to be solved on a daily basis, using heuristics is the only viable alternative.

In this work, we have presented an LS for solving the DSBDCP. The LS shows clear advantages over a commercial solver in terms of computational effort. Currently, the LS is implemented as simple local descent. Mechanisms for both handling infeasible solutions and penalizing them during the search as well as mechanisms for diversification and intensification will be included to escape local optima.

References

[1] Reza Tavakkoli-Moghaddam, Farid Taheri, Mohammad Bazzazi, M Izadi, and Farrokh Sassani. Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *Computers & Operations Research*, 36(12):3224–3230, 2009.

A Branch and Price Algorithm for the Heterogeneous Fleet Multi-depot Multi-trip Vehicle Routing Problem with Time Windows

Munise Kubra Sahin[1] and Hande Yaman[2]

[1]ORSTAT, KU Leuven, Leuven, Belgium, ✉ munisekubra.sahin@kuleuven.be
[2]ORSTAT, KU Leuven, Leuven, Belgium, ✉ hande.yaman@kuleuven.be

The increase in urban population accompanied by the growth of e-commerce aggravates the already problematic issue of last-mile deliveries, causing more congestion, pollution and traffic accidents in central urban areas. Many countries have implemented pull and push measures to cope with the negative effects of freight vehicles on the quality of life in urban areas. Push measures, such as defining restricted zones where large vehicles are prohibited during busy hours, discourage the use of large vehicles. Pull measures, such as providing as direct a route as possible for bikes, promote cycling by improving the convenience of bicycle use. With this aim, some European cities such as Copenhagen and Antwerp have constructed bicycle bridges to provide short-cuts for bikers. These measures and incentives drove companies to deploy more sustainable vehicles such as cargo bikes for last-mile deliveries. However, cargo bikes do not provide an all-embracing solution to the delivery companies due to their limited speed on larger roads and their low load capacity compared to traditional vehicles. Hence most delivery companies using cargo bikes operate with a heterogeneous fleet to make use of the compatibility of different vehicle types to different routes. With the integration of small vehicles to the fleet, obliging vehicles to perform at most one route, as in the traditional setting of the VRP, leads to an oversized fleet and inefficient utilization of workdays. Relaxation of this assumption for a heterogeneous fleet and allowing vehicles to perform several trips per day result in a heterogeneous fleet multi-trip VRP. Due to this multi-trip feature, using a depot outside the city center becomes more costly, encouraging companies such as DHL and UPS to use several micro-depots in central areas.

Inspired by these new challenges the companies face, we study the heterogeneous fleet multi-depot multi-trip VRP with time windows under shared depot resources where small and large vehicles have different travel times in certain areas. We formulate this problem using workday variables and propose a branch and price algorithm whose performance is enhanced by a new heuristic algorithm based on the reduction in the graph size. The proposed algorithm introduces a new way to compute the completion bounds using the iterative structure of the state-space augmenting algorithm and eliminates the need for solving a separate relaxation. We conduct experiments on modified small and medium-size instances from Solomon's benchmark set. The results of our computational experiments show that the proposed algorithm is very effective and can solve instances with up to 40 customers, two depots and two types of vehicles.

Session Session 3A: robust optimization

Thursday 9 June 2022, 10:30-12:10

Lecture Hall I

Recoverable Robust Periodic Timetables

Vera Grafe[1] and Anita Schöbel[2]

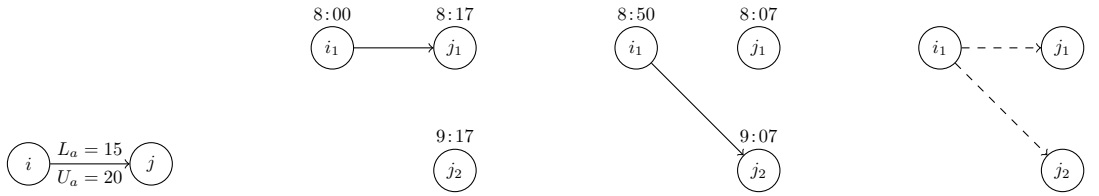
[1]Department of Mathematics, TU Kaiserslautern, Kaiserslautern, Germany, ✉ grafe@mathematik.uni-kl.de
[2]Department of Mathematics, TU Kaiserslautern, Kaiserslautern, Germany, ✉ schoebel@mathematik.uni-kl.de

An important aspect of optimising public transport is finding a good timetable. Often a *periodic* timetable is desirable, i.e. a timetable which repeats in a regular pattern (e.g. every hour). On the one hand, short travelling times are important from the passengers' point of view. The problem of finding a periodic timetable with minimal travelling times is known as the *Periodic Event Scheduling Problem (PESP)* and is well researched, see e.g. [4]. On the other hand, tight timetables without buffer times are prone to delays, which are inevitable in practice and highly dissatisfactory for the passengers. Hence, a good timetable should also have some degree of delay resistance. Many concepts and ideas on how to increase the robustness of a timetable against delays exist, see [6]. However, none of these approaches uses the promising concept of *recoverable robustness* introduced by [5]. The aim is to find a timetable with small travelling times such that in every delay scenario from a given uncertainty set $\mathcal{U}$ it is possible to *recover* the solution, i.e. to find a disposition timetable which is still acceptable for the passengers, e.g. the total delay should be below some given bound. A timetable which can be recovered for every delay scenario is called recovery robust. Hence, our aim is to combine two problems: The PESP and the Delay Management problem (DM).

We briefly describe the PESP, which is used for periodic timetabling. In the PESP we are given a period T together with a set of events $\underline{\mathcal{E}}$, which correspond to the arrival or the departure of a traffic line at some station. Furthermore, we have activities $\underline{\mathcal{A}}$, which represent processes between the events. Together, we obtain an *event-activity-network* (EAN) $\underline{\mathcal{N}} = (\underline{\mathcal{E}}, \underline{\mathcal{A}})$ in which the events are represented as nodes and the activities as arcs. We distinguish several different types of activities. *Driving activities* model a train line driving from one station to another, while *waiting activities* represent a line waiting at a station. Passengers have the possibility to transfer between different lines, which is included by the *transfer activities*. Additionally, we have passenger weights: w_a is the number of passengers using activity $a \in \underline{\mathcal{A}}$ and w_i is the number of passengers arriving at their destination $i \in \underline{\mathcal{E}}$. A timetable with period T is a mapping $\pi \colon \underline{\mathcal{E}} \to \{0, \dots, T-1\}$ assigning a time to every event. Every activity has a lower bound L_a and an upper bound U_a for its duration, i.e. for every $a = (i, j) \in \underline{\mathcal{A}}$ a feasible timetable π has to fulfil $\pi_j - \pi_i + z_a T \in [L_a, U_a]$ for some $z_a \in \mathbb{Z}$. The objective is to minimise the total travelling time.

In practice, delays occur and make the timetable infeasible. Hence, the Delay Management problem consists of two tasks: finding a disposition timetable, i.e. a new mapping of times to events respecting the source delays, and deciding which transfers between different trains should be maintained and which should be cancelled. See [2] for a survey on delay management.

The difficulty when integrating PESP and DM is that delays in general do not occur periodically, and hence, DM is not considered in the periodic EAN, but in a non-periodic network. I.e. the events do not represent the arrival or departure of a *line* (which repeats every T minutes), but of a single *trip*. Usually, PESP and DM are solved sequentially, i.e. DM has a timetable as input. This timetable is used to *roll out* the periodic EAN $\underline{\mathcal{N}} = (\underline{\mathcal{E}}, \underline{\mathcal{A}})$ to a non-periodic network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ for some planning horizon $I = [t_{\min}, t_{\max}]$. If K is the number of periods in I, every periodic event $i \in \underline{\mathcal{E}}$ has K realisations $i_1, \dots, i_K$ in the non-periodic network. The set of activities $\mathcal{A}$ depends on the timetable, as can be seen in the small example in Figure 1. In our paper, we integrate PESP and DM. PESP is a periodic problem in the EAN, while Delay Management uses the aperiodic network $\mathcal{N}$. However, we have to consider both problems in the same network. We decided to use the aperiodic network $\mathcal{N}$. Hence, we face two challenges: Rolling out the EAN without a timetable, and finding a periodic timetable in a non-periodic network. We solve the first problem by adding all possible activities when rolling out the network (as in Figure 1d) and choosing a subset while simultaneously fixing the timetable. Hence, we solve an assignment problem. The periodicity is ensured by synchronisation constraints between events corresponding to the same periodic



(a) Periodic activity (b) Rolled out activity for one timetable (c) Rolled out activity for another timetable (d) Both possible rolled out activities

Figure 1: Problem of rolling out a periodic EAN without knowing the timetable

event. The problem of finding a periodic timetable in this rolled out network, called *Periodic Timetabling in Aperiodic Network (PTTA)*, was introduced in [3], where it was also shown to be equivalent to PESP. Based on this model we now develop an integer programming formulation for the *Recoverable Robust Periodic Timetabling Problem (RRPT)*, i.e. the problem of finding a periodic timetable and a disposition timetable for every scenario $r \in \mathcal{U}$, such that the sum of nominal travelling time and worst-case delay summed over all passengers is minimised.

For the integer programming formulation we need variables π_i for $i \in \mathcal{E}$ determining the time of every event and binary variables u_a for $a \in \mathcal{A}$ for choosing the activities in the assignment problem of PTTA. We have to solve the delay management problem in every scenario. Hence, for every $r \in \mathcal{U}$ we need variables x_i^r determining the time of event $i \in \mathcal{E}$ in the disposition timetable and binary variables y_a^r for every transfer activity a, determining if the transfer is maintained. Since the disposition timetable depends on the choice of the activities in the assignment problem, we have coupling constraints between PTTA and DM. Because of the dependence on the choice of u_a and y_a we need different types of big-M-constraints and auxiliary variables for linearising quadratic constraints. The IP formulation is analysed and experiments are presented.
Apart from introducing an integer programming formulation for RRPT, we analyse how the concept of recovery robustness differs from the established concept of strict robustness (see [1]) in the described railway setting. Namely, we identify similarities and differences of both concepts in special cases. Furthermore, we sketch an iterative heuristic to solve the integrated problem RRPT.

References

[1] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[2] Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay propagation and delay management in transportation networks. In *Handbook of Optimization in the Railway Industry*, pages 285–317. Springer, 2018.

[3] Vera Grafe and Anita Schöbel. Solving the Periodic Scheduling Problem: An Assignment Approach in Non-Periodic Networks. In Matthias Müller-Hannemann and Federico Perea, editors, *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*, volume 96 of *Open Access Series in Informatics (OASIcs)*, pages 9:1–9:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[4] Christian Liebchen. *Periodic timetable optimization in public transport*. PhD thesis, TU Berlin, 2006.

[5] Christian Liebchen, Marco E. Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Lecture Notes in Computer Science*, 5868:1–27, 2009.

[6] Richard M. Lusby, Jesper Larsen, and Simon Bull. A survey on robustness in railway planning. *European Journal of Operational Research*, 266:1–15, 2018.

Improving Affine Routing Formulations for Robust Network Design

Yacine Al-Najjar[1,2], Walid Ben-Ameur[1], and Jérémie Leguay[2]

[1]Samovar, Telecom SudParis, Institut Polytechnique de Paris, France., ✉ walid.benameur@telecom-sudparis.eu
[2]Huawei Technologies, Paris Research Center, France., ✉ yacine.alnajjar@huawei.com✉ jeremie.leguay@huawei.com

Introduction

Given the dynamic nature of traffic in telecommunications networks, we investigate the variant of the robust network design problem where we have to determine the capacity to reserve on each link so that each demand vector belonging to a polyhedral set can be routed. The objective is either to minimize congestion or a linear cost. Routing is assumed to be fractional and dynamic (i.e., dependent on the current demand vector).

It has been proved in [1] that the problems above are $\Omega(\frac{\log n}{\log \log n})$ hard to approximate under the ETH conjecture, where n is the number of nodes. On the other hand, upper bounds can be obtained in polynomial-time by considering static/oblivious routing. In fact, it is shown in [6] that static routing leads to an $O(\log n)$ approximation for the dynamic routing variant of the problem.

Several approaches have been proposed to improve the solutions given by a static routing such that the multi-static routing [3] or the affine routing [4, 5]. Here we will present some of approaches to improve the quality of the affine routing solution [2]. Roughly speaking, affine routing consists in restraining the flow of each commodity $h \in \mathcal{H}$ to affinely depend on the demand scenario $d \in \mathcal{D}$ where $\mathcal{H}$ is the set of commodities and $\mathcal{D}$ is the uncertainty set (a polyhedral set representing the set of plausible traffic vectors). This can be done as follows. For each $h \in \mathcal{H}, e \in E$ (where E is the set of network links), the flow related to h and sent through e denoted by $f_{h,e}(d)$ is given by $f_{h,e}(d) = x^0_{h,e} + \sum_{h' \in \mathcal{H}} x^{h'}_{h,e} d_{h'}$ where $x^0_{h,e}$, $x^{h'}_{h,e}$ for $h' \in \mathcal{H}$ are $1 + card(H)$ are new variables that are subject to optimization. A first version where path variables are considered is proposed in [4], while [5] focused on node-arc formulations (i.e., using Kirchhoff's laws).

Relaxing the flow conservation constraints

In this section, we present some improvements of the node-arc formulation by relaxing the flow conservation constraints. Such improvements permit us to further reduce the cost of the solution and minimize the gap with the solution given by the dynamic routing. The standard formulation with an affine routing is denoted by $\mathcal{F}_=$ ("=" means that we have equalities in the flow conservation constraints). Let $\mathcal{F}_+$ be the formulation obtained by replacing the flow conservation constraints by the following inequalities.

$$\sum_{e \in \delta_+(v)} f_{h,e}(d) - \sum_{e \in \delta_-(v)} f_{h,e}(d) \begin{cases} \geq d_h, & \text{if } v = s(h) \\ \geq 0 & \text{if } v \neq s(h), t(h) \end{cases} \tag{1}$$

where $s(h)$ (resp. $t(h)$) is the source (resp. sink) of commodity h while $\delta_+(v)$ (resp. $\delta_-(v)$ denote the set of edges going out of (resp. into) v. In [2] we show the somewhat surprising result that this relaxation leads to a valid solution of the dynamic routing problem. Said another way, by interpreting $f^h_e(d)$ as a capacity that has to be reserved on link e to be used by commodity h, we prove that these capacities are sufficient to carry the demand vector d (one has then to solve a simple flow problem, one for each commodity h, to effectively route it on the these reserved capacities). Moreover, we provide a simple example where the optimal solution of $\mathcal{F}_+$ is strictly better than the optimal solution of $\mathcal{F}_=$. Our numerical experiments also show that this can lead to some improvement on realistic instances.

Another way to relax the flow conservation constraints consists in replacing them by the following inequalities:

$$\sum_{e\in\delta_+(v)} f_{h,e}(d) - \sum_{e\in\delta_-(v)} f_{h,e}(d) \begin{cases} \leq -d_h, & \text{if } v = t(h) \\ \leq 0 & \text{if } v \neq s(h), t(h) \end{cases} \tag{2}$$

The obtained formulation can then be called the $\mathcal{F}_-$ formulation. The solution of $\mathcal{F}_-$ can also be shown to be a valid solution for dynamic routing. There is also a simple example where the optimal solution of $\mathcal{F}_-$ is strictly better than the optimal solution of $\mathcal{F}_=$. $\mathcal{F}_-$. While both $\mathcal{F}_-$ and $\mathcal{F}_+$ dominate $\mathcal{F}_=$, they are not comparable (for some instances $\mathcal{F}_-$ provides better results than $\mathcal{F}_+$ and vice-versa).

Extended graph formulation

We have seen that formulations $\mathcal{F}_-$ and $\mathcal{F}_+$ can be strictly better than $\mathcal{F}_=$ (i.e., closer to $\mathcal{F}_{dyn}$, the formulation related to dynamic routing in its full generality). The difference between $\mathcal{F}_-$ and $\mathcal{F}_+$ lies in the sign of the terms $\sum_{e\in\delta_+(v)} f_{h,e}(d) - \sum_{e\in\delta_-(v)} f_{h,e}(d)$ for $v \in V \setminus \{s(h), t(h)\}$ required to be negative for $\mathcal{F}_-$ and positive for $\mathcal{F}_+$. We propose here a stronger formulation that is still easy to solve, where the features of $\mathcal{F}_-$ and $\mathcal{F}_+$ are combined in some way. For each commodity $h \in \mathcal{H}$, and for each node $v \in V \setminus \{s(h), t(h)\}$, we add to G the two directed edges $(t(h), v)$ and $(v, s(h))$. We also add an edge directed from $t(h)$ to $s(h)$. For each commodity h, an $s(h)t(h)$ flow f_h is considered in the extended graph. Notice that the extra edges we added $(t(h), v)$, $(v, s(h))$ and $(t(h), s(h))$ can only be used by commodity h. Flow conservation constraints can be expressed as follows.

$$f_{h,(v,s(h))}(d) + \sum_{e\in\delta_+(v)} f_{h,e}(d) - f_{h,(t(h),v)}(d) - \sum_{e\in\delta_-(v)} f_{h,e}(d) = 0 \text{ if } v \neq s(h), t(h) \tag{3}$$

$$\sum_{e\in\delta_+(s(h))} f_{h,e}(d) - \sum_{v\in V\setminus\{s(h)\}} f_{h,(v,s(h))}(d) = d_h. \tag{4}$$

Notice that $\delta_+(v)$ and $\delta_-(v)$ contain only edges belonging to G. Observe that there are no explicit capacity limitations for the edges not belonging to E (the added edges of type $(v, s(h))$ and $(t(h), v)$). However, positivity is required for the flow on these edges. It is easy to see that $\mathcal{F}_+$ (resp. $\mathcal{F}_-$) is a special case of $\overline{\mathcal{F}}$ since the term $\sum_{e\in\delta_+(v)} f_{h,e}(d) - \sum_{e\in\delta_-(v)} f_{h,e}(d)$ (resp. $\sum_{e\in\delta_-(v)} f_{h,e}(d) - \sum_{e\in\delta_+(v)} f_{h,e}(d)$) is positive in $\mathcal{F}_+$ (resp. $\mathcal{F}_-$) and can be seen as the flow going through an additional edge $(t(h), v)$ (resp. $(v, s(h))$). In other words, by considering only edges of type $(t(h), v)$ (resp. $(v, s(h))$) and solving $\overline{\mathcal{F}}$ we get $\mathcal{F}_+$ (resp. $\mathcal{F}_-$) . In [2] we provide a simple example where the optimal solution of $\overline{\mathcal{F}}$ is strictly better than the optimal solution of $\mathcal{F}_+$ and of $\mathcal{F}_-$. Our numerical experiments also show that this can lead to an improvement on realistic instances. Other improvements based on the combination of aggregation (either by source or by destination) with the affine approach are also proposed in [2] and will be presented in our talk.

References

[1] Y. Al-Najjar, W. Ben-Ameur, and J. Leguay. On the approximability of robust network design. *Theoretical Computer Science*, 860:41–50, 2021.

[2] Y. Al-Najjar, W. Ben-Ameur, J. Leguay, and J. Elias. Affine routing for robust network design. *Networks*, (published on-line), 2021.

[3] Walid Ben-Ameur. Between fully dynamic routing and robust stable routing. In *6th International Workshop on Design and Reliable Communication Networks*, pages 1–6. IEEE, 2007.

[4] A. Ouorou and J.P. Vial. A model for robust capacity planning for telecommunications networks under demand uncertainty. In *Workshop on Design and Reliable Communication Networks*, 2007.

[5] M. Poss and C. Raack. Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks*, 61(2):180–198, 2013.

[6] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 255–264, 2008.

On the Approximability of Robust Network Design in Undirected Graphs

Yacine Al-Najjar[1,2], Walid Ben-Ameur[1], and Jérémie Leguay[2]

[1]Samovar, Telecom SudParis, Institut Polytechnique de Paris, France., ✉ walid.benameur@telecom-sudparis.eu
[2]Huawei Technologies, Paris Research Center, France., ✉ yacine.alnajjar@huawei.com✉ jeremie.leguay@huawei.com

Introduction

In this talk we will present results published in [1] on the approximability of the robust network design problem in undirected graphs. Given the dynamic nature of traffic in telecommunication networks, we investigate the variant of the robust network design problem where we have to determine the capacity to reserve on each link so that each demand vector belonging to a polyhedral set can be routed. The objective is either to minimize congestion or a linear cost. Routing is assumed to be fractional and dynamic (i.e., dependent on the current traffic vector).

The robust network design problem where a linear reservation cost is minimized was proved to be co-NP hard in [10] when the graph is directed. A stronger co-NP hardness result is given in [7] where the graph is undirected (this implies the directed case result). Some exact solution methods for robust network design have been considered in [8, 12]. Notice that since solving this dynamic routing problem is generally difficult (as shown in this paper), and dynamic routing is difficult to implement in practice, other routing strategies have been proposed in literature such as static/oblivious routing [3, 5] and less conservative variants such as those of [2, 4, 13, 14]). However, we will only focus on dynamic routing.

The robust network design problem

Consider an undirected graph $G = (V, E)$, a set of commodities $\mathcal{H}$ (each commodity $h \in \mathcal{H}$ has a source $s(h)$ and a destination $t(h)$), a cost vector $\lambda \in \mathbb{R}^E$ representing the cost of reserving one unite of capacity on each edge $e \in E$ of the network and a polytope $\mathcal{D}$ describing all possible demand vectors $d \in \mathcal{D}$. We consider here the case where $\mathcal{D}$ is given by a set of inequalities. More precisely, $\mathcal{D} = \{d \in \mathbb{R}^{\mathcal{H}}_+ | Ad \leq b\}$ where A is a matrix and b is a vector.

We aim to chose the capacities u_e to reserve on each edge $e \in E$ such that, for all demand vectors $d \in \mathcal{D}$, there exists a multicommodity flow that serves all demands and does not exceed capacities u_e.

We denote by $\mathcal{U}(\mathcal{D})$ the set of capacities vectors $u \in \mathbb{R}^E_+$ that are big enough to serve all demands in $\mathcal{D}$. The "linear cost" problem consists in finding a capacity vector $u \in \mathcal{U}(\mathcal{D})$ minimizing the linear cost $\min_{u \in \mathcal{U}(\mathcal{D})} \sum_{e \in E} \lambda_e u_e$. Given capacities c_e on each edge e, the "congestion" problem consists in minimizing the congestion $\min_{u \in \mathcal{U}(\mathcal{D})} \max_{e \in E} \frac{u_e}{c_e}$. It is worth to insist again on the fact that the routing for the commodities is dynamic (i.e. it can depend on each demand vector $d \in \mathcal{D}$)

Our contributions

- We first prove that the robust network design problem with minimum congestion cannot be approximated within any constant factor. The reduction is based on the PCP theorem and some connections with the Gap-3-SAT problem. The same reduction also allows to show inapproximability within $\Omega(\log \frac{n}{\Delta})$ where Δ is the maximum degree in the graph and n is the number of vertices.
- Using the ETH conjecture [11], we prove a $\Omega(\frac{\log n}{\log \log n})$ lower bound for the approximability of the robust network design problem with minimum congestion. This implies that the well-known $O(\log n)$ approximation ratio that can be obtained using the result in [15] is tight.

- We show that any α-approximation algorithm for the robust network design problem with linear costs directly leads to an α-approximation for the problem with minimum congestion. The proof is based on Lagrange relaxation. We obtain that robust network design with minimum congestion can be approximated within $O(\log n)$. This was already proved in [15] in a different way.
- An important consequence of the Lagrange-based reduction and our inapproximability results is that the robust network design problem with linear reservation cost cannot be approximated within any constant ratio. This answers a long-standing open question stated in [6].
- Another consequence is a new proof for the existence of instances for which the optimal static solution can be $\Omega(\log n)$ more expensive than a solution based on dynamic routing, when a linear cost is minimized. This was already proved in [9] in a different way.
- We show that even if only two given paths are allowed for each commodity, there is a constant k such that the robust network design problem with minimum congestion or linear costs cannot be approximated within k.

References

[1] Y. Al-Najjar, W. Ben-Ameur, and J. Leguay. On the approximability of robust network design. *Theoretical Computer Science*, 860:41–50, 2021.

[2] Y. Al-Najjar, W. Ben-Ameur, J. Leguay, and J. Elias. Affine routing for robust network design. *Networks*, (published on-line), 2021.

[3] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. *Journal of Computer and System Sciences*, 69(3):383–394, 2004.

[4] W Ben-Ameur. Between fully dynamic routing and robust stable routing. In *6th International Workshop on Design and Reliable Communication Networks*, pages 1–6. IEEE, 2007.

[5] W. Ben-Ameur and H. Kerivin. New economical virtual private networks. *Communications of the ACM*, 46(6):69–69, 2003.

[6] C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–129, 2007.

[7] C. Chekuri, F.B. Shepherd, G. Oriolo, and M-G. Scutella. Hardness of robust network design. *Networks*, 50(1):50–54, 2007.

[8] G. Claßen, A. M. C. A. Koster, M. Kutschka, and I. Tahiri. Robust metric inequalities for network loading under demand uncertainty. *Asia-Pacific Journal of Operational Research*, 32(5), 2015.

[9] N. Goyal, N. Olver, and F. Shepherd. Dynamic vs. oblivious routing in network design. In *European Symposium on Algorithms*, pages 277–288. Springer, 2009.

[10] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 389–398, 2001.

[11] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[12] S. Mattia. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54(3):619–643, 2013.

[13] A. Ouorou and J.P. Vial. A model for robust capacity planning for telecommunications networks under demand uncertainty. In *Workshop on Design and Reliable Communication Networks*, 2007.

[14] M. Poss and C. Raack. Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks*, 61(2):180–198, 2013.

[15] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 255–264, 2008.

Tractable Policies for Multi-Stage Adjustable Robust Optimization

Simon Thomä[1], Grit Walther[1], and Maximilian Schiffer[2]

[1]Chair of Operations Management, RWTH Aachen, 52072 Aachen, Germany
✉ simon.thomae@om.rwth-aachen.de, ✉ walther@om.rwth-aachen.de
[2]TUM School of Management & Munich Data Science Institute, Technical University of Munich, 80333 Munich, Germany
✉ schiffer@tum.de

We study piecewise affine policies for multi-stage adjustable robust optimization (ARO) problems with non-negative right-hand side uncertainty. In multi-stage robust optimization, uncertainty vectors $\boldsymbol{\xi}$ are drawn from an uncertainty set $\mathcal{U}$ and revealed over the course of a time horizon with T discrete stages. In each stage i, a fraction $\boldsymbol{x}^i$ of the decision vector $\boldsymbol{x}$ is fixed after the realization of uncertainties $\boldsymbol{\xi}^i$. Herein, $\boldsymbol{x}^i$ may only depend on uncertainties that were revealed up to stage i. These constraints of only using past information are called *nonanticipativity* constraints and are key for multi-stage optimization. Specifically, we provide new policies for problems that are formally given by

$$\begin{aligned} Z_{AR}(\mathcal{U}) = \min_{\boldsymbol{x}(\boldsymbol{\xi})} \max_{\boldsymbol{\xi}\in\mathcal{U}} \quad & \boldsymbol{c}^\intercal \boldsymbol{x}(\boldsymbol{\xi}) \\ & \boldsymbol{A}\boldsymbol{x}(\boldsymbol{\xi}) \geq \boldsymbol{D}\boldsymbol{\xi} + \boldsymbol{d} \quad \forall \boldsymbol{\xi} \in \mathcal{U} \end{aligned} \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{l\times n}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{D} \in \mathbb{R}^{l\times m}_+$, $\boldsymbol{d} \in \mathbb{R}^l_+$ and $\mathcal{U} \subset \mathbb{R}^m_+$.

For such problems, finding solutions for $\boldsymbol{x}$ is hard in general. Accordingly, many existing approaches focus on restricting $\boldsymbol{x}$ to more tractable function spaces. One class of well known restrictions are affine policies proposed by Ben-Tal et al. [1], which remain optimal for some special variants of Problem (1) [5, 7, 8]. To the best of our knowledge, no results on approximation guarantees exist for the general multi-stage adjustable problem of (1), although a lot of recent work has focused on its two-stage variant, providing $O(\sqrt{m})$ policies for general uncertainty sets $\mathcal{U}$ and $\boldsymbol{c}, \boldsymbol{x}$ non-negative [2, 4]. Moreover, improved bounds in the two-stage problem exist for many common uncertainty sets, including hypersphere uncertainty, ellipsoid uncertainty, p-ball uncertainty and intersections thereof [2, 3], as well as budgeted and generalized budgeted uncertainty [2, 3, 6].

With this work, we generalize some of these ideas and extend the existing literature in multiple ways.

- To the best of our knowledge, we present the first policies for the multi-stage robust optimization problem (1) that yield optimality bounds of $O(\sqrt{m})$ on general instances. Moreover, we show even tighter bounds for many commonly used uncertainty sets, see Table 1.
- In contrast to existing policies for the two-stage problem, our policies do not rely on the assumption of $\boldsymbol{c}, \boldsymbol{x}$ being non-negative, which allows modelling many additional instance classes.
- We show that we can find solutions for our policies with a linear program (LP) that can efficiently be solved using state of the art solvers. Furthermore, we present comprehensive numerical experiments that show significant improvements in computational times as well as improvements of the objective value for many instances.

In Table 2, we show results for instances that are modified versions of the tests in [2] where decisions and uncertainties were assigned to $\sqrt{m}$ stages. While our policies perform about 6% worse than affine policies for budgeted uncertainty sets with a budget of $\sqrt{m}$, they perform better by almost a factor of two for hypersphere uncertainty sets. Additionally, we find improvements in the solution time by up to a factor of 20 for both uncertainty types.

Concluding, our policies not only extend the theoretical bounds for multi-stage adjustable robust optimization problems; they can also be found by orders of magnitude faster than affine adjustable policies on many instances while achieving similar performances.

No.	Uncertainty Set $\mathcal{U}$	Bound	Asymptotic Bound
1	$\left\{\boldsymbol{\xi} \in \mathbb{R}^m_+ \middle\| \|\boldsymbol{\xi}\|_2^2 \leq 1\right\}$	$\sqrt[4]{m}\frac{\sqrt{m-1}}{\sqrt{2\left(m-\sqrt{m}\right)}}$	$O(\sqrt[4]{m})$
2	$\left\{\boldsymbol{\xi} \in [0,1]^m \middle\| \|\boldsymbol{\xi}\|_1 \leq k\right\}$	$\frac{k(m-1)}{m+k(k-2)}$	$O\left(\min\left\{k, \frac{m}{k}\right\}\right)$
3	$\{\boldsymbol{\xi} \in \mathbb{R}^m \mid \boldsymbol{\xi}^\intercal \boldsymbol{\Sigma} \boldsymbol{\xi} \leq 1\}$	$\left(\frac{a}{2} + \frac{\sqrt{1-a}}{\sqrt[4]{am^2+(1-a)m}}\right)^{-1}$	$O\left(m^{\frac{2}{5}}\right)$
4	$\left\{\boldsymbol{\xi} \in \mathbb{R}^m_+ \middle\| \|\boldsymbol{\xi}\|_p \leq 1\right\}$	$\frac{2}{p}(p-1)^{\frac{p-1}{p}} m^{\frac{p-1}{p^2}}$	$O\left(m^{\frac{p-1}{p^2}}\right)$
5	$\left\{\boldsymbol{\xi} \in \mathbb{R}^m_+ \middle\| \|\boldsymbol{\xi}\|_p \leq 1, \|\boldsymbol{\xi}\|_q \leq r\right\}$	$\min\left\{r^{\frac{1-p}{p}} m^{\frac{p-1}{pq}}, r^{\frac{1}{q}} m^{\frac{q-1}{q^2}}\right\}$	$O\left(\min\left\{r^{\frac{1-p}{p}} m^{\frac{p-1}{pq}}, r^{\frac{1}{q}} m^{\frac{q-1}{q^2}}\right\}\right)$
6	general $\mathcal{U} \subset \mathbb{R}^m$	$2\sqrt{m}+1$	$O\left(\sqrt{m}\right)$

Table 1: Performance bounds of the piecewise affine policy for different uncertainty sets.
We prove specific bounds for uncertainty sets of the forms 1. hypersphere uncertainty; 2. budgeted uncertainty; 3. ellipsoid uncertainty ,with $\boldsymbol{\Sigma} := (1-a)\mathbf{1} + a\boldsymbol{J}$ where $\mathbf{1}$ is the unity matrix and $\boldsymbol{J}$ the matrix of all ones; 4. p-norm ball, with $p \geq 1$; 5. intersection of two norm balls, with $m^{\frac{1}{q}-\frac{1}{p}} \geq r \geq 1$.

m	Avg	$T_{our}(s)$	$T_{aff}(s)$
10	0.85	0.00	0.01
20	0.76	0.01	0.15
30	0.70	0.05	1.01
40	0.66	0.21	3.41
50	0.63	0.72	11.13
60	0.61	1.74	30.93
70	0.59	4.12	62.88
80	0.58	8.97	134.42
90	0.56	16.27	251.98
100	0.55	29.45	482.67

(a) hypersphere uncertainty

m	Avg	$T_{our}(s)$	$T_{aff}(s)$
10	1.06	0.01	0.02
20	1.06	0.04	0.10
30	1.06	0.16	0.64
40	1.06	0.64	3.08
50	1.06	2.27	12.42
60	1.06	6.58	32.44
70	1.06	13.81	119.18
80	1.06	28.81	364.65
90	1.06	59.15	950.59
100	1.06	102.27	1903.73

(b) budgeted uncertainty

Table 2: Average relative objective value $\frac{Z_{our}}{Z_{aff}}$ of our policies compared to the affine adjustable policies introduced in [1] and computation times on modified test instances from [2].

References

[1] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, mar 2004.

[2] Aharon Ben-Tal, Omar El Housni, and Vineet Goyal. A tractable approach for designing piecewise affine policies in two-stage adjustable robust optimization. *Mathematical Programming*, 182(1-2):57–102, 2020.

[3] Dimitris Bertsimas and Hoda Bidkhori. On the performance of affine policies for two-stage adaptive optimization: a geometric perspective. *Mathematical Programming*, 153(2):577–594, 2015.

[4] Dimitris Bertsimas and Vineet Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical Programming*, 134(2):491–531, 2012.

[5] Dimitris Bertsimas, Dan A. Iancu, and Pablo A. Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, may 2010.

[6] Omar El Housni and Vineet Goyal. On the optimality of affine policies for budgeted uncertainty sets. *Mathematics of Operations Research*, 46(2):674–711, may 2021.

[7] Dan A. Iancu, Mayank Sharma, and Maxim Sviridenko. Supermodularity and affine policies in dynamic robust optimization. *Operations Research*, 61(4):941–956, aug 2013.

[8] David Simchi-Levi, Nikolaos Trichakis, and Peter Yun Zhang. Designing response supply chain against bioattacks. *Operations Research*, 67(5):1246–1268, sep 2019.

Session Session 3B: energy

Thursday 9 June 2022, 10:30-12:10

Lecture Hall III

Loads scheduling for energy community Demand Response on Smart Grids

Mariam Sangaré[1], Michael Poss[2], Eric Bourreau[3], and Amaury Pachurka[4]

[1]LIRMM, University of Montpellier, Montpellier, France, ✉ msangare@lirmm.fr
[2]LIRMM, University of Montpellier, Montpellier, France, ✉ michael.poss@lirmm.fr
[3]LIRMM, University of Montpellier, Montpellier, France, ✉ eric.bourreau@lirmm.fr
[4]BEOGA, Montpellier, France, ✉ amaury.pachurka@beoga.fr

1 Introduction

The energy transition is a means to meet the objectives set by the Paris Agreement [3]. Faced with the challenges of the energy transition, many innovative companies are emerging and propose continuous technological progress. This paper focuses on optimizing collective self-consumption in an energy community composed of various members: households and public premises (stadium). Each member can produce, consume, exchange, sell (within the community or on the grid) and/or store photovoltaic energy. The considered community remains connected to the public grid and uses it. This study aims finally to provide optimal planning of the controllable loads for each member in the planning horizon. Moreover, it gives a plan of energy exchanges and a management scheme for electricity storage units installed in the community.

2 Loads scheduling

We consider a collective self-consumption community composed of n members. Each member can have production and storage equipment. In addition, each member performs a set of tasks with electrical devices. These tasks are detailed as follows. The objective is to schedule the tasks to minimize the use of the grid. According to [1], controllable loads are those with flexible and programmable operation. The controllable loads are grouped into two categories: type A and B loads, which reorganization over time allows increasing the community's energy efficiency.

- **Type A loads**: these are related to tasks whose execution allows to regulate the temperature of certain environments to ensure human comfort. As an example of such tasks, we have heating and water heaters. In practice, an individual regulates the temperature of a room to reach a comfort zone temperature and then maintains this zone until a certain time of the day.
- **Type B loads**: these relate to tasks that must be done within some given time windows and with fixed periodic levels of electrical consumption. They are generated by the use of appliances such as washing machines, dryers, and electric vehicles. For each corresponding task, the user will indicate the different time windows where she could execute the task and indicate the periodic consumption levels in each time window.
- **Type C loads** these are loads that must be executed without delay after the request and necessarily with the required levels of electrical consumption. As type C-tasks, we can mention lighting, cooking, television. For each member, we estimate the periodic accumulation of the consumption of those tasks in the planning horizon.

The proposed solution consists of determining the starting period and the periodic consumption levels to be used for each type A task and also choosing the best schedules to operate the type B tasks while respecting the community's operating constraints.

3 Resolution method

In a first step, we developed a mixed-integer linear programming (MILP) model inspired by the model proposed in [2]. However, the MILP is not efficient for large problem instances. We then developed a heuristic based on a column generation algorithm, convexifying the feasible set corresponding to the schedules of tasks of type A. That heuristic allows us to generate only the schedules likely to improve the restricted master problem (RMP) instead of explicitly generating them all. We begin this resolution approach by generating some feasible schedules to get the dual values by solving the RMP. Then, we solve a pricing problem at each iteration to get the potentially improving plans (with a negative reduced cost). Next, we add these columns to the RPM and solve it. We repeat this process until no improving solution is returned or until the maximum iteration is reached. Finally, we solve the RMP with integrality constraints to get the result of the heuristic

4 Results

The used instances are built with data from the BEOGA's Smart Lou Quila demonstrator located in Cailar in the Guard. Smart Lou Quila is composed of seven members. The other instances are built by duplicating those members. The planning horizon is consists of a day sliced into 48 periods. The MILP's solving time is `time_limit`=5600s, the column generation's pricing problem has a maximum time `time_limit`=200s, and the maximum number of GC iterations is `maxIter`=10 for each instance. Finally, the RMP with integrality constraints has a `time_limit`=3600s. The results are reported in Table 1 where $|N|$ denotes the number of members in the community. *obj* and obj_{GC} represent respectively the sum of the electricity extracted from the main grid during the planning horizon returned by the MILP and the column generation. Output " *** " means that no feasible solution has been found after the `time_limit`. We notice that the MILP approach is more efficient for the small instances while the column generation is more efficient for the large ones.

	MILP's solutions			Column generation solutions	
$\|N\|$	*obj*	*Gap*	CPU	obj_{GC}	CPU
7	111.38kWh	0.08%	5605.4	112.00kWh	110.44
28	481.68kWh	0.2%	5601.66	482.59kWh	1311.08
56	974.89kWh	0.4%	5601.78	976.87kWh	1035.71
112	***	***	***	1964.57kWh	1882.25
224	***	***	***	3904.91kWh	1162.68

Table 1: Comparison between the solutions of the two approaches.

References

[1] Raffaele Carli, Mariagrazia Dotoli, Jan Jantzen, Michael Kristensen, and Sarah Ben Othman. Energy scheduling of a smart microgrid with shared photovoltaic panels and storage: The case of the ballen marina in samsø. *Energy*, 198, 2020.

[2] Anjos F. Miguel, Luce Brotcorne, Martine Labbé, and maria restrepo ruiz. Load Scheduling for Residential Demand Response on Smart Grids. In *VAME 2017 - Variational Analysis and Applications for Modelling of Energy Exchange*, Perpignan, France, May 2017.

[3] United Nations. The paris agreement.

Supporting Energy Communities - Operational Research and Energy Analytics

Paula Carroll[1,3], Cristian Aguayo[5,6], Sandeep Araveti[1,3], Luce Brotcorne[6], Imre Drovtar[4], Bernard Fortz[5,6], Evita Kairiša[4], Tarmo Korõtko[4], Anna Mutule[4], and Conor Sweeney[2,3]

[1]School of Business, University College Dublin, Ireland, ✉ paula.carroll@ucd.ie
[2]School of Mathematics and Statistics, University College Dublin, Ireland, ✉ conor.sweeney@ucd.ie
[3]Energy Institute, University College Dublin, Ireland
[4]SmaA, Institute of Physical Energetics, Riga, Latvia
[5]Computer Science Department, Université libre de Bruxelles, Belgium
[6]INRIA, France

We present an overview of the Supporting Energy Communities - Operational Research and Energy Analytics (SEC-OREA) project which enables local energy communities (LECs) to participate in the decarbonisation of the energy sector by developing advanced efficient algorithms and analytics technologies.

LECs require accurate information on renewable energy generation to develop their full potential at both planning and operational stages. SEC-OREA is an open data driven project. As part of the project, we plan to go through the path of creating an energy community modelling framework, simulating different energy community load profiles in four countries, considering each country's low voltage network restrictions and average annual household consumption. At the same time, the emphasis is placed on the accuracy of the data and the assessment of real world conditions. Generated load profiles will be validated, comparing the communities' load profiles with their county's typical load curve and average households' electricity consumption.

Climate reanalysis combine past observations with models to generate consistent time series of multiple climate variables. Reanalysis datasets are widely used for simulating renewable energy availability such as assessing the potential of solar photovoltaic, and wind power generation. There is growing demand from industry, research and other sectors for high quality and long-term gridded climate datasets with high temporal and spatial resolution.

We explore available climate services to gather energy-relevant pan-European indicators of climate trends and variability. We assess these open data sources against meteorological observations to determine the best suited climate data sources for particular use cases such as the location where an LEC may decide to locate. The climate data is used to estimate renewable energy scenarios at high time and geographic resolution by converting weather variables such as wind speed to estimates of wind power for different wind turbine technologies. We use better data to model renewable energy generation, to understand and create dynamic scenarios of electricity consumption. We create better ensemble models of climate dependent electricity generation from renewable energy sources, and consumer electricity demand.

We create a set of mathematical optimization models to efficiently solve the multilateral economic dispatch decisions of the LEC Renewable Energy Sources in a fair manner. More precisely we explicitly model the interactions among the stakeholders by relying on Nash and Stackelberg equilibriums. These problems result in bilevel optimisation problems involving possibly muti-leader or multi follower.

In particular, we extend the classical Unit Commitment (UC) problem to take into account the specific characteristics of LECs. We develop methods to solve realistic size instances of these UC deterministic optimisation problems, and extend these methods to address uncertainty given by the different electricity consumption and generation scenarios.

For the evaluation of mathematical optimisation models, an LEC power system model is used. The LEC power system is modelled using a universal prosumer model as shown in Fig 1. This enables us to simulate power flows in various grid topologies using a single modelling object. The prosumer modelling object enables to incorporate customised control blocks, to simulate individual asset controllers but also to execute internal optimisation and aggregation logic of the LEC, and provides an interface for integration with external utilities, e.g higher-level mathematical optimisation models or data sources for forecasts.

This approach provides a simple and flexible way to model power flow in LEC power systems, thus reducing the required resources for object modelling and control system integration.

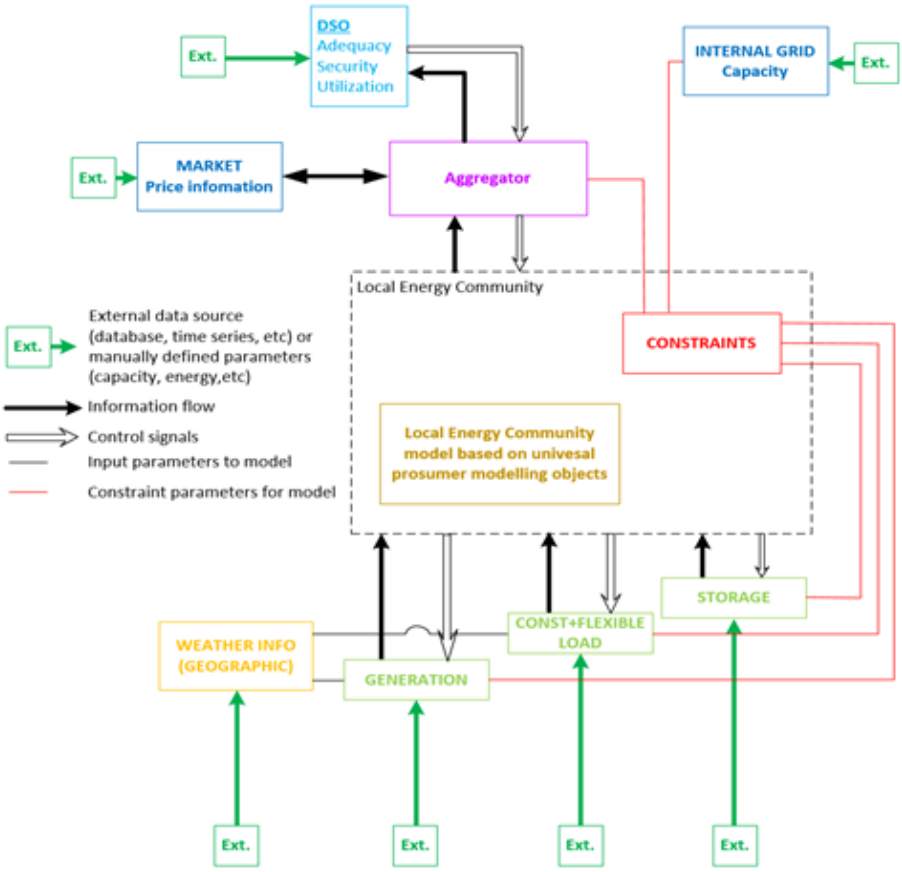


Figure 1: LEC Modelling Framework

We evaluate the implications of the LEC activity and net demand on sample grid topologies. The model provides support to the Distribution System Operator (DSO) in understanding the impacts of, and requirements for LECs on the low voltage distribution network. Different optimisation models are investigated on their performance to meet objectives of different LEC parties but also provide benefits for the DSO. This understanding supports better LEC and DSO decisions on asset reinforcement, network power flow and congestion management.

We address challenges in operation and planning of the grid, studying various climate, energy and dispatch optimisation scenarios useful in understanding of LEC impact on distribution systems. The scenarios will cover operation of different technologies (either individually or in combination). Concerns will be addressed for operation, short-term planning and investment planning from the perspective of different stakeholders. Thus, solutions will address adequacy, security, utilisation and investment decisions, developing solutions to enhance economy and sustainability of grid development.

We provide recommendations for an overarching LEC enabling framework to ensure safe reliable efficient sustainable operation of the LEC and low voltage network. Our framework will allow LEC members to take ownership of the energy transition, benefit from the new technologies we develop and so reduce their bills and their carbon footprint. We provide business model analyses, efficient scalable multilateral economic dispatch and energy analytics algorithms, and integrated climate/LEC/Low Voltage models to support our climatology, meteorological services, smart city, municipality and energy agencies stakeholder decision makers.

Acknowledgements: This work emanates from research supported by the ERA-NET Cofund grant under the CHIST-ERA IV Joint Call on Novel Computational Approaches for Environmental Sustainability (CES), project “Supporting Energy Communities - Operational Research and Energy Analytics” (SEC-OREA). Sandeep Araveti is funded by the Irish Research Council. The work of Cristian Aguayo and Bernard Fortz is supported by the Fonds de la Recherche Scientifique - FNRS under Grant R801020F.

Exploiting fundamental network flow theory to solve a bilevel energy market problem

Stephan Marnach[1] and Arie M.C.A. Koster[2]

[1]Department of Mathematics, RWTH Aachen University, Aachen, Germany, ✉ marnach@math2.rwth-aachen.de
[2]Department of Mathematics, RWTH Aachen University, Aachen, Germany, ✉ koster@math2.rwth-aachen.de

In this talk we study a simple model for *demand side management* in the energy market. A distribution network operator (DNO) is tasked with covering the energy demand of consumers in his sub-network. To that end the DNO can buy energy from the power market or provide energy from renewable sources under his control. However, renewable energy is lost when not used. The DNO's problem is now to dynamically adjust the energy prices in each period to stimulate a shift in energy consumption in his favor, e.g. shifting consumption to times where there is surplus of renewable energy. Each consumer, on the other hand, tries to minimize his energy bill. To be able to adjust to non-uniform prices, consumers do not have fixed demand but rather choose a cummulative-demand curve from a set of curves bounded by a lower and an upper stair function.

The idea behind this approach is to improve the incorporation of renewable energy sources in the energy market via economic incentives for both producers and consumers. The model is to be used as a rough proof on concept, constrained by our current ability to solve multilevel mathematical problems, rather than a realistic real world model.

The mathematical problem emerging from the model is a bilevel problem with bilinear objective functions and linear constraints. The DNO constitutes the upper level and the consumers the lower level. The bilinear part of the respective objectives consists of a product of energy price and realized consumption. When fixing the energy prices, the lower level problems turn into linear problems. Thus they fulfill Slater's constraint qualification and thereby admit two of the most common reformulation techniques, a *KKT-reformulation* and a *strong-duality reformulation*. In both cases the dual problem of each lower level problem is added to the formulation. The *KKT-reformulation* then ensures optimality for the lower level problems by enforcing complementary slackness of the pairs of primal and dual variables via additional constraints. The *strong-duality reformulation* achieves the same with an equation forcing the primal-dual pairs to assume solutions with zero duality gap [1].

Applying the *KKT-reformulation* to our model yields a single level mathematical problem with complementarity constraints (MPCC) featuring a quadratic objective function whereas the *strong-duality reformulation* results in a single level quadratically constrained quadratic problem (QCQP). Both non-linear, non-convex and thus very challenging to solve.

We propose to build on the *strong-duality reformulation* to arrive at a mixed integer (MIP) formulation of the original problem. This reformulation is based on an interpretation of the lower level problem as a *minimum cost flow* problem. The number of non-continuous variables of the MIP formulation only depends on T, the number of time-periods considered.

First we show that one can interpret the lower level problems as a simple *minimum-cost-flow* problem that has at most two priced arcs on any given elementary cycle. Applying the well known *negative cycle optimality condition* allows us to reverse optimize the prices set by the upper level. Given a price vector and a corresponding optimal flow, we can raise the prices, yet preserve optimality of the flow as long as we do not create negative cycles in the residual network. By virtue of the simple structure of the underlying graph we see that we can avoid negative cycles if we do not disturb the natural ordering of the given prices. We can use this insight to determine that in any optimal solution of the bilevel problem the entries of the price vector can take at most T different values. Those values can be calculated in linear time.

Using the now discrete prices, we can linearize all quadratic terms in the *strong-duality reformulation* via *big*-$\mathbb{M}$ constraints and binary variables, without loss of optimality. The resulting problem is a MIP with at most $\frac{T(T+1)}{2}$ binary variables.

To strengthen the MIP formulation we introduce valid inequalities that cut off combinations of the binary variables values that are impossible to occur in optimal solutions.

Computational experiments show that all formulations, the QCQP, the MPCC, the MIP and the strengthened MIP are decent at finding strong feasible solutions. However, the MIP formulation, especially the strengthened one, beats the QCQP and MPCC formulations by orders of magnitude when it comes to closing the duality gap and proving optimality.

Acknoledgement

This research is supported by Bundesministerium für Wirtschaft und Energie (German Federal Ministry of Economics and Energy (BMWi)) as part of the project Flexibilitätswende (funding reference: 03E11015A).

References

[1] S. Dempe and A. B. Zemkoho. The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. *Mathematical Programming*, 138(1):447–473, Apr 2013.

A bilevel formulation for a reliable stochastic network expansion problem facing unavoidable unmet demands

Xavier Blanchot[1,2], François Clautiaux[1], Aurélien Froger[1], and Manuel Ruiz[2]

[1]Université de Bordeaux, UMR CNRS 5251, Inria Bordeaux Sud-Ouest, Talence,France, ✉ xavier.blanchot@u-bordeaux.fr, francois.clautiaux@math.u-bordeaux.fr, aurelien.froger@math.u-bordeaux.fr
[2]RTE, Paris La Défense, France, ✉ xavier.blanchot@rte-france.com, manuel.ruiz@rte-france.com

1 Industrial context

We propose in the current work a methodology to take into account a reliability constraint in some network expansion problems used in long term adequacy studies done by RTE, the french transmission system operator. Long term adequacy studies aim at analyzing the risk of imbalance on the electricity network between production capacity and demand to horizon from 15 to 30 years. The studies also propose some possible investment on production or transmission capacities on the network to face this risk of imbalance.

This can be modeled by a two-stage stochastic network expansion problem. Let $\mathcal{S}$ be a set of scenarios with probabilities $(p_s)_{s\in\mathcal{S}}$, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph modeling the network and $\mathcal{T}$ be a set of time steps. We denote by x the first-stage variables which model investment decisions on the network, and $\phi(x, s)$ the cost of the operational problem, for a given scenario $s \in \mathcal{S}$ and first-stage decision x. The stochastic network expansion problem can be modeled as follows:

$$\begin{cases} min\ c^T x + \sum_{s\in\mathcal{S}} p_s \phi(x, s) \\ s.t.\ Ax = b \\ \quad x \in \mathbb{R}^{n_{1,C}} \times \mathbb{N}^{n_{1,I}} \end{cases}$$

The operational problem models the optimal balance between production and demand on the network:

$$\phi(x, s) = \begin{cases} min\ g_s^T y_s + Me^T u_s \\ s.t.\ W_s y_s + Q_s u_s = d_s - T_s x \\ \quad y_s \in \mathbb{R}^{n_2}, u_s \in \mathbb{R}^{Card(\mathcal{N})\times Card(\mathcal{T})} \end{cases}$$

where variable $u_s \in \mathbb{R}^{Card(\mathcal{N})\times Card(\mathcal{T})}$ counts, for each node of the graph and each time step, the quantity of unmet demand, y_s represents all the other second-stage variables (e.g. production, flow). M is the fixed unit cost of an unmet demand, and e is a vector of $1's$.

2 Reliability constraint and bilevel formulation of the reliable stochastic network expansion problem

The cost associated to an unmet demand in the used formulations is fixed by legislation to 20.000€\ MWh. This cost might be not large enough to produce solutions in which the demand at each node and each time step in every scenario is satisfied. To enforce the reliability of the network, the legislation requires not to have more than a given constant α of unmet demands across all nodes and all time steps on the network in expectation over the scenarios.

This constraint couples all the scenarios. However, the operational problem models an optimal operational solution for a given scenario, and its solution is independent of other scenarios. Then, its solution

can not be chosen because of the relationship between scenarios caused by the reliability constraint. Said differently, the solution of the balance between production and demand in a particular scenario can not be suboptimal because of an other scenario. This leads to a mixed-integer bilevel formulation, where the reliability constraint appears in the upper level, and have to be satisfied only by optimal solutions of the lower level problems.

We denote by $\delta \in \{0,1\}^{Card(\mathcal{S})\times Card(\mathcal{N})\times Card(\mathcal{T})}$ the vector of binary variables which represent whether a demand is unmet or not at each node and each time step in every scenario. Let α be the maximum number of unmet demands imposed by legislation, R be a large enough constant, and ϵ a positive threshold above which a demand is said to be unmet. The reliable stochastic network expansion problem can be modeled as follows:

$$
\begin{cases}
min\ c^T x + \sum_{s\in S} p_s(g_s^T y_s + Me^T u_s) \\
s.t.\ Ax = b \\
\quad \delta_{s,n,t} \leq \frac{1}{\epsilon} u_{s,n,t}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \\
\quad R\delta_{s,n,t} \geq u_{s,n,t} - \epsilon, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \\
\quad \sum_{s\in\mathcal{S}} p_s \sum_{n\in\mathcal{N}} \sum_{t\in\mathcal{T}} \delta_{s,n,t} \leq \alpha \\
\quad (y_s, u_s) \in \operatorname{Arg\,min}\{g_s^T y'_s + Me^T u'_s : W_s y'_s + Q_s u'_s + T_s x = d_s\}, \forall s \in \mathcal{S} \\
\quad \delta \in \{0,1\}^{Card(\mathcal{S})\times Card(\mathcal{N})\times Card(\mathcal{T})} \\
\quad x \in \mathbb{R}^{n_{1,C}} \times \mathbb{N}^{n_{1,I}}, y \in \mathbb{R}^{n_2 \times Card(\mathcal{S})}, u \in \mathbb{R}^{Card(\mathcal{N})\times Card(\mathcal{T})\times Card(\mathcal{S})}
\end{cases}
$$

3 Solution method

Many solution methods for mixed-integer bilevel programs require some assumptions on the structure of the problem, such as only integer coupling variables [1], or no constraint on lower-level variables in the upper level. We propose a heuristic algorithm based on the relaxation of the upper-level coupling constraint, and a dichotomic search on investment costs. For a given bound on investment cost $c^T x \geq \Lambda$, we can efficiently solve the resulting problem with Benders decomposition thanks to the relaxation of the upper-level coupling constraint. Moreover, as the Benders cuts associated to the subproblems are valid on the whole feasible domain of x variables, we can warm-start the Benders decomposition at each iteration of the heuristic to accelerate its convergence. We compare the results of the propose algorithm, both in computation time and solution quality with the KKT-reformulation of the problem introduced in [2], and its resolution with SOS1-branching [3].

References

[1] Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. On the use of intersection cuts for bilevel optimization. *Mathematical Programming*, 172(1):77–103, November 2018.

[2] José Fortuny-Amat and Bruce McCarl. A Representation and Economic Interpretation of a Two-Level Programming Problem. *Journal of the Operational Research Society*, 32(9):783–792, September 1981.

[3] S. Siddiqui and S. A. Gabriel. An SOS1-Based Approach for Solving MPECs with a Natural Gas Market Application. *Networks and Spatial Economics*, 13(2):205–227, June 2013.

Session Session 3C: routing problems 3

Thursday 9 June 2022, 10:30-12:10

Lecture Hall IV

A Hybrid Variable Neighborhood Search Algorithm for the Multiple Roaming Salesman Problem

Masoud Shahmanzari[1]

[1]Faculty of Business, Ozyegin University, Istanbul, Turkey, ✉ masoud.shahmanzari@ozyegin.edu.tr

In this paper we study multiple Roaming Salesman Problem (m-RSP), a new variant of the recently introduced Roaming Salesman Problem (Shahmanzari et al., 2020), the goal of which is to determine daily tours for m traveling salesmen who collect time-dependent rewards from various cities during a planning horizon of length T_{max}. m-RSP is a generalization of the traditional traveling salesman problem (TSP) and has a wide range of real-world applications including touristic trip planning, election logistics, nurse routing, multi-period vehicle routing, and marketing campaigns.

Consider a network of G=(N,E) where N denotes the set of nodes including a central node indexed as 1 and E denotes the set of edges. During a planning horizon of T_{max} days, salesmen in set M should decide on which nodes to be visited. Among visited nodes, each salesman should decide which nodes to be included in the reward collection. The total number of visits for each salesman must not exceed the maximum number of permitted visits per day (p). Also, the duration of each daily tour should not exceed the daily maximum tour threshold, denoted by q. A node's reward must be collected by only one salesman per each day. Salesmen can end daily tours in any node in set N. However, it must be guaranteed that the route of today originates where the route of yesterday terminates. Inspired by real-world applications, we assume that every salesman must visit the central node every T_{away} days. Given that sales representatives in real life usually return to central offices frequently, this assumption makes m-RSP more realistic. Every edge in E is assigned a traveling cost and a traveling time, measuring the total traveling cost and time of daily tours for every salesman. The objective function seeks to maximize the total benefit defined as the difference between the collected rewards and the incurred routing costs for all salesmen.

A distinctive feature of m-RSP is that all salesmen are allowed to include open and closed tours on different days of the planning horizon. We present a toy-size m-RSP instance in Figure 1 to illustrate how the combination of closed and open daily tours improves the solution. The instance consists of 5 nodes and one depot. Let us assume travel time and travel costs between each pair of nodes are identical. These values are given in Figure 1. Moreover, the maximum daily tour duration is 8 hours. According to traditional TSP models where starting node and ending node should coincide (Figure 1.a), the objective value of the optimal solution is 22, spanning three days of the planning horizon. Compared with the previous solution, the m-RSP solution (Figure 1.b) provides improved efficiency where all nodes are visited, the total objective value is 20, and one day is saved.

m-RSP can be characterized as an extension of team orienteering problem (Chao et al., 1996) with static edge costs and time-dependent vertex rewards. During the planning horizon, each node can be visited more than once. However, a depreciation coefficient is applied for repeat visits. m-RSP seeks a closed or open tour for every salesman for each period with the objective of maximizing the total benefit. m-RSP is a selective routing problem, e.g., the salesmen are not required to visit all nodes, and the starting and ending nodes of daily tours for each salesman do not necessarily coincide. Moreover, each salesman can stay overnight in any node to start the tour of the next period. Each node is associated with a time dependent reward and a fixed visit duration. Finally, the total length of travel times between nodes and visits on each period cannot exceed a fixed maximum tour duration.

We propose a MILP model to tackle the m-RSP that includes relevant real-life assumptions, some of which are widely used in business context. We categorize decision variables into two distinct groups as routing variables and reward variables. Routing variables deal with determining order of visits and

terminal nodes in any period, being independent of reward collection in any node. Reward variables determine the amount of reward collected from visited nodes. Note that rewards are increased linearly in time as we get closer to the end of the planning horizon rather than the other way around. The objective function consists of maximizing collected rewards, including rewards corresponding to first and repeat meetings, and minimizing travelling costs. To make these two components of objective function compatible, we multiply traveling costs with a normalization coefficient β.

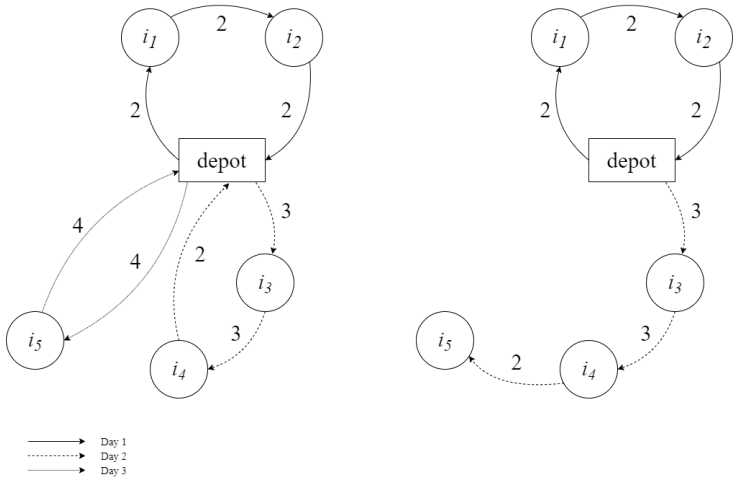


Figure 1: Comparison between a m-RSP solution and the corresponding optimal solution.

Commercial solvers such as GUROBI and CPLEX fall short of solving m-RSP large instances to optimality in a reasonable CPU time. To tackle large-size instances, we develop a new hybrid metaheuristic algorithm that consists of a Skewed Granular Tabu Search which is embedded in a Variable Neighborhood Search algorithm. The proposed method is experimentally validated on 50 real-life instances with actual travel times and distances. The computational results indicate that our method can produce near-optimal solutions very fast. Using an effective mathematical model and a hybrid metaheuristic, we demonstrate that promising results can be achieved to hopefully assist routing and scheduling managers in their strategic decision making.

References

Chao, I. M., Golden, B. L., & Wasil, E. A. (1996). The team orienteering problem. *European journal of operational research*, 88(3), 464-474.

Shahmanzari, M., Aksen, D., & Salhi, S. (2020). Formulation and a two-phase matheuristic for the roaming salesman problem: Application to election logistics. *European Journal of Operational Research*, 280(2), 656-670.

A Metaheuristic Algorithm for a Multi-period Orienteering Problem Arising in a Car Patrolling Application

Full paper

Giorgio Zucchi
School of Doctorate E4E, University of Modena and Reggio Emilia
R&D department, Coopservice s.c.p.a
Reggio Emilia, Italy
giorgio.zucchi@unimore.it

Victor Hugo Vidigal Corrêa
Department of Informatics, Federal University of Viçosa
Viçosa, Brazil
victor.vidigal@ufv.br

Manuel Iori
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia
Reggio Emilia, Italy
manuel.iori@unimore.it

André Gustavo dos Santos
Department of Informatics, Federal University of Viçosa
Viçosa, Brazil
andre@dpi.ufv.br

Mutsunori Yagiura
Graduate School of Informatics, Nagoya University
Nagoya, Japan
yagiura@nagoya-u.jp

ABSTRACT

This paper addresses a real-life multi-period orienteering problem arising in a large Italian company that needs to patrol a vast area in order to provide security services. The area is divided into clusters, and each cluster is assigned to a patrol. A cluster comprises a set of customers, each requiring different services on a weekly basis. Some services are mandatory, while others are optional. It might be impossible to perform all optional services, and each of them is assigned a score when performed. The challenge is to determine a set of routes, one for each patrol and each day, that maximizes the total collected score, while meeting a number of operational constraints, including minimum quality of service, hard time windows, maximum riding time, and minimum time between two consecutive visits for the same service at the same customer. To solve the problem, we propose an iterated local search that invokes at each iteration an inner variable neighborhood descent procedure. Computational tests performed on a number of real-life instances prove that the developed algorithm is very efficient and finds in a short time solutions that are consistently better than those in use at the company.

1 INTRODUCTION

Every day, private security guards need to inspect structures, parks, buildings, and many other facilities to check for any anomalies, in order to counter potential criminal actions or simply restore normal safety conditions following forgetfulness or breakdowns. In this paper, we study a real-life security problem in which patrols are required to perform a set of services at customers located in a vast area. Some services are mandatory, while others are optional. The optional services, when performed, induce a score, and the aim is to maximize the collected score, while meeting different operational constraints.

The problem originates from the everyday activity of Coopservice, a large service provider company located in Italy (https://www.coopservice.it/). Counting on more than 25 000 employees, Coopservice operates a number of different services, including logistics, transportation, cleaning, maintenance, and security. The company also operates car patrolling services in a number of provinces all around Italy. An example of the patrolling activity performed in the province of Reggio Emilia is provided in Figure 1.

The customers are geographically dispersed in the area and are consequently divided into clusters. Each cluster is assigned to a patrol, who performs every day a route to visit customers and executes the required services. The cluster division does not vary from a day to the other, but the routes performed inside the clusters may change. Indeed, customers may require different services according to the day of the week, following the contract stipulated

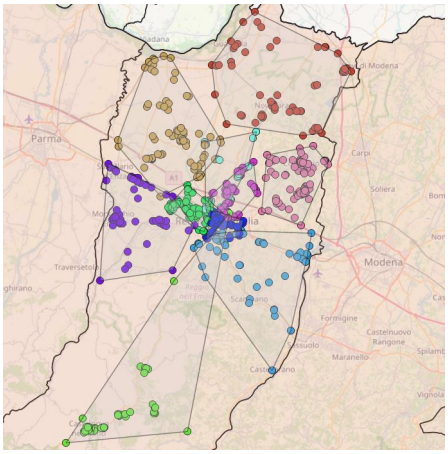


Figure 1: Customers and clusters for the patrolling of the Reggio Emilia province

with the company. More in detail, each customer may require multiple services, and, for each such service, multiple visits during the same period. Some services, such as the closing or opening of a commercial activity, are mandatory, whereas others, such as the inspection of an area or a building, are optional. The optional services create a score when performed, and the company is interested in maximizing the collected score.

The resulting optimization problem is very difficult, as it involves a number of operational constraints. First of all, the services should be performed within hard time windows, and the routes should not exceed a maximum riding time. In addition, a customer might require multiple visits for the same service in the same period. In such a case, two consecutive visits should be separated by at least a threshold time (i.e., 90 minutes or so). This constraint is indeed very challenging, as it requires us to schedule endogenous time windows, induced by the consecutive visits, inside the exogenous time window received in input. Moreover, the company is interested in maintaining a minimum quality of service (QoS) level. The QoS level is computed as the ratio between the number of optional services that have been performed and the number of optional services that have been required. Ideally, the QoS level should be balanced among all customers at the end of the week.

Our aim is to determine a set of routes, one for each period and each cluster, by keeping unchanged the cluster configuration received in input and maximizing the total collected score. The resulting problem is thus a multi-period orienteering problem, which is a generalization of the well-known orienteering problem (OP) [4]. The OP is known to be strongly NP-hard and difficult to solve in practice, and the problem we are facing is a challenging generalization of the OP that includes different additional constraints. For this reason, we decided to solve the problem by means of a metaheuristic algorithm.

We developed an iterated local search (ILS), a metaheuristic that obtained in recent years relevant results on a huge number of optimization problems [7]. The ILS receives in input the set of customers, the set of services to be performed, the cluster configuration, and all details of the instance. It builds an initial solution by means of a constructive heuristic. Then, while the time limit is not reached, it introduces perturbations on the current solution and then improves it by means of a variable neighborhood descent (VND) procedure [6] based on five neighborhood operations.

Computational tests on a set of real-life instances provided by the company prove that the developed ILS works very well in practice. The solutions obtained by the ILS consistently improve the ones in use at the company in terms of the total score. In addition, the average QoS level is also improved.

The remainder of the paper is organized as follows. Section 2 contains a brief literature review of orienteering problems and car patrolling applications. Section 3 formally describes the problem we solve. Section 4 presents the ILS algorithm proposed in this paper. Section 5 shows the results obtained and, finally, Section 6 gives some concluding remarks and hints for future research directions.

2 LITERATURE REVIEW

Car patrolling is a security measure widely used to protect large areas from criminal activity. It consists of guards (patrols) using vehicles to move between points of interest in a region and taking actions that may prevent or respond to crimes. Samanta et al. [9] surveys the police patrolling problem considering every component involved in this complex operation and divides it into three categories: (i) resource allocation, (ii) district design, and (iii) route design. Among these categories, the route design is the one that most resembles our problem, since it is concerned with how the routes are selected and how they affect the patrols' efficiency. Many techniques are used to solve this type of problem and many study cases arise in this context. Some examples are the maximization of the vehicle patrolling coverage in Israel [1], and the minimization of patrols' idle time and unpredictability in London and Chicago [2].

In general, the problems evaluated in [9] differ from ours, due to how Coopservice needs to provide its patrolling service. In fact, in our routing design, not necessarily all clients need to be visited, so our problem is more similar to an orienteering problem (OP). The literature on OPs is very rich and we found plenty of applications. The first study on the OP dates back to 1984 [10], where the OP was presented as a generalization of the traveling salesman problem. Since it is an NP-hard problem, most studies use heuristic methods to solve the OP and its variants. Gendreau et al. [3] discuss why it is so difficult to design high-quality heuristics for this type of problem. The score of a location and the distance to reach it are independent and often in contrast to one another, so it is difficult to select the locations that are part of an optimal solution. Therefore, simple construction heuristics may direct the algorithm towards undesirable directions and are not sufficient to explore large parts of the solution space.

A hybrid heuristic composed of a greedy randomized adaptive search procedure (GRASP) and a variable neighborhood search (VNS) has been proposed by [8] to solve a generalization of the OP. This variant has constraints related to mandatory visits and incompatibilities among nodes. Those constraints mean that there is a set of nodes that must be visited in order for the solution to be considered feasible, while visiting some other nodes is optional. In addition, incompatible nodes cannot share the same route. The hybrid heuristic takes advantage of the multi-start feature of the GRASP to generate initial solutions that are then optimized with the VNS. The authors report that the heuristic was able to find 128 optimal solutions on a set of 131 instances and required, on average, only 0.8% of the time required by an integer linear programming model solved with commercial software.

Recent applications of the OP were studied also for tourism trip planning. For example, Vansteenwegen et al. [11] developed an iterated local search (ILS) metaheuristic to a problem that requires determining a set of touristic locations to visit, while satisfying time limits and budget criteria. The local search step is done through an insertion procedure in which feasible points are added into routes until a locally optimal solution is reached. In the shake step, some visited points are removed from routes according to some predefined parameters. The authors report that, in their experiments, the average gap between the optimal solution value and the ILS one is 2.1%. In [5], the goal is to optimize touristic routes considering constraints such as visit redundancy avoidance and time windows, where some attractions might have a shorter visiting time than others. An integer linear programming model and an ILS metaheuristic

are proposed. The authors report that the ILS could almost match the results from the solver Gurobi, used to execute the model, for the smaller instances, while for the larger instances it could provide better solutions in most cases.

Outside the scope of metaheuristic algorithms, in [12] an approximation algorithm for a variant of the team orienteering problem (TOP) was proposed. In addition to the basic TOP constraints and the basic objective of maximizing the collected score, their problem includes a set of new features to better model Internet of things applications. The new features of the problem are the following: a limited budget is imposed on the vehicles to perform the routes, the node costs are included in the path cost function in addition to the edge costs, and the nodes can be served by multiple vehicles. Computational experiments proved that the developed algorithm provided up to a 17.5% increase in the collected score than the state-of-the-art algorithms for this same TOP variant.

3 PROBLEM DESCRIPTION

The area to be patrolled is divided into clusters, but the cluster configuration is not part of the optimization process. For this reason, we formally describe our problem for a unique cluster served by a unique car.

The problem we face can be viewed as a multi-period orienteering problem with time windows (MPOPTW). In the MPOPTW, we are given a graph $G = (V_0, A)$. The set of vertices is defined as $V_0 = \{0, 1, \ldots, n\}$, where 0 is the depot at which the single vehicle starts and ends each route, and $V = \{1, \ldots, n\}$ is the set of customer locations. The graph is complete, and with each arc $(i, j) \in A$, we associate a traveling time t_{ij}. The time matrix is asymmetric.

Let T be the set of services provided by the company. A standard service time is given for each service $t \in T$. This is denoted by q_t and gives the time duration necessary for a patrol to stay at a customer location to execute the service. The set of services is partitioned as $T = M \cup U$, where M is the set of mandatory services and U is the set of optional services. A score w_t is associated with each optional service $t \in U$, which represents the level of importance of the service.

The activities should be executed on a given set $D \subseteq \{1, \ldots, 7\}$ of periods. Each period corresponds to the working hours from 22:00 of a day to 06:00 of the next day in our instances, as all activities are performed at night time. Each customer $v \in V$ requires services on a subset $D_v \subseteq D$ of periods. Formally, we denote by $T_{vd} \subseteq T$ the set of services to be performed at customer v on period d. This set is partitioned as $T_{vd} = M_{vd} \cup U_{vd}$, where $M_{vd} \subseteq M$ comprises mandatory services and $U_{vd} \subseteq U$ optional ones.

Let n_{vdt} be the number of times service t is required by customer v in period d, and let $\widetilde{n}_{vdt}$ be the number of services that have been actually performed. We define the QoS level as $QoS = \sum_{v \in V,\, d \in D_v,\, t \in T_{vd}} \widetilde{n}_{vdt} / n_{vdt}$. This index represents the ratio of services that have been performed in the entire set of periods, and it should be at least a required value $QoS_{\min}$ (which is set to 75% in our instances).

Every service t required by a customer v in a day d has a time window $[e_{vdt}, l_{vdt}]$. This defines the earliest and latest possible times to start the execution of each of the n_{vdt} visits for that customer in that period. The time window defines a hard constraint, so late arrivals are forbidden and waiting on site is imposed in case of early arrivals. A time window is also imposed on the depot and corresponds to the total working time (from 22:00 of a day to 06:00 of the next day in our instances).

For some services, such as closing or opening a commercial activity, the time window is strict (e.g., 10 minutes) and just one visit per night is required. This typically happens for mandatory services. For other services, such as checking a private house, the time window is usually loose (e.g., several hours during the night) but multiple visits are required in a period. This typically happens for optional services. In the latter case, if two or more visits are performed for the same service at the same customer in the same period, the start times of any two of such visits should be separated by at least a given threshold δ (which is equal to 90 minutes in our instances). This is imposed to enforce a balanced patrol of the customer during the execution of a route.

For each period, a patrol starts its route at the depot, performs visits to customers to execute the services, and then returns to the depot. The total riding time, which comprises traveling, service, and waiting times, is the difference between the start and end times of the route, and it should not exceed a given upper bound $Q_{\max}$.

To summarize, the aim of the MPOPTW is to define a set of routes, one per period, in such a way that (i) constraints on hard time windows, the QoS level, the time distance between consecutive visits, and the total riding time are satisfied; (ii) all mandatory services are performed; and (iii) the score of the optional services that have been actually performed is maximized.

The problem is of interest not only because of its real-life application, but also because it is very general and models a large number of other possible applications arising in the context of car patrolling and attended home services. In the next section, its solution is pursued by means of a metaheuristic algorithm.

4 PROPOSED METHODOLOGY

To solve the problem, we developed an ILS metaheuristic [7]. The ILS receives as input the set of customers, the set of services to be performed, the cluster configuration, and other details of the real-life application. It builds an initial solution using a constructive heuristic and then applies perturbations and local search iteratively in the current solution until a time limit is reached. An acceptance function decides at each iteration whether to keep the current solution or to move to a newly-generated one. In our algorithm, it always chooses the best solution among the two, and if their fitness values are the same, our algorithm keeps the current one. The overall ILS procedure is presented in Algorithm 1. Each step is described in detail in the remaining part of this section. The local search is performed by means of a VND, an algorithm that sequentially invokes a set of neighborhoods [6].

4.1 Solution evaluation

The objective of the problem is to maximize the total score achieved with the optional services performed by the patrol. However, in order to guide our algorithm, we use a fitness function that, together with the score, takes into account also the riding time of the routes. Let $\mathcal{S}(\sigma)$ be the total score of a given route σ and $\mathcal{T}(\sigma)$ be its riding

Algorithm 1 ILS algorithm

```
1: procedure ILS(T_max)
2:     s* ← ConstructiveHeuristic
3:     s* ← VND(s*)
4:     while elapsedTime ≤ T_max do
5:         s' ← Perturbation(s*)
6:         s'' ← VND(s')
7:         s* ← Accept(s*, s'') ▷ the best of s* and s'' with respect to F
8:     end while
9:     return the best feasible solution found during the search
10: end procedure
```

time. The fitness function used by the proposed ILS algorithm to evaluate a solution s is a weighted average of these values, namely

$$\mathcal{F}(s) = \alpha \sum_{\sigma \in s} \mathcal{S}(\sigma) - \beta \sum_{\sigma \in s} \mathcal{T}(\sigma), \tag{1}$$

where α and β are weights to be calibrated. By doing this, we expect that the algorithm favors shorter routes during the optimization in order to add more services later on, thus improving the score.

4.2 Constructive heuristic

An initial solution is constructed by a greedy algorithm. The services of each period are sorted in non-decreasing order of the start time of their time windows. A route is constructed for each period in two phases: first, the mandatory services are inserted sequentially, in the order in which they were sorted (this is always feasible for the mandatory services in the instances provided by the company); later, while possible, optional services are appended one by one in the solution (i.e., an optional service is appended if the solution remains feasible, otherwise it is skipped). Algorithm 2 summarizes the steps of this heuristic. In the ILS, the solution obtained by Algorithm 2 is optimized by invoking a VND procedure to the solution built by the constructive heuristic.

Algorithm 2 The greedy constructive heuristic

```
1: procedure Constructive Heuristic(M, U)
2:     for each period d ∈ D do
3:         M_d, U_d ← services in M and U for period d
4:         Sort M_d and U_d in non-decreasing order of e_vdt
5:         σ_d ← AppendAllMandatory(M_d)
6:         while CanAppendOptionals(σ_d, U_d) do
7:             σ_d ← AppendOptionals(σ_d, U_d)
8:         end while
9:     end for
10:    return s = {σ_1, ..., σ_D}
11: end procedure
```

4.3 VND heuristic

A VND procedure is used to find a locally optimal solution using a sequence of different neighborhoods N_k ($k = 1, \ldots, k_{\max}$). Algorithm 3 shows the main steps of the VND heuristic. Starting with the first neighborhood ($k = 1$), the heuristic explores the solution space searching through the sequence of neighborhoods in a deterministic way, controlled by the neighborhood change procedure presented in Algorithm 4. The steps of these algorithms are detailed in the following.

At each step of the VND, a neighbor s' of the current solution s is selected from the neighborhood $N_k(s)$. A neighbor is always selected by a first improvement move in the fitness function $\mathcal{F}$ defined in (1). If there is no better neighbor in the kth neighborhood (i.e., the current solution is locally optimal with respect to this neighborhood), the algorithm changes to the next neighborhood, N_{k+1}. If, instead, a better neighbor is found, the algorithm returns to the first neighborhood. The process continues while there is a neighborhood to be explored, that is, it stops when the current solution is locally optimal with respect to all neighborhoods.

For the VND, we implemented five classical neighborhood operators from the traveling salesman and vehicle routing literature. **Swap**: this operator swaps the positions of two visits inside a route. **2-opt**: this operator reverses the visiting order between two visits in a route. **Relocate**: this operator moves a visit to another position in the route. **Swap unrouted**: this operator swaps the status of two services, an unrouted service takes place of a performed service. **Insertion unrouted**: this last operator inserts a visit in a route to perform an unrouted service, increasing the number of services performed by the patrol. Note that the first three operators, Swap, 2-opt and Relocate, may improve the fitness function by reducing the riding time of a route, as they do not change the collected score. The operator Swap unrouted may instead improve the fitness function by increasing the score or by reducing the riding time. The last operator, Insertion unrouted, may improve the score at the expense of an increase in the riding time.

Note that all described neighborhoods consist of intra-period movements, in which they change routes of each period independently. A solution may be further improved by performing inter-period movements, exchanging services from different periods. For example, a customer requiring services in more than one period that is currently served in only a subset of those periods may have the visits changed without changing the total score, perhaps decreasing the riding time. Inter-period movements are costly to be evaluated because of the large number of neighbors. Hence, they are not fully explored in a deterministic way, but are considered in the perturbation step, described next.

Algorithm 3 Variable neighborhood descent heuristic

```
1: procedure VND(s)
2:     k ← 0
3:     s* ← s
4:     while k ≤ k_max do
5:         Let s be an improved sol. in N_k(s*) if any; otherwise s ← s*
6:         s*, k ← NeighborhoodChange(s*, s, k)
7:     end while
8:     return s*
9: end procedure
```

4.4 Perturbation procedure

The perturbation procedure is introduced to escape from the locally optimal solution obtained by the VND used in the local search step. Two inter-period operators are used, which both attempt to modify the current solution in the set of periods.

Algorithm 4 Neighborhood change procedure

```
1: procedure NeighborhoodChange(s*, s, k)
2:     if F(s) > F(s*) then
3:         s* ← s
4:         k ← 1
5:     else
6:         k ← k + 1
7:     end if
8:     return s*, k
9: end procedure
```

Two periods are randomly chosen and an arbitrary number of shaking movements are applied, limited by a maximum number of tries and a maximum number of successful movements. The **Swap inter-period** operator randomly chooses a service in the route of a chosen period and tries to swap it with every other service in a successive period. The **Move inter-period** chooses one service routed in a period and tries to insert it in every position of the route of a different period. In either case, if a try succeeds in finding an improved solution, then the move is applied. Only feasible moves are considered.

5 COMPUTATIONAL EVALUATION

In this section, we present the computational results that we obtained. First, we briefly describe the instances we address. Then, we report the calibration of the main ILS parameters. Finally, we present the results obtained by the ILS. The algorithm was coded in Python 3.7.3 and executed on an Intel Xeon CPU E5-2640 v3 2.60 GHz with 64 GB of memory, running under Windows 10 Pro 20H2 64-bits.

5.1 Instances

The company provides security services in a number of provinces of Italy. We were provided with the data of four of such provinces. Each of them is different in the number of customers and frequency of tasks. Table 1 reports for each province, in order, the number of instances, the number of periods (column $|D|$), the total number of customers (column $|V|$), and the total number of services (column $|T|$). The traveling times have been obtained by using the OSRM application.

Table 1: Details of the real-life instances

Province	Instances	$\|D\|$	$\|V\|$	$\|T\|$
Parma	5	7	175	1382
Pescara	5	7	160	987
Sassari	9	7	122	1044
Roma	8	7	121	1234

5.2 Parameter calibration

As the size of the instances changes by province, we defined a different time limit $T_{\max}$ (the maximum time allowed for the ILS execution) for each of them, based on the number of customers. We set $T_{\max}$, in minutes, to 120, 120, 60, and 60 for Parma, Pescara, Sassari, and Roma instances, respectively.

As the fitness function $\mathcal{F}$ defined by (1) used to guide the algorithm has two parameters, α and β, we ran preliminary tests for all combinations of $\alpha = [1, 2, 3, 4, 5]$ and $\beta = [0.1, 0.3, 0.5, 0.7, 0.9]$. For any combination, we evaluated the reported solutions considering (i) the total score, (ii) the mean of the distances traveled by the patrols, (iii) the mean of the patrol times, (iv) the mean of the waiting times, and (v) the QoS level obtained. For each instance, a ranking of the parameter combination results was created. To merge all rankings into a unique one, we assigned scores for each ranking as follows: from position 1 to 5 of a ranking the score is 5, from 6 to 10 the score is 4, from 11 to 15 the score is 3, from 16 to 20 the score is 2, and from 21 to 25 the score is 1. Then, we summed the scores for each combination. Note that the higher is the ranking, the larger is the sum. The winning combination was determined as $\alpha = 5$ and $\beta = 0.9$ and was used in all successive experiments.

5.3 Computational experiments

To compare the solutions obtained by the ILS with the ones in use at the company, several criteria have been considered: $\overline{km}$, the average distance traveled by the patrols; $\overline{dv}$, the average distance between two consecutive visits; $\overline{\mathcal{T}}$, the average riding time; $\overline{uc}$, the average number of unvisited customers; $\overline{QoS}$, the average QoS; $\overline{\delta}$, the average time between two visits at the same customer; and $\mathcal{S}$, the total collected score.

Table 2 reports the above parameters computed for the solutions in use at the company. It is important to highlight that the solutions by the company do not always satisfy the constraint on the desired interval of $\delta = 90$ minutes between two visits to the same customer.

Table 2: Evaluation of the solutions in use at the company

Province	$\overline{km}$	$\overline{dv}$	$\overline{\mathcal{T}}$	$\overline{uc}$	$\overline{QoS}$	$\overline{\delta}$	$\mathcal{S}$
Parma	156.85	2.51	431.63	1.40	92.56	85.39	4117.57
Pescara	53.42	1.14	339.69	21.12	48.63	30.74	693.43
Sassari	40.81	2.39	217.21	1.12	92.72	45.38	718.86
Roma	92.56	5.65	379.76	0.50	83.12	87.73	1393.28

Since the ILS algorithm contains a random factor in the perturbation step, we have run the algorithm multiple times for each instance. Table 3 reports the average results obtained for five runs of the ILS, and Table 4 shows the comparison with the real-life solutions currently adopted by the company. Table 5 reports the difference reported in Table 4, but in terms of percentages in order to better highlight the results (we have decided to exclude column $\overline{uc}$ from Table 4 because all values were -100%).

The results obtained by the ILS show several improvements with respect to the real ones. The only negative result is noticed for the distance $\overline{km}$ traveled by the vehicles. This happens because, in order to maximize the score, more optional services have been performed, and this requires more km to be traveled. The two measures used in our guide fitness function, $\mathcal{T}$ and $\mathcal{S}$, were highly improved. The average travel time was reduced by 10 to 42% for the tested instances, while the total score was increased by 11 to 96%. Moreover, although not directly used in the fitness function, other criteria were also improved, as they were used to choose the weights

α and β in this function. Note that in our solutions, no customer is left unvisited. Moreover, almost all tasks are performed, as the $\overline{QoS}$ is above 97%. The distance $\overline{dv}$ between two visits is similar to the one obtained by the company, which indicates that the neighborhood operators were able to find routes as good as those designed by the experienced workers of the company. Furthermore, the δ parameter, the interval between two consecutive visits to the same customer, is always respected in the solutions of our ILS algorithm, which we recall is not the case in the solutions by the company.

Table 3: Evaluation of the solutions obtained by the ILS

Province	$\overline{km}$	$\overline{dv}$	$\overline{\mathcal{T}}$	$\overline{uc}$	$\overline{QoS}$	$\overline{\delta}$	$\mathcal{S}$
Parma	179.69	2.17	335.46	0	98.22	153.51	4572.60
Pescara	127.78	3.32	197.35	0	98.29	93.25	1365.20
Sassari	69.98	4.29	170.64	0	98.52	122.85	1052.60
Roma	115.61	5.40	338.04	0	97.10	131.08	1750.60

Table 4: Absolute differences between ILS and company

Province	$\overline{km}$	$\overline{dv}$	$\overline{\mathcal{T}}$	$\overline{uc}$	$\overline{QoS}$	$\overline{\delta}$	$\mathcal{S}$
Parma	22.84	−0.34	−96.17	−1.40	5.66	68.12	455.03
Pescara	74.36	2.18	−142.34	−21.12	49.65	62.50	671.77
Sassari	29.17	1.90	−46.57	−1.12	5.80	77.46	333.74
Roma	23.05	−0.25	−41.71	−0.50	13.98	43.34	357.32

Table 5: Percentage differences between ILS and company

Province	$\overline{km}$	$\overline{dv}$	$\overline{\mathcal{T}}$	$\overline{QoS}$	$\overline{\delta}$	$\mathcal{S}$
Parma	14%	−13%	−22%	6%	79%	11%
Pescara	139%	190%	−42%	102%	203%	96%
Sassari	71%	79%	−21%	6%	171%	46%
Roma	24%	−4%	−10%	17%	49%	26%
Average	**62%**	**63%**	**−23%**	**58%**	**125%**	**44%**

6 CONCLUSIONS AND FUTURE RESEARCH

This paper presented a study on a car patrolling application that arose from a large Italian company, which has to plan routes to perform mandatory and optional services at customers. The resulting optimization problem is a challenging variant of a multi-period orienteering problem. Because of its difficulty, we decided to solve it with an iterated local search (ILS) metaheuristic, which was enriched with a variable neighborhood descent.

Through a computational analysis, we observed that the ILS consistently improved the company solutions. The improvement obtained was remarkable for both the total collected score and for the overall quality of service (a measure of the percentage of optional services that were performed). In the province of Pescara, just to give an example, the quality of service was remarkably increased by 102%. Another important result obtained by the ILS comes from the fact that all solutions were feasible with respect to the constraint that imposes a required elapsed time between two consecutive visits to the same customer. This constraint is indeed not always satisfied in the solutions by the company.

We present four suggestions for future studies. First, the modification of the cluster: the current clusters were provided by the company, but we foresee that changing their configuration might help improve the solutions even further. Second, the insertion of dynamic and stochastic features in the problem: in the current version of the problem, dynamic occurrences such as alarm triggering or unexpected urgent services are not considered; by embedding them into a new problem that considers dynamic and stochastic aspects, we may obtain a more sophisticated model, to be used on-the-fly during the execution of the activities. Third, a more elaborated evaluation of the quality of service: in our study, the quality of service is evaluated using an overall measure of the number of services performed, which may introduce unfairness as some customers may have been poorly served while others fully served; introducing a quality of service measured per single customer might improve the fairness of the resulting solutions. Finally, the development of an integer linear programming model represents another interesting topic for future research. This model might be used to provide proven optimal solutions, or at least to assess the quality of the solutions found by the metaheuristic.

ACKNOWLEDGMENTS

We acknowledge financial support by Coopservice and by JSPS KAKENHI Grant No. 20H02388. A special thank to Hang Dong, Nicolas Porto Campana and Matteo Magnavacchi for their support to the research.

REFERENCES

[1] Ramon Auad and Rajan Batta. 2017. Location-coverage models for preventing attacks on interurban transportation networks. Annals of Operations Research 258, 2 (2017), 679–717.
[2] Huanfa Chen, Tao Cheng, and Sarah Wise. 2017. Developing an online cooperative police patrol routing strategy. Computers, Environment and Urban Systems 62 (2017), 19–29.
[3] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. 1998. A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research 106, 2-3 (1998), 539–545.
[4] Bruce L. Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. Naval Research Logistics 34, 3 (1987), 307–318.
[5] Felix Gündling and Tim Witzel. 2020. Time-dependent tourist tour planning with adjustable profits. In 20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020), Dennis Huisman and Christos D. Zaroliagis (Eds.), Vol. 85. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 14:1–14:14.
[6] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Pérez. 2019. Variable neighborhood search. In Handbook of Metaheuristics. Springer, 57–97.
[7] Helena Ramalhinho Lourenço, Olivier C. Martin, and Thomas Stützle. 2019. Iterated local search: Framework and applications. In Handbook of Metaheuristics. Springer, 129–168.
[8] Pamela J. Palomo-Martínez, M. Angélica Salazar-Aguilar, Gilbert Laporte, and André Langevin. 2017. A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints. Computers & Operations Research 78 (2017), 408–419.
[9] Sukanya Samanta, Goutam Sen, and Soumya Kanti Ghosh. 2021. A literature review on police patrolling problems. Annals of Operations Research (2021).
[10] Theodore Tsiligirides. 1984. Heuristic methods applied to orienteering. Journal of the Operational Research Society 35, 9 (1984), 797–809.
[11] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk van Oudheusden. 2009. Metaheuristics for tourist trip planning. In Metaheuristics in the Service Industry. Springer, 15–31.
[12] Wenzheng Xu, Weifa Liang, Zichuan Xu, Jian Peng, Dezhong Peng, Tang Liu, Xiaohua Jia, and Sajal K. Das. 2021. Approximation algorithms for the generalized team orienteering problem and its applications. IEEE/ACM Transactions on Networking 29, 1 (2021), 176–189.

An efficient heuristic for solving the traveling salesman problem with time windows under various objectives

Mengdie Ye[1] and Michael Schneider[2]

[1,2]Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, Germany,
✉{ye|schneider}@dpo.rwth-aachen.de

1 Introduction

The traveling salesman problem with time windows (TSPTW) arises both in scheduling and routing applications such as school bus transportation, postal deliveries and single-machine scheduling. The problem aims to find an optimal Hamiltonian tour with respect to different objectives, which depend on the goals of the specific application. A classical variant is to minimize the total travel cost or time (TSPTW-T). Excellent solution methods, including exact and heuristic ones, have been introduced to solve this variant [3, 4]. Other important objectives include but are not limited to: the minimization of completion time or makespan (TSPTW-C) [1], and the minimization of total duration (TSPTW-D) [5].

In this paper, we introduce a new variant maximizing the minimum slack (TSPTW-S), i.e., the smallest time buffer between arrival time and the end of the time window over all customers. Thus, we target four variants in total: TSPTW-T, TSPTW-C, TSPTW-D and TSPTW-S. To solve these variants, we develop a two-phase heuristic. We apply an efficient move evaluation procedure that is introduced by [6] for the vehicle routing problem with time windows (VRPTW) and embed the procedure within the local search. The resulting algorithm is able to solve all considered variants. For each variant, we conduct extensive numerical experiments to assess the quality of the solutions obtained by our algorithm against the best known solutions from the literature.

2 Solution method

Our two-phase heuristic is based on the general variable neighborhood search (GVNS) framework of [2] and works iteratively in two phases. The first is a constructive phase, which tries to find a feasible solution using a standard variable neighborhood search (VNS). This phase iteratively calls a combination of a shaking and a local search procedure until a feasible solution is found or the time limit is reached. We modify the shaking procedure by replacing the *level-based* random *1-shift* moves of [2] with a *level-based* destruction and construction procedure of [3], and we consider a *1-shift* neighborhood in the local search. The second phase is an improvement phase, which aims to improve the feasible solution found by the first phase using a GVNS, i.e., a VNS with variable neighborhood descent (VND) as local search. The GVNS calls iteratively the same shaking procedure that is used in the first phase and the VND to update the overall best solution. We consider six neighborhoods in total in the VND: *1-shift backward*, *1-shift forward*, *1-swap*, *2-shift backward*, *2-shift forward*, *2-opt*. The phase terminates if no improvement has been found in $iter_{max}$ consecutive iterations or a time limit t_{max} is reached. In both phases, we adapt the constant-time move evaluation procedure from [6] in the local search to speed up the search.

3 Preliminary results

We conduct experiments for each variant on multiple well-known TSPTW benchmarks to evaluate the performance of the proposed two-phase GVNS. In total seven benchmark sets are considered. Table 1 presents a sample summary of the computational results on two of the considered sets, namely *AFG* and *GDE*. For each variant, we report the best known solutions (BKS), the solutions from the state-of-the-art heuristics from the literature, and the solutions obtained by our heuristic. Note that for TSPTW-S, there

exist no BKS from the literature. We solve the problem exactly using a constraint programming (CP) algorithm. Thus, the set of BKS consists of the best solutions obtained using both the CP algorithm and our two-phase heuristic. Each row in the table reports the aggregated results for all instances included in the set.

Table 1: Aggregated results on benchmark sets *AFG* and *GDE*.

	TSPTW-T					TSPTW-C					TSPTW-D					TSPTW-S				
		Start-of-the-art		GVNS			Start-of-the-art		GVNS			Start-of-the-art		GVNS			Exact CP		GVNS	
Inst.	BKS	*gap%*	*time*	*gap%*	*time*	BKS	*gap%*	*time*	*gap%*	*time*	BKS	*gap%*	*time*	*gap%*	*time*	BKS	*gap%*	*time*	*gap%*	*time*
AFG	5550.80	**0.000**[4]	4.9	0.040	1.9	9568.63	**0.000**[1]	0.4	0.001	0.4	8393.94	0.460[5]	-	**0.006**	2.0	433.52	**0.000**	1078.2	-0.040	5.3
GDE	**432.08**	0.070[3]	1.1	0.070	0.6	575.03	**0.000**[1]	0.3	**0.000**	0.2	**505.60**	0.410[5]	-	0.000	1.2	56.26	-0.050	197.7	**0.000**	1.4

Note[1]: All experiments, including solving TSPTW-S using CP, were run on a computing cluster with an Intel Xeon E5-2430v2 processor with 2.50 GHz. [3] and [4] used a 2.53 GHz processor, [1] and [5] used a 3.40 GHz processor.
Note[2]: For each instance, our heuristic is run 30 times and the time limit t_{max} is 120 seconds. The $gap\%$ is computed as $(100 \cdot \frac{best-BKS}{BKS})\%$, where *best* is the solution obtained by the state-of-the-art heuristic or by our heuristic. The average runtime is presented in seconds.

The main observation is that our two-phase GVNS is competitive with the state-of-the-art heuristics for all variants. Our heuristic can find solutions with the same quality for TSPTW-T and TSPTW-C in the *GDE* set, and a gap below 0.04% exists for these two variants in the *AFG* set. For TSPTW-D, our heuristic beats the simple VND heuristic in [5] by providing better solutions for both sets with an overall gap below 0.006% to the BKS. For TSPTW-S, our heuristic finds feasible solutions for all instances that can be solved exactly by the CP algorithm for both sets, and is able to improve the solutions in the *GDE* set. Moreover, the runtime of our heuristic is significantly smaller than that of the CP algorithm in both sets.

4 Conclusions

We develop a two-phase GVNS heuristic to solve multiple TSPTW variants with different objectives efficiently. The proposed heuristic performs well for all considered variants and is competitive with respect to the start-of-the-art heuristics that are developed separately for each variant. Future work will concentrate on the development of improved strategies to further optimize the performance of our heuristic.

References

[1] Khalid Amghar, Jean-François Cordeau, and Bernard Gendron. *A General Variable Neighborhood Search Heuristic for the Traveling Salesman Problem with Time Windows under Completion Time Minimization.* CIRRELT, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, 2019.

[2] Rodrigo Ferreira da Silva and Sebastián Urrutia. A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010.

[3] Korhan Karabulut and Mehmet Fatih Tasgetiren. A variable iterated greedy algorithm for the traveling salesman problem with time windows. *Information Sciences*, 279:383–395, 2014.

[4] Nenad Mladenović, Raca Todosijević, and Dragan Urošević. An efficient general variable neighborhood search for large travelling salesman problem with time windows. *Yugoslav Journal of Operations Research*, 23(1):19–30, 2013.

[5] Christian Tilk and Stefan Irnich. Dynamic programming for the minimum tour duration problem. *Transportation Science*, 51(2):549–565, 2017.

[6] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. *Computers & Operations Research*, 40:475–489, 2012.

A Decomposition Branch-and-Cut Algorithm for the Maximum Cross-Graph k-Club Problem

Hao Pan
hao.pan@okstate.edu
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA

Balabhaskar Balasundaram
baski@okstate.edu
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA

Juan S. Borrero
juan.s.borrero@okstate.edu
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, Oklahoma, USA

ABSTRACT

The analysis of social and biological networks often involves modeling clusters of interest as *cliques* or their graph-theoretic generalizations. The *k-club* model, which relaxes the requirement of pairwise adjacency in a clique to length-bounded paths inside the cluster, has been used to model cohesive subgroups in social networks and functional modules/complexes in biological networks. However, if the graphs are time-varying, or if they change under different conditions, we may be interested in clusters that preserve their property over time or under changes in conditions. To model such clusters that are conserved in a collection of graphs, we consider a *cross-graph k-club* model, a subset of nodes that forms a k-club in every graph in the collection. In this paper, we consider the canonical optimization problem of finding a cross-graph k-club of maximum cardinality. We introduce algorithmic ideas to solve this problem and evaluate their performance on some benchmark instances.

KEYWORDS

Cross-graph mining, temporal networks, k-clubs, integer programming

1 INTRODUCTION

In graph-based data mining (or graph mining), a node models a data item with different attributes, and two nodes are joined by an edge if they are "close" to each other based on similarity measures. Graph mining in social and biological networks involves modeling clusters of interest using cliques and their graph-theoretic generalizations. In these graphs, a cohesive/tight-knit subset is a group whose member nodes are believed or verified to intimately cooperate with each other towards some specific goal. Cohesive subgroups in social networks could be identified for use in recommender systems, marketing campaigns, community detection, influence maximization, and so forth [3]. In biological networks like protein interaction networks, gene co-expression networks, and metabolic networks, clusters and network motifs are commonly used to identify functional modules that could represent protein complexes, transcriptional modules, or signaling pathways [12].

The clique and its graph-theoretic relaxations have been extensively studied and used as cluster models in diverse fields [19]. Major categories include distance based relaxations k-clique and k-club [6], and edge count, degree, and density based relaxations k-defective clique [25], k-plex [5], and quasi-clique [15], respectively.

A significant body of literature on optimization methods for cluster detection seeks to find a subset of nodes satisfying a graph property while optimizing a measure of fitness like cluster size or weight. One common characteristic shared by optimization approaches to graph mining is that they identify cohesive subgraphs, critical nodes, most central actors, or other graph structures of interest in a single graph. However, in many settings the graphs are time-varying as the underlying dynamic systems they are modeling evolve over time. In this case, a single graph is typically a snapshot that reflects node relationships at the point in time it is recorded.

Alternatively, relationships between a group of nodes may be different under different conditions. Jointly mining node relationships under different conditions might uncover novel clusters that cannot be found by individually analyzing each condition. An example in cross-market customer segmentation is finding customers who have similar behaviors across different markets as a more robust cohesive subgroup than those found in a single market [21]. Similarly, systems biologists are interested in finding groups of co-expressing genes or interacting proteins that are conserved under different biological conditions or between different species [20].

In this paper we consider a *cross-graph k-club* model to represent clusters that are conserved in a collection of graphs. Note that the graph collection may represent temporal graphs with an implicit ordering, or may be obtained under different (experimental) conditions without any natural ordering. Although our focus is on clusters that induce low-diameter subgraphs, one may investigate any clique relaxation or another graph property in the same setting.

2 CROSS-GRAPH k-CLUBS

For a simple graph G, we use $V(G)$ and $E(G)$ to denote its node and edge sets respectively. For simplicity we use uv to denote an edge $\{u, v\} \in E(G)$. For a subset of nodes $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced by S, obtained by deleting nodes outside S and their incident edges. We denote by $\text{dist}_G(i, j)$ the minimum number of edges on a path connecting nodes i and j in graph G and its diameter as $\text{diam}(G) := \max\{\text{dist}_G(i, j) : i, j \in V(G)\}$.

Definition 2.1 ([14]). Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-clique* if $\text{dist}_G(i, j) \leq k$ for every pair of nodes $i, j \in S$.

A k-clique S allows two vertices u and v to be in S even if every path between u and v of length at most k in G includes vertices outside S (see Figure 1). By contrast, in a k-club, at least one of those paths should be contained in S as the definition below states.

Definition 2.2 ([16]). Given a graph G and a positive integer k, a subset of nodes $S \subseteq V(G)$ is called a *k-club* if $\text{diam}(G[S]) \leq k$.

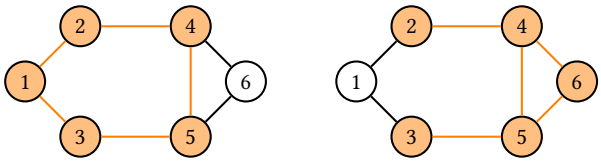


Figure 1: The set $\{1, 2, 3, 4, 5\}$ forms a 2-club; the set $\{2, 3, 4, 5, 6\}$ forms a 2-clique, but does not induce a 2-club [2].

For low values of parameter k, typically no more than four, the k-club can be an appropriate choice for modeling cohesive social subgroups or tightly knit clusters. We define the cross-graph counterpart of the k-club, based on the cross-graph quasi-clique model introduced by Pei et al [21], which also appears to be the earliest formal study of a cross-graph cluster model. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_p\}$ denote a collection of p simple, undirected graphs, all defined on a common node set denoted by $V(\mathcal{G})$.

Definition 2.3. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *p-graph k-club* if S is a k-club in each graph in the collection $\mathcal{G}$.

This paper focuses on *the maximum p-graph k-club problem*, which seeks to find a p-graph k-club of maximum cardinality in $\mathcal{G}$. We use the prefix "p-graph" when we know or wish to specify that there are p graphs in the collection. Otherwise, in line with past usage, we simply refer to it as a cross-graph k-club [21].

3 LITERATURE REVIEW

The (1-graph) maximum k-club problem is NP-hard for every value of parameter k fixed in the problem [8]. Consequently, the maximum p-graph k-club problem is NP-hard for every fixed positive integer k as it includes the maximum k-club problem as special case. Shahinpour and Butenko [23] provide a comprehensive survey on the complexity results and algorithmic approaches for the maximum k-club problem. Given the focus of this paper, we limit our review to integer programming (IP) formulations of the maximum k-club problem and prevailing works on cross-graph models.

The first IP formulation for the maximum k-club problem seen in the literature was the chain formulation given by Bourjolly et al [8]. The chain formulation introduces a binary variable for each path of length at most k connecting a nonadjacent pair of nodes i and j, in addition to binary variables indicating membership in the k-club. For the special case of $k = 2$, the binary variables for the paths are unnecessary as each path of length 2 between i and j is uniquely identified by the common neighbor internal to that path. Thus, we obtain the so-called common neighbor formulation for the maximum 2-club problem. Path enumeration gets increasingly challenging as k takes values larger than two, and for arbitrary k it can take up to $O(n^{k+1})$ binary variables and constraints to fully describe the chain formulation on a graph with n nodes.

For the maximum 3-club problem, Almeida and Carvalho [4] introduced a compact neighborhood formulation, as well as a node cut set formulation with exponentially many constraints in the worst case. Veremyev and Boginski [26] introduced two polynomial-sized IP formulations for the maximum k-club problem, one using binary variables and the other using integer variables, obtained by linearizing a polynomial formulation. Both are described by $O(kn^2)$ variables and constraints on an n-node graph, and are known to be the first compact IP formulations for the maximum k-club problem for general k.

Moradi and Balasundaram [13, 17] proposed a branch-and-cut algorithm that is based on a delayed application of canonical hypercube cuts to eliminate integral solutions to the initial relaxation (a maximum k-clique formulation) that do not correspond to a k-club. They also introduced an iterative graph decomposition framework based on variable fixing to solve the problem on potentially small subgraphs of the original graph and to take advantage of intermediate solutions in preprocessing (vertex deletion) between iterations.

Salemi and Buchanan [22] introduced a cut-like formulation and a path-like formulation using length-bounded separators and connectors. Their study generalizes for arbitrary k, some of the aforementioned formulations for the special cases, when $k = 2, 3$. Generally, the cut-like formulation could use exponentially many constraints, but only $|V|$ binary variables. This formulation and the associated decomposition branch-and-cut algorithm demonstrated effective computational performance making it the current state-of-the-art mathematical programming approach to solve the maximum k-club problem for arbitrary k.

Although previous works on cross-graph models in the literature are limited, we mention some examples that are related and motivated our study. To extract reliable patterns across multiple pieces of data, Pei et al [21] mined cross-graph quasi-cliques and developed algorithms to enumerate them across multiple graphs. Jiang and Pei [11] extended this work to the problem of finding frequent cross-graph quasi-cliques. They seek to enumerate maximal node subsets that form quasi-cliques in at least a minimum number of graphs in the collection. We must clarify that the terminology has been reused as these are "degree-based" quasi-cliques [18], and not "density-based" quasi-cliques [15].

Sim et al [24] introduced an approach to clustering stocks that exhibit homogeneous financial ratio values by mining the complete set of cross-graph quasi-bicliques in a bipartite graph. This bipartite graph has stocks as nodes in one partition and different features of the stock data in the other partition. The cross-graph quasi-biclique model was used to handle the issue of missing values in stock data.

4 IP FORMULATIONS

We begin with an IP formulation that is a direct extension of the cut-like formulation for the maximum k-club problem to the cross-graph setting. Then we propose a new formulation based on what we refer to as 'pairwise peeling.'

4.1 A conjunctive cut-like formulation

We use the following additional notations in the formulation. Let $\overline{G}$ denote the complement graph of G. Given a graph G and a pair of nonadjacent nodes u and v, a subset of nodes S is called a length-k

u, v-separator if $\text{dist}_{G\setminus S}(u,v) > k$, where $G \setminus S$ denotes the graph obtained from G by deleting the nodes in S along with its incident edges. In other words, every path of length at most k in G between u and v uses nodes from S. By $\mathcal{S}_G(u,v)$, we denote the collection of all length-k u, v-separators that are minimal by exclusion. For the case $k = 2$, the unique minimal length-2 u, v-separator is the set of common neighbors, *i.e.*, nodes adjacent to both u and v in G. For a subset of nodes C, we use the short form $x(C)$ to denote $\sum_{u\in C} x_u$. Consider the following optimization problem:

$$\max\ x(V(\mathcal{G})) \tag{1a}$$

$$\text{s.t. } x_u + x_v - x(S) \le 1 \quad \forall S \in \mathcal{S}_G(u,v), uv \in E(\overline{G}), G \in \mathcal{G} \tag{1b}$$

$$x_u \in \{0,1\} \quad \forall u \in V(\mathcal{G}). \tag{1c}$$

Formulation (1) is a conjunction of the cut-like formulation of the maximum k-club problem introduced by Salemi and Buchanan [22], across all the graphs in the collection. Henceforth, we refer to formulation (1) as the conjunctive cut-like formulation (CCF). It is readily verified that the CCF is a correct formulation in the sense that x is an incidence vector of a cross-graph k-club if and only if it is feasible to the CCF.

Formulation (1) can be strengthened by noting that if a node w that belongs to some minimal length-k u, v-separator of graph $G_i \in \mathcal{G}$ (*i.e.*, $w \in S \in \mathcal{S}_{G_i}(u,v)$) is also at a distance strictly greater that k from either u or v in some other graph G_j in the collection, then w cannot be in a cross-graph k-club that contains both u and v. Consequently, constraints (1b) can be replaced by

$$x_u + x_v - x(S \cap D_{uv}) \le 1, \tag{2}$$

where,

$$D_{uv} \coloneqq \big\{w \in V(\mathcal{G}) \setminus \{u,v\} : \text{dist}_G(u,w) \le k \text{ and } \text{dist}_G(v,w) \le k\ \forall G \in \mathcal{G}\big\} \tag{3}$$

is the set of nodes that are at a distance of at most k from u and v in all the graphs in $\mathcal{G}$. The validity of constraints (2) follows from the observation that if $x_u = x_v = 1$, then $x(S \setminus D_{uv}) = 0$ as no nodes from the set $S \setminus D_{uv}$ can be included in a cross-graph k-club containing u and v. Alternately, we can think of $S \cap D_{uv}$ as further minimalizing the separator S by removing nodes that are not on any path of length at most k between u and v, in some graph in the collection. Observe that the resulting formulation is at least as tight as the CCF. Moreover, there are instances where $x(S \cap D_{uv}) < x(S)$ for at least one separator $S \in \mathcal{S}_G(u,v)$, as illustrated next.

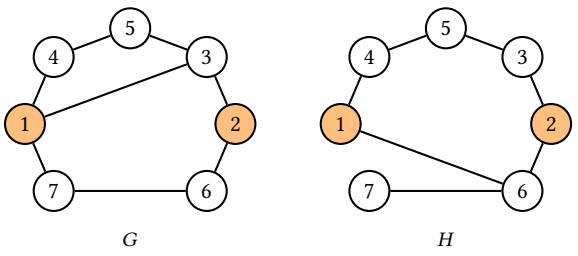


Figure 2: Inequality $x_1 + x_2 \le 1$ is valid for the problem on $\mathcal{G} = \{G, H\}$ when $k = 2$.

Consider the maximum 2-graph 2-club problem on the graph collection in Figure 2. Formulation (1), for node pair 1 and 2, includes the constraints $x_1 + x_2 - x_3 \le 1$ due to G and $x_1 + x_2 - x_6 \le 1$ due to H. However, we can tighten the first constraint by intersecting the separator $\{3\}$ with $D_{1,2} = \{5, 6, 7\}$ to obtain the constraint $x_1 + x_2 \le 1$ that dominates both previous constraints; note that $\text{dist}_H(1,3) = 3$.

4.2 A conjunctive formulation based on pairwise peeling

Based on the idea of intersecting the length-k u, v-separator S in constraint (1b) with D_{uv} to obtain a tighter constraint (2), we can envision an approach in which we further tighten the constraints with respect to each u, v pair, by *recursively* deleting nodes which are too far away from either u or v in any graph in the collection. As the deletion of nodes tends to have a domino effect on pairwise distances in graphs, leading to more nodes meeting the condition for deletion, the resulting inequalities will be at least as strong as their counterpart in constraints (2). However, it is important to recognize that this operation is node pair specific, *i.e.*, the graph collection obtained by deleting nodes based on a particular u, v pair is only valid for generating constraints with respect to that pair. This is because nodes deleted based on u and v might be within distance k of a different node pair.

To illustrate this idea, consider the maximum 2-graph 3-club problem on the graph collection in Figure 3. Constraints (2) are listed below for the node pair 1 and 6, for graphs G and H, by noting that $D_{1,6} = \{3, 4, 5\}$, $\mathcal{S}_G(1,6) = \big\{\{2,3\}, \{2,5\}, \{3,4\}, \{4,5\}\big\}$, and $\mathcal{S}_H(1,6) = \big\{\{3\}, \{4\}\big\}$.

$$\begin{aligned} x_1 + x_6 - x_3 &\le 1 \\ x_1 + x_6 - x_5 &\le 1 \\ x_1 + x_6 - x_3 - x_4 &\le 1 \\ x_1 + x_6 - x_4 - x_5 &\le 1 \\ x_1 + x_6 - x_3 &\le 1 \\ x_1 + x_6 - x_4 &\le 1 \end{aligned}$$

However, the inequality $x_1 + x_6 \le 1$ that can replace all of the foregoing constraints for the node pair 1 and 6 can be derived as follows: observe that $\text{dist}_H(2,6) = 4 > 3$, thus if we want to simultaneously include nodes 1 and 6 in a 2-graph 3-club, then we cannot include node 2 and it can be deleted from G and H. Then, the $\text{dist}_{G\setminus\{2\}}(1,4) = 4 > 3$, and consequently we cannot include node 4 either. Upon deleting nodes 2 and 4 from G and H, we find that nodes 1 and 6 are disconnected in H; so, $x_1 + x_6 \le 1$ is valid.

Algorithm 1 formalizes the idea illustrated by the foregoing example to generate tighter constraints, and we refer to it as the *pairwise peeling algorithm*. We denote by $\mathcal{J}$ the node pairs that are nonadjacent in some graph in the collection $\mathcal{G}$, *i.e.*,

$$\mathcal{J} \coloneqq \Big\{\{u,v\} \subset V(\mathcal{G}) : u \ne v, uv \in E(\overline{G}) \text{ for some } G \in \mathcal{G}\Big\}.$$

The algorithm takes a graph collection $\mathcal{G}$, a positive integer k, and a node pair $uv \in \mathcal{J}$ as input, and creates an auxiliary graph collection $\mathcal{G}_{uv}$ by recursively deleting from every graph in the collection, nodes that are more than distance k from either u or v in some graph in the collection. The constraints for the node pair u and v

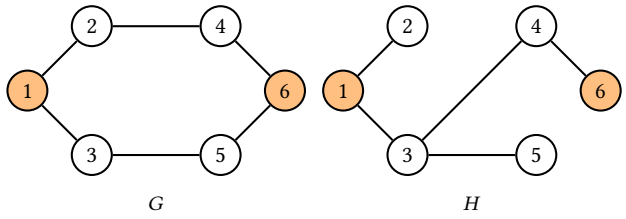


Figure 3: Inequality $x_1 + x_6 \leq 1$ is valid for the problem on $\{G, H\}$ when $k = 3$.

can then be generated based on the minimal separators of graphs in this auxiliary collection $\mathcal{G}_{uv}$. Thus, we can replace constraints (1b) by the following based on the pairwise peeled collection:

$$x_u + x_v - x(S) \leq 1 \quad \forall S \in \mathcal{S}_G(u,v) \text{ and } \forall G \in \mathcal{G}_{uv} \text{ such that } uv \in E(\overline{G}), uv \in \mathcal{J}. \quad (4)$$

In Algorithm 1, the do-while loop executes at most $|V(\mathcal{G})| + 1$ times, as we delete at least one node in each iteration. In one execution of the do-while loop, the nested for-loops execute $O(p|V(\mathcal{G})|)$ times. Computing distances from a single node w and deleting w if necessary, can be completed in $O(|V(G)| + |E(G)|)$. Although the worst-case complexity is unattractive, we did not find this algorithm to be time-consuming when compared to the impact it has on overall running time in our computational experiments.

Algorithm 1: Pairwise Peeling

Input: $\mathcal{G}, k, uv \in \mathcal{J}$
Output: $\mathcal{G}_{uv}$

```
do
    W ← ∅
    for G ∈ 𝒢 do
        for w ∈ V(𝒢) \ (W ∪ {u, v}) do
            if dist_G(u, w) > k or dist_G(v, w) > k then
                W ← W ∪ {w}
                delete w from every graph in 𝒢
while W ≠ ∅;
return 𝒢_uv ← 𝒢
```

Proposition 4.1. *If constraints* (1b) *in formulation* (1) *are replaced by constraints* (4)*, the resulting formulation is correct for the maximum p-graph k-club problem.*

The claim follows from the observation that the incidence vector of a p-graph k-club satisfies constraints (4) and every binary vector satisfying these constraints also satisfies constraints (1b).

Proposition 4.2. *The pairwise peeling algorithm will delete the same set of nodes independent of the order in which the graphs in $\mathcal{G}$ are processed by the algorithm.*

Proof. Suppose for a specific $uv \in \mathcal{J}$, $(w_1, w_2, \ldots, w_q)$ is the order in which nodes were deleted using an ordering π of the graphs in $\mathcal{G}$. Then, w_1 is too far from either u or v in some graph in the original collection, and hence, must be deleted by Algorithm 1 using any other ordering of graphs in $\mathcal{G}$. If w_2 was deleted following w_1 when using π, then in any other ordering, after w_1 is deleted, we know that w_2 must be too far from either u or v, and therefore, must also be deleted. By repeating this argument, $\{w_1, w_2, \ldots, w_q\}$ must be deleted under any ordering that is different from π. As π is arbitrary, we can conclude that the final outcome of Algorithm 1 is independent of the order in which graphs in $\mathcal{G}$ are processed. □

Henceforth, we refer to the new formulation as the pairwise peeled cut-like formulation (PPCF). For each $uv \in \mathcal{J}$, constraint (4) is at least as strong as constraint (2) (which in turn dominates constraint (1b)). Through our computational experiments reported in the next section, we assess the gains made by using Algorithm 1 to generate potentially stronger constraints.

5 COMPUTATIONAL EXPERIMENTS

The goal of our computational study is to compare the performance of a general purpose IP solver when using CCF and PPCF to solve the maximum p-graph k-club problem. As either formulation uses exponentially many constraints in the worst case, we generate them in a delayed fashion and implement two decomposition branch-and-cut (BC) algorithms that use the same master problem based on cross-graph k-cliques defined below.

Definition 5.1. A subset of nodes $S \subseteq V(\mathcal{G})$ is called a *p-graph k-clique* if S is a k-clique in each graph in the collection $\mathcal{G}$.

5.1 Overview of the solvers

Like the single-graph counterparts, a cross-graph k-clique is a graph-theoretic relaxation of a cross-graph k-club. The maximum p-graph k-clique problem is equivalent to the classical maximum clique problem on the *power-intersection graph* of $\mathcal{G}$, *i.e.*, the graph with node set $V(\mathcal{G})$ and edge set containing every pair of distinct nodes that are at distance at most k in every graph in the collection. We can then define the complementary edge set $\hat{E}$ containing all the conflicting pairs of nodes as:

$$\hat{E} := \left\{ \{u,v\} \subseteq V(\mathcal{G}) : \text{dist}_G(u,v) > k \text{ in some graph } G \in \mathcal{G} \right\},$$

and we use it in our master IP formulation (5):

$$\max x(V(\mathcal{G})) \quad (5a)$$

$$x_u + x_v \leq 1 \qquad \forall uv \in \hat{E} \quad (5b)$$

$$x_u \in \{0,1\} \qquad \forall u \in V(\mathcal{G}) \quad (5c)$$

The two decomposition BC algorithms, referred to as CCF and PPCF henceforth, would add constraints (1b) and (4), respectively, whenever a feasible solution to the master problem (5) that is not a cross-graph k-club is encountered anywhere in the BC tree. Our implementations are also enhanced by preprocessing (see Algorithm 2) based on a feasible solution S to the problem obtained using the "DROP heuristic" for k-clubs [7, 22] on the *intersection graph* J with node set $V(\mathcal{G})$ and edge set $\bigcap\{E(G) : G \in \mathcal{G}\}$.

Note that a k-club in J is a cross-graph k-club of $\mathcal{G}$. Then, in the power-intersection graph of $\mathcal{G}$ denoted by J^k, if a node u has fewer than $|S|$ neighbors, it cannot belong to a cross-graph k-club larger than S. (Because if it did, node u would have degree at least $|S|$ in

Algorithm 2: Preprocessing

Input: $\mathcal{G}, k$
Output: a preprocessed graph collection $\mathcal{G}$
obtain the intersection graph J of all graphs in $\mathcal{G}$
compute a k-club S of J using DROP heuristic
obtain the power-intersection graph J^k of $\mathcal{G}$
CorePeel($\mathcal{G}, J^k, |S|$)
CommunityPeel($\mathcal{G}, J^k, |S|$)
recursively delete edge uv from *every* graph $G \in \mathcal{G}$ *and* J^k in which it is present, if u and v are in distinct connected components of some graph $G \in \mathcal{G}$ or J^k
return $\mathcal{G}$

J^k). Hence, *core peeling* [1] recursively deletes nodes with degree less than $|S|$ in J^k, and correspondingly from every graph in $\mathcal{G}$.

After core-peeling, as long as $V(J^k)$ is not empty, J^k will be an $|S|$-core as every node that survives will have at least $|S|$ neighbors. Now, a pair of nodes u and v that are adjacent in J^k can belong to a cross-graph k-club larger than S only if they have at least $|S| - 1$ common neighbors in J^k. If not, the edge uv is deleted from J^k and from every graph in the collection where u and v are adjacent in the *community peeling* [27] step.

Graph J^k may contain more connected components after core and community peeling than before. As a result, there may exist an edge $uv \in E(G)$ for some $G \in \mathcal{G}$ whose end points u and v belong to different connected components of J^k. These edges can be removed from every $G \in \mathcal{G}$ containing the edge. Doing so may disconnect a graph $G \in \mathcal{G}$ so that not only u and v belong to different components, but so do some other nodes a and b that are adjacent in J^k; then, we can delete edge ab from J^k. Hence, we can recursively delete edges from every graph in the expanded collection $\mathcal{G} \cup \{J^k\}$ until all of them have identical connected components (in terms of the node subsets inducing the components). The preprocessing steps can be implemented to run in time that is linear or quadratic in the size of the input [22, 27], and their running time is negligible in practice compared to the BC algorithm.

To avoid unnecessary constraints (5b) added for pairs of nodes which reside in different components of J^k, we extend formulation (5) by introducing a binary variable for each of the connected components of J^k and enforce that nodes selected must belong to the same component. In the master problem, only conflict constraints related to a pair of nonadjacent nodes in a same connected component would be enumerated. Similarly, any node pair for which a lazy constraint is generated must also belong to the same connected component of J^k.

5.2 Preliminary results

We report computational experiments conducted on 64-bit Linux® compute nodes with dual Intel® "Skylake" 6130 CPUs with 96 GB RAM. Optimization models are solved using Gurobi™ Optimizer v9.0.1 [10] and the algorithms are implemented in C++. We consider the following parameter values in our experiments: $k \in \{2, 3, 4\}$ and $p \in \{2, 3, 4, 5\}$.

To generate test instances for this preliminary computational study, we generate graphs using the algorithm described in [8] (BG graphs). These graphs are known to be challenging for the (single-graph) maximum k-club problem. We first generate a set of 10 graphs $\{G_1, G_2, \ldots, G_{10}\}$, each with 200 nodes, by specifying an edge density and running BG algorithm 10 times. From our preliminary results, BG_1 (generated with edge density of 0.15%) and BG_4 (generated with edge density of 0.1%) are challenging instances when $k = 2$. When $k = 3$ or 4, BG_2 (generated with edge density of 0.05%) and BG_3 (generated with edge density of 0.025%) are challenging instances. For each p, k pair, we report results averaged over instances obtained from the collection of 10 graphs with the edge density that is most challenging for the chosen value of k.

Each row of Table 1 reports results averaged over $11 - p$ runs with collections $\{G_1, \ldots, G_p\}, \{G_2, \ldots, G_{p+1}\}, \ldots, \{G_{11-p}, \ldots, G_{10}\}$. Except for BG_1 instance when $p = 2$ and $k = 2$, where both approaches did not reach optimality, PPCF based BC takes a significantly shorter running times than CCF for most of the instances. For example, on BG_2 instance when $p = 4$ and $k = 3$, CCF took 1079.11 seconds while PPCF only took 47.11 seconds (over 20 times faster than CCF). Overall, PPCF is over 6 times faster than CCF in terms of average running time on challenging instances.

Table 1: Comparison of CCF and PPCF on BG instances.

			CCF				PPCF				
k	p	**Instance**	obj	time(s)	#LC[a]	#NCT[b]	obj	time(s)	#LC	#SLC[c]	#NCT
2	2	BG_4	9.44	88.01	12116.22	2.26	9.44	54.77	8085.56	6887.00	0.53
2	2	BG_1	≥16.33[d]	7200.23	20353.00	4.32	≥16.22	6973.67	17958.22	1310.33	4.00
2	3	BG_4	4.63	55.22	12564.38	2.27	4.63	38.41	9807.00	9802.25	0.00
2	3	BG_1	7.88	3074.61	24496.13	4.39	7.88	760.30	19702.88	3214.00	3.48
2	4	BG_4	2.43	49.74	11712.43	2.27	2.43	44.59	10442.00	10441.86	0.00
2	4	BG_1	4.43	1982.95	25226.43	4.39	4.43	338.92	20934.71	5719.86	2.94
2	5	BG_4	2.00	40.70	10166.33	2.27	2.00	36.36	9253.67	9253.67	0.00
2	5	BG_1	2.50	1533.07	25121.67	4.41	2.50	201.43	21074.00	8851.50	2.36
3	2	BG_2	12.00	3008.56	47557.00	4.54	12.00	576.44	26125.44	6079.22	3.46
3	3	BG_2	3.13	1748.18	45065.50	4.52	3.13	105.67	21304.25	12592.63	1.91
3	4	BG_2	3.00	1079.11	40636.29	4.55	3.00	47.11	17284.29	15254.00	0.61
3	5	BG_2	3.00	818.45	36544.33	4.47	3.00	40.02	15971.67	15687.17	0.10
4	2	BG_3	8.78	192.51	23791.11	3.60	8.78	40.72	12169.78	10365.56	0.79
4	3	BG_3	4.00	99.02	18502.13	3.37	4.00	31.06	11116.88	10921.00	0.01
4	4	BG_3	4.00	59.55	13925.57	3.21	4.00	30.74	9216.86	8992.71	0.00
4	5	BG_3	4.00	46.07	11001.67	3.16	4.00	29.12	7622.17	7406.50	0.00

[a] Average number of lazy constraints added.
[b] Average number of negative terms in a lazy constraint.
[c] Average number of strengthened lazy constraints added.
[d] On instances not solved to optimality we report the lower-bound provided by the best solution found.

Table 1 also shows the number of lazy constraints added by each algorithm on average. For each instance, we observe fewer lazy constraints used in PPCF than CCF. Together with the fact that PPCF took much less time than CCF for each instance, this observation is indicative of the strength of using constraints (4) over constraints (1b). On average, CCF added over 1.5 times more lazy constraints than PPCF. Column #SLC under PPCF reports the number of strengthened lazy constraints added, *i.e.*, these are strictly constraints (4). On average, over 70% of lazy constraints added in PPCF are of this type. For BG_4 instance when $p = 5$ and $k = 2$, all lazy constraints added in PPCF are of this type.

The columns labeled #NCT in Table 1 report the average number of terms with coefficient -1 on the left hand side of the added lazy constraints. This is another indirect indicator of the strength of the lazy constraints—generally, the smaller this number, the stronger the constraint. For most of the instances, we observed a significantly

smaller value under PPCF than CCF. Note that the value of #NCT is zero for five instances under PPCF. The lazy constraints of this type are actually conflict constraints with no negative terms on the left hand side. The foregoing observations strongly suggest that PPCF approach based on pairwise peeling constraints significantly improves our ability to find maximum p-graph k-clubs.

Figure 4 shows the performance profiles [9] based on running times of CCF and PPCF algorithms for solving the maximum p-graph k-club problem on the challenging instances. The red (dash-dotted) and blue (dotted) lines plot the fraction $f_i(\tau)$ of instances solved within a factor τ of the shortest running time (over all values of parameters p, k, and all challenging instances) by solvers i = CCF and PPCF. The profiles clearly suggest that the computational benefits of using lazy constraints strengthened by pairwise peeling is quite significant.

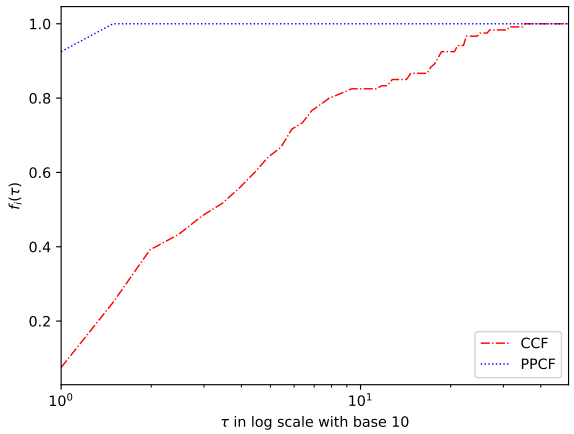


Figure 4: Performance profiles of CCF and PPCF algorithms.

6 CONCLUDING REMARKS

In this paper, we consider cross-graph k-clubs for modeling low-diameter clusters that are conserved in a collection of graphs. We introduce a pairwise peeling approach designed to strengthen constraints in a straightforward conjunction of a known formulation. This approach helps with scale-reduction and integrates information across graphs in the collection to produce tighter constraints. These claims are supported by the empirical evidence obtained from our preliminary experiments. The decomposition branch-and-cut algorithm based on this approach seems to be promising, and capable of handling moderately large instances and graph collections.

ACKNOWLEDGEMENT

The computing for this project was performed at the High Performance Computing Center at Oklahoma State University supported in part through the National Science Foundation grant OAC-1531128. The research of the third author was partially supported by the Office of Naval Research under Grant N00014-19-1-2329.

REFERENCES

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende. 1999. On maximum clique problems in very large graphs. In *External memory algorithms and visualization*, J. Abello and J. Vitter (Eds.). DIMACS Series on Discrete Mathematics and Theoretical Computer Science, Vol. 50. American Mathematical Society, Boston, MA, USA, 119–130.
[2] R. D. Alba. 1973. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology* 3, 1 (1973), 113–126.
[3] R. Alhajj and J. Rokne (Eds.). 2018. *Encyclopedia of Social Network Analysis and Mining*. Springer, New York.
[4] M. T. Almeida and F. D. Carvalho. 2012. Integer models and upper bounds for the 3-club problem. *Networks* 60 (2012), 155–166. Issue 3.
[5] B. Balasundaram, S. Butenko, and I. V. Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (January-February 2011), 133–142.
[6] B. Balasundaram, S. Butenko, and S. Trukhanov. 2005. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization* 10, 1 (August 2005), 23–39.
[7] J.-M. Bourjolly, G. Laporte, and G. Pesant. 2000. Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research* 27 (2000), 559–569.
[8] J.-M. Bourjolly, G. Laporte, and G. Pesant. 2002. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal of Operational Research* 138 (2002), 21–28.
[9] E. D. Dolan and J. J. Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical programming* 91, 2 (2002), 201–213.
[10] Gurobi Optimization, LLC. 2020. Gurobi Optimizer Reference Manual. http://www.gurobi.com
[11] D. Jiang and J. Pei. 2009. Mining Frequent Cross-Graph Quasi-Cliques. *ACM Transactions on Knowledge Discovery in Data* 2, 4 (2009), 16:1–42.
[12] B. H. Junker and F. Schreiber (Eds.). 2008. *Analysis of Biological Networks*. Wiley, New York.
[13] Y. Lu, E. Moradi, and B. Balasundaram. 2018. Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters* 12, 8 (November 2018), 1959–1969.
[14] R. D. Luce. 1950. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* 15, 2 (1950), 169–190.
[15] Z. Miao and B. Balasundaram. 2020. An Ellipsoidal Bounding Scheme for the Quasi-Clique Number of a Graph. *INFORMS Journal on Computing* 32, 3 (2020), 763–778. Codes/instances online at: https://github.com/baski363/Quasi-clique.
[16] R. J. Mokken. 1979. Cliques, Clubs and Clans. *Quality and Quantity* 13, 2 (1979), 161–173.
[17] E. Moradi and B. Balasundaram. 2018. Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters* 12, 8 (November 2018), 1947–1957.
[18] G Pastukhov, A. Veremyev, V. Boginski, and O. A. Prokopyev. 2018. On maximum degree-based-quasi-clique problem: Complexity and exact approaches. *Networks* 71, 2 (2018), 136–152.
[19] J. Pattillo, N. Youssef, and S. Butenko. 2013. On clique relaxation models in network analysis. *European Journal of Operational Research* 226, 1 (2013), 9–18.
[20] J. Pei, D. Jiang, and A. Zhang. 2005. Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE, 353 – 356.
[21] J. Pei, D. Jiang, and A. Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05)*. ACM, New York, NY, USA, 228–238.
[22] H. Salemi and A. Buchanan. 2020. Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation* 12, 3 (2020), 493–528.
[23] S. Shahinpour and S. Butenko. 2013. Distance-based clique relaxations in networks: s-clique and s-club. In *Models, Algorithms, and Technologies for Network Analysis*, B. I. Goldengorin, V. A. Kalyagin, and P. M. Pardalos (Eds.). Vol. 59. Springer, New York, 149–174.
[24] K. Sim, G. Liu, V. Gopalkrishnan, and J. Li. 2011. A case study on financial ratios via cross-graph quasi-bicliques. *Information Sciences* 181, 1 (2011), 201–216.
[25] S. Trukhanov, C. Balasubramaniam, B. Balasundaram, and S. Butenko. 2013. Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Computational Optimization and Applications* 56, 1 (September 2013), 113–130. https://doi.org/10.1007/s10589-013-9548-5
[26] A. Veremyev and V. Boginski. 2012. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research* 218, 2 (2012), 316–326.
[27] A. Verma, A. Buchanan, and S. Butenko. 2015. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing* 27, 1 (2015), 164–177.

Session Session 4A: spanning trees

Thursday 9 June 2022, 13:30-15:10

Lecture Hall I

The Minimum Spanning Tree Problem under Explorable Uncertainty

Corinna Mathwieser[1] and Eranda Çela[2]

[1]Lehr- und Forschungsgebiet Kombinatorische Optimierung, RWTH Aachen University, Aachen, Germany, ✉ mathwieser@combi.rwth-aachen.de
[2]Department of Discrete Mathematics, Graz University of Technology, Graz, Austria, ✉ cela@math.tugraz.at

18 November 2021

Many real world problems do not allow to work with precise data. Instead, parts of the input are uncertain or only known approximately. Different approaches to deal with uncertainty include stochastic optimization, where the input data is known to follow a specific probability distribution, robust optimization which aims to find good solutions for all possible inputs and explorable uncertainty. In the setting of explorable or queryable uncertainty which was introduced by Kahan [4], it is possible to obtain more precise or even exact data upon request, e.g. by making further measurements or performing a user survey. However, any such query causes additional exploration cost, say time, money or other resources needed to carry out measurements.

This talk deals with the *Minimum Spanning Tree Problem under Explorable Uncertainty* ($MST\text{-}U$), which was first introduced by Erlebach et al. [3]. In an instance of $MST\text{-}U$, each edge is equipped with an uncertainty set and a query cost. The uncertainty set of an edge is an open interval which is guaranteed to contain the edge's weight or a singleton set containing only the edge's weight. In the latter case we refer to an edge as trivial. An edge query (we also say update) reveals the edge's true weight. A set of queries which allows to compute a minimum spanning tree with certainty is called feasible query set. The goal is to find a feasible query set of minimum query cost. The queries may be chosen adaptively, i.e. we are allowed to choose the next update based on the previous outcomes of edge queries. In general, we consider two types of algorithms: deterministic algorithms and randomized algorithms. An algorithm's performance is measured in terms of its competitive ratio, i.e. the ratio between the query cost of the algorithm's solution and the optimal query cost.

Megow et al. [5] introduce a general algorithm framework, based on which they provide a deterministic algorithm with performance ratio 2 which was shown by [3] to be the smallest possible competitive ratio for deterministic algorithms. Moreover, they introduce the randomized algorithm $RANDOM$ which relies on the same framework and achieves a performance ratio of $1 + \frac{1}{\sqrt{2}}$. The best known bound for randomized algorithms is 1.5 which was observed by Erlebach and Hoffmann in [2] and remains true even on triangles. We consider the special case of cactus graphs for which we introduce the randomized algorithm $RANDOM_C$ and show that $RANDOM_C$ achieves a competitive ratio of 1.5 for the specified instance type.

Erlebach et al. [3] also introduce a different problem related to $MST\text{-}U$, namely the *Minimum Spanning Tree Problem under Vertex Uncertainty* ($V\text{-}MST\text{-}U$), where vertices are points in the plane with uncertain locations and the edge weights correspond to the distances of the respective end vertices. So far, $V\text{-}MST\text{-}U$ has only been considered in the deterministic setting and for uniform query costs, i.e. instances where all vertices have identical query costs. We prove that no randomized algorithm can have a performance guarantee better than 2.5. Moreover, we deal with the question of transferability of results concerning $MST\text{-}U$ to the vertex uncertainty setting, especially with respect to general query costs and randomized algorithms. We discuss several fundamental differences between $MST\text{-}U$ and $V\text{-}MST\text{-}U$ which impede the adaption of algorithms in a straightforward way. For the special case of instances where no two cycles share a non-trivial vertex however, we introduce the algorithm $V\text{-}RANDOM_C$ and

show that $V\text{-}RANDOM_C$ achieves a competitive ratio of at most 3.

Note that even for an omniscient solver who knows all outputs of edge queries a priori, finding an optimal query set is not a task which allows for an immediate obvious solution. The problem of computing an optimal query set if the uncertainty sets as well as the exact edge weights are given is called *verification problem* for $MST\text{-}U$ and was studied by Erlebach and Hoffmann in [1]. They establish a connection between $MST\text{-}U$ and the Minimum Bipartite Vertex Cover Problem which they use to show that the verification problem for $MST\text{-}U$ with uniform query costs is solvable in polynomial time, while the verification problem for the vertex uncertainty problem $V\text{-}MST\text{-}U$ is NP-hard. We show that the verification problem for $MST\text{-}U$ with general query costs can be solved in polynomial time by adapting the algorithm of [1] to the non-uniform case.

Even though deterministic and randomized algorithms with good performance guarantee are known for $MST\text{-}U$, it turns out that such results can no longer be achieved once we consider more specific trees instead of general spanning trees. More precisely, we introduce the *Minimum Spanning Star Problem under Explorable Uncertainty* ($MSS\text{-}U$). $MSS\text{-}U$ is defined analogously to $MST\text{-}U$ except that we are now aiming to find queries which allow the identification of a spanning star of minimum weight instead of a general minimum spanning tree. For $MSS\text{-}U$ we derive a negative result with respect to competitive analysis, i.e we show that no algorithm for $MSS\text{-}U$ can achieve a constant competitive ratio.

References

[1] T. Erlebach and M. Hoffmann. Minimum spanning tree verification under uncertainty. In *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 164–175, 2014.

[2] T. Erlebach and M. Hoffmann. Query-competitive algorithms for computing with uncertainty. *Bulletin of the European Association for Theoretical Computer Science*, 116:n.pag, 2015.

[3] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *Proceedings of Symposium on Theoretical Aspects of Computer Science*, pages 277–288, 2008.

[4] S. Kahan. A model for data in motion. *23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 267–277, 1991.

[5] N. Megow, J. Meißner, and M. Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017.

Learning to Prune Instances of Steiner Tree Problem in Graphs

Jiwei Zhang[1] and Deepak Ajwani[2]

[1]University College Dublin, Dublin, Ireland, ✉ jiwei.zhang@ucdconnect.ie
[2]School of Computer Science, University College Dublin, Dublin, Ireland, ✉ deepak.ajwani@ucd.ie

1 Introduction

In the Steiner Tree Problem, we are given a set of nodes in a graph and the goal is to find a tree sub-graph of minimum weight that contains all nodes in the given set (possibly including additional nodes). This is an NP-hard problem and as such, solving large instances of this problem optimally (or close to optimally) remains a computational challenge.

In recent years, a machine learning approach called learning-to-prune has been successfully used in a range of combinatorial optimization problems such as maximum clique enumeration [1], k-median [2], travelling salesperson problem [3] etc. The idea behind this approach is to train a learning model that can confidently predict the values of a large number (but not all) of variables based on features derived from algorithmic and optimization literature. The training is done on smaller instances generated from the same distribution as the larger input. Fixing a large number of variables reduces the search space significantly and the remaining problem can be solved using mixed integer linear programming (MILP) solvers or other exact algorithms.

In this paper, we show that a careful application of the learning-to-prune approach enables us to reduce the running time of a commercial MILP solver on Steiner tree problem instances of I160 dataset from SteinLib [5] with only a marginal loss (less than 1%) in the objective function value of the solution.

2 Methodology

In the context of Steiner tree problem, the learning-to-prune approach predicts which edges can be removed from the graph such that the solution on the remaining graph still remains quite close to optimal.

Feature	Importance
LP	0.461665
Normalized Weight	0.108099
Variance	0.082869
Degree Centrality Max	0.052033
Eigenvector Centrality Max	0.047566
Betweenness Centrality Max	0.047523
Degree Centrality Min	0.045958
Local Rank j	0.041085
Eigenvector Centrality Min	0.039031
Betweenness Centrality Min	0.038027
Local Rank i	0.036145

Table 1: Feature Importance

In applying the learning-to-prune framework, our first challenge is to find a good Integer Linear Programming (ILP) formulation of the problem. We use the formulation from Wong [4]. In this formulation, the problem is considered as a multi-commodity flow problem with polynomial number of constraints and thus, its linear programming (LP) relaxation can be solved quite fast even for larger instances of the problem. The LP values of the variables can be used as a highly discriminative feature in the learning-to-prune approach.

Apart from the LP values (**F1**), the other features that we use in our learning-to-prune model include **(F2)** normalized edge weight, **(F3)** variance of normalized edge weight, **(F4, F5)** normalized local rank of the edge at the two incident nodes, **(F6, F7)** min and max degree centrality of the two incident nodes, **(F8, F9)** min and max betweenness centrality of the two incident nodes and **(F10, F11)** min and max Eigenvector centrality of the two incident nodes.

In terms of the classification techniques, we used Support Vector Machine (SVM) for the prediction task; the results are similar with other models such as Random forests, Logistic Regression, Gaussian

	Objective	Original ILP Time	Feature Computation Time	Objective (Hard Pruning)	ILP Runtime (After Pruning)	Objective Increase %	Runtime Decrease %
i160-344	8307.0	27245.214428	54.171017	8324.0	157.707098	0.204647	99.222329
i160-244	5076.0	7762.750552	25.680089	5103.0	47.128251	0.531915	99.062081
i160-345	8327.0	70653.843429	51.934421	8327.0	242.822062	0.000000	99.582816
i160-343	8275.0	20897.368129	50.549097	8275.0	114.900505	0.000000	99.208275
i160-342	8348.0	91351.384691	60.099658	8355.0	1384.392882	0.083852	98.418751
i160-313	9159.0	3832.540745	15.259630	9159.0	84.731579	0.000000	97.390994
i160-241	5086.0	6446.484885	24.782818	5086.0	32.778224	0.000000	99.107094
i160-341	8331.0	52473.687999	53.653058	8331.0	104.738673	0.000000	99.698150
i160-245	5084.0	3014.052865	28.052475	5084.0	15.946626	0.000000	98.540201
i160-242	5106.0	4817.800565	27.340318	5106.0	42.810647	0.000000	98.543921

Table 2: Evaluation by Hard Pruning

Naive Bayes. We observe that in this classification model, LP values are found to be very important (Table 1), but the prediction is not solely reliant on this feature.

For the edges that are predicted to not be in the optimal solution by our classifier and whose LP value is also zero, we can prune them out from further consideration (hard pruning) or constrain that atmost a small number of edges can be taken from that set (soft pruning).

3 Evaluation

In the benchmark SteinLib [5] dataset, we found that there are only 55 problem instances for which the LP relaxation of the considered ILP formulation does not return integral solutions. Thus, we only focus on these instances and select 80% of them with the smallest running times as training and use the remaining 20% of the instances with the largest running time as the test dataset. This is because we want to show that our model generalizes from smaller instances to larger and more complex instances in this dataset.

Table 2 presents the results of the learning-to-prune approach on the 10 test instances. We first note that on these larger and more complex instances, the time to compute all the features including the LP values is very small compared to the time to run the original ILP. More importantly, the running time of the learning-to-prune approach (including the time to compute features and then running the ILP with hard pruning constraint) is around 99% smaller than the original ILP time on these instances. In 7 of these 10 instances, the hard pruning is able to find the optimal solution. In the remaining three instances, the resultant increase in the objective function value because of the mistakes in the pruning process is still very small (less than 0.6%).

In applications where we wish to reduce the optimality gap even further, we can use the soft pruning approach. When applied on the instance i160-344, the soft pruning approach that allows just one edge from the pruned set finds the optimal solution of the original problem. The running time of this approach on this instance is around 3000 seconds, which is still considerably less than the original ILP time of around 27000 seconds, but more than the time of the hard pruning approach (around 150 seconds).

References

[1] J. Lauri and S. Dutta. "Fine-grained search space classification for hard enumeration variants of subset problems." AAAI 2019: Vol.33. No.01

[2] D. Tayebi, S. Ray and D. Ajwani. "Learning to Sparsify instance of k-median and related problems." ALENEX 2022

[3] J. Fitzpatrick, D. Ajwani and P. Carroll. "Learning to Sparsify Travelling Salesman Problem Instances." CPAIOR 2021: 410-426.

[4] R. T. Wong. "A dual ascent approach for Steiner tree problems on a directed graph." Mathematical programming 28.3 (1984): 271-287.

[5] T. Koch, A. Martin and S. Voß, "SteinLib: An Updated Library on Steiner Tree Problems in Graphs", http://elib.zib.de/steinlib

Lagrangian Relaxations of Constrained Spanning Forests

Sávio S. Dias[1] and Luidi G. Simonetti[2]

[1]PESC/Coppe - Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, ✉ sdias@cos.ufrj.br
[2]PESC/Coppe - Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, ✉ luidi@cos.ufrj.br

Let $G = (V, E)$ be an undirected graph, $w \in \mathbb{R}_+^{|E|}$ be a vector of edge weights in G, and a positive integer $2 \leq k \leq n$, where $n = |V|$. The Minimum-Weight Constrained Spanning Forest Problem (CFP, for short) aims to find a minimum cost spanning forest $F = (V, E_F)$, $E_F \subseteq E$, such that each connected component is a tree spanning at least k vertices. In the literature, F is often called a k-capacitated forest or a k-forest. The CFP is $\mathcal{NP}$-hard for $3 \leq k \leq \lfloor \frac{n}{2} \rfloor$ [5], since if $k = 2$, the problem is reduced to minimum-weight edge covering, and if $k \geq \lceil \frac{n}{2} \rceil$, the problem is reduced to the minimum spanning tree.

The CFP is a graph partition problem with minimum size requirements, where each partition weights its minimum spanning tree. It is closely related to a series of tree problems and graph partition problems abroad the literature, and has applications in telecommunications [4]; political districting [8]; data micro aggregation [7]; and wiring optimization of the robotic skin design [2].

Ali and Huang [1] studied a variation where the amount of trees is an input parameter, and the trees must not differ in vertices by more than one. Guttmann-Beck and Hassin [4] presented another similar problem, but in this case, the number of trees and the number of vertices for each tree are input data. Also, the Capacitated Minimum Spanning Tree Problem [3] aims at finding a minimum spanning tree such that each subtree from a root has capacity at most Q. There is also an online version of the CFP with different weight vector for every planning period [9]. Note that, in some of those problems, the adaptation of our approaches is straightforward, needing adjustments on existing constraints or new ones. Also, for the Online CFP, we may run our algorithms for every planning period.

In this work, we review two models from the literature to identify the current state-of-the-art. For that, we present a dominance study and computational results. Also, we introduce a reduction test, used as preprocessing, for cutting out suboptimal edges from the input graph. Consider Ji [6] model as $\mathcal{P}_1$, using a set of decision variables $x \in \mathbb{B}^{|E|}$ for edge selection of the input graph. Still, let $S \subseteq V$ be a subset of vertices, $\delta(S) \subset E$ be the subset of edges with exactly one endpoint in S, $E(S) \subseteq E$ be the subset of edges with both endpoints in S, $\varphi(S) = \delta(S) \cup E(S)$, and $x[E_P] = \sum_{e \in E_P} x_e$ be the sum of variables implied by the edge subset $E_P \subseteq E$.

$$(\mathcal{P}_1)\ \min \quad w^T x \tag{1}$$

$$\text{s.t.:} \quad x[E(S)] \leq |S| - 1, \qquad \forall S \subseteq V \tag{2}$$

$$x[\varphi(S)] \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq V, |S| \geq 2 \tag{3}$$

$$x[\delta(v)] \geq 1, \qquad \forall v \in V \tag{4}$$

$$x[E(V)] \geq |V| - \left\lfloor \frac{|V|}{k} \right\rfloor \tag{5}$$

In $\mathcal{P}_1$, the objective function aims to minimize the weight of the selected edges. Subcycle Elimination Constraints (SECs) (2) were used to ensure an acyclical graph. Then, inequalities (3) impose a lower bound on the number of edges that a k-forest with $|S|$ vertices must have. Finally, inequalities (4)-(5) are specific cases of previous inequalities. They cut out isolated vertices (4), and limit the minimum number of edges that the forest must have (5). The sole purpose of those inequalities is to serve as a starting bound in the cutting plane loop.

As another way of exploring the possible bounds for the problem, we present two Lagrangian Relaxations using sets of exponential inequalities. This method is particularly interesting, since it provides useful information as preprocessing to other methods, such as the Branch-and-Cut (B&C). Such information includes the fixation/discarding of variables, an initial pool of cuts, and starting (primal and dual) bounds.

Consider the problem of dualizing the exponentially many inequalities (3) from $\mathcal{P}_1$ in a Lagrangian fashion. Now, let $\mu \in \mathbb{R}_+^{|\mathcal{S}|}$ be the dual variables associated with those inequalities, also called lagrangian multipliers, where $\mathcal{S} = \{S \subseteq V, |S| \geq 2\}$. Although this Lagrangian Relaxation Problem (LRP) represents a valid lower bound over the CFP, to the best of our knowledge, its solution is not well defined in polynomial algorithmic terms. For that very reason, we considered two well-defined LRPs:

$$\begin{aligned} \mathcal{L}_1(\mu) = \min \quad & \sum_{e \in E} \left(w_e - \sum_{S \in \mathcal{S}: e \in \varphi(S)} \mu_S \right) x_e + \sum_{S \in \mathcal{S}} \mu_S \left(|S| - \left\lfloor \frac{|S|}{k} \right\rfloor \right) \\ \text{s.t.:} \quad & \text{(2) and (4)} \end{aligned} \tag{6}$$

where $\mathcal{L}_1(\mu)$ is the minimum-weight forest cover problem, which can be optimally solved in strongly polynomial-time using the algorithm described in White [10]. Thus, the best possible bound obtainable from $\mathcal{L}_1(\mu)$ is given by $\mathcal{L}_1^*(\mu) = \max_{\mu \in \mathbb{R}_+^{|\mathcal{S}|}} \{\mathcal{L}_1(\mu)\}$. And,

$$\begin{aligned} \mathcal{L}_2(\mu, \pi) = \min \quad & \sum_{e \in E} \left(w_e - \sum_{S \in \mathcal{S}: e \in \varphi(S)} \mu_S - \pi_u - \pi_v \right) x_e + \sum_{S \in \mathcal{S}} \mu_S \left(|S| - \left\lfloor \frac{|S|}{k} \right\rfloor \right) + \sum_{v \in V} \pi_v \\ \text{s.t.:} \quad & \text{(2) and (5)} \end{aligned} \tag{7}$$

where $\pi \in \mathbb{R}_+^{|V|}$ are the lagrangian multipliers associated with inequalities (4). Then, $\mathcal{L}_2(\mu, \pi)$ models a minimum-weighted spanning forest problem, which is solved through Kruskal's algorithm, and has the best possible bound given by $\mathcal{L}_2^*(\mu, \pi) = \max_{\mu \in \mathbb{R}_+^{|\mathcal{S}|}, \pi \in \mathbb{R}_+^{|V|}} \{\mathcal{L}_2(\mu, \pi)\}$.

Both $\mathcal{L}_1^*(\mu)$ and $\mathcal{L}_2^*(\mu, \pi)$ are solved using the Subgradient Method (SM), with some adaptations to handle the many dualized exponential inequalities. Then, since convergence of the pure SM is limited, we defined stop criteria in order to use the good-quality information obtained as a warm-start to the B&C algorithm. Our results show promising performance when compared to the previous B&C in Ji [6], while also providing a way to tackle larger instances where even solve the linear relaxation is hard.

References

[1] Agha Iqbal Ali and Chung-Hsing Huang. Balanced spanning forests and trees. *Networks*, 21(6): 667–687, 1991.

[2] Davide Anghinolfi, Giorgio Cannata, Fulvio Mastrogiovanni, Cristiano Nattero, and Massimo Paolucci. Application and experimental validation of pheromone design in ant colony optimization: the problem of robot skin wiring. *Applied Artificial Intelligence*, 28(3):292–321, 2014.

[3] Bezalel Gavish. Formulations and algorithms for the capacitated minimal directed tree problem. *Journal of the ACM (JACM)*, 30(1):118–132, 1983.

[4] Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for minimum tree partition. *Discrete Applied Mathematics*, 87(1-3):117–137, 1998.

[5] Celina Imielińska, Bahman Kalantari, and Leonid Khachiyan. A greedy heuristic for a minimum-weight forest problem. *Operations Research Letters*, 14(2):65–71, 1993.

[6] Xiaoyun Ji. Graph partition problems with minimum size constraints. *Rensselaer Polytechnic Institute, Troy, New York*, 2004.

[7] Michael Laszlo and Sumitra Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge & Data Engineering*, 17(7):902–911, 2005.

[8] Anuj Mehrotra, Ellis L Johnson, and George L Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1114, 1998.

[9] Jiawei Qian and David P Williamson. An o (logn)-competitive algorithm for online constrained forest problems. In *International Colloquium on Automata, Languages, and Programming*, pages 37–48. Springer, 2011.

[10] Lee J White. Minimum covers of fixed cardinality in weighted graphs. *SIAM Journal on Applied Mathematics*, 21(1):104–113, 1971.

Towards Stronger Lagrangean Bounds for Stable Spanning Trees

Phillippe Samer
University of Bergen
Bergen, Norway
samer@uib.no

Dag Haugland
University of Bergen
Bergen, Norway
dag@ii.uib.no

ABSTRACT

Given a graph $G = (V, E)$ and a set C of unordered pairs of edges regarded as being in conflict, a stable spanning tree in G is a set of edges T inducing a spanning tree in G, such that for each $\{e_i, e_j\} \in C$, at most one of the edges e_i and e_j is in T. The existing work on Lagrangean algorithms to the NP-hard problem of finding minimum weight stable spanning trees is limited to relaxations with the integrality property. We have recently initiated the combinatorial and polyhedral study of fixed cardinality stable sets [17], which motivates a new formulation for stable spanning trees based on Lagrangean Decomposition. By optimizing over the spanning tree polytope of G and the fixed cardinality stable set polytope of the conflict graph $\hat{G} = (E, C)$ in the subproblems, we are able to determine stronger Lagrangean bounds (equivalent to dualizing exponentially-many subtour elimination constraints), while limiting the number of multipliers in the dual problem to $|E|$. This naturally asks for more sophisticated dual algorithms, requiring the fewest iterations possible, and we derive a collection of Lagrangean dual ascent directions to this end.

KEYWORDS

Stable spanning trees, conflict-free spanning trees, Lagrangean decomposition, dual ascent, fixed cardinality stable sets.

1 INTRODUCTION

Given an undirected graph $G = (V, E)$, with edge weights $w : E \to \mathbb{Q}$, and a family C of unordered pairs of edges that are regarded as being in conflict, a stable (or conflict-free) spanning tree in G is a set of edges T inducing a spanning tree in G, such that for each $\{e_i, e_j\} \in C$, at most one of the edges e_i and e_j is in T. The minimum spanning tree under conflict constraints (MSTCC) problem is to determine a stable spanning tree of least weight, or decide that none exists. It was introduced by [8, 9], who also prove its NP-hardness.

Different combinatorial and algorithmic results about stable spanning trees explore the associated conflict graph $\hat{G} = (E, C)$, which has a vertex corresponding to each edge in the original graph G, and where we represent each conflict constraint by an edge connecting the corresponding vertices in $\hat{G}$. Note that each conflict-free spanning tree in G is a subset of E which corresponds both to a spanning tree in G and to a stable set (or independent set, or co-clique: a subset of pairwise non-adjacent vertices) in $\hat{G}$. Therefore, one can equivalently search for stable sets in $\hat{G}$ of cardinality exactly $|V| - 1$ which do not induce cycles in the original graph G.

We have recently initiated the combinatorial study of stable sets of cardinality exactly k in a graph [17], where k is a positive integer given as part of the input. There are appealing research directions around algorithms, combinatorics and optimization for problems defined over fixed cardinality stable sets. Also from an applications perspective, conflict constraints arise naturally in operations research and management science. Stable spanning trees, in particular, model real-world settings such as communication networks with different link technologies (which might be mutually exclusive in some cases), and utilities distribution networks. In fact, the latter is a standard application of the quadratic minimum spanning tree problem [1], which generalizes the MSTCC one.

Exact algorithms to find stable spanning trees have been investigated for a decade now, building on branch-and-cut [6, 18], or Lagrangean relaxation [7, 20] strategies. Consider the natural integer programming (IP) formulation for the MSTCC problem:

$$\min \sum_{e \in E} w_e x_e \tag{1}$$

$$\text{s.t.} \sum_{e \in E(S)} x_e \leq |S| - 1, \quad \text{for each } S \subsetneq V, S \neq \emptyset, \tag{2}$$

$$\sum_{e \in E} x_e = |V| - 1, \tag{3}$$

$$x_{e_i} + x_{e_j} \leq 1, \qquad \text{for each } \{e_i, e_j\} \in C, \tag{4}$$

$$x_e \in \{0, 1\}, \qquad \text{for each } e \in E. \tag{5}$$

While a considerable effort in the development of branch-and-cut algorithms led to more sophisticated formulations and contributed to a better understanding of our capacity to solve MSTCC instances by judicious use of valid inequalities, the existing Lagrangean algorithms are limited to the most elementary approach. Namely, a relaxation scheme dualizing conflict constraints (4), which thus has the integrality property. We review other aspects of the corresponding references in Section 2.

The present paper takes the standpoint that the development of a full-fledged Lagrangean strategy to find stable spanning trees is an unsolved problem. While we recognize different merits of previous work, we argue that it is worth investigating stronger Lagrangean bounds for the MSTCC structure: exploring more creative relaxation schemes, designing improved dual methods, all the while harnessing the progress in IP computation.

The main idea of this paper is to offer an alternative starting point for this problem. In Section 3, we call attention to a stronger relaxation scheme, based on Lagrangean *Decomposition*. We explain how classical results from the literature guarantee the superiority of such a reformulation: both with respect to the quality of dual bounds, when compared to the straightforward relaxation, and with regard to the number of multipliers, when compared to an alternative framework to determine the same bounds (relax-and-cut

dualizing violated subtour elimination constraints (2) dynamically). The decomposition naturally leads to the dual ascent paradigm to solve the Lagrangean dual problem, and Section 4 is devoted to presenting maximal ascent directions. These are fundamental ingredients in tailored methods guaranteeing monotone bound improvement when optimizing the Lagrangean dual.

We see the opportunity for renewed interest in Lagrangean Decomposition in light of the progress in mixed-integer linear programming (MILP) computation. Given the impressive speedup of MILP solvers over the past two decades, Dimitris Bertsimas and Jack Dunn are among a group of distinguished researchers who make a case for (exact) optimization over integers as the natural, correct model for several tasks within machine learning and towards interpretable artificial intelligence. This is the theme of their recent book [2]; see also [3, 4]. We are interested in exploring whether this philosophy (challenging assumptions previously deemed computationally intractable) should also imply less hesitation towards designing Lagrangean algorithms that exploit subproblems for which, albeit strongly NP-hard, specialized solvers attain good performance. In the particular case of the MSTCC problem, one could leverage state-of-the-art branch-and-cut algorithms for stable sets (in particular, of fixed cardinality) to find stable spanning trees more efficiently by means of Lagrangean Decomposition, such as we outline in this paper.

In summary, our contributions are the following.

(1) We bring attention to the quality of different Lagrangean bounds for the MSTCC problem as an inviting margin for designing improved algorithms, and we discuss the advantages, in theory, of a reformulation based on Lagrangean Decomposition.
(2) We determine a collection of Lagrangean dual ascent directions for optimizing the Lagrangean dual problem corresponding to the new MSTCC reformulation, hence contributing towards a new family of algorithms and dual bounds for the problem.

2 DRAWBACKS OF EXISTING LAGRANGEAN APPROACHES FOR MSTCC

The work of [20] contributes in many research directions about stable spanning trees, including particular cases which are polynomially solvable, feasibility tests, several heuristics, and two exact algorithms based on Lagrangean relaxation. The first formulation is the straightforward one we mentioned, dualizing all conflict constraints (4); they denote the corresponding dual bound L^*. The second approach relaxes a subset of inequalities (4): using an approximation to the maximum edge clique partitioning problem [10], this scheme dualizes a subset of conflict constraints such that the remaining conflict graph is a collection of disjoint cliques; the resulting dual bound is denoted ℓ^*. The authors argue that the latter reformulation is stronger than the former, and present extensive computational results justifying their claims.

Unfortunately, the Lagrangean dual bounds L^* and ℓ^* in [20] are in fact identical, as we show next. The first relaxation clearly has the integrality property, as the remaining constraints correspond to a description of the spanning tree polytope or, equivalently, to bases of the graphic matroid of G. The second relaxation scheme is designed so that the conflict constraints which remain in the subproblem of relaxation ℓ^* induce a collection of disjoint cliques in $\hat{G}$. The subproblem thus corresponds to the intersection of two matroids: the graphic matroid of G and the partition matroid of subsets of E that intersect the enumerated cliques in $\hat{G}$ at most once. It follows that the second relaxation also has the integrality property [15, Theorem III.3.5.9], and consequently, L^* and ℓ^* both equal the optimal objective function value in the continuous relaxation of (1)−(5) [15, Corollary II.3.6.6]. In this perspective, the computational results in Tables 2–4 of [20] diverge from what Lagrangean duality theory prescribes.

Recently, [7] presented thorough computational experiments of a new Lagrangean algorithm for the MSTCC problem. They use the same relaxation scheme dualizing all conflict constraints, and focus on a combination of dual ascent and the subgradient method to compute the Lagrangean bound, namely, L^* in [20], equal to the LP-relaxation of (1) − (5). In Table 1 of [7], the performance of the new algorithm is compared to the results published in [20]. That is, the issue we analyse above regarding the computational results of [20] is repeated as a baseline of the new numerical evaluation.

Another drawback of the new algorithm is that dual ascent steps are intertwined with subgradient optimization. While *not* incorrect, this choice undermines the advantages of a strategy to solve the dual problem in fewer iterations. A passage from a classical work of Guignard and Rosenwein [14] is conclusive: *"An ascent procedure may also serve to initialize multipliers in a subgradient procedure. This scheme is particularly useful at the root node of an enumeration tree. However, an ascent method cannot guarantee improved bounds over bounds obtained by solving the Lagrangean dual with a subgradient procedure."*

Moreover, the ascent steps rely on a greedy heuristic, and not on *maximal ascent directions*, *i.e.* optimal step size in a direction of bound increase; see Definition 4.1. In the algorithm of [7], if a conflicting pair of edges exists in a Lagrangean solution, the multiplier adjustment is derived from the observation that the dual bound shall improve by at least the increased cost of replacing one of the edges by its cheapest successor (in a list of edges ordered by current costs). The authors remedy the resulting low adjustment values by alternating subgradient optimization iterations and the ascent procedure.

We stress again that references [7] and [20] have many virtues and present concrete contributions to the MSTCC literature. Our only remark is that the first Lagrangean strategy designed to improve upon the LP-relaxation bound is matter-of-factly yet to be introduced. In the next sections, we offer an interesting approach to tackle this challenge.

3 LAGRANGEAN DECOMPOSITION

Renaming the variables in (4) as y, and introducing linking constraints $x_e = y_e$ for each $e \in E$, we have the same formulation. Now, dualizing the linking constraints with Lagrangean multipliers $\lambda \in \mathbb{Q}^{|E|}$, we arrive at the Lagrangean Decomposition formulation:

$$z(\lambda) \stackrel{\text{def}}{=} \min_{x \in \mathcal{F}_{\text{sp.tree}}(G)} (w-\lambda)^\top x \quad + \quad \min_{y \in \mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)} \lambda^\top y \qquad (6)$$

where $\mathcal{F}_{\text{sp.tree}}(G)$ is given by

$$\sum_{e\in E(S)} x_e \le |S|-1, \quad \text{for each } S\subsetneq V, S\neq\emptyset, \tag{7}$$

$$\sum_{e\in E} x_e = |V|-1, \tag{8}$$

$$x_e \in \{0,1\}, \qquad \text{for each } e\in E, \tag{9}$$

and $\mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)$ is given by

$$\sum_{e\in E} y_e = |V|-1, \tag{10}$$

$$y_{e_i} + y_{e_j} \le 1, \quad \text{for each } \{e_i, e_j\}\in C, \tag{11}$$

$$y_e \in \{0,1\}, \quad \text{for each } e\in E. \tag{12}$$

The Lagrangean dual problem is to determine the tightest such bound:

$$\zeta \stackrel{\text{def}}{=} \max_{\lambda\in\mathbb{Q}^{|E|}} \{z(\lambda)\}. \tag{13}$$

The first systematic study of Lagrangean Decomposition as a general purpose reformulation technique was presented by Guignard and Kim [13]. They indicate earlier applications of variable splitting/layering, especially [19] and [16]. See also the outstanding presentation in [12, Section 7].

One of the main virtues of the decomposition principle over traditional Lagrangean relaxation schemes is that the bound from the Lagrangean decomposition dual is equal to the optimum of the primal objective function over the intersection of the convex hulls of both constraint sets [13, Corollary 3.4]. The decomposition bound is thus equal to the strongest of the two Lagrangean relaxation schemes corresponding to dualizing either of the constraint sets.

In our application to the MSTCC problem, we recognize the integrality of the spanning tree formulation described by (7) – (8) over $x\in\mathbb{Q}^{|E|}$. Hence the decomposition bound matches that of the stronger scheme where constraints (10) – (11) enforcing fixed cardinality stable sets are kept in the subproblem (which is thus *convexified*), and all subtour elimination constraints (7) are dualized. This means that we can compute stronger Lagrangean bounds, while limiting the number of multipliers in the dual problem to $|E|$, instead of dealing with exponentially-many multipliers *e.g.* in a relax-and-cut approach.

We defend the advantages of breaking the original problem into two parts, exploiting their rich combinatorial and polyhedral structures, so as to derive stronger dual bounds. The price of this strategy is to solve a strongly NP-hard subproblem, which naturally leads to more sophisticated dual algorithms, requiring the fewest iterations possible. Customized dual ascent is a technique that integrates naturally with Lagrangean decomposition [13], and may be the key ingredient towards effective computation of such stronger bounds.

4 LAGRANGEAN DUAL ASCENT

In this section we present the main contributions of the paper. In what follows, let $\mathfrak{e}_i\in\mathbb{R}^m$ denote the standard unit vector in the i-th direction, let **conv** S denote the convex hull of a set S, and let **ext** Q denote the set of extreme points of a given polyhedron Q. We let

$$\mathcal{P}_{\text{sp.tree}}(G) \stackrel{\text{def}}{=} \textbf{conv}\, \mathcal{F}_{\text{sp.tree}}(G)$$

denote the spanning tree polytope of a graph G, and let

$$\mathfrak{C}(G,k) \stackrel{\text{def}}{=} \textbf{conv}\, \mathcal{F}_{\text{kstab}}(G,k)$$

denote the polytope of stable sets of cardinality k in G. Note that $\mathcal{P}_{\text{sp.tree}}$ and $\mathfrak{C}$ are bounded (polytopes contained in the 0,1 hypercube), and do not contain extreme rays.

The Lagrangean dual function $z:\mathbb{Q}^{|E|}\to\mathbb{Q}$ is an implicit function of λ. It is determined by the the lower envelope of

$$\Big\{(w-\lambda)^\intercal x^r + \lambda^\intercal y^s \;:\; x^r\in\textbf{ext}\,\mathcal{P}_{\text{sp.tree}}(G), \; y^s\in\textbf{ext}\,\mathfrak{C}(\hat{G},|V|-1)\Big\}.$$

Hence, it is piecewise linear concave, and differentiable almost everywhere, with breakpoints at λ' such that the optimal solution of $z(\lambda')$ is not unique.

Such breakpoints are the key ingredient in the dual ascent paradigm to solve a Lagrangean dual problem. In particular, the following kind of point deserves special attention to guide progress in this framework.

Definition 4.1. A **maximal ascent direction** of the Lagrangean dual function $z:\mathbb{Q}^m\to\mathbb{Q}$ at λ^r is a vector $u\in\mathbb{Q}^m$ in a direction of increase from $z(\lambda^r)$, *i.e.*

$$z(\lambda^r+u) > z(\lambda^r),$$

such that $\lambda+u$ is a breakpoint of z, *i.e.*

$$z(\lambda^r+u) \ge z(\lambda^r+\alpha u), \text{ for all } \alpha\in\mathbb{Q}.$$

A maximal ascent direction determines an optimal multiplier adjustment in a given direction of increase of the Lagrangean dual function. It need not correspond to a *steepest* ascent direction from $z(\lambda^r)$, in general.

The technique of optimizing the Lagrangean dual function by means of ascent directions uses the formulation structure to determine monotone bound improving sequences of multipliers. It was pioneered by [5] and [11] in the context of the facility location problem. An actual algorithm of this kind thus relies on analysing the specific problem and the information available from subproblem solutions. Nevertheless, we found it instructive to summarize and systematically review the following instructions as a *guiding principle* of Lagrangean decomposition based dual ascent:

i. Update multiplier λ_e corresponding to a violation $x_e\neq y_e$
ii. in the sense of improving the Lagrangean dual bound,
iii. analysing the implications of changes in λ_e *alone*,
iv. so as to induce alternative subproblem solutions
v. while avoiding bound-decreasing effects.

Although one cannot hope for a pragmatic, problem-independent algorithm, this principle is the intuitive foundation of the arguments that follow.

So as not to overload notation, we omit the transposition symbol in the remainder of the text, whenever it is clear from the context *e.g.* in vector products like $(w-\lambda^r)^\intercal\, x^r$.

Theorem 4.2. *Let (x^r, y^r) be an optimal solution of subproblem $z(\lambda^r)$, such that $x^r_e = 0 < 1 = y^r_e$. Define the non-negative quantities*

$$\Delta^r_e \stackrel{def}{=} \min\left\{\lambda^r y : y \in \mathcal{F}_{kstab}(\hat{G}, |V|-1), y_e = 0\right\} - \lambda^r y^r, \tag{14}$$

$$\partial^r_e \stackrel{def}{=} \min\left\{(w-\lambda^r)x : x \in \mathcal{F}_{sp.tree}(G), x_e = 1\right\} - \left(w - \lambda^r\right) x^r. \tag{15}$$

If $\min\left\{\Delta^r_e, \partial^r_e\right\} \neq 0$, then $\min\left\{\Delta^r_e, \partial^r_e\right\} \cdot \mathfrak{e}_e$ is a maximal ascent direction of z at λ^r.

Proof.

(i.) Given that $x^r_e = 0$ and $y^r_e = 1$, increasing λ^r_e corresponds to increasing the dual bound, until alternative optimal solutions where that hypothesis fails are induced. Specifically,

$$z(\lambda^r + \epsilon\mathfrak{e}_e) > z(\lambda^r) \tag{16}$$

for all $\epsilon > 0$ such that

$$x^r \in \arg\min\left\{\left(w - \left(\lambda^r + \epsilon\mathfrak{e}_e\right)\right)x : x \in \mathcal{F}_{\text{sp.tree}}(G)\right\}, \tag{17}$$

$$y^r \in \arg\min\left\{\left(\lambda^r + \epsilon\mathfrak{e}_e\right) y : y \in \mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)\right\}. \tag{18}$$

As long as ϵ can be made positive, $\epsilon\mathfrak{e}_e$ is a direction of increase from $z(\lambda^r)$. The necessity of conditions (17) and (18) follows from noting that the contribution of the e-th variables x_e and y_e to z,

$$\left(w_e - \left(\lambda^r_e + \epsilon\mathfrak{e}_e\right)\right) x_e + \left(\lambda^r_e + \epsilon\mathfrak{e}_e\right) y_e,$$

remains constant as we increase ϵ after x_e joins, or y_e leaves, an optimal solution. For, if $x_e = y_e = 1$, meaning that the coefficient of edge e is attractive enough in (17), any further increase $+\epsilon y_e$ is cancelled by $-\epsilon x_e$. Moreover, if $x_e = y_e = 0$, which means that the coefficient of vertex e is no longer attractive enough in (18), further increasing ϵ in $\left(\lambda^r_e + \epsilon\mathfrak{e}_e\right) y_e = 0$ has no effect.

(ii.) To determine ϵ such that we find a breakpoint of z, we use the limiting conditions (17), (18).
For x^r to no longer be the unique optimum in (17), the cost of edge e decreases so much that an alternative solution $\tilde{x} \in \mathcal{F}_{\text{sp.tree}}(G)$ which includes e is determined. Note that $\tilde{x}$ is well-defined, as the choice of edges in a minimum spanning tree where e is fixed *a priori* does not depend on the cost of e (all other costs are kept unchanged). Also note that, since the existing solution is such that $x^r_e = 0$, the cost of $\tilde{x}$ is no less than that of x^r. The difference is precisely ∂^r_e in (15).
If $\partial^r_e = 0$, the bound cannot be improved by adjusting λ^r_e, as an alternative minimum spanning tree including e is readily available; equivalently, we should have $\epsilon = 0$ in part (i). If $\partial^r_e > 0$, it is the maximum increase in λ^r_e (*i.e.* decrease in the cost of edge e in the x subproblem) before $\tilde{x}$ becomes optimal and z starts to decrease. That is, enforcing (17) yields

$$\epsilon \leq \partial^r_e\ . \tag{19}$$

(iii.) For y^r to no longer be the unique optimum in (18), the cost of vertex e increases so much that an alternative fixed cardinality stable set $\tilde{y} \in \mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)$ which does not include e is determined.
Analogous to the situation in part (ii), $\tilde{y}$ is well-defined because the multipliers corresponding to all other vertices are kept constant: choosing $\tilde{y}$ amounts to finding a minimum cost fixed cardinality stable set in $\hat{G}-e$. Also, its cost is no less than that of y^r, the existing optimal solution to the y subproblem. The difference is exactly Δ^r_e in (14).
If $\Delta^r_e = 0$, no bound improvement by changing λ^r_e is possible, as an alternative fixed cardinality stable set of least cost not including e is readily available; *i.e.* we should have $\epsilon = 0$ in part (i). On the other hand, if $\Delta^r_e > 0$, it is the maximum increase in λ^r_e before $\tilde{y}$ becomes optimal and z stops increasing. That is, enforcing (18) yields

$$\epsilon \leq \Delta^r_e\ . \tag{20}$$

(iv.) In conclusion, if $\min\left\{\Delta^r_e, \partial^r_e\right\} = 0$, then $\epsilon = 0$ and $\epsilon\mathfrak{e}_e$ fails to be a direction of increase from $z(\lambda^r)$. Otherwise, we combine bounds (19) and (20) into (16):

$$\forall \epsilon > 0,\ z(\lambda^r + \min\left\{\Delta^r_e, \partial^r_e\right\} \cdot \mathfrak{e}_e) \geq z(\lambda^r + \epsilon\mathfrak{e}_e),$$

showing that $\lambda^r + \min\left\{\Delta^r_e, \partial^r_e\right\} \cdot \mathfrak{e}_e$ is a breakpoint of z, and $\min\left\{\Delta^r_e, \partial^r_e\right\} \cdot \mathfrak{e}_e$ is a maximal ascent direction.

□

To determine a minimum spanning tree with edge $e = \{i, j\}$ fixed *a priori* in part (ii), we may *contract* that edge in G. If the contraction operator is defined so as to allow parallel edges between the new vertex ij and $k \in N(i) \cap N(j)$, where $N(u) \subset V$ denotes the neighbourhood of vertex u, we must ensure that not more than one edge between two vertices is chosen (*e.g.* in Kruskal's algorithm; this is not an issue in Prim's method). Now, if the contraction operator forbids parallel edges, we make an unambiguous choice in the original graph G by recognizing the proper edge ($\{i, k\}$ or $\{j, k\}$) yielding the correct spanning tree.

A maximal ascent direction from Lagrangean solutions where $x^r_e = 1$ but $y^r_e = 0$ is derived by an argument analogous to that of Theorem 4.2. The next proof if thus significantly streamlined.

Theorem 4.3. *Let (x^r, y^r) be an optimal solution of subproblem $z(\lambda^r)$, such that $x^r_e = 1 > 0 = y^r_e$. Define the non-negative quantities*

$$\Delta^r_e \stackrel{def}{=} \min\left\{\lambda^r y : y \in \mathcal{F}_{kstab}(\hat{G}, |V|-1), y_e = 1\right\} - \lambda^r y^r, \tag{21}$$

$$\partial^r_e \stackrel{def}{=} \min\left\{(w-\lambda^r)x : x \in \mathcal{F}_{sp.tree}(G), x_e = 0\right\} - \left(w - \lambda^r\right) x^r. \tag{22}$$

If $\min\left\{\Delta^r_e, \partial^r_e\right\} \neq 0$, then $\min\left\{\Delta^r_e, \partial^r_e\right\} \cdot (-\mathfrak{e}_e)$ is a maximal ascent direction of z at λ^r.

Proof. Decreasing λ^r_e corresponds to increasing the dual bound, in this case. Hence, $\epsilon(-\mathfrak{e}_e)$ is a direction of increase from $z(\lambda^r)$, as long as ϵ can be made positive in

$$z\left(\lambda^r + \epsilon(-\mathfrak{e}_e)\right) > z\left(\lambda^r\right), \tag{23}$$

where

$$x^r \in \arg\min\left\{\left[w - \left(\lambda^r + \epsilon(-\mathfrak{e}_e)\right)\right]x : x \in \mathcal{F}_{\text{sp.tree}}(G)\right\}, \tag{24}$$

$$y^r \in \arg\min\left\{\left[\lambda^r + \epsilon(-\mathfrak{e}_e)\right] y : y \in \mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)\right\}. \tag{25}$$

For y^r to no longer be the unique optimum in (25), the cost of vertex e decreases enough for an alternative solution including e to be determined. Since all other multipliers are kept constant, such point $\tilde{y} \in \mathcal{F}_{\text{kstab}}(\hat{G}, |V|-1)$ corresponds to a minimum cost stable

set of cardinality $|V| - 2$ in $\hat{G} - N[e]$, that is, the conflict graph where the closed neighbourhood of vertex e is removed. As the existing solution is such that $y^r_e = 0$, the cost of $\tilde{y}$ is no less than that of y^r. The difference is precisely Δ^r_e in (21).

Now, for x^r to no longer be the unique optimum in (24), the cost of edge e increases as far as determining an alternative minimum spanning tree not including e. Let $\tilde{x} \in \mathcal{F}_{\text{sp.tree}}(G)$ denote that point, which corresponds to a minimum spanning tree in $G - e$, since all other multipliers are held constant. The cost of $\tilde{x}$ is no less than that of x^r, the existing optimal solution to the x subproblem. The difference is exactly ∂^r_e in (22).

If $\min\left\{\Delta^r_e, \partial^r_e\right\} = 0$, then $\epsilon = 0$, and $\epsilon(-\mathfrak{e}_e)$ fails to be a direction of increase from $z(\lambda^r)$. Otherwise, we have

$$\forall \epsilon > 0,\ z\big(\lambda^r + \min\left\{\Delta^r_e, \partial^r_e\right\} \cdot (-\mathfrak{e}_e)\big) \geq z\big(\lambda^r + \epsilon(-\mathfrak{e}_e)\big),$$

showing that $\lambda^r + \min\left\{\Delta^r_e, \partial^r_e\right\} \cdot (-\mathfrak{e}_e)$ is a breakpoint of z, and $\min\left\{\Delta^r_e, \partial^r_e\right\} \cdot (-\mathfrak{e}_e)$ is a maximal ascent direction. □

5 CONCLUDING REMARKS

We bring attention to a research question that we consider both attractive and promising. Stable spanning trees comprise appealing combinatorial and polyhedral structures, and designing a Lagrangean algorithm that may yield stronger dual bounds to optimal stable trees is an open problem. This paper presents the first steps in a sensible direction: Lagrangean Decomposition inducing a non-integral relaxation, coupled with carefully designed dual ascent.

Our development relies on the solid foundation that the pioneers of Lagrangean duality in IP have laid, through which we are able to justify the shortcomings of existing approaches and the virtues of the one we propose. We also make an effort for our exposition of the design principle of Lagrangean dual ascent to be fairly tutorial, and for the main proof of the maximal ascent direction to be instructive.

The definitive proof of concept should be actually computing the stronger bounds and finding optimal stable spanning trees in computationally challenging benchmark instances more efficiently. We are currently crafting an implementation of the method outlined in this paper. Regardless of the success of our current efforts and one particular algorithm, we stand in the position put forth at the end of the Introduction. In light of the progress in MILP computation, it seems worthwhile to further investigate the strategy of Lagrangean Decomposition based on harder subproblems, possibly replacing the common sense boundary of weakly NP-hard choices by the weaker requirement that our choice be *computationally tractable*.

ACKNOWLEDGMENTS

This research is partly supported by the Research Council of Norway through the research project 249994 CLASSIS.

REFERENCES

[1] Arjang Assad and Weixuan Xu. 1992. The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)* 39, 3 (1992), 399–417. https://doi.org/10.1002/1520-6750(199204)39:3<399::AID-NAV3220390309>3.0.CO;2-0

[2] Dimitris Bertsimas and Jack Dunn. 2019. *Machine learning under a modern optimization lens.* Dynamic Ideas LLC, Charlestown.

[3] Dimitris Bertsimas, Angela King, and Rahul Mazumder. 2016. Best subset selection via a modern optimization lens. *The Annals of Statistics* 44, 2 (2016), 813 – 852. https://doi.org/10.1214/15-AOS1388

[4] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. 2020. Sparse Regression: Scalable Algorithms and Empirical Performance. *Statist. Sci.* 35, 4 (2020), 555 – 578. https://doi.org/10.1214/19-STS701

[5] Ole Bilde and Jakob Krarup. 1977. Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem. In *Studies in Integer Programming*, P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser (Eds.). Annals of Discrete Mathematics, Vol. 1. Elsevier B.V., North-Holland Publishing Company, 79–97. https://doi.org/10.1016/S0167-5060(08)70728-3

[6] Francesco Carrabs, Raffaele Cerulli, Rosa Pentangelo, and Andrea Raiconi. 2021. Minimum spanning tree with conflicting edge pairs: a branch-and-cut approach. *Annals of Operations Research* 298, 1 (2021), 65–78. https://doi.org/10.1007/s10479-018-2895-y

[7] Francesco Carrabs and Manlio Gaudioso. 2021. A Lagrangian approach for the minimum spanning tree problem with conflicting edge pairs. *Networks* 78, 1 (2021), 32–45. https://doi.org/10.1002/net.22009

[8] Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. 2009. Determining a Minimum Spanning Tree with Disjunctive Constraints. In *Algorithmic Decision Theory*, Francesca Rossi and Alexis Tsoukias (Eds.). Lecture Notes in Computer Science, Vol. 5783. Springer, Berlin, Heidelberg, 414–423. https://doi.org/10.1007/978-3-642-04428-1_36

[9] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. 2011. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics* 159, 16 (2011), 1726 – 1735. https://doi.org/10.1016/j.dam.2010.12.016

[10] Anders Dessmark, Jesper Jansson, Andrzej Lingas, Eva-Marta Lundell, and Mia Persson. 2007. On the Approximability of Maximum and Minimum Edge Clique Partition Problems. *International Journal of Foundations of Computer Science* 18, 02 (2007), 217–226. https://doi.org/10.1142/S0129054107004656

[11] Donald Erlenkotter. 1978. A Dual-Based Procedure for Uncapacitated Facility Location. *Operations Research* 26, 6 (1978), 992–1009. http://www.jstor.org/stable/170260

[12] Monique Guignard. 2003. Lagrangean relaxation. *Top* 11, 2 (2003), 151–200. https://doi.org/10.1007/BF02579036

[13] Monique Guignard and Siwhan Kim. 1987. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming* 39 (1987), 215–228. https://doi.org/10.1007/BF02592954

[14] Monique Guignard and Moshe B. Rosenwein. 1989. An application-oriented guide for designing Lagrangean dual ascent algorithms. *European Journal of Operational Research* 43, 2 (1989), 197–205. https://doi.org/10.1016/0377-2217(89)90213-0

[15] George L Nemhauser and Laurence A Wolsey. 1999. *Integer and combinatorial optimization.* Wiley-Interscience series in discrete mathematics and optimization, Vol. 55. John Wiley & Sons, Inc. https://doi.org/10.1002/9781118627372

[16] Celso Ribeiro and Michel Minoux. 1986. Solving hard constrained shortest path problems by Lagrangean relaxation and branch-and-bound algorithms. *Methods of Operations Research* 53 (1986), 303–316.

[17] Phillippe Samer and Dag Haugland. 2021. Fixed cardinality stable sets. *Discrete Applied Mathematics* 303 (2021), 137–148. https://doi.org/10.1016/j.dam.2021.01.019

[18] Phillippe Samer and Sebastián Urrutia. 2015. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters* 9, 1 (2015), 41–55. https://doi.org/10.1007/s11590-014-0750-x

[19] Fred Shepardson and Roy E. Marsten. 1980. A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem. *Management Science* 26, 3 (1980), 274–281. https://doi.org/10.1287/mnsc.26.3.274

[20] Ruonan Zhang, Santosh N. Kabadi, and Abraham P. Punnen. 2011. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization* 8, 2 (2011), 191 – 205. https://doi.org/10.1016/j.disopt.2010.08.001

Session Session 4B: interdiction problems

Thursday 9 June 2022, 13:30-15:10

Lecture Hall III

A tri-level Network Protection Problem with weight control

Pierre Hosteins[1] and Margarida Carvalho[2]

[1]COSYS-ESTAS, University Gustave Eiffel, Villeneuve d'Ascq, France, ✉ pierre.hosteins@univ-eiffel.fr
[2]Dep. of Comp. Sc. and Op. Res., Polytechnique Montréal, Montréal, Canada, ✉ carvalho@iro.umontreal.ca

1 Introduction

Network Interdiction Problems (NIPs) are problems where an attacker tries to disable some network elements (usually edges or vertices) submitted to an attack budget, while a defender reacts to such an attack in order to optimise some kind of connectivity on the network. Classic examples are, among many others, the Maximum Flow Interdiction or Shortest Path Interdiction problems, where an attacker disables edges to minimise the maximum flow or maximise the length of the shortest path between two nodes. Such problems have many possible applications, such as assessing the robustness of a transportation or energy distribution network, studying the structure of social networks or biological networks and of course military and security applications. Given their attacker-defender structure, they are often modeled as bi-level optimisation problems. We refer the reader to the survey [2] for an overview of NIPs and the Mathematical Programming techniques used to solve them.

Though the focus is often on finding the most critical set of elements to disable from an attacker's point of view, a few works exist in the literature that instead focus on how to protect the network from the malign intervention of an attacker. These problems very often take the form of a *fortification* of a subset of graph elements, i.e. rendering some graph elements impervious to attacks, see e.g. [1]. To our knowledge, no approach has been studied to act instead on the relative deletion costs of the different graph elements. However, we feel that in many situations it should be possible to allocate more resources to protect a specific part of the network with respect to other parts, therefore requiring a larger effort on the part of the attacker to disable the respective graph elements. We refer to such a problem as the Network Protection Problem with weight control (NPPwc) in the following.

2 Tri-level formulation and cutting plane reformulation

Next we model the NPPwc as a tri-level problem over a graph $G = (V, E)$. We consider that the defender has a protection budget W, the attacker an attack budget K and that both edges and nodes can be interdicted. Define the following variables:

- w_i, w_{ij}: deletion costs (either continuous or integer) for nodes $i \in V$ and edges $(i, j) \in E$;
- v_i, v_{ij}: binary variables equal to 1 if $i \in V$ or $(i, j) \in E$ is deleted from G, 0 otherwise;
- x: general variables of the follower to optimise the connectivity measure on the network.

We call $X(v)$ the domain of definition of the lower level (defender) variables. The NPPwc is:

$$\max_{w} C_d \tag{1}$$

$$\sum_{i \in V} w_i + \sum_{\{i,j\} \in E} w_{ij} \leq W \tag{2}$$

$$C_d = \min_{v} C_a(w) \tag{3}$$

$$\sum_{i \in V} w_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_{ij} \leq K \tag{4}$$

$$C_a = \max_{x} C'_d(x) \tag{5}$$

$$x \in X(v), \tag{6}$$

where C_d is the objective value of the defender at the higher level, C_a is the objective of the attacker at the second level and finally C'_d is the objective of the defender at the lower level. We will concentrate on problems where the lower level is a polynomial problem and the two lower levels can be transformed into a single level (by, e.g., inner dualisation). The problem becomes bi-level, where the lower level is the traditional NIP and the upper problem tries to allocate the deletion costs optimally. We consider both the possibility of continuous and integer values for the deletion costs w.

We write the model in a simpler form which enumerates all the second level solutions with their respective objective value. Define $\mathcal{S}$ as the set of second level (attacker) solutions: a $s \in \mathcal{S}$ is defined by parameters $\bar{v}_i^s$ and $\bar{v}_{ij}^s$ equal to 1 if the graph elements are deleted in solution s, along with the connectivity C_s associated to s. We can reformulate the above MINLP as:

$$\max_{w} C_d \tag{7}$$

$$\sum_{i \in V} w_i + \sum_{\{i,j\} \in E} w_{ij} \ \leq W \tag{8}$$

$$C_d \leq C_s + (1 - \pi_s)(C_{max} - C_s) \qquad s \in \mathcal{S} \tag{9}$$

$$(K + \varepsilon)\pi_s \geq K + \varepsilon - \sum_{i \in V} w_i \bar{v}_i - \sum_{\{i,j\} \in E} w_{ij} \bar{v}_{ij} \qquad s \in \mathcal{S} \tag{10}$$

with π_s binary variables which activate the constraint for a solution $s \in \mathcal{S}$ if its attack budget is less than the maximum budget K and where ε is small enough to find the optimal solution ($\varepsilon = 1$ for integer weights). Therefore, each solution has an associated variable and constraint. We can adopt a Cutting Plane (CP) algorithmic approach, i.e. start with an empty (or very small) set $\mathcal{S}$ and solve the model iteratively by adding violated cuts. At each iteration, we add a binary variable π_s and we can branch on it: the branching node where $\pi_s = 1$ is closed immediatly since we cannot improve on the solution found while the node where $\pi_s = 0$ forces the model to make the newly found attacker's solution infeasible in the next iteration. The procedure converges when the model become infeasible.

3 Conclusions and perspectives

We will assess the above algorithmic framework for solving the protection version of several classic NIPs on a set of benchmark instances. It is known that models with continuous upper level variables but integer lower level variables suffer in general from the non-existence of an optimal solution but we will prove its existence. Depending on the time and results available, we will also try to refine the cuts obtained to lift them as much as possible, or to apply existing sophisticated approaches for fortification problems, as e.g. [1]. We will also derive the Σ_p^2-completeness of the application of our model to some classic NIPs (with integer attack weights) in order to justify a CP algorithm to solve our bi-level approach. This demonstration first shows that several NPPwc incarnations are equivalent, on specially structured graphs, to an Interdiction Knapsack Problem with weight control, which we show to be Σ_p^2-complete.

References

[1] L. Lozano and J. C. Smith. A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing*, 29:123–139, 2017.

[2] J. C. Smith and Y. Song. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 283:797–811, 2020.

A Two-stage Network Interdiction-Monitoring Game

Di H. Nguyen[1], Yongjia Song[2], and J. Cole Smith[3]

[1]Department of Industrial Engineering, Clemson University, Clemson, USA, ✉ din@clemson.edu
[2]Department of Industrial Engineering, Clemson University, Clemson, USA, ✉ yongjis@clemson.edu
[3]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, USA, ✉ colesmit@syr.edu

We study a two-stage network interdiction-monitoring game involving an defender and an attacker. The attacker's goal is to reach a target node represented by the sink node undetected. In the first stage, the defender removes (by interdicting) a subset of arcs on the network given an interdiction budget. In the second stage, the problem is a two-person simultaneous zero-sum game similar to the one studied by Washburn and Wood [6], in which the defender randomly selects from a set of feasible monitoring decisions subject to a budget independent of the interdiction budget. The probability that the attacker is detected when traversing an arc that is monitored is known to both players. Overall, the defender seeks an interdiction-monitoring strategy that minimizes the probability that the attacker reaches the target node. We refer to this problem as the *network interdiction in a simultaneous game*, or *NISG* for short. There is an extensive body of literature on network interdiction problems in which the defender employs a deterministic strategy [3, 7, 8], or a probabilistic strategy [1, 2, 4, 5]. Applications for this problem include homeland security and cyber/physical security, where modifying the network, i.e., removing a subset of arcs, can be costly. Thus, a strategy that combines interdiction and monitoring decisions may be more desirable. This paper contributes an exact formulation to the NISG and a decomposition-based solution appproach to solving the NISG.

An Integer Programming Formulation to the NISG. We assume that the game takes place over an acyclic network $G = (N, A)$. Let decision variables x represent the defender's interdiction decisions in the first stage, where $x_a = 1$ if arc $a \in A$ is interdicted and $x_a = 0$ otherwise. Letting B_I denote the interdiction budget and d_a be the cost to interdict arc a, $\forall a \in A$, we define the feasible region of the first-stage problem as $X = \{x : \sum_{a \in A} d_a x_a \leq B_I, x \in \{0,1\}^{|A|}\}$.

With respect to the second stage, we denote by B_M the defender's monitoring budget and by m_a the cost to monitor arc a, $\forall a \in A$. Let $\mathcal{P}$ and $\mathcal{M}$ represent the set of all source-sink paths and the set of all feasible monitoring decisions, respectively. Additionally, denote by $\mathcal{P}(x)$ the set of all feasible source-sink paths and by $\mathcal{M}(x)$ as the set of feasible monitoring decisions for a fixed x, i.e., source-sink paths and monitoring decisions that do not include an interdicted arc. Let y_M represent the probability that the defender selects monitoring decision M, $\forall M \in \mathcal{M}(x)$. Let z_P represent the probability that the attacker selects path P, $\forall P \in \mathcal{P}(x)$. The NISG can be formulated as follows:

$$\text{NISG}(\mathcal{P}, \mathcal{M}) : \min_{x \in X} \left\{ \min_{y} \max_{z} \sum_{P \in \mathcal{P}(x)} F_P(x, y) z_P \right\} \tag{1a}$$

$$\text{s.t.} \quad \sum_{M \in \mathcal{M}(x)} y_M = 1 \tag{1b}$$

$$\sum_{P \in \mathcal{P}(x)} z_P = 1 \tag{1c}$$

$$y_M \geq 0 \qquad \forall M \in \mathcal{M}(x) \tag{1d}$$

$$z_P \geq 0 \qquad \forall M \in \mathcal{P}(x), \tag{1e}$$

where $F_P(x, y)$ denotes the probability that the attacker reaches the sink node undetected via path P given interdiction x and probabilistic monitoring strategy y. Letting $F_{P,M}$ denote the probability that the attacker reaches the sink node undetected via path P given monitoring decision M, the NISG$(\mathcal{P}, \mathcal{M})$ is equivalent to the following:

$$\text{F-NISG}(\mathcal{P}, \mathcal{M}) : \ \min\ u \tag{2a}$$

$$\text{s.t. } u \geq \sum_{M \in \mathcal{M}} F_{P,M} y_M - \sum_{a \in P} x_a \qquad \forall P \in \mathcal{P} \tag{2b}$$

$$\sum_{M \in \mathcal{M}} y_M = 1 \tag{2c}$$

$$y_M \geq 0 \qquad \forall M \in \mathcal{M} \tag{2d}$$

$$x \in X. \tag{2e}$$

However, as solving the F-NISG$(\mathcal{P}, \mathcal{M})$ requires the enumeration all paths in $\mathcal{P}$ and all feasible monitoring decisions in $\mathcal{M}$, we propose a decomposition-based solution approach that utilizes Benders cut in the first stage, and column and row generation in the second stage.

A Reformulation of the NISG. Using an epigraph formulation, we reformulate the first stage as follows: $\min\{\theta : \theta \geq u(x), x \in X\}$, where $u(x)$ represents the optimal objective function to the second stage given a fixed x. Denote by $H_{P,M}$ a linear approximation of $F_{P,M}$, $\forall P \in \mathcal{P}, M \in \mathcal{M}$. The second-stage problem given $x = \hat{x}$ is given by:

$$\text{H-SG}(\hat{x}, \mathcal{P}, \mathcal{M}) : u(\hat{x}) = \min u \tag{3a}$$

$$\text{s.t. } u \geq \sum_{M \in \mathcal{M}} H_{P,M} y_M - R \sum_{a \in P} \hat{x}_a \qquad \forall P \in \mathcal{P} \tag{3b}$$

$$\sum_{M \in \mathcal{M}} y_M = 1 \tag{3c}$$

$$y_M \geq 0 \qquad \forall M \in \mathcal{M}, \tag{3d}$$

where the row-generation problem becomes a shortest-path problem, and the column-generation problem can be solved as a knapsack problem.

An Algorithmic Solution Approach. We begin with subsets $\overline{\mathcal{P}} \subseteq \mathcal{P}$ and $\overline{\mathcal{M}} \subseteq \mathcal{M}$. Our algorithm iteratively obtains a feasible interdiction solution and a lower bound on the optimal objective function by solving the first-stage problem. Given a fixed interdiction solution, we solve the second-stage problem using row and column generation, updating subsets $\overline{\mathcal{P}}$ and $\overline{\mathcal{M}}$ as necessary. Benders cuts corresponding to the second stage's optimal objective values are added to the first stage. The algorithm terminates when the objective function values in the first and second stage are within some permissible tolerance.

Finally, we will also discuss some preliminary results on the efficacy of our algorithm on randomly generated NISG instances.

References

[1] N. Basilico, N. Gatti, and F. Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial intelligence*, 184:78–123, 2012.

[2] D. Bertsimas, E. Nasrabadi, and J. B. Orlin. On the power of randomization in network interdiction. *Operations Research Letters*, 44(1):114–120, 2016.

[3] P. M. Ghare, D. C. Montgomery, and W. C. Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, 18(1):37–45, 1971.

[4] T. Holzmann and J. C. Smith. The shortest path interdiction problem with randomized interdiction strategies: Complexity and algorithms. *Operations Research*, 69(1):82–99, 2021.

[5] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 327–334, 2011.

[6] A. Washburn and R. K. Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

[7] R. D. Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964.

[8] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

On Orienteering Interdiction Games

Kübra Tanınmış[1], Eduardo Álvarez-Miranda[2,3], and Markus Sinnl[1,4]

[1]Institute of Production and Logistics Management, Johannes Kepler University Linz, Linz, Austria, ✉ {kuebra.taninmis_ersues, markus.sinnl}@jku.at
[2]School of Economics and Business, Universidad de Talca, Campus Curicó, Chile, ✉ ealvarez@utalca.cl
[3]Instituto Sistemas Complejos de Ingeniería, Chile,
[4]JKU Business School, Johannes Kepler University Linz, Linz, Austria

Extended Abstract

The orienteering problem (OP) is a fundamental routing problem where the aim is to find a tour on a graph such that a given maximum tour length is not exceeded and the total prize collected from the visited nodes is maximized [3]. It is a variant of the traveling salesman problem (TSP); instead of minimizing the the total tour length, our goal is to find a tour that maximizes the collected prize while respecting a budget on the tour length. This NP-hard problem has applications in several areas such as scheduling, logistics, and tourism and there exist many variants of the problem, see, e.g., the survey [8]. In this work, we introduce orienteering interdiction games (OIGs), which correspond to the incorporation of a competition (game) setting in the context of the OP. Interdiction games involve two players, denoted as *leader* and *follower*, who act sequentially. The follower optimizes a given problem, and the leader tries to undermine the objective of the follower by partially or completely interdicting the usage of some resources of the follower's optimization problem. A recent survey on network interdiction models is presented in [7]. There exists considerably less work on interdiction games in a routing context, the work [5] considers a traveling salesman game and in [6] a multi-depot vehicle routing games is studied.

In our OIG, both the leader and the follower look for a tour which fulfills their given budget. The two tours are over the same graph, and whenever a node is in the tour of the leader, the follower cannot collect the prize of this node. Like in the classical OP, the follower aims to maximize the collected prizes of her or his tour, while the leader tries to minimize the objective the follower can achieve. The proposed problem can model applications such as determining police patrolling strategies against criminal activities [1] and urban campaigning during elections by door-to-door canvassing [4].

Formally, our OIG is defined as follows. We are given a complete graph with node set V, a prize $p_i > 0$ associated with each node $i \in V$, and cost/length d_{ij} associated with each edge $\{i, j\} \in V \times V$. The goal of the leader is to find a tour starting from its depot $\rho_\ell \in V$ and with a total distance no greater than B_ℓ, so that it minimizes the maximum profit that the follower can achieve by performing a tour, starting from $\rho_f \in V$ and with a total distance no greater than B_f.

Let $\mathbf{x}^k \in \{0,1\}^{|V|}$ be a vector of decision variables so that $x_i^k = 1$ if player $k \in \{\ell, f\}$ visits node $i \in V$ in their tour, and $x_i^k = 0$ otherwise. Likewise, $\mathbf{y}^k \in \{0,1\}^{|V \times V|}$ be a vector of decision variables so that $y_{ij}^k = 1$ if player k visits node $j \in V$ after visiting node $i \in V \setminus \{j\}$, and $y_{ij}^k = 0$ otherwise. Also let $(\mathbf{x}^k, \mathbf{y}^k)$ denote the route associated with $\mathbf{x}^k$ and $\mathbf{y}^k$. The problem can be formulated as mixed-integer bilevel programming problem as follows.

$$
\begin{aligned}
z^* = \min_{\mathbf{x}^\ell, \mathbf{y}^\ell} \max_{\mathbf{x}^f, \mathbf{y}^f} & \sum_{i \in V} p_i (1 - x_i^\ell) x_i^f \\
\text{s.t.} \quad & \sum_{i,j \in V \times V} d_{ij} y_{ij}^\ell \leq B_\ell \\
& \sum_{i,j \in V \times V} d_{ij} y_{ij}^f \leq B_f \\
& (\mathbf{x}^\ell, \mathbf{y}^\ell) \in \mathcal{P}_\ell \\
& (\mathbf{x}^f, \mathbf{y}^f) \in \mathcal{P}_f
\end{aligned}
$$

Here $\mathcal{P}_\ell$ and $\mathcal{P}_f$ denote the sets of feasible tours, i.e., cycles that do not contain subtours, where each node is visited at most once, and which pass through ρ_ℓ and ρ_f, respectively.

In order to tackle this challenging bilevel programming problem, we propose a single-level reformulation based on so-called *interdiction cuts*. This technique was introduced in [2] for interdiction games fulfilling a certain *monotonicity* assumption and we show how to adapt it to our OIG. Based on this reformulation, we develop a branch-and-cut algorithm to solve the OIG and introduce various enhancements for the algorithm. We test our approach on instances created from a set of TSPLIB instances by considering several leader's budget levels. In addition, we analyze the obtained results with the aim of gaining some managerial insights. From an algorithmic standpoint, the preliminary results (over 24 TSPLIB instances with sizes between 17 and 100 nodes and node prizes randomly generated) show that the proposed enhancements improve the efficiency of the branch-and-cut significantly. In Table 1, we provide the average results under four of our settings, assuming that the leader's maximum allowed distance is equal to the follower's, i.e., $B_\ell = B_f$. In the first one, interdiction cuts are generated for integer points only. In the second one, fractional points are separated too, but in a heuristic way. Next, we integrate a heuristic for speed up the integer separation. Lastly, we generate multiple interdiction cuts for the same integer/fractional solution of the master problem. We provide in the table the average solution times, optimality gaps, root gaps, number of interdiction cuts and the incumbent objective value.

Table 1: Preliminary results of the B&C algorithm using interdiction cuts.

Setting description	Time(s.)	Gap(%)	RootGap(%)	nICcuts	zBest
Integer separation only	2246.13	20.65	64.23	62.88	799.58
Integer and fractional separation	1778.29	16.75	54.39	20.92	789.08
Heuristics to speed up the separation	1673.29	15.85	57.45	48.91	764.65
Multiple ICs added for the same point	1732.51	15.06	56.11	52.75	781.50

References

[1] Miguel Camacho-Collados and Federico Liberatore. A decision support system for predictive police patrolling. *Decision Support Systems*, 75:25–37, 2015.

[2] Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing*, 31(2):390–410, 2019.

[3] Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[4] Donald P Green, Alan S Gerber, and David W Nickerson. Getting out the vote in local elections: results from six door-to-door canvassing experiments. *The Journal of Politics*, 65(4):1083–1096, 2003.

[5] Leonardo Lozano, J Cole Smith, and Mary E Kurz. Solving the traveling salesman problem with interdiction and fortification. *Operations Research Letters*, 45(3):210–216, 2017.

[6] Mir Ehsan Hesam Sadati, Deniz Aksen, and Necati Aras. The r-interdiction selective multi-depot vehicle routing problem. *International Transactions in Operational Research*, 27(2):835–866, 2020.

[7] J Cole Smith and Yongjia Song. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 283(3):797–811, 2020.

[8] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: a survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

A Monte Carlo Tree Search for Dynamic Shortest-Path Interdiction

Alexey Bochkarev[1] and J. Cole Smith[2]

[1]Department of Industrial Engineering, Clemson University, Clemson, SC, USA 29634
✉ abochka@g.clemson.edu
[2]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA 13244
✉ colesmit@syr.edu

The foundation for our work is the problem of Shortest-Path Interdiction: a two-stage, two-player zero-sum game, played over a directed weighted graph [2, 3]. We study the *Dynamic* Shortest-Path Interdiction problem (DSPI) [5]: a variant of this game, where two players, the Evader and the Interdictor, take steps in turns. The Evader seeks to take a shortest-path between two given nodes, while the Interdictor tries to maximize the cost of this path by attacking the arcs. When an arc is interdicted, its cost increases by a pre-defined amount. The Interdictor starts the game, and can interdict any subset of arcs, subject to a cardinality (budget) constraint. The Evader follows and traverses one arc during each turn. The players take turns until the Evader reaches the given terminal node. Potential applications include counter-terrorist activities, infrastructure reliability problems, and vulnerability analysis, as well as natural disaster response, analysis of social effects, and others.

The decision variant of the problem was shown to be NP-hard [5]. To the best of our knowledge, while bounds on the optimal objective have been proposed in the aforementioned paper along with an exact (exponential-time) dynamic programming solution approach, no heuristic exists in the literature that yields high quality solutions (valid sequences of moves corresponding to the provided objective values). This work contributes an algorithm that fills in this gap.

Note, that it is not forbidden for the Evader to visit a node more than once. Moreover, it is sometimes optimal to do so [5]. This complexity implies that even for graphs with dozens of nodes, enumerating all game states to build the whole decision tree might become impractical. Following some ideas from the Reinforcement Learning research [1, 7] and strategy employed to design a powerful Go playing machine [6], we adapt Monte Carlo Tree Search (MCTS) approach. We first formalize the concept of a game tree, where each node comprises the current game state (Evader's position, current turn, and interdiction set), available players' actions, and costs information. We guide the tree construction by cost-to-go estimates obtained from random simulations of the players' turns, so that the more promising nodes are explored first. We also incorporate tree pruning mechanism leveraging the bounds presented in the literature [5] and using a classical approach for two-agent zero-sum games, called alpha–beta pruning [4]. The game tree encodes learned policies both for the Interdictor and the Evader for the specific instance. The algorithm updates the tree iteratively, choosing the most promising nodes, expanding the tree, and running simulations in the forward pass, and propagating the bounds and the obtained cost information from child to parent nodes in the backward pass.

The proposed framework yields a sequence of turns for both players as follows. First, the tree is built by running a pre-determined number of learning episodes, and the Interdictor's turn is chosen as the most promising child node of the root node. Then, the root node is updated to point to the chosen game state, and the procedure is repeated for the other player, and so on. By limiting the number of learning episodes per move we can trade the depth of analysis and solution quality for runtime. The resulting procedure is a reinforcement learning algorithm in a sense that it does not require a pre-compiled dataset to devise a good solution for the given instance, but learns parts of the solution by interacting with the environment model and receiving *rewards* (in the form of costs in this case).

We highlight the practicality of these ideas in a series of numerical experiments, highlighting the performance of the heuristic for different instance characteristics and algorithm parameters. This analysis suggests that Monte Carlo Tree Search constitute a promising framework for designing a game playing heuristic for DSPI, which can be fine tuned to accommodate for the amount of available computational resources.

There are several possible lines of research aimed to build upon these results. First, the MCTS algorithm can be improved using the many ideas already discussed in the literature (e.g., [1]): caching the frequently visited game states, improving the reliability of node assessment, possibly involving deep neural networks, and many others. Then, the algorithm can be adapted for other variants of interdiction problems. Finally, it would be interesting to investigate ways to retain the information encoded in the game tree across different instances (which in fact would be equivalent to learning heuristic algorithms).

References

[1] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[2] D. R. Fulkerson and Gary C. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13(1):116–118, December 1977.

[3] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[4] Donald E Knuth and Ronald W Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, page 34, 1975.

[5] Jorge A. Sefair and J. Cole Smith. Dynamic shortest-path interdiction. *Networks*, 68(4):315–330, 2016.

[6] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, second edition edition, 2018.

Session Session 4C: routing problems 4

Thursday 9 June 2022, 13:30-15:10

Lecture Hall IV

Optimizing the product distribution within buildings

Germán Paredes-Belmar[1], Guillermo Latorre-Núñez[2], and Andrés Bronfman[3]

[1]School of Industrial Engineering, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile, ✉ german.paredes@pucv.cl
[2]Department of Industrial Engineering, Universidad del Bío-Bío, Concepción, Chile, ✉ glatorre@ubiobio.cl
[2]Department of Engineering Sciences, Universidad Andres Bello, Santiago, Chile, ✉ abronfman@unab.cl

The last-mile delivery (LMD) is defined as the final stage to deliver products from depots, hubs, or warehouses to the customer. In the LMD, many packages are performed inside buildings (residential, institutional, educational, and commercial buildings), especially in urban centers. A wide variety of products, such as food, furniture, certified letters, etc., must be delivered directly to the customer. For example, in office buildings, most office supplies are delivered directly to the offices because usually there is not enough space for temporary storage in the building receptions. Thus, one or more deliverymen (DM) must deliver products inside the buildings (using a bag or cargo cart), starting at the reception, using the elevation system, and ending at the different demand points.

As a relevant antecedent, [5] and [4], estimate that the distribution time inside buildings represents about 87% of the total distribution time in districts with high population density. [6], indicates that the Company Camden Property Trust loses about 10 minutes of productivity per employee in the delivery of a single parcel within a condo with buildings. In the same way, since the Covid-19 pandemics, many orders are accumulated in the reception, especially in residential buildings. Then, the recipients go down to pick up their parcel, or the building staff member delivers the products to residents [2, 7]. Moreover, the elevators' social distancing implies that the goods distribution is slower than a normal situation within an office, university, or hospitals buildings [9].

From a different perspective, each type of building has a specific traffic pattern according to the moment of the day at which the delivery is made [1, 3]. Figure 1 shows a theoretical traffic pattern of an office building [8], indicating that the number of individual elevator calls represents the passenger demand for elevators. They are aggregated according to the demand direction (going up or going down). At the beginning of the day, there are multiple going-up calls (morning up-peak), and they are generated by the arrival of people who work in the building. At the end of the day, there is a down-peak caused by the people that leave the building. In the middle of the day, there are two groups of up-peaks and down-peaks caused by two different lunchtimes (i.e., from 12:00 PM-01:00 PM, and from 01:00 PM to 02:00 PM). This pattern is usually known as bidirectional traffic. There is inter-floor traffic for the rest of the day, where the number of calls is similar. All traffic patterns affect the travel times within a building. Naturally, the traffic pattern differs according to the type of the building and its use, e.g., residential, educational, hospital, public institution, parking, office, etc. [8].

Consequently, the DM distribution time within buildings depends on the arrival time, the building pattern traffic, the elevation system, the building type, and its particular characteristics, etc. Thus, the waiting times and vertical travel times within a building depend on the day's moment. For a DM, these times could represent a significant percentage of the total time within a building, e.g., the total distribution time is higher if a DM arrives at a peak hour.

We define the *vertical routing* term to refer to all those activities required for goods transportation to customers located within buildings. It involves mainly waitings, vertical and horizontal movements. We address time-dependent waiting and vertical times. It can be noticed that the vertical routing problem with time dependency is a particular case of the well-known Time-Dependent Traveling Salesman Problem (TDTSP). In our research, the network of arcs and nodes within a building has particular characteristics, such as the traffic patterns, the time dependency in nodes and arcs, and the network *fishbone* shape. We focus our research on the vertical routing problem by introducing, modeling, and solving the time-dependent vertical routing problem (TDVeRP). To the best of our knowledge, this problem has not been studied in the traveling salesman or last-mile distribution literature. We propose a mixed-linear integer programming (MILP) model and a Genetic Algorithm to solve a set of building instances. A set of

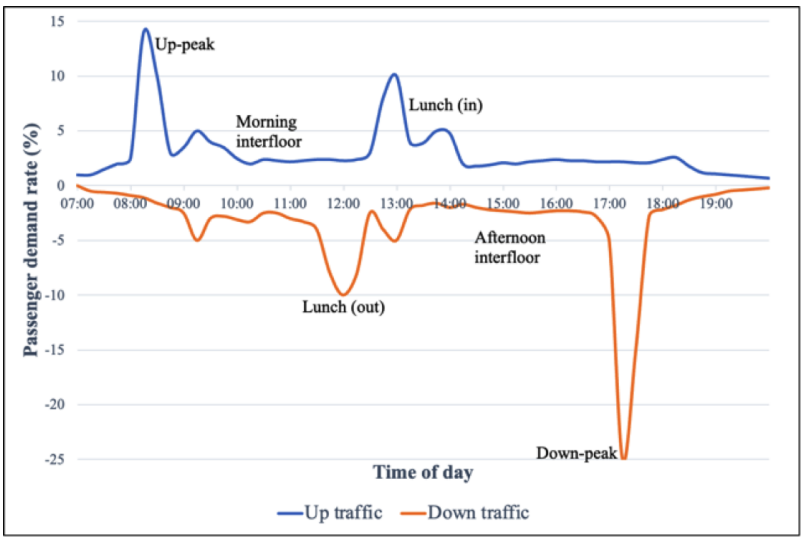


Figure 1: Traffic pattern for an office building (based on [8])

detailed results is presented in two different buildings, considering different variations in the number of customers, DM arrival time, and distribution of customers within a building.

References

[1] Gina Barney and Lutfi Al-Sharif. *Elevator traffic handbook: theory and practice*. Routledge, 2015.

[2] A. Brucker, P. Massa, and D. Degenshein. Covid-19 vs. building and unit access: Who gets in?, 2020.

[3] Cebrail Ciflikli and Emre Oner Tartan. A model for the visualization and analysis of elevator traffic. *Transportation Planning and Technology*, 42(8):868–880, 2019.

[4] Anne Goodchild and Barbara Ivanov. The final 50 feet of the urban goods delivery system. *System*, 54:55, 2017.

[5] Haena Kim, Linda Ng Boyle, and Anne Goodchild. Delivery process for an office building in the seattle central business district. *Transportation Research Record*, 2672(9):173–183, 2018.

[6] L. Kusisto. Web-shopping deluge boxes in landlords: Camden property trust takes a tough stand and stops accepting package deliveries, 2015.

[7] R.J. Sobelsohn. Coops, condos and covid-19, 2020.

[8] G.R. Strakosch and R. S. Caporale. *The vertical transportation handbook*. John Wiley & Sons, 2010.

[9] David Swinarski. Modelling elevator traffic with social distancing in a university classroom building. *Building Services Engineering Research and Technology*, 42(1):82–97, 2021.

The Driver Aide Problem: Coordinated Logistics for Last-Mile Delivery

S. Raghavan[1] and Rui Zhang[2]

[1]Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, Maryland 20742, USA, ✉ raghavan@umd.edu
[2]Leeds School of Business, University of Colorado Boulder, Colorado 80309, USA, ✉ rui.zhang@colorado.edu

Last-mile delivery is a critical portion of logistics networks accounting for approximately 30-35% of the costs. As delivery volumes have increased delivery vehicle route times have become unsustainably long. To address this issue many logistics companies (including FedEx and UPS) have resorted to using a "Driver Aide" to assist with deliveries.

In the "Driver Aide Problem", a delivery vehicle is equipped with a "driver" and an "aide". The aide can assist the driver in two modes. As a "Jumper" the aide works with the driver, thus reducing the service time at a given location. As a "Helper" the aide works independently at a location while the driver leaves (to deliver packages at other locations) and then returns. Given a set of delivery locations, travel times, and service times (based on the aide mode) the goal is to determine both the delivery route as well as the most effective mode to use the aide at different locations to minimize the total delivery time. We model this problem as an integer program with an exponential number of variables and constraints and develop a branch-cut-and-price approach for solving it. We discuss our computational experience and insights based on data provided by an industrial partner.

Optimization for queuing networks with correlated arrival flows and moving servers as models of car sharing systems

Chesoong Kim[1]

[1]Department of Business Administration, Sangji University, Wonju, 26339, Republic of Korea, ✉ dowoo@sangji.ac.kr

During the past few years, car sharing services have quickly developed in many countries for client transportation, especially in urban areas where there are many potential clients who are ready to pay for the private mobility via the short-term use of a vehicle on a per trip basis. The use of well managed car sharing services is profitable for individuals. They have enough mobility without buying or leasing of expensive cars, their maintenance, possible repair, refueling, parking, paying taxes and insurance, etc. The social profit is provided via more efficient use of parking places, the increase in the throughput of the roads and a decrease in the probability of a traffic jam, reduction of carbon emission, etc. Car sharing services may be a profitable business if they are well built and managed. One of the most important problems, which have to be resolved while starting or developing car sharing service, is to determine the number of required cars. The problem of fleet management is quite challenging and complicated. The importance and profitability of car sharing services made them popular both in real life and scientific literature.

We consider a car sharing system as an open queuing network. Customers are associated with clients. Servers correspond to the cars. Each idle car can be located in a corresponding node (zone of the operational area). If the car is busy, its current location in the network is assumed be temporarily undefined for customers because they cannot observe it. We present an exact analysis of the constructed model of a car sharing system. The study of queuing systems and networks with moving servers has not received proper attention due to the mathematical complexity of such systems and networks. As the simplest examples of such systems (with the exception of the trivial examples of systems with server's vacations, unreliable systems, and polling systems, in which a server sequentially connects to the existing queues according to a certain schedule), we can note the following two systems. One of them is the tandem system with moving servers that are dynamically redistributed between stations. A part of the available servers is permanently assigned to tandem stations, and then the remaining part is dynamically redistributed between stations depending on the ratio of the number of users present at the corresponding station. The servers are distributed over the nodes of the network and from time to time they interchange the nodes, taking the existing queues of users with them.

An essential advantage of the queuing networks that considered in the paper is the consideration of a quite general, marked Markovian arrival process (MMAP) of customers that allows us to take into account random fluctuations of the intensity of customer arrival in various nodes. Such fluctuations take place in the overwhelming majority of real car sharing systems. At the same time, the majority of the research is implemented under the assumption that the arrivals occur according to a stationary Poisson process that has a constant arrival rate. An essential novelty of the model proposed for analysis within this study is that the queuing network takes into account the server's mobility and their unavailability at any node during the service process. It is worth to note also that we allow the realistic scenario when the arriving customer meets idle servers in the target node, but he/she balks the network with a certain probability, which depends on the node and the number of available cars. Such a situation is typical in real car sharing systems because the client can refuse to occupy the available car, e.g., because his/her walking distance to the nearest car seems to be too long or the available car is not suitable to him/her by some reason.

The client of a car sharing system is considered as a customer that receives service in a queuing network. The queuing network consists of K nodes. The total number of servers in the network is equal to N. The servers can move and change their location. The numbers of idle and busy servers at each node are observed. These numbers can be changed at any instant of the start or finish of the trip of a client. We distinguish the arriving customers by the type according to the node at which a customer appears.

Namely, a type-k customer arrives at the k-th node of the network. The arrival process is assumed to be defined by the MMAP (Marked Markovian Arrival Process).

The goal of the analysis is the computation of the performance measures of the system under the fixed number of available cars N and estimation of possible variation of these measures when the number N is changed. The results of this analysis can be used for the choice of the optimal value of N. Also, these results can be helpful in answering the question: whether or not the existing distribution of the nodes of starting and finishing journey is satisfactory or it has to be somehow varied. Such variation seems to be possible, e.g., via differentiation of the tariffs for the use of cars depending on the time and the nodes of starting and finishing the journey. In the paper, the expressions for computation of the key performance measures of the system are given and numerical results are presented. They characterize the dependence of some performance measures of the network and the nodes on the total number of cars (fleet size of car-sharing system) and correlation in the arrival process.

Keywords: Queueing network, movable servers, car-sharing, marked Markovian arrival process

An improved decomposition-based heuristic for truck platooning

Boshuai Zhao[1] and Roel Leus[2]

[1]ORSTAT, KU Leuven, Leuven, Belgium, ✉ boshuai.zhao@kuleuven.be
[2]ORSTAT, KU Leuven, Leuven, Belgium, ✉ roel.leus@kuleuven.be

Truck platooning is a promising transportation mode to save fuel consumption in the freight industry. In this paper, we consider a truck platooning system for which we jointly optimize the truck route and schedule from the perspective of a central platform. We improve an existing decomposition-based heuristic by Luo and Larson [1], which iteratively solves a routing and a scheduling problem, with a cost modification step after each scheduling run. We propose different formulations for the routing and scheduling problems and embed these into Luo and Larson's framework, and we examine ways to improve their iterative process. In addition, we propose several preprocessing techniques to accelerate the solution of the scheduling problem and enable the resulting heuristic to deal with large instances. The computational results show that our procedure achieves better performance than the existing one. Moreover, based on sensitivity analysis, we find that long routes, cluster demands, and sparse networks can promote the formation of truck platoons.

References

[1] Fengqiao Luo and Jeffrey Larson. A Repeated Route-then-Schedule Approach to Coordinated Vehicle Platooning: Algorithms, Valid Inequalities and Computation. *Operations Research*, Preprint, 2021.

Session Session 5A: infrastructure networks

Thursday 9 June 2022, 15:40-17:20

Lecture Hall I

Optimal Investment Strategies for Continuous Time Opinion Dynamics in a Social Network

Swapnil Dhamal[1], Walid Ben-Ameur[2], and Tijani Chahed[3]

[1]Télécom SudParis, France, ✉ swapnil.dhamal@gmail.com
[2]Télécom SudParis, France, ✉ walid.benameur@telecom-sudparis.eu
[3]Télécom SudParis, France, ✉ tijani.chahed@telecom-sudparis.eu

Opinion dynamics is a natural phenomenon in a system of cognitive agents, and is a well-studied topic across several disciplines. Camps such as those for elections and marketing, harness this phenomenon to maximize the adoption of their opinions or products among the nodes in a social network. Since nodes update their opinions based on their neighbors' opinions, a camp aims to influence the opinions of influential nodes by investing on them by way of money, persuasion time, etc. Thus, given a budget constraint, the strategy of a camp comprises how much to invest and on which nodes. Problems related to maximizing opinion adoption in social networks have been extensively studied, however, much of the literature focuses on discrete time dynamics [2]. Among the works in the domain of continuous time opinion dynamics in social networks, most either consider simplistic models which always result in consensus, or practical models which are intractable to analyze. It is, thus, difficult to analytically study the investment strategies that a camp could devise so as to maximize its influence in the network. In this work, we propose a continuous time opinion dynamics model while accounting for camp investments, which we justify to be a logical extension of some of the well-accepted models, while also being tractable for analyzing a camp's optimal investment strategies.

In a discrete time setting, the nodes' opinions get updated in discrete time steps, while in a continuous time setting, the opinions are updated continuously with time. While there have been several models aiming to capture discrete time dynamics, the Friedkin-Johnsen model is among the most well-accepted ones that are analytically tractable. In an earlier work, we incorporated camp investment into it so as to study the model from a strategic perspective [1]. Taylor's model [3] is a continuous time equivalent of the Friedkin-Johnsen model. This work aims to extend this model to incorporate camp investment and present optimal investment strategies in a variety of settings. We now describe our model formally.

Let N be the set of all nodes, $v_i^{(t)}$ be the opinion value of a node i at time t, and w_{ij} be the weightage attributed by node i to the opinion of node j. Taylor's model is based on the hypothesis that the rate of influence on node i due to another node j at time t is proportional to the difference in their opinions at time t. As per the Taylor's model, this rate is given by $w_{ij}(v_j^{(t)} - v_i^{(t)})$. Similar to [1], we consider that node i has a biased opinion of its own (say v_i^0) and a weightage attributed to it (say w_{ii}^0); in the current context, bias could be understood as a node's reluctance to change its opinion. Hence, the rate at which the opinion value gets drawn towards the bias owing to that bias, can be written as $w_{ii}^0(v_i^0 - v_i^{(t)})$. In order to incorporate a camp's investment into the Taylor's model, we consider that the camp aims to drive the opinion values of the nodes in the social network to be as high as possible. A hypothesis on similar lines as [1] is that the rate of change of a node i's opinion at time t owing to the camp's investment on it, is directly proportional to the investment at time t (say $x_i^{(t)}$) and the weightage attributed to the camp by node i (say w_{ig}). We hence consider this rate to be $w_{ig}x_i^{(t)}$, which is in line with the multilinear nature of the Taylor's model. Combining all the aforementioned factors, we can write the rate of change of node i's opinion value as

$$\frac{dv_i^{(t)}}{dt} = \sum_{j\in N} w_{ij}(v_j^{(t)} - v_i^{(t)}) + w_{ii}^0(v_i^0 - v_i^{(t)}) + w_{ig}x_i^{(t)} = -\left(\sum_{j\in N} w_{ij} + w_{ii}^0\right) v_i^{(t)} + \sum_{j\in N} w_{ij}v_j^{(t)} + w_{ii}^0 v_i^0 + w_{ig}x_i^{(t)}$$

Let Q be the diagonal matrix with its i^{th} diagonal element being $\left(\sum_{j\in N} w_{ij} + w_{ii}^0\right)$, and let W be the matrix whose cell (i,j) has element w_{ij}. Denote $A = W - Q$. Let B and C be diagonal matrices

with their i^{th} diagonal elements as w_{ig} and w^0_{ii}, respectively. Let $V^{(t)}, X^{(t)}, V^0$ be vectors whose i^{th} components are $v_i^{(t)}, x_i^{(t)}, v_i^0$, respectively. Thus, the above equation can be written in matrix form as

$$\frac{dV^{(t)}}{dt} = AV^{(t)} + BX^{(t)} + CV^0$$

With certain practically relevant assumptions such as $w_{ij} \geq 0, w^0_{ii} > 0$, it can be shown that matrix A is nonsingular and the real parts of its eigenvalues are negative. Consider the initial conditions that the initial time $t_0 = 0$ and the vector of initial opinion values $V^{(t_0)} = V^0$. Solving the differential equation with these initial conditions, we get that the opinion vector at the end of time T is

$$V^{(T)} = e^{AT}V^0 + e^{AT}\int_0^T e^{-At}(BX^{(t)} + CV^0)dt = \left(e^{AT} - A^{-1}C + e^{AT}A^{-1}C\right)V^0 + e^{AT}\int_0^T e^{-At}BX^{(t)}dt$$

Since it is not practical for a camp to constantly change its investments as a continuous function of time, we split the time period from 0 to T into M equal intervals, where the investment stays constant within each interval k. Let the investment vector corresponding to interval k be denoted by $X^{[k]}$. So, we can formulate the fundamental problem of maximizing the sum of opinion values at time horizon T as

$$\underset{(X^{(t)})_{t\in[0,T]}}{\arg\max}\ \mathbf{1}'V^{(T)} = \underset{(X^{(t)})_{t\in[0,T]}}{\arg\max}\ \mathbf{1}'e^{AT}\int_0^T e^{-At}BX^{(t)}dt = \underset{(X^{[k]})_{k\in\{1,\ldots,M\}}}{\arg\max}\ \mathbf{1}'e^{AT}\sum_{k=1}^{M}\int_{\frac{(k-1)T}{M}}^{\frac{kT}{M}} e^{-At}BX^{[k]}dt$$

In the course of simplifying the above objective function, we arrive at the following quantity: $e^{AT}A^{-1}\left(-e^{-\frac{kT}{M}A} + e^{-\frac{(k-1)T}{M}A}\right)$, which we denote by $L^{[k]}$. Let its element corresponding to cell (i,j) be $\ell_{ij}^{[k]}$, which could be viewed as the influencing power of node j on node i in interval k. Note that $\ell_{ij}^{[k]}$ could be non-zero even if j does not have a direct link to i (owing to indirect influence via other nodes). Let $r_j^{[k]} = \sum_{i\in N}\ell_{ij}^{[k]}$, which could be viewed as the combined influencing power of node j over all the nodes in the network in interval k. It can hence be shown that the above objective function simplifies to

$$\underset{(X^{[k]})_{k\in\{1,\ldots,M\}}}{\arg\max}\ \sum_{k=1}^{M}\sum_{j\in N} x_j^{[k]}w_{jg}r_j^{[k]}$$

Now, the above objective function can lead to different optimal investment strategies for different types of budget constraints. For instance, if there is only an overall budget constraint with regard to the investment that can be made over all intervals and all nodes, an optimal strategy is to invest on only one node and in one interval, namely, $\arg\max_{j,k} w_{jg}r_j^{[k]}$. If there are constraints on investment per node (or per interval), an optimal strategy is to invest on nodes (or intervals) in decreasing values of $w_{jg}r_j^{[k]}$ while satisfying the constraints. In a more general setting wherein there are constraints on both investment per node and investment per interval in addition to an overall budget constraint, the problem can be transformed into a minimum cost flow problem which can be solved using one of the relevant algorithms.

The other problems that we study in this work are that of maximizing the sum of opinion values under diminishing returns on investments, maximizing the cumulative opinion value over the entire time period, optimizing on the number of intervals, and maximizing the sum of opinion values under uncertainty (i.e., when the parameters' values are not exactly known). We believe that this work lays a theoretical foundation for studying investment strategies in a social network in the continuous time framework, and explores the immense research potential that this area has to offer.

References

[1] Swapnil Dhamal, Walid Ben-Ameur, Tijani Chahed, and Eitan Altman. Optimal investment strategies for competing camps in a social network: A broad framework. *IEEE Transactions on Network Science and Engineering*, 6(4):628–645, 2018.

[2] Adrien Guille, Hakim Hacid, Cécile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM SIGMOD Record*, 42(1):17–28, 2013.

[3] Michael Taylor. Towards a mathematical theory of influence and attitude change. *Human Relations*, 21(2):121–139, 1968.

A Mixed Integer Linear Programming model for CO_2 emissions minimization in a waste transfer Facility Location Problem

Giulia Caselli
School of Doctorate E4E, University of Modena and Reggio Emilia, Italy
giulia.caselli@unimore.it

Giomaria Columbu
School of Doctorate E4E, University of Modena and Reggio Emilia, Italy
Iren Ambiente S.p.A., Reggio Emilia, Italy
giomaria.columbu@unimore.it

Manuel Iori
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Italy
manuel.iori@unimore.it

Carlo Alberto Magni
School of Doctorate E4E, University of Modena and Reggio Emilia, Italy
magni@unimore.it

ABSTRACT

In this work, we solve a real-world facility location problem by means of a mixed integer linear programming model. The problem is faced by an Italian multi-utility company operating in the sector of waste management. The company works in several Italian regions to collect and treat the urban waste through a network of facilities. In this problem, a set of demand points is given with a predicted quantity of waste to be collected and a fixed number of visits required over a predetermined time horizon. The flow of different classes of recyclable waste must be optimized by deciding whether and where to open additional intermediate transfer facilities among a set of dedicated points. The aim is to minimize the CO_2 emissions involved in the process, including emissions from the use of additional facilities and the transport of waste across the network. We provide a mathematical formulation for the problem, and use it to solve a real-world case study. An optimal solution is obtained with a significant reduction in CO_2 emissions and a well-structured network, proving the efficacy of the model.

1 INTRODUCTION

Waste management is a general term referring to the set of activities related to collection, transport, treatment and disposal of waste, and, in addition, control and prevention actions across the whole process. The increasing amount and complexity of waste generated by modern societies has indeed raised major sustainability-related concern around governments, firms, and individuals. As a consequence, waste management has been recently connected to environmental issues, as stated, for example, by Tolaymat et al. [27], who referred to waste management as the link between all the subjects involved in the waste production network and the societal entities taking care of environmental goals. The significant environmental impact of the waste industry is well known and reduction measures have been already introduced in many systems (e.g., ReVelle [24]) to cut the amount of greenhouse gas emissions along the process (e.g., products' recycling and salvage, collection routing optimization).

Many strategic and operational problems related to each phase of the waste management process have been studied in the literature for decades, confirming a high interest of both researchers and practitioners in this field. Operational problems refer to short-term optimization decisions like routing and scheduling problems. In this Operations Research context, the literature on the classical Vehicle Routing Problem (VRP) applied to waste collection contexts is huge (e.g., Golden et al. [13]). Strategic problems refer, instead, to medium- and long-term design and management decisions to optimize the waste collection and treatment network, including the location of different facilities (e.g., collection points, intermediate transfer facilities, final treatment facilities, and landfills). The Operations Research area of Facility Location Problems (FLPs) is addressed in this context (e.g., Van Engeland et al. [29]).

This work deals with a specific FLP in the context of urban waste collection, that is, the collection of urban waste from multiple sources and its transport to the treatment or disposal plants. The activity is typically managed by municipal services or by public or private corporations. In this work, we study the strategic problem of transfer facility location to minimize CO_2 emissions, by considering the transfer phase of urban plastics and paper waste carried out by Iren Ambiente S.p.A., an Italian multi-utility private company.

Iren Ambiente is a division of Iren Group, an industrial holding company operating in the Italian market of multi-utilities. Iren Ambiente manages the operations of waste collection, treatment and disposal, designs waste treatment and disposal systems, and controls renewable energy systems in several areas of Italian regions (mainly Emilia Romagna, Piemonte, Liguria, Lombardia, and Sardegna). Our case study refers to the region of Emilia Romagna.

The specific problem studied in this work only considers the transfer of waste, excluding downstream and upstream processes of waste production and treatment or disposal (and the related CO_2 emissions), even tough the entire waste management process is managed by the company in its assigned areas.

The problem is a particular Capacitated Facility Location Problem (CFLP) (i.e., a FLP where facilities have limited capacity), which we solve by means of a Mixed Integer Linear Programming (MILP) model. The goal is to determine the optimal network in real-world scenarios, evaluating how many intermediate transfer locations must be opened and where they must be located to minimize the

total CO_2 emissions produced during the process (i.e., facilities' and vehicles' emissions) over a fixed time horizon.

The rest of the paper is organized as follows. A summary of the related literature is given in Section 2. In Section 3, we provide a detailed description of the problem and present our mathematical model. Section 4 reports the results obtained by the proposed solution method on our real-world case study. Concluding remarks are provided in Section 5.

2 LITERATURE REVIEW

Reverse logistics commonly refers to the set of activities and processes related to the flow of raw material, inventory, and finished goods other than waste from the point of consumption back to the origin point. Back in 1998, Carter and Ellram [3] already referred to reverse logistics as the practice whereby firms can become more environmentally efficient by, for example, recovering material and recycling products. A review on reverse logistics is given by Govindan et al. [15]. A restricted field of study on environment-related reverse logistics problems is labelled in the literature as "green logistics". A review on green logistics and related combinatorial optimization problems is given by Sbihi and Eglese [25].

Waste management is the reverse logistic sub-field of study focusing only on waste, and more commonly on solid waste (e.g., Beliën et al. [2]). In their recent survey, Van Engeland et al. [29] reviewed the literature of the so-called waste reverse supply chain, identified as the overlapping subject between waste management and the broader reverse logistic. Collection, transportation, recovery, and disposal of waste are included in the waste reverse supply chain, where several problems are solved with the aim of creating value at three different levels of management decisions: (i) long-term strategic decisions of waste network design; (ii) medium-term decisions (e.g., waste quantities and capacity allocation), and (iii) short-term operational decisions like routing and scheduling. In our work, we are interested only in the first decision level.

Focusing on the area of long-term strategic network design problems, Van Engeland et al. [29] surveyed the extensive literature of the period 1995-2020, providing a classification of strategic network design problems and their combinatorial optimization solution methods based on several characteristics: single- or multi-period decisions, single- or multi-product problems, single- or multi-objective optimization, with specific constraints and different objective functions. They found that 60% of collected works dealt with single-objective functions, representing cost minimization or profit maximization in around 95% of the cases. Only one out of 133 articles surveyed in this work was categorized as a carbon emissions-related single-objective model. In this regard, our work is an attempt to extend this branch of the literature. In multi-objective models, instead, environmental goals are often included and balanced with economic ones, such as the minimization of CO_2 emissions or energy use. Talaei et al. [26] solved a bi-objective cost-emissions minimization facility location-allocation problem for a closed-loop supply chain. They developed an ϵ-constrained method where the higher priority cost function was used as the objective function, and the CO_2 emissions function formed the ϵ-based constraints. Sometimes, the social impact is also included in multi-objective problems. For example, Govindan et al. [14] proposed a multi-objective MILP model for the closed-loop supply chain network of a generic product recovery system, including in the objective function the minimization of carbon emissions and social impact, together with the maximization of revenues. Mirdar Harijani et al. [21] developed a multi-objective model for sustainable recycling of municipal solid waste with similar economic, environmental, and social goals.

In the Operations Research literature, the strategic problems defined above are classified under the broad category of FLPs; they have been of great interest since the 1960s. In their book, Farahani and Hekmatfar [9] defined FLP as the problem of locating a set of facilities (resources) to minimize the cost of satisfying some set of demands (of the customers) with respect to some set of constraints. The authors surveyed different FLP families and related discrete and continuous optimization algorithms, along with case studies from private and public firms.

As reported by Verter [30], the classical FLP has been extended in a number of ways by, for example: (i) increasing the number of products, from single- to multi-commodity FLPs (see, e.g., Liu et al. [19], who studied a complex multi-commodity CFLP involving sustainability concerns); (ii) increasing the number of facility echelons, that is, the types of facility to locate (e.g., Gendron and Semet [11]); (iii) increasing the number of time periods included in the model, defining dynamic FLPs where the facility location is determined at each period so as to minimize the total cost over time (e.g., Nickel and Saldanha-da Gama [23]); and (iv) incorporating possible scale and scope economies in the cost function (e.g., Wu et al. [31]) and uncertainties (e.g., Correia and Saldanha-da Gama [4]).

For what concerns FLPs in the waste management industry, Adeleke and Olukanni [1] surveyed important models and solution algorithms, published between 2006 and 2020 and adapted to deal with several optimization problems. These problems are usually formulated by MILP models, but then solved in practice by means of heuristic algorithms able to find near-optimal solutions in a limited time. In the following, we mention some case studies in the urban waste management contexts. Ghiani et al. [12] studied a bin allocation problem in Italy where the aim is to minimize the total number of activated waste collection sites. The problem was solved by means of a MILP model and a constructive heuristic. Lee et al. [18] proposed several mathematical models for the waste management system of Hong Kong, in which they minimize the total cost for the municipal solid waste management system (i.e., daily waste management costs, transportation costs, net of the revenues from incinerators). Dimitrijević et al. [6] applied a bi-objective optimization model to a landfills' location problem belonging to the class of so-called Undesired FLPs, where the economic objective asks for minimizing total costs (i.e., costs generated by establishing new facilities and satisfying the demand) while the social objective concerns the total number of end users undesirably influenced by new landfills. Gambella et al. [10] studied a facility location and waste flow allocation problem. They developed a stochastic programming model that was applied to solve a real-world Italian case study.

A relevant part of the current literature on waste transfer FLPs includes environmental concerns in the problem statement and in the model formulation. A summary of the main concepts and models for the so-called Green FLP was included in 2017 by Martínez and Fransoo [20]. The focus of their work is on the transportation performance of firms in terms of both costs and emissions, which is

strongly determined by the design of the network. In the models under review, the main sources of CO_2 emissions associated with the location of facilities derive from both mobile sources (transportation) and stationary sources (production, storage, and handling). In 2002, environmental qualitative and quantitative evaluations have been combined with a MILP model and multi-criteria methods by Vaillancourt and Waaub [28] to solve waste facility location problems similar to our problem. Valuable results have been obtained on a case study of the city of Montreal, Canada. Some more recent real-world examples of green FLPs are the works by Mohsenizadeh et al. [22], who solved a bi-objective cost-pollution transfer stations location problem in Ankara, by Kudela et al. [17] who addressed a case study for the Czech Republic, and by Eiselt and Marianov [8], who minimized a bi-objective cost-pollution function in a real-world Chilean landfills' location problem.

As stressed by Martínez and Fransoo [20], not many companies have implemented in practice facility locations strategies to reduce their environmental impact, especially as primary goal, although a considerable amount of theoretical work is already available in the literature. In this respect, our work is an attempt to provide an example of application of green FLPs in practice and contribute to the relevant literature.

3 PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

In this section, we describe the CFLP addressed by Iren Ambiente, introduce the mathematical notation, and present the proposed MILP model that we developed.

With the aim of optimizing the waste transfer logistics in a specific area, the number of demand points to be visited (i.e., locations, number of visits, and types and quantities of waste) is assumed to be known and fixed over a limited time horizon (one year) as estimated from past data and budgets. The number of intermediate transfer locations and final treatment locations to open and the flow across the network are the decisions to be made in the problem. Representing a real-world scenario, we assume that some facilities are already open and must stay open even in the optimized scenario. One or more locations are given as candidate locations where a facility can be opened (e.g., areas already owned by the company and activated in the past). The goal is to find the optimal waste transfer network (i.e., locations and flow) that minimizes the total CO_2 emissions involved in the overall process of collection, transfer, and delivery of waste to final treatment plants.

A set I of customers to be visited (i.e., demand points where waste must be collected) is given over a limited time horizon. The total estimated amount of waste produced by each customer over the considered time period must be collected and delivered to final treatment facilities, either directly or passing by one or more intermediate transfer facilities. We call J the set of all candidate facility locations, and J_e the subset of locations where a facility already exists and is open. In addition, we call J_1 the subset of intermediate candidate facility locations. We denote by H the set of recyclable waste types, such as paper, plastics, and glass (i.e., we deal with a multi-commodity CFLP).

Let q_{ih} indicate the total estimated quantity of waste type h to be collected from customer i over the considered time period, and let n_{ih} represent the corresponding required number of trips. The value of n_{ih} is predetermined by the municipality (e.g., paper waste is collected once or twice per week). For each facility location j, we define Q_j as the overall waste capacity and Q_{jh} as the capacity for waste type h, where $Q_j \leq \sum_{h \in H} Q_{jh}$. For example, if a location j has $Q_j = 100$ tons and $Q_{jh} = [70, 60]$ tons for $h = 2$, then we can dedicate half capacity to each waste (i.e., $[50, 50]$ tons), or we can accept an unbalanced solution without exceeding the Q_{jh} values (e.g., $[70, 30]$ tons).

For CO_2 emissions minimization, we introduce the parameters e_{ijh}, for $i \in I, j \in J, h \in H$, and e'_{jkh}, for $j \in J_1, k \in J, h \in H$. The former estimates transport emissions for waste type h from customer i to facility j, while the latter from intermediate facility j to facility k. Parameters e_{ijh} consider, for each customer i, the type of vehicle that serves it, the time required by the vehicle to go from customer i to facility j, its fuel consumption, the conversion factor, and the number n_{ih} of required trips over the selected time horizon (i.e., $[\frac{kgCO_2}{year}]$). Parameters e'_{jkh} consider the capacity of the loading vehicle in use at each facility j, its time to move from facility j to facility k, and its fuel consumption to compute the carbon emissions for a single tripe from j to k (i.e., $[\frac{kgCO_2}{ton}]$).

In addition, let p_{jh} and F_j represent, respectively, variable and fixed components of emissions generated by opening a new facility $j \in J$. For each facility we estimate a different percentage of emissions for fixed and variable components, depending on the energy consumption of the operating machines in the facility, i.e., the equipment used for loading, unloading, and moving waste. The greater the amount of waste collected by a single facility, the greater the environmental benefit from opening that facility.

To formulate our MILP model, we introduce a set of three-index variables x_{ijh} that indicate the fraction of waste of type h collected at customer $i \in I$ and transferred to location $j \in J$ in the overall considered period. The variable is continuous, implying that each customer demand can be fulfilled by one or more facilities. An additional set of non-negative continuous variables f_{jkh} represents the flow of waste transferred from intermediate location $j \in J_1$ to intermediate/final location $k \in J, j \neq k$. Then, binary variables y_j take the value 1 if location $j \in J$ is open, and 0 otherwise.

The problem is modelled as follows:

$$\min \sum_{i \in I} \sum_{j \in J} \sum_{h \in H} e_{ijh} x_{ijh} + \sum_{j \in J_1} \sum_{\substack{k \in J \\ j \neq k}} \sum_{h \in H} e'_{jkh} f_{jkh} + \\ + \sum_{k \in J} \sum_{h \in H} p_{kh} \Big(\sum_{i \in I} q_{ih} x_{ikh} + \sum_{\substack{j \in J_1 \\ j \neq k}} f_{jkh} \Big) + \sum_{j \in J} F_j y_j \quad (1)$$

$$\text{s.t.} \quad \sum_{j \in J} x_{ijh} = 1 \qquad i \in I, h \in H \quad (2)$$

$$\sum_{i \in I} q_{ih} x_{ikh} + \sum_{\substack{j \in J_1 \\ j \neq k}} f_{jkh} \leq Q_{kh} y_k \qquad k \in J, h \in H \quad (3)$$

$$\sum_{i \in I} \sum_{h \in H} q_{ih} x_{ikh} + \sum_{\substack{j \in J_1 \\ j \neq k}} \sum_{h \in H} f_{jkh} \leq Q_k y_k \qquad k \in J \quad (4)$$

$$\sum_{i \in I} q_{ih} x_{ikh} + \sum_{\substack{j \in J_1 \\ j \neq k}} f_{jkh} - \sum_{\substack{l \in J \\ l \neq k}} f_{klh} = 0 \qquad k \in J_1, h \in H \quad (5)$$

$$y_j = 1 \qquad j \in J_e \quad (6)$$

$$0 \leq x_{ijh} \leq 1 \qquad i \in I, j \in J, h \in H \quad (7)$$

$$f_{jkh} \geq 0 \qquad j \in J_1, k \in J, j \neq k, h \in H \quad (8)$$

$$y_j \in \{0, 1\} \qquad j \in J \quad (9)$$

The objective function (1) minimizes the total amount of CO_2 emissions involved in the process in one year: the first two terms count the emissions from vehicles' travels for customer-facility and facility-facility trips respectively, while the third and fourth terms consider the emissions generated by the working facilities, addressing variable and fixed components of CO_2 emissions respectively. Constraint (2) ensures that each waste demand is fulfilled by one or more facilities sharing its entire demand over the considered period. Constraints (3) and (4) represent the overall and waste-specific capacity constraints for each facility j. Constraint (5) guarantees the conservation of flow for each intermediate facility (i.e., the total incoming flow from customers and other intermediate facilities is equal to the flow going out to other facilities). Constraint (6) imposes that existing facilities are kept open by the model solution. Constraints (7)-(9) give the domain of the variables.

4 CASE STUDY

In this section, we study the performance of model (1)-(9) on a realistic instance from Iren Ambiente. Our model has been coded in the Mosel language and solved with FICO Xpress Solver 64bit v8.9.0 on an Intel Core i7, 1.80 GHz, with 16 GB of RAM memory, running under Windows 10 64 bits.

The company collects the waste of 34 municipalities serving approximately 460,000 inhabitants in the province of Reggio Emilia, for a total quantity of nearly 330,000 tons of urban waste per year. Such activity consists in moving vehicles from the depots, collecting waste from the municipalities and transporting them to the final treatment or disposal plants, and bringing back the vehicles to the original depots. This does not exclude the possibility of including intermediate transfer facilities, so as to allow waste flows from municipalities to intermediate points, between intermediate points, and from intermediate points to final plants. The company is currently building a new final treatment facility close to the Reggio Emilia district, where the entire volume of plastic and paper urban waste from the entire area of the district could be possibly conferred. The company wants to optimize the future waste collection network, also evaluating whether and where it could be convenient to open additional intermediate waste transfer facilities.

In this strategic decision, the company has been moved, first and foremost, by environmental issues rather than economic concern, under the increasing pressure of the European Union and other international entities for providing more environmentally efficient waste management solutions (see, e.g., [7] and [16]).

In our case study, we use the data on paper and plastic waste collected in the province of Reggio Emilia over a one-year time period (i.e., from October 2019 to October 2020), which is also the time horizon adopted for our model. A preliminary analysis has confirmed that the emergency situation due to Covid-19 pandemics did not affect the urban waste industry significantly, nor was Iren Ambiente's business specifically impacted. In Reggio Emilia district, there are several waste collection points distributed over 34 municipalities. As requested by the company, we aggregate points to find stable solutions by merging similar weekly repetitive patterns of waste collection services. Municipalities are clustered based on several characteristics (e.g., distance, number of inhabitants, balanced number of units per cluster, business constraints). From the clustering, we obtain 18 clusters.

Each cluster represents a service to be fulfilled over the considered time period; it is characterized by: (i) a total predicted quantity of waste to be collected for paper and plastic, respectively; and (ii) a required number of visits and a type of serving vehicle (assumed to serve at full capacity for the sake of simplicity), which, in our analysis, is significant only for the overall impact on CO_2 emissions. The total demand for waste collection from the 18 clusters in a year is equal to 27,532 tons of paper and 16,098 tons of plastic.

The overall capacity of the one final treatment location under construction is equal to 50,000 tons, distributed as 32,000 and 18,000 tons for paper and plastic, respectively. Two candidate locations for intermediate transfer facilities are considered following the strategic guidelines of the company and the clustering logic. For each possible location, 16 different options for its paper and plastic capacities are considered. Total capacities of intermediate candidate facilities range between 540 and 32,000 tons of waste.

In the computation of CO_2 emissions, we consider fuel consumption and gross weight of the different diesel Euro 6 vehicles (ranging between 3 and 15 liters per hour, and 3.5 and 38 tons, respectively). Fuel consumption is converted in CO_2 emissions with the conversion factor (2.63 kg CO_2 per l) taken by (DEFRA) [5]. The number of visits for each cluster is computed over the entire period starting from given weekly requirements (e.g., one or two visits per week for plastic). For emissions generated by building and opening a new facility, we consider some estimations of consumption and productivity done by the company, and a different percentage addressed to the fixed and variable components for each candidate location (e.g., 14% and 86%, respectively).

Our model was able to find an optimal solution in 0.83 seconds of CPU time. We compare two optimal solutions: the one generated by the model assuming additional intermediate transfer locations and the one generated by the model in the absence of intermediate facilities (i.e., the final treatment facility collects the whole waste flow of Reggio Emilia district). The networks resulting from the two solutions are represented in Figures 1 and 2. Red points indicate facilities, with names in black color for final treatment facilities and in purple color for intermediate transfer facilities. Green points indicate the centroids of the municipalities. Waste flows are represented by dashed lines and numbers (tons of waste), where plastic waste flows are in red and paper waste flows are in blue.

Figure 1 represents the optimal solution on the reduced instance with the only final treatment facility (i.e., "C1 RE") open. This first network connects all the clusters to the final treatment plant, with an overall amount of 1,407.7 tons of CO_2 emissions. The solution has been obtained by the model in 0.15 seconds of CPU time.

Figure 2 represents the optimal flow on the complete instance. The model opens two intermediate locations both for paper and

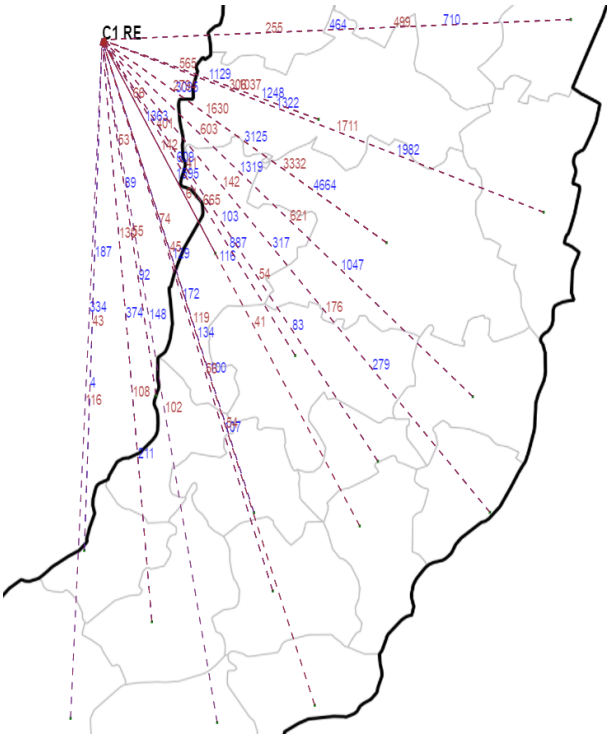


Figure 1: Solution without intermediate facilities

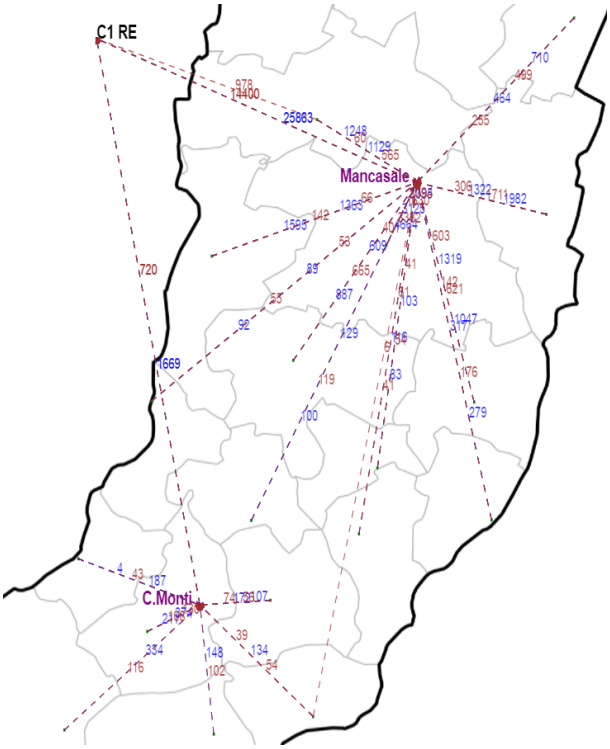


Figure 2: Solution with intermediate facilities

Table 1: Summary of case study results on plastic waste

	Plastic waste		
Facility	C [ton]	F^a [ton]	Sat.
Final *"C1 RE"*	18,000	16,098	89%
Intermediate *"Mancasale"*	14,400	14,400	100%
Intermediate *"C. Monti"*	720	720	100%

[a] Total incoming flow from collection points and intermediate facilities

Table 2: Summary of case study results on paper waste

	Paper waste		
Facility	C [ton]	F^a [ton]	Sat.
Final *"C1 RE"*	32,000	27,532	86%
Intermediate *"Mancasale"*	28,800	25,863	90%
Intermediate *"C. Monti"*	1,920	1,669	87%

[a] Total incoming flow from collection points and intermediate facilities

plastic waste in the two different candidate locations (i.e., "Mancasale" and "C. Monti"). In this extended network, we obtain a 25% reduction in the total amount of CO_2 emissions with respect to the value of the previous solution, and an efficient flow of waste. Almost all the waste flow is delivered to intermediate facilities, apart from a fraction generated by one cluster , which is the nearest one to the final plant in the north-west area of the district. The southern intermediate facility in "C. Monti" has lower capacities, in accordance to the lower quantities of waste produced by the clusters in the south of the district (mainly upland) with respect to the north area (which concentrates the waste produced by the main city of the district, Reggio Emilia, and other urban areas).

Tables 1-2 report the results obtained by the model on the case study in terms of overall capacity (C) and incoming flow (F) for each open intermediate location, expressed in tons of waste for plastic and paper, respectively. For the sake of clarity, results for the final treatment facility are also included, although its capacity was predetermined by the model input data. Plants' saturation (Sat.) is also reported, proving the efficiency of the solution with all facilities' capacity almost saturated. Indeed, they are all greater than 85%, and equal to 100% in the case of intermediate facilities for plastic waste. Note that saturation of travels between facilities is important because waste from different travels converge to intermediate facilities before being transported to the final one. The more the intermediate facilities are saturated, the more the second-level trips (i.e., from intermediate to final facility) can be aggregated.

5 CONCLUSIONS AND FUTURE RESEARCH

In this work, we have studied a real-world capacitated facility location problem occurring in an Italian multi-utility company. The goal is to define the optimal network of final treatment and intermediate transfer facilities in the district of Reggio Emilia (Italy) for plastic and paper waste collection to minimize CO_2 emissions generated by the on-road transfer of waste with heavy vehicles and by the opening of new facilities.

We have modelled the problem by means of a mixed integer linear programming formulation, where binary variables define which facility should be open among a set of candidate facility locations

and which capacity should be employed, and continuous variables define the flow between customers and facilities and between different facilities. The model enables dividing the waste demand of each customer among one or more facilities, ensuring that the total and waste-specific capacity constraints for each facility are satisfied.

The model is general, so as to be applied to different case studies of waste collection that appears in the literature and in different real-world contexts. On the other hand, the model considers environmental issues as the single objective. While not so common in the literature, it is nonetheless very important, considering the current situation of the increasing waste industry and the corresponding governments' attention to this topic.

We have solved a real-world case study of our industrial partner by means of the proposed mathematical model, obtaining significant results on an aggregated one-year instance. The model has provided an optimal solution in less than 1 second of computation, with a 25% reduction in CO_2 emissions with respect to the case in which only one final facility (currently under construction) is open. The resulting flow and saturation of new facilities are well-structured. To further test our model, it would be interesting also to solve each daily instance of the considered year, and then sum the results over the total number of days. That could provide more insight on the solution obtained for the single aggregated instance.

In view of the good results, we plan to further work on this problem. First, we plan to study a bi-objective formulation for the problem in order to consider economic as well as environmental goals. The idea is to investigate the investment in new facilities and measure the economic value created (as expressed by the project Net Present Value) also performing a sensitivity analysis for detecting the parameters that have the greatest impact on the objective function.

Moreover, we plan to provide a more extensive computational evaluation of our model, testing it on other real-world scenarios and on more complex random instances, to better evaluate the performance and scalability of the proposed model.

REFERENCES

[1] Olawale J Adeleke and David O Olukanni. 2020. Facility location problems: models, techniques, and applications in waste management. *Recycling* 5, 2 (2020), 10.

[2] Jeroen Beliën, Liesje De Boeck, and Jonas Van Ackere. 2014. Municipal solid waste collection and management problems: a literature review. *Transportation Science* 48, 1 (2014), 78–102.

[3] Craig R Carter and Lisa M Ellram. 1998. Reverse logistics: a review of the literature and framework for future investigation. *Journal of Business Logistics* 19, 1 (1998), 85.

[4] Isabel Correia and Francisco Saldanha-da Gama. 2019. Facility location under uncertainty. In *Location Science*. Springer, 185–213.

[5] Department for Environment Food & Rural Affairs (DEFRA). 2019. Measuring and reporting environmental impacts: guidance for businesses. https://www.gov.uk/guidance/measuring-and-reporting-environmental-impacts-guidance-for-businesses Last updated 31 January 2019.

[6] Branka Dimitrijević, Branislava Ratković, and Milica Šelmić. 2017. A multi-objective model for undesirable facility location. (2017).

[7] EC Directive et al. 2008. Directive 2008/98/EC of the European Parliament and of the Council of 19 November 2008 on waste and repealing certain Directives. *Official Journal of the European Union L* 312, 3 (2008).

[8] Horst A Eiselt and Vladimir Marianov. 2014. A bi-objective model for the location of landfills for municipal solid waste. *European Journal of Operational Research* 235, 1 (2014), 187–194.

[9] Reza Zanjirani Farahani and Masoud Hekmatfar. 2009. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media.

[10] Claudio Gambella, Francesca Maggioni, and Daniele Vigo. 2019. A stochastic programming model for a tactical solid waste management problem. *European Journal of Operational Research* 273, 2 (2019), 684–694.

[11] Bernard Gendron and Frédéric Semet. 2009. Formulations and relaxations for a multi-echelon capacitated location–distribution problem. *Computers & Operations Research* 36, 5 (2009), 1335–1355.

[12] Gianpaolo Ghiani, Demetrio Laganà, Emanuele Manni, and Chefi Triki. 2012. Capacitated location of collection sites in an urban waste management system. *Waste Management* 32, 7 (2012), 1291–1296.

[13] Bruce L Golden, Arjang A Assad, and Edward A Wasil. 2002. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In *The vehicle routing problem*. SIAM, 245–286.

[14] Kannan Govindan, Prakash C Jha, and Kiran Garg. 2016. Product recovery optimization in closed-loop supply chain to improve sustainability in manufacturing. *International Journal of Production Research* 54, 5 (2016), 1463–1486.

[15] Kannan Govindan, Hamed Soleimani, and Devika Kannan. 2015. Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. *European Journal of Operational Research* 240, 3 (2015), 603–626.

[16] Repubblica Italiana. 2006. Decreto Legislativo 3 aprile 2006, n. 152 (Codice Ambientale).

[17] Jakub Kudela, Radovan Šomplák, Vlastimír Nevrlỳ, Tomáš Lipovskỳ, Veronika Smejkalová, and Ladislav Dobrovskỳ. 2019. Multi-objective strategic waste transfer station planning. *Journal of Cleaner Production* 230 (2019), 1294–1304.

[18] CKM Lee, CL Yeung, ZR Xiong, and Sai Ho Chung. 2016. A mathematical model for municipal solid waste management–A case study in Hong Kong. *Waste Management* 58 (2016), 430–441.

[19] Weiwei Liu, Nan Kong, Mingzheng Wang, and Lingling Zhang. 2021. Sustainable multi-commodity capacitated facility location problem with complementarity demand functions. *Transportation Research Part E: Logistics and Transportation Review* 145 (2021), 102165.

[20] Josué C Velázquez Martínez and Jan C Fransoo. 2017. Green facility location. In *Sustainable Supply Chains*. Springer, 219–234.

[21] Ali Mirdar Harijani, Saeed Mansour, and Behrooz Karimi. 2017. A multi-objective model for sustainable recycling of municipal solid waste. *Waste Management & Research* 35, 4 (2017), 387–399.

[22] Melika Mohsenizadeh, Mustafa Kemal Tural, and Elcin Kentel. 2020. Municipal solid waste management with cost minimization and emission control objectives: A case study of Ankara. *Sustainable Cities and Society* 52 (2020), 101807.

[23] Stefan Nickel and Francisco Saldanha-da Gama. 2019. Multi-period facility location. In *Location Science*. Springer, 303–326.

[24] Charles ReVelle. 2000. Research challenges in environmental management. *European Journal of Operational Research* 121, 2 (2000), 218–231.

[25] Abdelkader Sbihi and Richard W Eglese. 2007. Combinatorial optimization and green logistics. *4OR* 5, 2 (2007), 99–116.

[26] Mohammad Talaei, Babak Farhang Moghaddam, Mir Saman Pishvaee, Ali Bozorgi-Amiri, and Sepideh Gholamnejad. 2016. A robust fuzzy optimization model for carbon-efficient closed-loop supply chain network design problem: a numerical illustration in electronics industry. *Journal of cleaner production* 113 (2016), 662–673.

[27] Thabet Tolaymat, Amro El Badawy, Reynold Sequeira, and Ash Genaidy. 2015. A system-of-systems approach as a broad and integrated paradigm for sustainable engineered nanomaterials. *Science of the Total Environment* 511 (2015), 595–607.

[28] Kathleen Vaillancourt and Jean-Philippe Waaub. 2002. Environmental site evaluation of waste management facilities embedded into EUGENE model: A multicriteria approach. *European Journal of Operational Research* 139, 2 (2002), 436–448.

[29] Jens Van Engeland, Jeroen Beliën, Liesje De Boeck, and Simon De Jaeger. 2020. Literature review: Strategic network optimization models in waste reverse supply chains. *Omega* 91 (2020), 102012.

[30] Vedat Verter. 2011. Uncapacitated and capacitated facility location problems. In *Foundations of Location Analysis*. Springer, 25–37.

[31] XY Wu, Gordon H Huang, Jianan Liu, and Jianbing Li. 2006. An interval nonlinear program for the planning of waste management systems with economies-of-scale effects—a case study for the region of Hamilton, Ontario, Canada. *European Journal of Operational Research* 171, 2 (2006), 349–372.

Towards the Solution of Robust Gas Network Optimization Problems Using the Constrained Active Signature Method

Timo Kreimeier
Humboldt-Universität zu Berlin
Berlin, Germany
timo.kreimeier@hu-berlin.de

Martina Kuchlbauer
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Erlangen, Germany
martina.kuchlbauer@fau.de

Frauke Liers
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Erlangen, Germany
frauke.liers@fau.de

Michael Stingl
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Erlangen, Germany
michael.stingl@fau.de

Andrea Walther
Humboldt-Universität zu Berlin
Berlin, Germany
andrea.walther@math.hu-berlin.de

ABSTRACT

This work studies robust gas network optimization under uncertainties in demand and in the physical parameters. The corresponding optimization problems are nonconvex in node pressures and flows along the pipes. They are thus very difficult to solve for realistic instance sizes. In recent approaches, an adaptive bundle method has been developed, where one solves the occurring adversarial problems via iteratively refined piecewise linear relaxations. These subproblems need to be solved always from scratch using mixed-integer linear programming (MIP). As alternative to the MIP solver, we employ here a nonsmooth optimization approach that allows a warm start strategy such that it can profit from the results obtained for coarser relaxations. We evaluate the approach for realistic gas network topologies and outline possibilities for future research.

1 INTRODUCTION

Resource-efficient distribution of energy is one of the grand challenges of modern times. In the current transformation of the energy system, natural gas is considered as a transition technology that is used to ensure stable and resilient energy supply. Furthermore, in the future current energy sources may be combined or even replaced by hydrogen. Optimization of the respective energy networks is of major importance [5].

In addition, optimization of gas network operation should be hedged against uncertainties that are inherent in the energy demands and in the physical parameters of gas transport. In particular when demand distributions are unknown (e. g. if they are market-driven) or when uncertainties cannot be measured easily (which for example is true for the roughness in the pipes), protection is sought in the sense of robust optimization. A particular mathematical challenge consists in the fact that the relation between gas pressure at the network nodes and gas flow along the pipes is nonconvex quadratic. Thus, in order to determine a best possible operation of the active elements in the network such as compressors and valves, a robust nonconvex optimization problem needs to be solved. To this end, several robust optimization approaches have been established, see, e.g., [1, 12]. An efficient solution approach is given by an adaptive bundle method [7] that can cope with the nonconvexities. It is integrated within an outer-approximation scheme that is able to decide on discrete as well as continuous decisions for operating the active elements [8]. To solve the adversarial problems that arise within the bundle method, the nonconvex expressions are replaced by iteratively refined piecewise linear relaxations. The latter are modeled via a mixed-integer linear optimization problem (MIP). In each iteration, the MIP is refined until a predefined error guarantee on the quality of the relaxation is given. Typically, after applying small changes in a MIP, the corresponding simplex based branched-and-bound approaches do not allow any warm start strategy as they usually cannot profit from earlier iterations. Therefore, in [7], the MIPs are always solved from scratch using available MIP solvers.

In this work, we advance this method by replacing the MIP solver by an approach called CASM for Constrained Active Signature Method. CASM is tailored to solve optimization problems where the objective function as well as the constraints are continuous and piecewise linear. In contrast to the MIP solvers that always need to solve the problems from scratch, the CASM approach allows a warm start based on the optimization results obtained for a coarser approximation of the nonconvex expressions. The goal of this work is to apply CASM to the problem of robust operation of gas network operation in order to evaluate its applicability.

The structure of this paper is as follows. In the next section, the considered problem stemming from gas transport is introduced. Section 3 presents CASM in more detail including also a description of the warm start option. Numerical results are discussed in Sec. 4. Section 5 contains a conclusion and an outlook.

2 THE GAS TRANSPORT PROBLEM

We consider a problem that arises in the context of gas networks, namely the stationary robust gas transport problem. For a profound explanation of models and solution approaches for gas transport problems, we refer to [5].

Here, we consider the problem of finding an optimal control that is robustly protected against the perturbation of physical parameters. We aim for a minimum-cost control of compressors. Constraints are thereby that all demands should be satisfied and no physical constraints should be violated. The control of active elements can be modeled as here-and-now variables at the first stage and the realization of physical states as wait-and-see variables at the second stage. The realization of physical states takes place after uncertain parameters realize themselves. As uncertain parameters, we consider demands and pressure loss coefficients, where the latter is due to uncertain frictions of the pipes. For every possible realization of the pressure loss coefficients, physical feasibility of the gas transport has to be maintained by the network operator. In the following, we model the arising robust gas transport problem from the point of view of the network operator.

We describe a gas network by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the arcs model pipes and compressors ($\mathcal{A} = \mathcal{A}_{pi} \cup \mathcal{A}_c$) and an incidence matrix $A \in \{-1, 0, 1\}^{|\mathcal{V}|\times|\mathcal{A}|}$. We denote the gas flow by $q \in \mathbb{R}^{|\mathcal{A}|}$, where its sign indicates the flow's direction. Further, squared pressure values are denoted by $\pi \in \mathbb{R}^{|\mathcal{V}|}$. To ensure uniqueness of the physical states, we fix the pressure value at one so-called root node. By $w(\Delta)$, we denote the costs of a control Δ of compressors. We use a linear compressor model, so that a value Δ_a induces a pressure increase of Δ_a at compressor a. In total, we consider the following robust optimization problem:

$$\min_{\Delta\in[\underline{\Delta},\overline{\Delta}]} \max_{(d,\lambda)\in\mathcal{U},\pi,q} w(\Delta) + \sum_{v\in V} \max\{0, \underline{\pi}_v - \pi_v, \pi_v - \overline{\pi}_v\} \tag{1a}$$

$$\text{s.t.} \quad Aq = d \tag{1b}$$

$$(A^T\pi)_a = \Delta_a \qquad \forall a \in \mathcal{A}_c \tag{1c}$$

$$(A^T\pi)_a = -\lambda_a q_a |q_a| \qquad \forall a \in \mathcal{A}_{pi} \tag{1d}$$

$$(q, \pi) \in \mathbb{R}^{|\mathcal{A}|} \times \mathbb{R}^{|\mathcal{V}|}. \tag{1e}$$

Thereby, the uncertainty set $\mathcal{U}$ is defined as follows:

$$\mathcal{U} := \{(d, \lambda) \mid \lambda \in [\underline{\lambda}, \overline{\lambda}], d_i \in [\underline{d}, \overline{d}], \sum_{i=1}^{n} d_i = 0\}.$$

Fixing the uncertain parameters to some values, there is a unique physical state, i.e., unique flow and pressure variables, that fulfills the physical constraints [1, 2]. Due to this fact, we can reformulate (1) as a box-constrained optimization problem writing the pressure as a function of the other parameters (see [7]):

$$\min_{\Delta\in[\underline{\Delta},\overline{\Delta}]} \max_{(d,\lambda)\in\mathcal{U}} \; w(\Delta) + \sum_{v\in V} \max\{0, \underline{\pi}_v - \pi_v(\Delta; d, \lambda), \pi_v(\Delta; d, \lambda) - \overline{\pi}_v\}.$$

In [7], an adaptive bundle method is developed to solve problems of this kind. For this purpose, the bundle method is applied to the outer minimization problem with the optimal value function of the inner maximization problem as objective function. As in every iteration of the bundle method, an approximate function evaluation is required, the inner maximization problem has to be approximately solved in every iteration. This inner adversarial problem is the following nonconvexly constrained optimization problem:

$$\max_{(d,\lambda)\in\mathcal{U},\pi,q} \sum_{v\in\mathcal{V}} \max\{0, \underline{\pi}_v - \pi_v, \pi_v - \overline{\pi}_v\} \tag{2a}$$

$$\text{s.t.} \quad Aq = d \tag{2b}$$

$$(A^T\pi)_a = \Delta_a \qquad \forall a \in \mathcal{A}_c \tag{2c}$$

$$(A^T\pi)_a = -\lambda_a q_a |q_a| \qquad \forall a \in \mathcal{A}_{pi} \tag{2d}$$

$$(q, \pi) \in \mathbb{R}^{|\mathcal{A}|} \times \mathbb{R}^{|\mathcal{V}|}. \tag{2e}$$

In [7] this adversarial problem is approximately solved via piecewise linear relaxation. The adaptive bundle method only allows for a certain error in the optimal objective value. As a relaxation that fulfills a requested error bound, for each of the pressure loss constraints, piecewise linear relaxation via the delta method [1, 3, 9] is used. In the adaptive bundle method [7], an error bound on the optimal objective value of the adversarial problem is requested and a consequent bound for the error in the pressure loss constraints is provided. As this theoretical bound turned out to be not very tight, the strategy in [7] is to allow for large errors in the constraints and to refine in case of a too large a posteriori error in the objective (cf. [7, Section 5.1.1, 5.1.2]). In [7], it is noted that the run time of the bundle method is largely determined by the solution of the adversarial problem up to the requested error, where the piecewise linearly relaxed adversarial problems are solved via MIP solvers. This motivates the development and analysis of an alternative solution strategy for these piecewise linear problems in the present paper. In particular, as by the use of the refinement strategy, sequences of refined relaxations are solved, a method that allows for warm start strategies has the potential to speed up computations.

3 THE OPTIMIZATION APPROACH CASM

In [6] the so-called Constrained Active Signature Method (CASM) for solving constrained piecewise linear optimization problems was introduced and analyzed in detail. Therefore, here we just introduce its main aspects briefly. Based on results contained, e.g., in [11], it follows that any continuous piecewise linear function $f : \mathbb{R}^n \mapsto \mathbb{R}$, $y = f(x)$, can be represented by a system of equations of the form

$$z = c + Zx + Mz + L|z| \,,$$
$$y = d + a^\top x + b^\top z \,,$$

where $z \in \mathbb{R}^s$ is the vector of so-called switching variables, $c \in \mathbb{R}^s$, $Z \in \mathbb{R}^{s\times n}$, strictly lower triangular matrices $M, L \in \mathbb{R}^{s\times s}$, $d \in \mathbb{R}$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}^s$. Here and throughout, $|z|$ denotes the componentwise absolute value of the vector z. Using a similar representation also for piecewise linear constraints and ignoring a possible constant shift in the objective, the piecewise linearly relaxed adversarial problems to be maximized for the function evaluations in the bundle method can be described by

$$\begin{aligned} \max_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad & a^\top x + b^\top z \\ \text{s.t.} \quad & 0 = g + Ax + Bz + C|z| \\ & 0 \geq h + Dx + Ez + F|z| \\ & z = c + Zx + Mz + L|z| \,, \end{aligned} \tag{3}$$

with additional constants $g \in \mathbb{R}^m, h \in \mathbb{R}^p, A \in \mathbb{R}^{m\times n}, B, C \in \mathbb{R}^{m\times s}, D \in \mathbb{R}^{p\times n}$ and $E, F \in \mathbb{R}^{p\times s}$ to describe the m piecewise

linear equality and p piecewise linear inequality constraints. For each x, we define the signature vector

$$\sigma(x) = (\text{sign}(z_i(x)))_{i=1\ldots s} \in \{-1, 0, 1\}^s \ .$$

The signature vectors yield the inverse images

$$P_\sigma \ \equiv \ \{x \in \mathbb{R}^n : \text{sign}(z(x)) = \sigma\} \quad \text{for} \quad \sigma \in \{-1, 0, 1\}^s \ ,$$

which are relatively open polyhedra that form collectively a disjoint decomposition of $\mathbb{R}^n$. The signatures $\sigma \in \{-1, 1\}^s$ are called definite and the associated P_σ are by continuity open.

Any uniformly convex continuous objective function must attain a unique minimizer x_σ on each one of the closed sets $\bar{P}_\sigma$. Furthermore, for a given definite σ, the optimization problem (3) restricted to the corresponding $\bar{P}_\sigma$ is smooth. These observations motivated the optimization strategy of CASM. To obtain a strictly convex continuous objective function, a quadratic regularization term is added to the target function in Eq. (3). Subsequently, standard KKT theory for the resulting smooth constrained quadratic optimization problem on $\bar{P}_\sigma$ can be applied yielding a system of $n + 2s + m + p$ linear equations and $n + 2s + m + p$ unknowns as necessary optimality conditions. It can then be verified by checking the signs of the corresponding Lagrange multipliers if the solution of this system of equations is indeed a minimizer of the original problem. If this is not the case, the computed solution can be used to determine a descent direction and also a new polyhedron $P_{\tilde{\sigma}}$ to be considered. When changing from P_σ to $P_{\tilde{\sigma}}$ one component σ_i of σ changes its sign such that at a point $x \in \bar{P}_\sigma \cap \bar{P}_{\tilde{\sigma}}$ one must have for the signature vector that $\sigma_i(x) = 0$ and $z_i(x) = 0$. Such a switching variable $z_i(x)$ is called active and the corresponding absolute value evaluation yields a nonsmooth contribution.

Hence, the nonsmooth optimization algorithm CASM solves a sequence of quadratic optimization problems where in each iteration of CASM, a linear system $Mv = w$ has to be solved where M is a usually very sparse $(n + 2s + m + p) \times (n + 2s + m + p)$ matrix with real-valued entries. Due to the guaranteed existence of a minimizer there always exists a solution v of the linear system but it must not be unique. Note, that v is related to a Newton step. All remaining steps in the algorithm are rather cheap linear algebra operations.

The convergence properties of the nonsmooth optimizer CASM are analyzed in [6] including also the derivation of optimality conditions for piecewise linear constrained optimization problems. These results extend the work on the unconstrained case presented in [4]. Since the convergence analysis in both papers is based on KKT theory, both algorithms terminate at local optimizers. In the nonconvex case it is not ensured that a global solution is found.

CASM expects a feasible starting point and feasibility is maintained throughout. In the example from the gas market considered here, a feasible starting point can be constructed in various ways. We consider cascades of up to three successive relaxed adversarial problems that result from the application of the bundle method to problem (1). Hence, we need a feasible starting point for each of the three different relaxations. We have no special previous knowledge for the starting point of an optimization cascade, i.e., the coarsest relaxation. Therefore, we determine a starting point by using the nominal values for d and λ. If these two variables are fixed, the physical states, i.e., the pressure π and the flow q, are uniquely determined as described in Section 2. Hence, they can be evaluated.

For the finer discretizations, i.e., the next two optimization tasks in the provided cascade, a warm start strategy will be used, which is an essential aspect of this paper. In the bundle method (cf. Sec. 2), the MIP is solved anew after each refinement, without the old solution having any influence since so far no warm start strategy is known for MIP solvers. In contrast to that, we perform a warm start when using CASM for the inner loop. That is, if the inner problem is solved for a given discretization, a new starting point for the next model with a finer discretization is calculated with the help of the previous solution. For this purpose, the calculated values for demand d and pressure loss coefficient λ are taken from the solution and new starting values for pressure and flow are determined for the refined model. This step coincides with the one for the starting point, i.e., the coarsest discretization.

4 NUMERICAL RESULTS

In this section, we present numerical results for the application of CASM to the adversarial problem of the robust gas transport problem. We hence determine the worst-case values of uncertain parameters for a given compressor control.

GasLib-Instances. We use data from a library of realistic gas network instances [10]. In detail, we conduct numerical experiments on the instances GasLib-11, GasLib-40 and GasLib-134, which model gas networks with 11, 40 and 134 nodes, respectively. For the network with 40 nodes, we distinguish between a non robust feasible and a robust feasible control. GasLib-134 thereby models the Greek gas network. For all test cases, we consider in the end a piecewise linear relaxation of the adversarial problem (2).

We investigate different choices of the compressor control Δ. First, we use arbitrarily chosen controls, which are not robust feasible. Second, we consider a robust feasible control implying that the optimal value of the adversarial problem is equal to 0. Furthermore, we use different choices of the piecewise linear relaxation. That is, we impose different allowed errors up to which the relaxed problem deviates from the original one in terms of the nonconvex pressure loss constraints leading to different discretizations in the piecewise linear approximation of the nonconvex term. As described in Sec. 2, the error bounds are possibly refined during one iteration of the applied bundle method. Therefore, we investigate here the applicability of CASM for such a cascade of refinements.

In detail, we solve the adversarial problem for GasLib-11 with an initial compressor control for two typical sizes of given error bounds. The adversarial problems for GasLib-40 are taken from runs of the adaptive bundle method. First, we applied the adaptive bundle method with an uncertainty set for demand d and pressure loss coefficients λ that is $[0.95d, 1.05d] \times [\lambda, 1.1\lambda]$. For this case, multiple refinements of the relaxation of the adversarial problem are requested in the bundle method's last iteration, when a robust feasible compressor control is investigated. To this series of adversarial problems, we applied CASM. Second, we enlarged the uncertainty set to $[0.9d, 1.1d] \times [\lambda, 1.5\lambda]$. In this case, multiple refinements are requested in an earlier iteration of the bundle method in which the compressor control is not robust feasible. We used these data for another series of adversarial problems to which we applied CASM. The adversarial problems for GasLib-134 are also taken from a run of the adaptive bundle method, namely for the

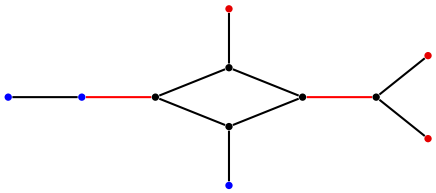


Figure 1: Topology of GasLib-11

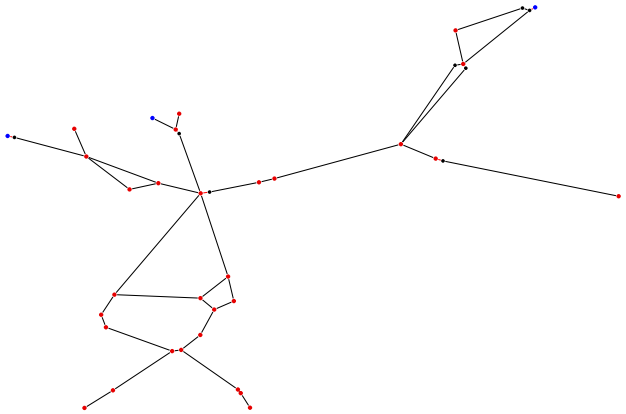


Figure 2: Topology of GasLib-40

	GasLib-11		GasLib-40 non robust feasible			GasLib-40 robust feasible		
relaxation	1.	2.	1.	2.	3.	1.	2.	3.
variables n	44		170					
equal. const. m	19		54					
inequal. const. p	70		314					
switching variables s	175	183	331	341	573	315	315	319
rows/columns of eq. system	484	500	1206	1226	1690	1174	1174	1182
iterations	65	23	939	556	731	472	193	204

Table 1: Complexity of different GasLib instances for their respective optimizations and iterations needed by CASM.

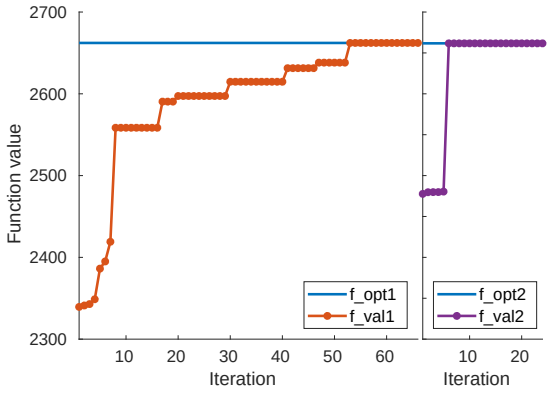


Figure 3: Optimization history of CASM for the GasLib-11 instance. Left: Coarse discretization. Right: Fine discretization.

uncertainty set $[0.8d, 1.2d] \times [\lambda, 2\lambda]$. Again, we applied CASM to multiple refinements that are requested for a compressor control that is not robust feasible. Sketches of the topologies of GasLib-11 and GasLib-40 can be seen in Figs. 1 and 2, respectively. There, sources where gas can be injected are depicted in blue, sinks where gas can be extracted in red, inner nodes, pipes and short pipes in black and compressor stations in red.

Complexity & results. As described in Sec. 3, for CASM the numbers n of variables, m of equality constraints, p of inequality constraints and s of absolute value evaluations are of primary importance for the complexity of the optimization problem. As explained briefly in Sec. 3 the size of the system of equations that must be solved in each iteration depends linearly on each of these numbers. For the different GasLib instances and their respective refinements, the corresponding numbers are given in Tab. 1, where the first column states the GasLib instance and the third one the relaxations. Note that the number of constraints is linear in the number of edges and nodes of the underlying model. As can be seen from this table, the improved approximations of the nonconvex term, i.e., the finer discretizations, only influence the number s of switching variables. The growth in the size of the system matrix fits perfectly to the number $(n+2s+m+p)\times(n+2s+m+p)$ stated in Sec. 3. The numbers of iterations needed by CASM are stated in the last column.

For the GasLib-11 instance, Fig. 3 shows the development of the function values during the optimization runs using CASM for the two discretizations of the nonconvex function. The red line with the label `f_val1` depicts the function values for the coarse discretization and the purple line (label `f_val2`) the function values for the fine discretization. The blue line labeled with `f_opt` illustrates the globally optimal function value, which is reached in both cases and changes from the first optimization to the second one by less than 0.5. The values of the respective optimal variables vary also. During the last iterations, there is a very small increase in the quadratic regularization term that is added to the piecewise linear objective in the CASM, see Sec. 3. However, the value of the nonregularized objective function remains constant, a fact that could be used for an improved termination criterion in the future.

The optimal values of the variables obtained from the optimization for the coarse discretization do not provide a feasible starting point for the fine optimization. Therefore, it is necessary to determine a new feasible starting point for CASM. A corresponding approach exploiting the results from the optimization for the coarse discretization was described at the end of Sec. 3. Figure 3 shows that this warm start strategy yields a larger initial function value for the finer discretization.

For the GasLib-40 instance, we proceed in a similar way. As before, we consider one of the adversarial problems from the bundle method, where subsequently refinements are made yielding a cascade of two refinements, i.e., three optimization runs are performed. From Tab. 1 we see, especially for the instance GasLib-40 with a non robust feasible compressor control, that the refinements induce

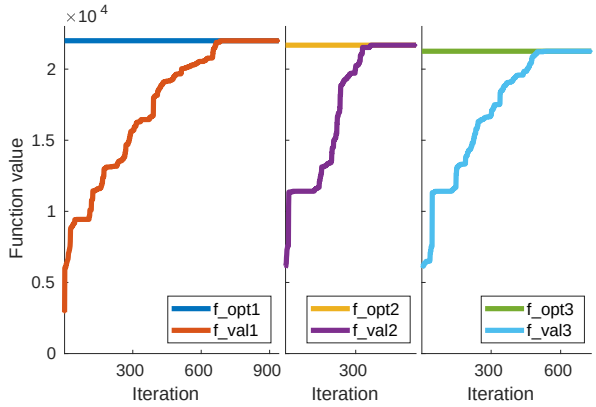


Figure 4: Optimization history of CASM for the non robust feasible GasLib-40 instance

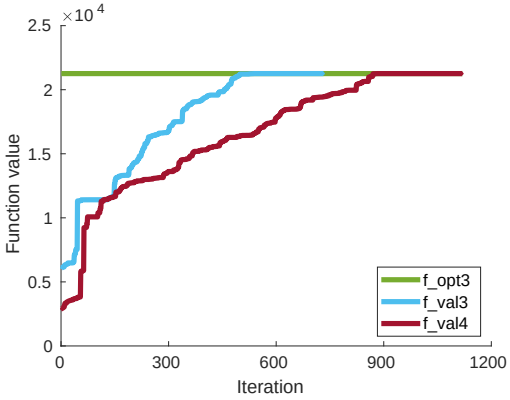


Figure 5: Comparison of the third optimization for non robust feasible GasLib-40 with and without warm start

a significant increase in the dimension of the system of equations and lead also to a noticeable effect on the run times. Despite the fact that the models become more complex with each refinement, the number of iterations does not increase in the same way.

Figure 4 illustrates the development of function values for the three optimization runs in red, purple and blue, respectively (cf. `f_val`). The constant lines depict the globally optimal function values that can be achieved for the respective discretization (cf. `f_opt`). Once more, the objective value at the starting point for the next level of discretization is larger than the initial value for the previous one.

To emphasize the effect of the warm start strategy, a comparison with an optimization not using the warm start is shown in Fig. 5 for the finest discretization from the non robust feasible GasLib-40 instance. The blue graph (`f_val3`) shows again the development of the function values using the warm start option. The dark red graph (`f_val4`) shows the optimization history of the function values for the case that an initial point is determined without exploiting the previously performed optimization such that the function value at the initial point is much smaller. In this case, 1118 iterations are necessary and hence less than for the three individual optimizations in total. However, the size of the system of equations is larger such that one iteration is much more expensive. In addition, from a technical point of view, the considered model with the adapted finer discretization can only be generated if the solution of the previous one is known. Otherwise, a refinement that leads to the same a posteriori error would be even more complex to solve (cf. Sec. 2) supporting also the warm start strategy proposed here.

Next, we consider the GasLib-40 instance with a robust feasible compressor control yielding for the first two relaxations of the three-part cascade the optimal function value 0.6965 and for the finest one the optimal function value 0 corresponding to a robust feasible compressor control. Here, CASM needs 472 iterations to solve the first model with the coarsest discretization to reach a local optimum that is not globally optimal. However, since CASM determines only locally optimal points this fits to the theoretical analysis of CASM as described in Sec. 3. The same behavior is observed also for the second relaxation, where the number of iterations is clearly reduced

	GasLib-134		
relaxation	1.	2.	3.
variables n	534		
equal. const. m	230		
inequal. const. p	784		
switching variables s	737	1107	1985
rows/columns of eq. system	3022	3762	5518
iterations	869	327	328
solves	902	327	328

Table 2: Overview of the complexity of the GasLib-134 instances for their respective optimizations and number of iterations and solves needed by CASM.

by the warm start (cf. Tab. 1). In the third optimization, however, the global optimum is found again.

As last test case, we consider the much larger GasLib-134 instance. Tab. 2 states the problem size and the numbers of iteration needed by CASM. The last line gives the number of equation solves required. In contrast to the previous instances, in some iterations the system of equations was not solved accurately enough resulting in an increase of the parameter in front of the quadratic regularization term to improve the conditioning of the system matrix M. This causes the difference between the number of iterations and the number of equation solves.

For this instance, the development of the function values is shown in Fig. 6. It is particularly noticeable that after the warm start in the second and third optimization, again the iterates just increase the value of the quadratic penalty term and keep the value of the nonregularized objective function constant. That is we observe the same behavior as for the GasLib-11 instance. Solving the finest discretization without a warm start, analogous to the GasLib-40 instance, the function value would be 361 at the initial iterate and 6339 iterations and 7727 equation solves are needed to reach the global optimum.

Finally, we discuss the sparsity structure of the matrix M of the linear system $Mv = w$ in more detail. Fig. 7 shows the nonzero entries of the matrix M in the first iteration of the first optimization of the GasLib-11 instance, where `nz` indicates the total number of nonzero entries of M. Almost all rows have between two and four nonzero entries. However, there are also some rows with up to 44 nonzero entries. These rows are directly related to the relaxation of Eq. (2d) since for the relaxation more evaluation of the absolute

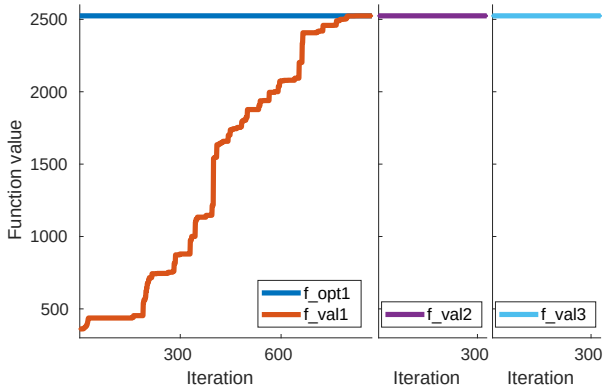


Figure 6: Optimization history of CASM for the non robust feasible GasLib-134 instance

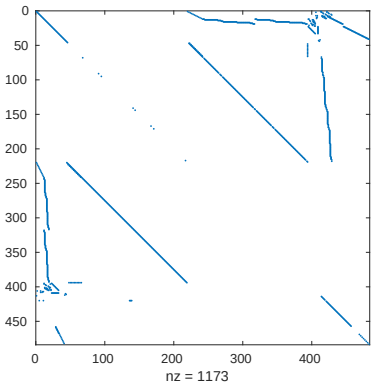


Figure 7: Sparsity of the matrix M in the first iteration of the first optimization for the GasLib-11 instance

value function are needed. Hence, M is denser in these areas. This sparsity of the matrix M is currently not exploited in the implementation. Therefore, we do not compare computation times with other solvers here but just state that the state-of-the-art MIP solver Gurobi, which is implemented in C, is in the range of seconds when solving the optimization problems, whereas a Matlab implementation of CASM is in the range of seconds for the GasLib-11. For the GasLib-40 instances, the current implementation of CASM needed a few minutes. For the optimizations of the GasLib-134 instance the solution was obtained after a bit less than 2 hours. Since solving the system of equations currently accounts for about 95% of the computing time exploiting sparsity could reduce these run times considerably.

5 CONCLUSION AND OUTLOOK

In this work, we have shown that adversarial problems in robust gas transport optimization can be solved with a warm start strategy obtained from nonsmooth optimization. This is a major advantage, as typically MIP solvers cannot profit from earlier iterations, if only a small part of the model is changed. The warm start ability is very advantageous if the piecewise linear relaxation of a nonconvex adversarial problem is required to have high quality, i.e., when many iterative refinements are necessary in order to approximate well the nonconvex functions via piecewise linear functions. Whereas currently the corresponding series of mixed-integer linear problems is always solved from scratch, CASM promises the ability of a warm start strategy such that a subsequent iteration profits from earlier ones, without the necessity to always start from scratch. This will be very helpful for large robust gas network instances.

However, it is necessary to improve the run times needed by CASM via algorithmical engineering in order to allow meaningful comparisons with other solvers. An essential aspect is to speed up the solution of the system of equations that is required in each iteration of CASM. The system matrix is usually very sparse such that sparse solvers would reduce the run time consierably. As can be seen from the numerical results, CASM reached the global optimum in three out of four test cases. Further studies are required to analyse this behaviour in more detail and also the impact on the outer optimization performed by the bundle method. Especially in the scenario considered here, where one solves a cascade of optimization problems with moderate refinements when going from one level to the next one, it should be possible to develop a globalisation strategy to ensure that a global optimum is reached which is required by the adaptive bundle method. Another goal will be to integrate CASM directly into the bundle method. This will also allow a more rigorous comparison to the MIP based approach.

ACKNOWLEDGMENTS

The authors thank the Deutsche Forschungsgemeinschaft for their support within Project B06 and Project B10 in the Sonderforschungsbereich / Transregio 154 Mathematical Modelling, Simulation and Optimization using the Example of Gas Networks.

REFERENCES

[1] D. Aßmann, F. Liers, and M. Stingl. 2019. Decomposable robust two-stage optimization: an application to gas network operations under uncertainty. *Networks* 74, 1 (2019), 40–61. https://doi.org/10.1002/net.21871

[2] M. Collins, L. Cooper, R. Helgason, J. Kennington, and L. LeBlanc. 1977/78. Solving the pipe network analysis problem using optimization techniques. *Management Sci.* 24, 7 (1977/78), 747–760. https://doi.org/10.1287/mnsc.24.7.747

[3] B. Geißler, A. Martin, A. Morsi, and L. Schewe. 2012. Using piecewise linear functions for solving MINLPs. In *Mixed integer nonlinear programming*. IMA Vol. Math. Appl., Vol. 154. Springer, New York, 287–314.

[4] A. Griewank and A. Walther. 2019. Finite convergence of an active signature method to local minima of piecewise linear functions. *Optim. Methods Softw.* 34, 5 (2019), 1035–1055. https://doi.org/10.1080/10556788.2018.1546856

[5] T. Koch, B. Hiller, M.E. Pfetsch, and L. Schewe (Eds.). 2015. *Evaluating gas network capacities*. MOS-SIAM Series on Optimization, Vol. 21. SIAM.

[6] T. Kreimeier, A. Walther, and A. Griewank. 2021. An active signature method for constrained abs-linear minimization. (2021). https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/474

[7] M. Kuchlbauer, F. Liers, and M. Stingl. 2021. Adaptive bundle methods for nonlinear robust optimization. *Informs Journal on Computing* (2021). https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/307

[8] M. Kuchlbauer, F. Liers, and M. Stingl. 2021. Outer approximation for mixed-integer nonlinear robust optimization. (2021). https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/414

[9] H.M. Markowitz and A.S. Manne. 1957. On the solution of discrete programming problems. *Econometrica* 25 (1957), 84–110. https://doi.org/10.2307/1907744

[10] M.E. Pfetsch, A. Fügenschuh, B. Geißler, N. Geißler, R. Gollmer, B. Hiller, J. Humpola, T. Koch, T. Lehmann, A. Martin, A. Morsi, J. Rövekamp, L. Schewe, M. Schmidt, R. Schultz, R. Schwarz, J. Schweiger, C. Stangl, M.C. Steinbach, S. Vigerske, and B.M. Willert. 2015. Validation of nominations in gas network optimization: models, methods, and solutions. *Optim. Methods Softw.* 30, 1 (2015), 15–53. https://doi.org/10.1080/10556788.2014.888426

[11] S. Scholtes. 2012. *Introduction to piecewise differentiable equations*. Springer, New York. https://doi.org/10.1007/978-1-4614-4340-7

[12] J. Schweiger and F. Liers. 2018. A decomposition approach for optimal gas network extension with a finite set of demand scenarios. *Optim. Eng.* 19, 2 (2018), 297–326. https://doi.org/10.1007/s11081-017-9371-4

Solving network-based energy expansion planning and portfolio optimization problems for industrial usage

Jan-Patrick Clarner[1], Christine Tawfik[1], Janina Zittel[1], and Thorsten Koch[1,2]

[1]Applied Algorithmic Intelligence Methods Department, Zuse Institute Berlin, Berlin, Germany,
✉ clarner@zib.de, tawfik@zib.de, zittel@zib.de, koch@zib.de
[2]Chair of Software and Algorithms for Discrete Optimization, Technische Universität Berlin, Berlin, Germany

The energy transition to more sustainable sources universally imposes major challenges on future energy supply. This is particularly valid for Europe and Germany. The shift from centralized, fossil-fuel based technologies to decentralized, regenerative technologies leads to an increased complexity of energy systems. This is also reflected in the underlying mathematical models. For long-term expansion planning in specific, the number of decision options increases significantly, such that advanced mathematical optimization methods are indispensable.

Energy production planning problems are traditionally represented by unit commitment or economic dispatch models where energy demands need to be satisfied by internal generators or external imports. Intertwined network structures and physical transport laws often need to be incorporated. There exists a variety of related academic models for optimizing energy production portfolios that include some or all of the above aspects. Industrial models, however, are significantly more demanding with respect to the high level of detail in which generators have to be modeled in order to capture real-world practices and provide relevant insights to decision makers. Furthermore, specific technological aspects, such as full steam extraction condensation cycles for heat and power production, encompass an exhaustive range of inter-dependent physical processes that can only be manifested by a high modeling granularity. To the best of our knowledge, the literature of unit commitment models has hitherto failed to provide a holistic framework for energy production systems. To bridge this gap, we hereby propose a new model class that incorporates the mentioned aspects, by formalizing a consistent and generic network-based energy modeling framework.

The model is based on a directed, time-expanded graph where nodes represent technological elements of an energy system and arcs represent transport options for the involved resources such as fuel, heat and power. For each time step, we define positive continuous variables on arcs that model resource flows and binary variables on nodes that represent the operational status. An on-status means that the node is active whereas an off-status means that the node is inactive. In the model, we have a spatial dimension and a temporal dimension. In terms of spatial linkage, constraints are induced by nodes imposing a feasible operation region on incoming and outgoing flow variables depending on the operational status. If the status is inactive, flow variables are set to zero. If the status is active, the feasible operation region of inflows and outflows depends on the specific technological element the node represents. This can be a characteristic curve that models fuel-to-heat conversion, a polytope that defines operation modes for combined heat and power plants or upper and lower limits of a power trading contract. The set does neither need to be convex nor connected.

Operational costs are induced analogously using a term that depends on inflows, outflow and the status variable of a node. The operation of certain nodes can depend on each other. For instance, certain generators cannot work simultaneously whereas the operation of one generator is only possible, if another generator is active. This is a relevant practical feature that is rarely found in existing modeling frameworks. To cover those cases, we use hyper edges that impose a feasible region on the operation variables of the respective nodes. Concerning the time dimension, there are several constraints such as storage constraints as well as minimum up and down times. Moreover, to cover the expansion planning aspect, we model investment decisions by using additional binary variables as bounds on the operational status variables of the respective technological element. We use piecewise-linear formulations and additional binary variables to reformulate the model as a mixed-integer linear program.

Realistic instances contain a high level of detail and need to be solved for time horizons of 20 to

30 years. This leads to a model size of hundreds of millions of variables and constraints, and hence a complexity that cannot be handled out-of-the-box using commercial solvers. In this way, the planning horizon is often restricted to several months. To cover a realistic planning horizon, we develop a coarse-to-fine hierarchical framework. The basic idea is to solve coarse-detailed models and to infer variable fixings for the more refined models, such that the model in its full level of detail becomes computationally tractable. To decrease the level of detail on higher levels, we relax certain constraints, apply linearizations and aggregate time steps. The latter means to decrease time granularity or to use average or representative time periods, which can be calculated using clustering, averaging and scaling methods. Our hierarchical framework consists of three levels. On the highest level, we determine investment decisions. On the intermediate planning level, we derive storage and operational variable fixings at specific time points. In this way, the lowest level (unit commitment) decomposes into several independent sub problems that can be solved in parallel. To solve the corresponding optimization problems in each level, we use rolling horizon techniques. To assess the quality of our methods, we perform computational experiments on real-world instances from the district heating network in Berlin. Results so far indicate that the hierarchical framework can solve medium-scale optimization problems of this model class faster up to a factor of 4 than commercial solvers, without significant losses on the solution quality in terms of the objective value.

Nevertheless, particular large-scale instances are still posing ever-increasing computational challenges to be addressed. The methods we propose so far mainly focus on the time dimension of the problem. To accelerate the solution algorithm even further and to make use of the full model structure, there is a potential room for improvement by explicitly incorporating the spatial dimension in the developed procedures. Future planned research will thoroughly investigate the topological aspects of the network, with the outlook of deriving useful information for the solution process, e.g., problem relaxations, bounds or warm-start solutions.

Session Session 5B: communication networks 1

Thursday 9 June 2022, 15:40-17:20

Lecture Hall III

Joint Optimization of Caching and Recommendation

Yi Zhao[1], Zhanwei Yu[1], and Di Yuan[1]

[1]Department of Information Technology, Uppsala University, Sweden, ✉{yi.zhao, zhanwei.yu, di.yuan}@it.uu.se

To alleviate the burden of communication networks and reduce the access latency of users, caching popular contents at network edges has been recognized as a promising solution [4]. Proactively storing contents at base stations (BSs) enables direct content delivery to users without involving the remote cloud. To efficiently utilize the cache with limited capacity, a content delivery system tends to cache the most popular contents that are frequently requested by users. A typical objective of caching optimization is to maximize the hit-ratio, which is the probability that a requested content is cached.

The popularity of contents is influenced by many factors, among which recommendation systems [3] play a major role. Embedded in content providing sites or applications, recommendation systems undertake the task of recommending contents that best match the interests of users. Since the user preferences have a large diversity, the recommendation is usually personalized. In practice, recommendation has dramatically boosted users' engagement and satisfaction. It is reported that up to 80% of requests on Netflix come from recommendation [3].

That recommendation has strong impact on content request calls for extending caching optimization to recommendation, as a reshaping tool of content popularity to improve the hit-ratio. In fact, the emergence of content delivery networks (CDNs) makes the joint optimization practically possible. Two widely known examples of CDN are Netflix Open Connect and Google Global Cache. CDNs enable more network-efficient and user-friendly solutions by optimizing jointly content caching and recommendation. Some studies [1, 2, 5] have taken steps in this direction, with the target of maximizing the hit-ratio. In these works, heuristics are derived based on problem decomposition, leading to sub-optimal solutions. However, these works only focus on recommendation based on the long-term interest of users. In practice, the system need to also account for the short-term interest, namely the content currently being consumed by the user. We use the term incumbent content to refer to the content currently being selected and consumed by a user, and we call this recommendation scheme as incumbent-aware recommendation.

We consider the joint optimization of caching and incumbent-aware recommendation for multiple users with one BS, while accounting for user satisfaction of recommendation. The set of users is denoted by $\mathcal{K}$. The cache is deployed at the BS and controlled by the content provider, with a limited capacity denoted by C. Denote by $\mathcal{I}$ the sets of all contents under consideration. Define a binary indicator x_i for caching decision: If content i is cached, $x_i = 1$, otherwise $x_i = 0$. For content $i \in \mathcal{I}$, denote by s_i its size; denote by $p_i^k \in [0, 1]$ the probability of user k requesting content i without recommendation. Denote by $p_{ji}^k \in [0, 1]$ the probability of user k requesting content i via recommendation with incumbent content j. The value of $\sum_{k\in\mathcal{K}} \left(p_i^k + \sum_{j\in\mathcal{I}\setminus i} y_{ji}^k p_j^k p_{ji}^k\right)$ reflects the popularity of content i with recommendation. The maximum length of the recommendation list is denoted by B. Denote by y_{ji}^k a binary indicator of recommendation decision: If content i is in the incumbent-aware recommendation list of content j for user k, $y_{ji}^k = 1$, otherwise $y_{ji}^k = 0$. To account for user satisfaction of the recommendation system, for user k, if the probability of content i being requested from the recommendation is lower than a threshold, denoted by β^k, then content i will be not recommended.

The joint optimization of caching and incumbent-aware recommendation is formulated below, where (1a) is the hit-ratio, and (1b)-(1d) are the cache capacity constraint, recommendation list length constraint, and user satisfaction constraint, respectively.

First, regarding the problem complexity, we have proved Theorem 1.

$$\max_{\boldsymbol{x},\boldsymbol{y}} \sum_{i\in\mathcal{I}} x_i \left(\sum_{k\in\mathcal{K}} \left(p_i^k + \sum_{j\in\mathcal{I}\setminus i} y_{ji}^k p_j^k p_{ji}^k \right) \right) \tag{1a}$$

$$\textbf{s.t.} \quad \sum_{i\in\mathcal{I}} s_i x_i \leq C \tag{1b}$$

$$\sum_{i\in\mathcal{I}\setminus j} y_{ji}^k \leq B, \quad j \in \mathcal{I}, k \in \mathcal{K} \tag{1c}$$

$$y_{ji}^k = 0, \quad \text{if } p_j^k p_{ji}^k < \beta^k, \;\; i,j \in \mathcal{I}, i \neq j, k \in \mathcal{K} \tag{1d}$$

$$y_{ji}^k \in \{0,1\}, \quad i,j \in \mathcal{I}, i \neq j, k \in \mathcal{K} \tag{1e}$$

$$x_i \in \{0,1\}, \quad i \in \mathcal{I} \tag{1f}$$

Theorem 1. *The joint optimization problem of caching and recommendation in (1) is NP-hard, even for the simplified case where $|\mathcal{K}| = 1$, $B = 1$, s_i are of uniform size $\forall i \in \mathcal{I}$, and (1d) is discarded.*

The set of cached contents at the BS, denoted by $\mathcal{I}_c$, is given by $\boldsymbol{x}$. Second, we consider the recommendation subproblem with given $\mathcal{I}_c$, derive a greedy-based algorithm, and prove its optimality, as stated in Theorem 2.

Theorem 2. *The recommendation subproblem with given $\mathcal{I}_c$ can be solved to optimum by a greedy-based algorithm.*

Third, we prove the monotone and submodular properties of the objective function with given $\mathcal{I}_c$.

Theorem 3. *Denote by $f(\mathcal{I}_c)$ as the optimal objective value of (1a) with given $\mathcal{I}_c$. Function $f(\mathcal{I}_c)$ is a monotone increasing submodular set function.*

Based on Theorem 3, we derive a polynomial-time iterative algorithm. The proposed algorithm greedily adds content i^* to $\mathcal{I}_c$ at each iteration, by maximizing $(f(\mathcal{I}_c \cup \{i\}) - f(\mathcal{I}_c))/s_i$. This process will in turn invoke the algorithm indicated in Theorem 2. We prove that the proposed algorithm has $1 - e^{-1}$ approximation guarantee, based on the submodularity, as claimed in Theorem 4.

Theorem 4. *Let $\boldsymbol{x}_{OPT}$ be the optimal $\boldsymbol{x}$ for the problem in (1), and $\boldsymbol{x}$ be the solution achieved by the proposed algorithm. It holds that $f(\boldsymbol{x}) > (1 - 1/e) f(\boldsymbol{x}_{OPT})$.*

At last, we assess the proposed algorithm using synthetic datasets as well as real-world datasets. Simulation results show its good performance in terms of improving the hit-ratio.

References

[1] Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, and Iordanis Koutsopoulos. Caching-aware recommendations: Nudging user preferences towards better caching performance. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[2] Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, and Iordanis Koutsopoulos. Jointly optimizing content caching and recommendations in small cell networks. *IEEE Transactions on Mobile Computing*, 18(1):125–138, 2019.

[3] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*, 6(4), dec 2016.

[4] Shan Zhang, Peter He, Katsuya Suto, Peng Yang, Lian Zhao, and Xuemin Shen. Cooperative edge caching in user-centric clustered mobile networks. *IEEE Transactions on Mobile Computing*, 17(8):1791–1805, 2018.

[5] Dongsheng Zheng, Yingyang Chen, Mingxi Yin, and Bingli Jiao. Cooperative cache-aware recommendation system for multiple internet content providers. *IEEE Wireless Communications Letters*, 9(12):2112–2115, 2020.

Logic-based Benders Decomposition for the Network Migration Problem

Maryam Daryalal[1] and Hamed Pouya[2]

[1]Department of Mechanical & Industrial Engineering, University of Toronto, Toronto, Canada, ✉ daryalal@mie.utoronto.ca
[2]Department of Mechanical & Industrial Engineering, University of Toronto, Toronto, Canada, ✉ h.pouya@utoronto.ca

Introduction. Telecommunication networks frequently face technological advancements and need to upgrade their infrastructure. In telecommunication industries, network migration is the process of upgrading the existing infrastructure of a deployed network. A telecommunication network is composed of a set of sites (demand points), and circuits that transmit the traffic between the sites. Migration of such a network is performed by upgrading the circuits one by one. In order to upgrade every circuit, two synchronized technicians migrate its two endpoints within a time window. The goal of *the network migration problem* (NMP) is to find the upgrade order of these circuits such that the associated costs are minimized. This is a defining step in the customer acquisition of telecommunications service suppliers, and its outcome directly impacts the network owners' purchasing behaviour. Migration of a network is a strategic decision that can lead to immense savings of 10 to 100 times in power and space [1].

The NMP can also be stated in the context of the vehicle routing problem with synchronized constraints (VRPS), where at least one vertex or arc requires simultaneous visits of vehicles, or successive visits resulting from some precedence constraints. In the NMP, vehicles and nodes correspond to technicians and sites, respectively. Since the tasks (i.e., upgrading the circuit endpoints) are defined over the nodes, the NMP is a special case of the vehicle routing problem with node synchronization constraints and an application of the VRPS in the telecommunications domain. In regards to the applicability of the exisiting studies on solving the VRPS problems to the NMP, an important issue is their *local* synchronisation assumption, meaning that the arrival of vehicles is synchronized at the same node. However, the NMP involves several sets of technicians distributed over multiple regions which can be synchronized with several other technicians from the same or other regions based on the location of the circuit endpoints. Besides, the only exact method that considers multiple depots and time windows (the same as the NMP), can solve small instances that are far from the need of telecommunication networks [2]. Therefore, the existing works in the literature of the VRPS are not suited for the NMP's level of complexity.

In this work, we develop the first exact solution method for the network migration problem in order to find the optimal planning solutions, i.e., the order of circuit upgrades, along with the technician assignment and routing decisions. To the best of our knowledge, both in the context of the telecommunications problems and as a VRPS, this is the first method that exactly solves the NMP with a certificate of optimality/infeasibility. We decompose the NMP into three problems and link them all by designing a logic-based Benders decomposition (LBBD) approach that benefits from a hybrid constraint programming-based column generation in its master problem and a constraint programming model in its subproblem. By doing so, we are able to leverage the power of different solution techniques for linear programming and combinatorial optimization, and delegate the task of solving each problem to the most suitable optimization paradigm. Given that the NMP can be viewed as a VRPS, our method can also be adapted to a wide class of integer programming problems. We also make several algorithmic enhancements and considerably improve the basic version of the algorithm that only uses the classical adaption of Benders decomposition and column generation.

Solution Methodology. Our solution approach relies on the notion of a *shift*. A shift is defined as a set of circuit endpoints migrated by a single technician during a maintenance window, together with any travels between the sites. We model the NMP as an ILP that returns a planning solution consisting of a set of shifts. The proposed formulation is amenable to the LBBD framework, meaning that we can decouple the problem into smaller subproblems that are easier to solve. Denote by $\boldsymbol{m}$, a vector of decision variables $m_{ss'w} \in \mathbb{Z}_+$ that determine the number of circuits between the pair of sites $\{s, s'\}$

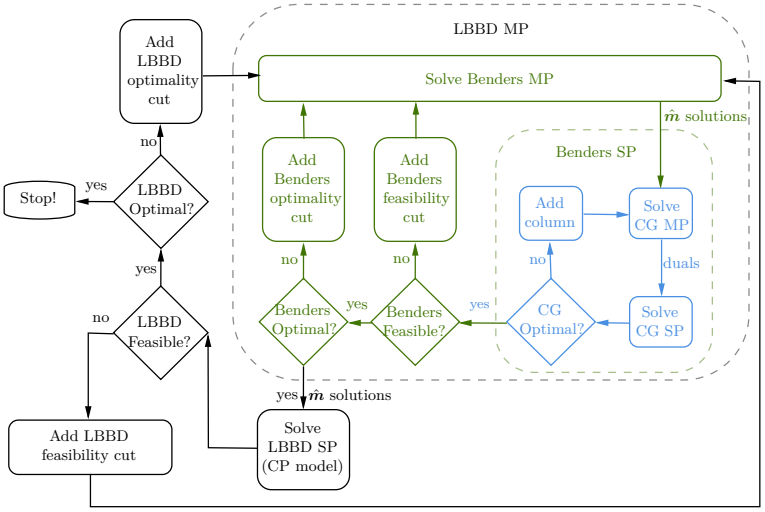


Figure 1: Overall framework of the LBBD method for the NMP.

migrated during a maintenance window w. Let $\mathcal{M}$ be the set of constraints defining a feasible $\boldsymbol{m}$, η_w be the migration cost at w, and $\text{SP}_w^{\text{LBBD}}(\boldsymbol{m})$ the problem of generating a set of shifts with minimum cost for a given $\boldsymbol{m}$. The problem can be formulated as follows:

$$\min \left\{ \sum_{\forall w} \eta_w \ : \ \boldsymbol{m} \in \mathcal{M},\ \eta_w \geq \text{SP}_w^{\text{LBBD}}(\boldsymbol{m}),\ \forall w,\ m_{ss'w} \in \mathbb{Z}_+ \forall s, s', w \right\}. \tag{1}$$

We design an LBBD where the master problem (LBBD MP) decides on the number of migrated circuits between the site pairs at each maintenance window, together with an estimation on the cost of such a plan, and the second-stage (recourse) problems ($\text{SP}_w^{\text{LBBD}}(\boldsymbol{m})$) verify if it is feasible to migrate the assigned number of circuits with the available resources, and if so what is the actual cost of this migration.

In its current form, LBBD MP is oblivious to the structure of the NMP. We add the LP relaxation of the subproblem to the LBBD MP to improve the performance of the algorithm. By relaxing the integrality constraints of the subproblems, we now have integer first-stage and continuous second-stage decision variables and the new problem is amenable to the Benders decomposition, with Benders MP the same as the LBBD MP. For the subproblem, considering the exponential number of possible shifts, we develop a column generation procedure that gradually generates improving shifts for the problem. The Benders feasibility and optimality cuts derived for this Benders decomposition are also valid for the LBBD MP. After the Benders decomposition converges, the first-stage solutions $\hat{\boldsymbol{m}}$ are passed to the $\text{SP}_w^{\text{LBBD}}(\boldsymbol{m})$ for checking their feasibility/optimality. We formulate the $\text{SP}_w^{\text{LBBD}}(\boldsymbol{m})$ (our LBBD SP) as a constraint programming (CP) model that creates a *plan*, i.e., a set of shifts corresponding to a set of technicians working during the maintenance window w, consisting of connected paths (for the travels) along with the number of circuit endpoints migrated between each site pair. By designing valid LBBD feasibility/optimality cuts we guarantee the exactness of our solution method. Figure 1 depicts the overall solution framework and its different levels of decomposition.

Results. We evaluate the proposed LBBD algorithm on instances defined over six real backbone and regional networks and provide detailed algorithmic analysis and discussions on the implementation choices, along with managerial insights on the trade-offs among the migration cost, resource usage, and the duration of the migration. Our comprehensive evaluations demonstrate the computational efficiency of the algorithm in obtaining quality solutions. We also show the merit of each incorporated optimization paradigm in achieving this performance.

References

[1] Ciena. The network modernization imperative, 2013.

[2] Jiliu Li, Hu Qin, Roberto Baldacci, and Wenbin Zhu. Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. *Transportation Research Part E: Logistics and Transportation Review*, 140:101955, 2020.

A branch-and-cut algorithm for the routing and spectrum allocation problem

Marcelo Bianchetti[1,2] and Javier Marenco[2]

[1]Computer Science Dept., FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina, ✉ mbianchetti@dc.uba.ar
[2]Computer Science Dept., FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina, ✉ jmarenco@dc.uba.ar

The routing and spectrum allocation problem (RSA) arises as a result of one of the most promising solutions to deal with huge data traffic demands in large communication networks, the *flexible grid* (flexgrid) technology [4] specified in the ITU-T standard G.694.1. In these networks, the frequency spectrum is divided into narrow frequency *slots*, and a sequence of consecutive slots forms a *channel* that can be switched in the network nodes to create a *lightpath* between two nodes. One of the simplest variations of the RSA consists in establishing the lightpaths for a set of end-to-end traffic demands that, assuming the same modulation format for each, are expressed in terms of the number of required slots. Since lightpaths are determined by a route and a selected channel that satisfies the volume, RSA involves finding a route and assigning frequency slots to each demand. To comply with the ITU recommendation, the following constraints must be respected in this setting: (I) *slot continuity*: the slots assigned to a certain demand must remain the same on all the links of the corresponding route; (II) *slot contiguity*: the slots allocated to each demand must be contiguous; and (III) *non-overlapping slot*: a slot can be assigned to various demands if the routes assigned to these demands do not share any link.

Formally the offline version of RSA can be stated as follows. We are given a digraph $G = (V, E)$ representing the optical fiber network, a fixed number $\bar{s} \in \mathbb{Z}_+$ of available slots, and a set of *demands* $D = \{d_i = (s_i, t_i, v_i)\}_{i=1}^{k}$, where each demand d_i, $i = 1, \ldots, k$, is composed by a source $s_i \in V$, a target $t_i \in V$, and a volume $v_i \in \mathbb{Z}_+$. If $d = d_i = (s_i, t_i, v_i) \in D$ is a demand, for some $i \in \{1, \ldots, k\}$, we define $s(d) = s_i$, $t(d) = t_i$, and $v(d) = d_i$. We define a *lightpath* for a demand $d_i = (s_i, t_i, v_i)$ to be a tuple (l, r, p), where $1 \leq l \leq l + v_i - 1 \leq r \leq \bar{s}$ and p is a (directed) path in G from s_i to t_i.

In this setting, RSA consists in establishing a lightpath associated to each demand, in such a way that lightpaths do not overlap. In other words, each demand d_i, for $i = 1, \ldots, k$, must be assigned a path p_i (regarded as a sequence of arcs) in G between s_i and t_i and an interval $[l_i, r_i]$ consisting of at least v_i consecutive slots in $[1, \bar{s}]$ in such a way that if $p_i \cap p_j \neq \emptyset$ then $[l_i, r_i] \cap [l_j, r_j] = \emptyset$, for any two demand indices $i \neq j$.

In the current work we show that this version of the problem belongs to the $\mathcal{NP}$-hard class. We survey the models present in the literature and classify some of them according to the most common criteria. In particular, we analyze the different approaches with which they try to solve the three major restrictions of RSA, i.e., contiguity, continuity and non-overlapping, as well as the used objective function. In [1] we have presented twelve integer programming models that solve the offline version of RSA, by trying with different families of variables and constraints, obtaining mixed results. In particular we considered two natural ways of modeling feasbible solutions within an integer programming approach: either we represent the routing with a set of variables and the slot allocation with a second set of variables, or we represent both decisions with a single set of variables. For each model we presented a few variations. The computational experiments, both real and random, suggested that one of the compact formulations that uses only one set of variables, namely the DSL-BF formulation is the one with the best results, namely

$$\min \quad \sum_{d \in D} \sum_{e \in E} \sum_{s \in S} u_{des}/v(d) \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in \delta^-(j)} u_{des} - \sum_{e \in \delta^+(j)} u_{des} = 0 \qquad \forall d \in D,\, \forall j \in V \backslash \{s(d), t(d)\},\, \forall s \in S \tag{2}$$

$$\sum_{e \in \delta^+(s(d))} \sum_{s \in S} u_{des} \geq v(d) \qquad \forall d \in D \tag{3}$$

$$\sum_{e \in \delta^-(s(d))} \sum_{s \in S} u_{des} = 0 \qquad \forall d \in D \tag{4}$$

$$\sum_{d \in D} u_{des} \leq 1 \qquad \forall e \in E, \forall s \in S \tag{5}$$

$$v(d)(u_{des} - u_{de,s+1}) \leq \sum_{s'=f}^{s} u_{des'} \qquad \forall d \in D, \forall e \in E, \forall s \in S,\ f = \max\{1, s - v(d) + 1\} \tag{6}$$

$$u_{des} \in \{0, 1\} \qquad \forall d \in D, \forall e \in E, \forall s \in S. \tag{7}$$

We concentrate on this formulation which, despite having a large number of variables, gave quite encouraging results and seems to be very interesting from a theoretical point of view. In particular we try to improve its performance by applying various techniques widely used in integer linear programming. Firstly, we intend to study the formulation and, exploiting some of its properties, propose three main theorems coming from symmetry considerations which allow us to generate valid inequalities that may be used as cutting planes in a branch-and-cut algorithm. We present more than sixty families of valid inequalities, equations and optimality cuts dealing with the non-overlapping restriction, and we continue with the implementation of a branch-and-cut algorithm using these families as cutting planes [3]. In order to select some of the inequalities from these families, we present several selection and filtering strategies. We calibrate the parameters for each procedure and compare between the different strategies and a generic branch-and-cut and branch-and-bound algorithm implemented by Cplex.

In spite of overcoming the two Cplex generic implementations, the results obtained with the branch-and-cut algorithm suggested that starting with a feasible solution, the algorithm could obtain better results faster. We review the state of the art related to heuristics for both RSA and SA, and then we propose a primal heuristic as well as some variations. Given that the studied model allows solutions with spurious cycles and paths, it was worth providing the proposed heuristics with an algorithm capable of improving those solutions.

The math-heuristic developed (FRSAP) iteratively solves an ILP model and not only quickly returns a near-optimal feasible solution, but is capable of detecting probably-infeasible instances with great precision. The experiments on more than 550 instances showed that the times and the quality of the solutions improved remarkably when adding the heuristics to the branch-and-cut, solving with optimality a quarter more instances than the best Cplex solver configuration (FCP), i.e., pre-resolution, heuristics, parallelization, and cutting planes. The number of sub-optimally solved instances was also around that difference, while it had memory problems half as many times as Cplex.

Algorithm	Time [hs]	Solved	Unknown	Feasible	Optimal	Infeasible	Prob. Inf.	Memory
FCP	35.45	256	67	33	137	86	0	20
FRSAP	26.37	312	11	52	170	90	64	1

Table 1: FRSAP vs FCP comparison over the 323 instances solved by some algorithm, fixing the time limit in 15 mins.

Since to the best of our knowledge there are no complete instances available in the literature, but only some topologies and parameters, and being a very common practice in the study of RSA and its variations to generate the benchmark, we implemented an instance generator script based on real topologies and parameters to be able to carry out the experiments. We also believe that it would be nice if these instances together with the script, which is available in [2], are the start of a standardized benchmark for RSA.

References

[1] F. Bertero, M. Bianchetti, and J. Marenco. Integer programming models for the routing and spectrum allocation problem. *TOP*, 322(10):891–921, Jul 2018.

[2] M. Bianchetti. exactasmache/rsainstances: Rsa instances, October 2020.

[3] M. Bianchetti and J. Marenco. Valid inequalities and a branch-and-cut algorithm for the routing and spectrum allocation problem. 2021.

[4] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *IEEE Communications Magazine*, 47(11):66–73, 2009.

New advances in Segment Routing optimisation

Jérôme De Boeck[1,2], Hugo Callebaut[1], and Bernard Fortz[1,2]

[1]Département d'informatique, Université libre de Bruxelles, Bruxelles, Belgium , ✉ {jerome.de.boeck,hugo.callebaut,bernard.fortz}@ulb.be
[2]INOCS, INRIA Lille Nord-Europe, Lille, France

The internet is composed of a set of autonomous systems (AS). An autonomous system is a set of routers under a single technical administration such as an internet service provider (ISP). To this day the internet still relies on packet switching. Transferred data is broken down into packets for more efficient transfer. The goal at the network layer of the internet is to provide the way to transport datagrams (IP packets), from their source to their destination.

The paths to follow are chosen by the internet routing protocols. These can be divided into two main groups: inter-domain and intra-domain routing protocols, also called respectively border gateway protocols (BGP) and interior gateway protocols (IGP). As their names indicate, BGPs are used to handle traffic between ASes and IGPs handle routing within ASes [1].

The branch that aims to improve the quality of service for IGP protocols is called Traffic Engineering (TE). From an optimisation point of view, the aim is to send each packet using the most efficient route across the network at that time by (re-)optimising the routing of the traffic, but leaving the network topology and hardware configuration unchanged [3].

One of the most important problems TE has to deal with is network congestion, because it almost always results in degradation of user experience. Network congestion occurs when the network has to carry more data than what it can handle. Usually TE techniques try to minimise the maximum link utilisation where the link utilisation is define as the total flow on the link divided by its capacity.

The most widely used IGP protocols rely on shortest path routing (SPR): traffic flowing from one router to another will always be routed along the shortest path. The shortest paths are computed using a link metric system, a weight is assigned to each link and these weights are used to determine the shortest path from one node to another. In case multiple shortest paths exist, a usual rule is to split traffic evenly on all outgoing arcs belonging to a shortest path. This technique is called *Equal Cost Multi-Path* (ECMP).

A lot of research have been devoted to TE with SPR protocols, see [1, 4, 6] for surveys on the topic.

Even when optimized, SPR protocols suffer a number of drawbacks. Demand matrices and networks change. As described in [7], operators do not like to change weights, because it can lead into transient instability until the distributed computation of the shortest paths converges. Moreover, SPR might be very far from an optimal routing where flow can be distributed freely in the network [8].

To provide more flexibility in routing and decrease the overhead induced by SPF protocols, Segment Routing (SR) was recently introduced. SR is a modern variant of source routing in computer networks, which is being developed within the SPRING and IPv6 working groups of the IETF [5, 10]. In a segment routed network, an ingress node may prepend a header to packets that contain a list of segments, which are instructions that are executed on subsequent nodes in the network. These instructions may be forwarding instructions, such as an instruction to forward a packet to a specific destination or interface. These instructions act on top of an existing protocol, like SPF ones. More precisely, an instruction to forward a packet to a specific router to reach a destination will be performed using the underlying protocol.

Here, we assume that routing in the underlying protocol, e.g. SPR with ECMP, is given, i.e. the link metric system is fixed and cannot be modified. We consider the problem of routing given demand matrix using node segments (i.e allowing to fix a set of nodes to visit to reach a destination from a given origin), with a given upper bound K on the number of segments used. For $K = 2$, the problem was formulated as a compact MIP in [2]. For $K \geq 2$, a compact MIP formulation was proposed in [9], but with a very poor linear relaxation. A path-based formulation was proposed in [11] and solved with a column-generation based heuristic.

In this talk, we introduce the notion of *forward graph* and extend it in the context of SR. We study theoretical properties of forward graphs and derive a pre-processing technique allowing to eliminate more than 90 % of the paths on realistic instances. These pre-processing techniques are made very efficient by the use of adequate data structures to represent the forward graphs. We also show with an extensive set of numerical experiments that the pre-processed models can be solved by state-of-the-art solvers without the help of column generation.

In future research, we plan to consider the simultaneaous optimization of SPR metrics and SR paths, and to consider uncertain demand matrices.

References

[1] A. Altın, B. Fortz, M. Thorup, and H. Ümit. Intra-domain traffic engineering with shortest path routing protocols. *Annals of Operations Research*, 204(1):65–95, April 2013.

[2] R. Bhatia, F. Hao, M. Kodialam, and T.V. Lakshman. Optimized network traffic engineering using segment routing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 657–665, 2015.

[3] A. Bley. An integer programming algorithm for routing optimization in IP networks. In Dan Halperin and Kurt Mehlhorn, editors, *Algorithms - ESA 2008*, pages 198–209, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] A. Bley, B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pióro, A. Tomaszewski, and H. Ümit. Optimization of OSPF routing in IP networks. In A. M. C. A. Koster and X. Muñoz, editors, *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks*, chapter 8, pages 199–240. Springer, 2010.

[5] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois. The segment routing architecture. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.

[6] B. Fortz. Applications of meta-heuristics to traffic engineering in IP networks. *International transactions in Operational Research*, 18(2):131–147, 2011.

[7] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124, 2002.

[8] B. Fortz and M. Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29(1):13–48, 2004.

[9] R. Hartert. *Fast and scalable optimization for segment routing.* PhD thesis, UCL-Université Catholique de Louvain, 2018.

[10] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois. A declarative and expressive approach to control forwarding paths in carrier-grade networks. In *SIGCOMM*, 2015.

[11] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure. Cg4sr: Near optimal traffic engineering for segment routing with column generation. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1333–1341, 2019.

Session Session 5C: location problems

Thursday 9 June 2022, 15:40-17:20

Lecture Hall IV

Analysing the Complexity of Facility Location Problems with Capacities, Revenues, and Closest Assignments

Christina Büsing*
Combinatorial Optimization
RWTH Aachen University, Germany
buesing@combi.rwth-aachen.de

Timo Gersing†
Combinatorial Optimization
RWTH Aachen University, Germany
gersing@combi.rwth-aachen.de

Sophia Wrede†
Combinatorial Optimization
RWTH Aachen University, Germany
wrede@combi.rwth-aachen.de

ABSTRACT

Facility Location Problems with Capacities, Revenues, and Closest Assignments (FLP-CRCA) are an extension of the well known, strongly $\mathcal{NP}$-complete Facility Location Problem (FLP). With this extension, we recognise that facilities have an upper capacity on the customers to be served, but also need to generate a minimum revenue to be operated economically. Furthermore, we acknowledge that customers have a strong preference towards their closest facility.

We show that finding a feasible solution for FLP-CRCA is already strongly $\mathcal{NP}$-complete if the underlying graph forms a star, but that the problem can be solved efficiently on paths and cycles. In the case where the number of facilities is fixed, we propose a pseudo-polynomial algorithm and show that the problem is weakly $\mathcal{NP}$-complete under this condition. Our results also hold for FLPs with closest assignments and *either* capacities *or* revenues.

1 INTRODUCTION

Facility Location Problems (FLPs) are one of the most fundamental problems in combinatorial optimization [15]. In their most basic version, facilities providing some kind of service for customers need to be opened at potential sites and customers are assigned to these facilities. The aim is to minimize the total cost consisting of opening cost for the facilites and the service cost or traveling cost for assigning the customers. Part of this problem's success is based on the broad applicability to real-world problems [1, 16]. In practice, further constraints need to be considered. Upper capacities on the facilities, i.e., the number of customers or the demand a facility can serve, prevent that customers wait too long at facilities. Such problems are referred to as Capacitated FLPs (CFLPs). One main difference between CFLPs and FLPs is that now customers can not always be served by their closest facility. In reality, however, customers are often free to choose their facility and prefer their closest one. This potentially leads to some facilities being overloaded, while others only serve few customers. In order to prevent such situations, the property of *closest assignments* is demanded: a constraint stating that each customer is to be assigned to their closest open facility; cf. [8, 10] for reviews of closest assignment constraints in integer programming. The closest assignment and upper capacities improve the service quality for the customers. Facilities, however, are often assumed to be operated by individual owners. In order to economically survive, they have to generate a minimum threshold of revenues by serving customer demands. This is achieved by introducing lower bounds on the revenue a facility accumulates. A real world example which can be modeled through Facility Location Problems *with capacities, revenues and closest assignments* is the optimized distribution of pharmacies in a given area under the following assumptions: due to time and space constraints only a limited number of customers can be served; a minimum revenue has to be generated by serving citizens due to the financial independence; citizens use their closest open pharmacy.

The basic FLP is $\mathcal{NP}$-complete in the strong sense [6, 14], which makes the computation of optimal solutions in acceptable time unlikely. This complexity result can be extendend to all of the problem's generalizations. However, if the FLP is defined on a graph, where the assignment costs are equal to the distances between the customer nodes and the facility nodes, the FLP can be solved efficiently if the graph is a tree [6]. For CFLPs, (pseudo-)polynomial algorithms are known in special cases [11, 12]. To the best of the authors' knowledge, the literature on FLPs with Closest Assignments (and Capacities) focuses on strengthening integer programming formulations or developing heuristics [2–4, 7, 8, 10, 13, 17]. Research on the computational complexity of FLPs with Closest Assignments mixed with capacity- and revenue-constraints is still missing.

In this paper, we analyse Facility Location Problems with Capacities, Revenues, and Closest Assignments (FLP-CRCA) and derive some settings that are $\mathcal{NP}$-complete and some that are polynomially solvable. More specifically, we show that:

- finding a feasible solution for the FLP-CRCA is strongly $\mathcal{NP}$-complete on star graphs - contrary to the general FLP, where an optimal solution on trees can be found in polynomial time [6] (Section 3).
- the FLP-CRCA on paths and cycles can be solved efficiently, contrary to the CFLP (Section 4).
- for a fixed number of facilities, there is a pseudo-polynomial algorithm and the problem becomes weakly $\mathcal{NP}$-complete (Section 5).
- these complexity results hold for FLPs with Closest Assignments and *either* capacities *or* revenues.

With this work we close the research gap regarding the computational complexity of Facility Location Problems with Closest Assignments mixed with capacitiy- and revenue-constraints.

*Supported by the German Federal Ministry of Education and Research (grant no. 05M16PAA) within the project "HealthFaCT - Health: Facility Location, Covering and Transport", by the Freigeist-Fellowship of the Volkswagen Stiftung and by the German research council (DFG) Research Training Group 2236 UnRAVeL.
†Supported by the German Federal Ministry of Education and Research (grant no. 05M16PAA) within the project "HealthFaCT - Health: Facility Location, Covering and Transport".

2 PROBLEM DEFINITION AND NOTATION

In this paper, we study the following setting. As an underlying structure consider an undirected graph $G = (V, E)$, where V denotes the set of nodes and E the set of edges. The nodes represent the locations of customers as well as the locations for potential facilities, while the edges represent the street network connecting these sites.

The costs of assigning node $v \in V$ to facility $f \in V$ are denoted by $\tau(v, f) \in \mathbb{Z}_{>0}$ and are defined to be the distance of a shortest (v, f)-path with respect to weights $\delta_e > 0$ on the edges $e \in E$. To each node $v \in V$, we assign parameters, $(r_v, d_v, R_v, C_v, c_v) \in \mathbb{Z}^5_{\geq 0}$, where $r_v \in \mathbb{Z}_{\geq 0}$ represents the *revenue* a customer generates for the serving facility and $d_v \in \mathbb{Z}_{\geq 0}$ represents the customer's *demand* a serving facility has to satisfy. Furthermore, if a facility opens at node v, it has to accumulate a minimum amount of revenue, denoted with $R_v \in \mathbb{Z}_{\geq 0}$ and the aggregated demand must not exceed the capacity $C_v \in \mathbb{Z}_{\geq 0}$. Lastly, parameter c_v denotes the costs of opening a facility at node v.

With these considerations, we define the optimization problem studied here.

Definition 2.1 (FLP-CRCA). We are given an undirected graph with five non-negative parameters for each node and positive weights on the edges, $G = (V, E, (r_v, R_v, d_v, C_v, c_v)_{v \in V}, (\delta_e)_{e \in E})$. Then, the FLP-CRCA consists in finding a subset $F \subseteq V$ of nodes for opening facilities and an assignment $\Lambda : V \to F$ of customers to these facilities such that

(1) for each open facility its lower bound on revenue and upper bound on capacity are not violated, i.e., $R_f \leq \sum_{v \in \Lambda^{-1}(f)} r_v$ and $\sum_{v \in \Lambda^{-1}(f)} d_v \leq C_f$ for all $f \in F$,
(2) each customer is assigned to their closest open facility, that is, there exist no $v \in V$ and $f \in F$ with $\tau(v, f) < \tau(v, \Lambda(v))$,
(3) the cost of opening facilities and distances of the customers is minimized, that is $\sum_{f \in F} c_f + \sum_{v \in V} \tau(v, \Lambda(v)) \to \min$.

Note that we assume, that the demand of a customer cannot be split between two different facilities.

3 COMPLEXITY OF THE FLP-CRCA

It is common knowledge that the CFLP is strongly $\mathcal{NP}$-complete. However, to emphasize the difference in the complexity of the FLP-CRCA and the CFLP, we first provide a reduction showing that it is already strongly $\mathcal{NP}$-complete to construct any solution for the CFLP at all. This demonstrates that the CFLP's complexity is independent of a potential underlying graph defining the costs – in contrast to the FLP-CRCA, as we will see in Section 4.

Theorem 3.1. *Finding a feasible solution for CFLP is strongly $\mathcal{NP}$-complete.*

Proof. We reduce from the strongly $\mathcal{NP}$-complete problem 3-Partition [9] to a CFLP-instance. Let $\mathcal{I} = (A, (s_a)_{a \in A}, B)$ be an instance of 3-Partition, with A being a set of $3m$ elements, sizes $s_a \in \mathbb{Z}_{\geq 0}$ for each element $a \in A$, such that $s_a \in (B/4, B/2)$ and $\sum_{a \in A} s_a = mB$, for a bound $B \in \mathbb{Z}_{\geq 0}$. The task is to partition A into disjunct subsets $A_1, \ldots, A_m$ such that $\sum_{a \in A_i} s(a) = B$ holds for all $i \in \{1, \ldots, m\}$. We construct from this a CFLP-instance without costs $\mathcal{I}' = (J, (d_j)_{j \in J}, I, (C_i)_{i \in I})$ with customers J, demands d_j, facilities I, and capacities C_i. For each $a \in A$, we define one facility and one customer, that is, $J = I = A$. Furthermore, we set the demands equal to the sizes $d_a = s_a$ and the capacities equal to the bound $C_a = B$.

Note that, there exists a partitioning $A_1, \ldots, A_m$ of A that is a solution to instance $\mathcal{I}$ of 3-Partition *iff* there exists a feasible solution to constructed CFLP-instance $\mathcal{I}'$.

CFLP is in $\mathcal{NP}$ since we can test the feasibility of an assignment of customers such that each open facility's capacity constraint is met in linear time w.r.t. the size of set A. □

Considering a lower bound on the revenue is a generalization of CFLPs. The closest assignment condition, however, brings a certain structure to feasible solutions of FLP-CRCA-instances. We will see in the next section that this makes the considered problem easier on certain graph classes. However, already considering trees or stars as underlying networks leads to an $\mathcal{NP}$-complete problem.

Theorem 3.2. *Finding a feasible solution for FLP-CRCA on stars is strongly $\mathcal{NP}$-complete.*

Proof. We reduce again from 3-Partition, this time to an FLP-CRCA-instance. Let $\mathcal{I} = (A, (s_a)_{a \in A}, B)$ be an instance of 3-Partition as defined in the proof of Theorem 3.1. Any such instance will be transformed into an FLP-CRCA-instance $\mathcal{I}' = (V, E, (r_v, R_v, d_v, C_v, c_v)_{v \in V}, (\delta_e)_{e \in E})$ as follows. The set of nodes V contains one node a for every element $a \in A$ and one extra node ξ; hence, $|V| = |A|+1$. Set $R_a = C_a = B$ and $r_a = d_a = s_a$, for each $a \in V \setminus \{\xi\}$. For node ξ, set $r_\xi = R_\xi = d_\xi = C_\xi = 0$. Next, introduce the set of edges $e \in E$. Connect the nodes so that the underlying graph is an $|A|$-star, where node ξ is the center. Set $\delta_e = 1$ for all edges $e \in E$. Note, due to the choice of parameters in $\mathcal{I}'$, all facilities need to be opened at a leaf in a feasible solution. Due to the underlying star structure, each customer is either indifferent between the facilities or uses the facility located at its own node.

Again, there exists a partitioning $A_1, \ldots, A_m$ of A that is a solution to instance $\mathcal{I}$ of 3-Partition *iff* there exists a feasible solution to constructed FLP-CRCA-instance $\mathcal{I}'$.

Finding the nearest facility for a node takes $O(|V|)$ on stars. Hence, the problem is still in $\mathcal{NP}$, as testing whether each customer is served by their nearest open facility can be done in $O(|V|^2)$. □

Thus, no (pseudo-)polynomial algorithm exists, unless $\mathcal{P} = \mathcal{NP}$. However, in the next section, we will see that, in contrast to the CFLP, feasible solutions on paths and cycles can be computed efficiently.

4 POLYNOMIAL SPECIAL CASES

If the underlying graph is either a path or a cycle, any instance of the FLP-CRCA can be solved efficiently. We show this via a reduction to the polynomially solvable Shortest s-t-Path Problem on Directed Acyclic Graphs.

4.1 FLP-CRCA on Paths

We first consider the case where the underlying graph of the FLP-CRCA is a path $p = (1, \ldots, n)$. Since we have positive edge weights, paths have the important property that all customers between the most-left and most-right customer of the same facility must be also served by it. That is, for every solution $(F, \Lambda : V \to F)$, the set of

customers $\Lambda^{-1}(f)$ assigned to a facility $f \in F$ forms an interval $\{l_i, \dots, u_i\}$ containing f. Hence, the customers can be partitioned into such intervals and every solution can be represented by a sequence of triples $(l_i, f_i, u_i)_{i=1}^k$, where f_i denotes the location of the i-th facility and nodes l_i, u_i the most-left and most-right customer served by it. Note that, for $f_1 < \dots < f_k$, we have $l_i = u_{i-1} + 1$, with $u_0 := 0$, for all $i \in [k]$. Hence, it suffices to consider tuples (f_i, u_i) for a complete representation of a solution. However, while any feasible solution corresponds to a sequence of tuples $(f_i, u_i)_{i=1}^k$, not every sequence represents a feasible solution.

Definition 4.1. We call $(f, u) \in V^2$ a *feasible tuple* if $f \le u$. We call a sequence of feasible tuples $(f_i, u_i)_{i=1}^k$ a *feasible sequence* if it meets the following properties:

(a) it holds $u_{i-1} < f_i$ and $u_k = n$,
(b) the customers served by facility f_i yield $\sum_{v=u_{i-1}+1}^{u_i} r_v \ge R_{f_i}$ and $\sum_{v=u_{i-1}+1}^{u_i} d_v \le C_{f_i}$,
(c) for $i \ge 2$, facility f_i is not closer to customer u_{i-1} than facility f_{i-1} and facility f_{i-1} is not closer to customer $u_{i-1}+1$ than facility f_i, that is, $\tau(u_{i-1}, f_i) \ge \tau(u_{i-1}, f_{i-1})$ as well as $\tau(u_{i-1}+1, f_i) \le \tau(u_{i-1}+1, f_{i-1})$.

The relation between a feasible sequence and a solution to the FLP-CRCA is elaborated in the next lemma.

Lemma 4.2. *There is a 1-1 correspondence between feasible solutions of the FLP-CRCA on paths and feasible sequences $(f_i, u_i)_{i=1}^k$.*

Proof. We already stated above that every solution (F, Λ) can be represented by a sequence $(f_i, u_i)_{i=1}^k$ with $f_1 < \dots < f_k$ and $\{f_1, \dots f_k\} = F$ as well as $u_i = \max\{v \in [n] | \Lambda(v) = f_i\}$. This sequence meets Property (a), as otherwise the sets $\Lambda^{-1}(f_i)$ would not form intervals. Furthermore, Properties (b) and (c) follow directly from Properties (1) and (2) in Definition 1.1.

Conversely, every feasible sequence $(f_i, u_i)_{i=1}^k$ defines a solution $F = \{f_1, \dots f_k\}$ with $\Lambda^{-1}(f_i) = \{u_{i-1} + 1, \dots, u_i\}$. This solution is feasible, as Property (1) in Definition 1.1 follows from Property (b); Property (2) is fulfilled, as no customer in the interval $\Lambda^{-1}(f_i)$ wants to deviate from facility f_i *iff* this is true for the customers at the boundaries, which is ensured by Property (c). □

For finding a feasible solution to the FLP-CRCA, we construct a directed auxiliary graph $G' = (V', A', (w_a)_{a \in A'})$, in which each $s-t$-path corresponds to a feasible sequence. We introduce one node for each feasible tuple together with two extra nodes s, t, i.e., $V' = \{(f, u) \in [n]^2 \mid f \le u\} \cup \{s, t\}$. Starting at node s, the first arc that we choose on our $s-t$-path corresponds to the first tuple in our sequence. Hence, we have $(s, (f, u)) \in A'$ *iff* tuple (f, u) meets Property (b). Afterwards, choosing an arc from (f', u') to (f, u) corresponds to (f, u) being the successor of (f', u') in our sequence. Hence, this arc exists *iff* Properties (a) – (c) are fulfilled. The last tuple in the sequence needs to meet $u = n$, thus $((f, n), t) \in A'$ for all $f \in [n]$. In conclusion,

$$\begin{aligned} A' = {} & \{(s, (f, u)) \in \{s\} \times (V' \setminus \{s, t\}) \mid \text{ fulfills (b)}\} \\ & \cup \{((f', u'), (f, u)) \in (V' \setminus \{s, t\})^2 \mid \text{ fulfills (a) – (c)}\} \\ & \cup \{((f, u), t) \in (V' \setminus \{s, t\}) \times \{t\} \mid u = n\}. \end{aligned}$$

FLP-CRCA-instance on a path:

R_v	2	1	5	3
r_v	2	2	2	4
C_v	3	1	6	4
d_v	2	1	2	3
c_v	5	1	6	2

Auxiliary Graph G' :

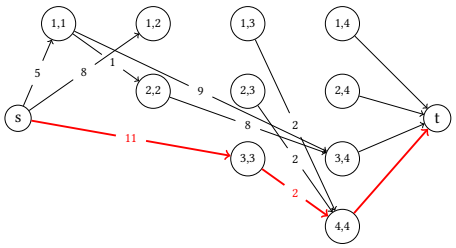


Figure 1: An instance of FLP-CRCA with an optimal solution, opening facilities 3 and 4, of value 13 and the corresponding auxiliary graph G' with a shortest path of the same value.

Finally, we define weights on the arcs such that the cost of an $s-t$-path equals the cost of the corresponding solution:

$$w_a = \begin{cases} 0, & \text{if } a = ((f, u), t) \in A' \\ c_f + \sum_{v=1}^{u} \tau(f, v), & \text{if } a = (s, (f, u)) \in A' \\ c_f + \sum_{v=u'+1}^{u} \tau(f, v), & \text{if } a = ((f', u'), (f, u)) \in A'. \end{cases}$$

For an example of the relationship between an FLP-CRCA-instance on paths and its auxiliary graph, see Figure 1.

The following lemma states that there exists a solution to the FLP-CRCA on paths *iff* there exists an $s-t$-path in the corresponding auxiliary graph G'. In this case, the cost of an optimal solution is equal to the cost of a shortest $s-t$-path.

Lemma 4.3. *There is a cost-preserving 1-1 correspondence between solutions of the FLP-CRCA and $s-t$-paths in G'.*

Proof. By construction of auxiliary graph G', every $s-t$-path $p' = (s, (f_1, u_1), \dots, (f_k, n), t)$ corresponds to exactly one feasible sequence $(f_i, u_i)_{i=1}^k$, and thus, due to Lemma 4.2, also to one solution (F, Λ) with $F = \{f_1, \dots f_k\}$ and $\Lambda^{-1}(f_i) = \{u_{i-1} + 1, \dots u_i\}$.

For the cost of p', it holds

$$\begin{aligned}\sum_{a\in p'} w_a &= w_{(s,(f_1,u_1))} + \sum_{i=2}^{k} w_{((f_{i-1},u_{i-1}),(f_i,u_i))} + w_{((f_k,n),t)}\\ &= c_{f_1} + \sum_{v=1}^{u_1}\tau(f_1,v) + \sum_{i=2}^{k}(c_{f_i} + \sum_{v=u_{i-1}+1}^{u_i}\tau(f_i,v))\\ &= \sum_{f\in F}(c_f + \sum_{v\in\Lambda^{-1}(f)}\tau(f,v))\\ &= \sum_{f\in F} c_f + \sum_{v\in V}\tau(v,\Lambda(v)),\end{aligned}$$

which is exactly the cost of solution (F,Λ). □

Checking whether an arc is feasible and computing its weight takes $O(n)$ steps for each of the $O(n^4)$ possible arcs. Hence, we can transform any FLP-CRCA instance on paths in such an auxiliary graph in $O(n^5)$ steps. For computing a shortest $s-t$-path, first note that G' is an acyclic graph. This can be seen by sorting the nodes in lexicographic order, i.e., $s,(1,1),(1,2),\dots,(1,n),(2,2),\dots,(n,n),t$, and recognizing that there exists no backward-arc due to (a). Since single-source shortest paths in directed acyclic graphs can be computed in $O(|V'|+|A'|)$ [5] , we can compute a shortest path in $O(n^4)$ steps. Altogether, we obtain the following.

Theorem 4.4. *An optimal solution to FLP-CRCA can be computed in $O(n^5)$ steps.*

In practice, constructing the graph and finding a shortest path can be sped up significantly by doing both in parallel. For this, we iterate over the nodes in the lexicographic ordering. When reaching node (f,u), we only check whether there exist outgoing arcs if node (f,u) was reached before, as those arcs do not belong to a shortest path otherwise.

From a modeler's perspective, this result is useful if the real world street network can be modeled as a path. For example, when deciding where to open pharmacies in rural areas, where multiple villages are connected by the same road, and each village is too small such that at least one pharmacy could economically operate in each of them. Real world examples are mountain passes with hotels or small villages along the road in Switzerland.

Besides the practical applicability, it is also interesting to analyse the theoretical impact of the constraints on the computational complexity on basic graph structures. In the next subsection, for example, we study how the algorithm on paths can be extended to cycles.

4.2 FLP-CRCA on Cycles

In order to solve the FLP-CRCA on cycles, we reduce it to multiple subproblems that are similar to the FLP-CRCA on paths.

Consider an instance on a cycle $G=(V,E)$ and an optimal solution $(F^*,\Lambda^*:V\to F^*)$. Note that there exists an edge $e^*\in E$ such that the above solution is also feasible for the FLP-CRCA on the path $G_{e^*}=(V,E\backslash\{e^*\})$. To see this, consider a forest within G that connects each customer $v\in V$ with its serving facility $\Lambda^*(v)$ via a shortest path. The missing edges in the forest are exactly those that can be deleted from E.

For now, assume that we are given e^*. Additionally, assume w.l.o.g. $e^*=\{n,1\}$, that is, G_{e^*} is the path $p_{e^*}=(1,\dots,n)$. In the best case, we can compute a shortest path in the auxiliary graph G'_{e^*}, as defined in the previous subsection, that corresponds to (F^*,Λ^*). However, it is possible that the computed shortest path in G'_{e^*} corresponds to a solution $(F',\Lambda':V\to F')$ that is not feasible for the original instance on G, as customer n might prefer facility $\Lambda'(1)$ over $\Lambda'(n)$ or customer 1 might prefer facility $\Lambda'(n)$ over $\Lambda'(1)$. Hence, we want to restrict ourselves to paths in G'_{e^*} that can be "glued together" at the first tuple (f_1,u_1) and the last tuple (f_k,n) such that the corresponding solution is feasible on G. For this, we require that (f_1,u_1) can be a successor of (f_k,n) in accordance to Definition 4.1. Unfortunately, this yields a shortest path problem in which the feasible paths depend on the first chosen arc $(s,(f_1,u_1))$, which is usually problematic. We resolve this issue by fixing a potential last arc $((f_k,n),t)$ and removing all arcs $(s,(f_1,u_1))$ from G'_{e^*} for which (f_1,u_1) is not a feasible successor of (f_k,n). Then every $s-(f_k,n)$-path in the resulting graph corresponds to a solution that is feasible to the original problem. Moreover, the path corresponding to the optimal solution (F^*,Λ^*) is contained in the graph resulting from fixing arc $((\Lambda^*(n),n),t)$ and can thus be found by computing a shortest $s-(\Lambda^*(n),n)$-path.

Naturally, we neither know $\Lambda^*(n)$, nor do we have e^* in advance. However, testing all possible combinations $e^*\in E$ and $\Lambda^*(n)\in V$, we are guaranteed to find one yielding an optimal solution. In summary, we solve n^2 shortest path problems, each requiring $O(n^4)$ steps. Deleting arcs $(s,(f_1,u_1))$ from G'_e for a fixed last arc $((f_k,n),t)$ can be done on the fly in constant time for each arc while computing the shortest path. As stated in the previous subsection, computing an auxiliary graph G'_e requires $O(n^5)$ steps. Note that $G'_{\{i,i+1\}}$ can be computed efficiently from $G'_{\{i-1,i\}}$ by reusing most of the graph. This leads us to the following statement.

Theorem 4.5. *The FLP-CRCA on cycles can be solved in $O(n^6)$.*

For a better understanding of the procedure for solving FLP-CRCA-instances on cycles, consider the following example.

Example 4.6. Consider an FLP-CRCA instance on a cycle with parameters as introduced in Figure 1; cf. Figure 2. When fixing arc $((3,4),t)$, arc $(s,(3,3))$ has to be removed since tuple $(3,3)$ is not a feasible successor of $(3,4)$; cf. auxiliary graph $G'_{\{4,1\}}$ in Figure 2. An optimal solution of value 14 can be achieved by opening facilities 1 and 4. Note that, arc $(s,(3,3))$ has also to be removed when fixing arc $((4,4),t)$: otherwise, customer 1 would deviate from facility 3 to facility 4 in the underlying cycle.

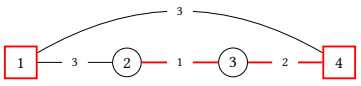
5 FIXED NUMBER OF FACILITIES

If the number of open facilities $k\in\mathbb{N}$ is fixed, we refer to the problem as the k-FLP-CRCA. That is, we consider instances of the FLP-CRCA and are only interested in optimal solutions where k facilities are opened and k is not part of the input.

Lemma 5.1. *Finding a feasible solution for k-FLP-CRCA on stars is at least weakly $\mathcal{NP}$-complete, already for $k=2$.*

Proof. The claim can be seen by a reduction from the weakly $\mathcal{NP}$-complete problem Partition [9]. In Partition, a finite set A is considered; each element $a\in A$ has a weight, $s_a\in\mathbb{Z}^+$. The question

FLP-CRCA-instance on a cycle with parameters R_v, r_v, C_v, d_v, c_v as considered in Figure 1:



Auxiliary Graph $G'_{\{4,1\}}$:

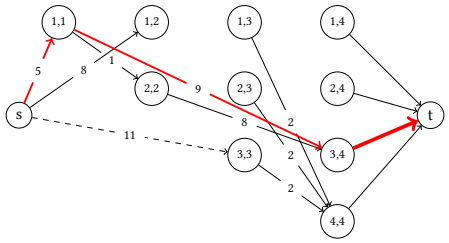


Figure 2: An instance of FLP-CRCA on a cycle with deleted edge $e^* = \{4, 1\}$. Note that, the optimal solution found in Figure 1 is not feasible here.

is whether A can be partitioned into two disjunct sets $A', A \setminus A'$ such that $\sum_{a\in A'} s_a = \sum_{a\in A\setminus A'} s_a =: B$.

For a given Partition instance $\mathcal{I} = (A, (s_a)_{a\in A})$, we construct a k-FLP-CRCA instance with $k = 2$ on a star analogously to the proof of Theorem 3.2. We introduce one leaf node for each element $a \in A$ and one center node ξ. We set $d_a = r_a = s_a$ and $C_a = R_a = B$, for $a \in V \setminus \{\xi\}$ as well as $d_\xi = C_\xi = r_\xi = R_\xi = 0$. Again, we define $\delta_e = 1$ for all edges $e \in E$ of the star

Note that, there exists a partitioning $A', A \setminus A'$ of A that is a solution to instance $\mathcal{I}$ of Partition *iff* there exists a feasible solution to constructed k-FLP-CRCA-instance $\mathcal{I}'$.

As a special case of FLP-CRCA, the problem is in $\mathcal{NP}$. □

In the following, we introduce a dynamic program that solves any k-FLP-CRCA-instance in pseudo-polynomial time, thus proving that the problem is indeed weakly $\mathcal{NP}$-complete. To that end, we fix a set of facilities and test in pseudo-polynomial time whether we find a closest assignment satisfying the revenue- and demand-constraints. Let $F \in \binom{V}{k}$ be a fixed set of facilities. We first determine for each customer $v \in V$ the set of closest facilities $F_v \subseteq F$ to which v can be assigned to. Afterwards, we compute labels

$$b : \{1, \dots, |V|\} \times \bigtimes_{f\in F} \left(\{0, \dots, C_f\} \times \{0, \dots, R_f\}\right) \to \mathbb{R} \cup \{\infty\},$$

where $b(i, C'_{f_1}, R'_{f_1}, \dots, C'_{f_k}, R'_{f_k})$ states the minimum cost of assigning customers $\{1, \dots, i\} \subseteq V$ so that facility $f \in F$ accumulates demand of at most C'_f and revenue of at least R'_f, that is the value of the subproblem

$$\min_{\Lambda:[i]\to F} \left\{ \sum_{f\in F} c_f + \sum_{v\in[i]} \tau(v, \Lambda(v)) \,\middle|\, \begin{aligned} \Lambda(v) &\in F_v && \forall v \in [i] \\ \sum_{v\in\Lambda^{-1}(f)} d_v &\le C'_f && \forall f \in F \\ \sum_{v\in\Lambda^{-1}(f)} r_v &\ge R'_f && \forall f \in F \end{aligned} \right\}. \quad (1)$$

Note that, the labels with finite cost and parameters $i = |V|$, $C'_f \in \{0, 1, \dots, C_f\}$ and $R' = R_f$ for all $f \in F$ are exactly the labels corresponding to feasible solutions in (1) of considered k-FLP-CRCA-instance with fixed set of facilities F; the cost of an optimal solution for fixed F is $b(|V|, C_{f_1}, R_{f_1}, \dots, C_{f_k}, R_{f_k})$.

To compute the labels, we use the recursion formula

$$\begin{aligned} &b(i, C'_{f_1}, R'_{f_1}, \dots, C'_{f_k}, R'_{f_k}) \\ &= \min_{f_j\in F_i} \{b(i-1, \dots, C'_{f_j} - d_i, R'_{f_j} - r_i, \dots, C'_{f_k}, R'_{f_k}) + \tau(i, f_j)\} \end{aligned}$$

together with the base values

$$b(0, C'_{f_1}, R'_{f_1}, \dots, C'_{f_k}, R'_{f_k}) = \begin{cases} \sum_{f\in F} c_f, & \text{if } C'_f \ge 0,\ R'_f \le 0\ \forall f \in F \\ \infty, & \text{otherwise.} \end{cases}$$

This leads to the following result.

Theorem 5.2. *The k-FLP-CRCA is weakly $\mathcal{NP}$-complete and can be solved in $O(\binom{|V|}{k} \cdot k \cdot |V| \cdot \prod_{f\in F}(C_f \cdot R_f))$ steps by using the above dynamic program.*

Proof. We show that the values computed by the recursion formula are equal to the values of the subproblems (1).

For the base values, assigning no customers leaves us with the opening costs $\sum_{f\in F} c_f$. Furthermore, assigning no customers meets the capacity- and revenue-constraints of subproblem (1) *iff* $C'_f \ge 0$ and $R'_f \le 0$ holds for all $f \in F$.

When assigning customer i to a fixed facility $f' \in F_i$, the minimum cost for assigning $[i]$ and respecting capacities and revenues $(C'_f, R'_f)_{f\in F}$ in subproblem (1) is given by

$$\tau(i, f') + \min_{\Lambda\in\mathcal{P}} \left\{ \sum_{f\in F} c_f + \sum_{v\in[i-1]} \tau(v, \Lambda(v)) \right\},$$

with

$$\mathcal{P} = \left\{ \Lambda : [i-1] \to F \,\middle|\, \begin{aligned} \Lambda(v) &\in F_v && \forall v \in [i-1] \\ \sum_{v\in\Lambda^{-1}(f)} d_v &\le C'_f && \forall f \in F \setminus \{f'\} \\ \sum_{v\in\Lambda^{-1}(f')} d_v &\le C'_{f'} - d_i \\ \sum_{v\in\Lambda^{-1}(f)} r_v &\ge R'_f && \forall f \in F \setminus \{f'\} \\ \sum_{v\in\Lambda^{-1}(f')} r_v &\ge R'_{f'} - r_i \end{aligned} \right\},$$

which is by induction exactly the value of

$$b(i-1, \dots, C'_{f'} - d_i, R'_{f'} - r_i, \dots, C'_{f_k}, R'_{f_k}) + \tau(i, f').$$

Taking the minimum cost over all $f' \in F_i$ yields the optimal assignment, which proves the correctness of the recursion formula.

For each set of facilities $F \in \binom{V}{k}$, computing the closest facilities $F_i \subseteq F$ for a customer $i \in V$ can be done in $O(k)$ steps if we compute all distances in a preprocessing step. Computing a label requires comparing $|F_i| \le k$ values and can thus be done in $O(k)$ steps. For every set of facilities $F \in \binom{V}{k}$, we compute $|V| \cdot \prod_{f\in F}(C_f \cdot R_f)$ many labels, which yields the time-complexity stated above.

The weak $\mathcal{NP}$-completeness of the k-FLP-CRCA follows together with Lemma 5.1. □

Thus, as in the general Facility Location Problem, the FLP-CRCA is easier to solve if the number of open facilities is fixed. Note that this result is independent of the underlying network.

6 CONCLUSION

We showed that Facility Location Problems with Capacities, Revenue, and Closest Assignments (FLP-CRCA) are already computationally intractable if the underlying graph forms a star - contrary to the famous Facility Location Problem, which is known to be tractable on trees [6]. On paths and cycles, however, the FLP-CRCA turns out to be computationally tractable - unlike Capacitated Facility Location Problems. This discrepancy is caused by the closest assignment property, which brings a special structure to the solutions for FLP-CRCA-instances. Furthermore, we showed that, if the number of open facilities is fixed, the FLP-CRCA is weakly $\mathcal{NP}$-complete for any underlying graph class. All results presented here also hold for Facility Location Problems with Closest Assignments and *either* capacities *or* revenues.

Further work includes the study of approximation algorithms and analysing the complexity of further graph classes.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their constructive feedback.

REFERENCES

[1] Amir Ahmadi-Javid, Pardis Seyedi, and Siddhartha S Syam. 2017. A survey of healthcare facility location. *Computers & Operations Research* 79 (2017), 223–263. https://doi.org/10.1016/j.cor.2016.05.018

[2] Herminia I. Calvete, Carmen Galé, José A Iranzo, José-Fernando Camacho-Vallejo, and Martha-Selene Casas-Ramírez. 2020. A matheuristic for solving the bilevel approach of the facility location problem with cardinality constraints and preferences. *Computers & Operations Research* 124 (2020), 105066. https://doi.org/10.1016/j.cor.2020.105066

[3] Massimiliano Caramia and Renato Mari. 2016. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters* 10 (2016), 997–1019. https://doi.org/10.1007/s11590-015-0918-z

[4] Martha-Selene Casas-Ramírez, José-Fernando Camacho-Vallejo, and Iris-Abril Martínez-Salazar. 2018. Approximating solutions to a bilevel capacitated facility location problem with customer's patronization toward a list of preferences. *Appl. Math. Comput.* 319 (2018), 369–386. https://doi.org/10.1016/j.amc.2017.03.051 Recent Advances in Computing.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms.* MIT Press.

[6] Gérard Cornuèjols, George L. Nemhauser, and Laurence A. Wolsey. 1990. The uncapacitated facility location problem. *Discrete Location Theory* (1990). url: https://pdfs.semanticscholar.org/1c16/97671bc4a9a85e8619bc0c8d3e40c1bd27cb.pdf.

[7] Lázaro Cánovas, Sergio García, Martine Labbé, and Alfredo Marín. 2007. A strengthened formulation for the simple plant location problem with order. *Operations Research Letters* 35, 2 (2007), 141–150. https://doi.org/10.1016/j.orl.2006.01.012

[8] Inmaculada Espejo, Alfredo Marín, and Antonio M. Rodríguez-Chía. 2012. Closest assignment constraints in discrete location problems. *European Journal of Operational Research* 219, 1 (2012), 49–58. https://doi.org/10.1016/j.ejor.2011.12.002

[9] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-completeness.* W. H. Freeman & Co., New York, USA.

[10] Ross A Gerrard and Richard L Church. 1996. Closest assignment constraints and location models: Properties and structure. *Location Science* 4, 4 (1996), 251–270. https://doi.org/10.1016/S0966-8349(97)00001-6

[11] Edward Kh Gimadi and Anna A. Kurochkina. 2019. Time complexity of the ageev's algorithm to solve the uniform hard capacities facility location problem. In *Optimization and Applications - 9th International Conference, OPTIMA 2018, Revised Selected Papers (Communications in Computer and Information Science),* Yury Kochetov, Michael Khachay, Yury Evtushenko, Vlasta Malkova, Mikhail Posypkin, and Milojica Jacimovic (Eds.). Springer-Verlag GmbH and Co. KG, Germany, 123–130. https://doi.org/10.1007/978-3-030-10934-9_9 9th International Conference on Optimization and Applications, OPTIMA 2018 ; Conference date: 01-10-2018 Through 05-10-2018.

[12] Edward Gimadi, Alexandr Shtepa, and Oxana Tsidulko. 2019. Improved Exact Algorithm for the Capacitated Facility Location Problem on a Line Graph. In *2019 15th International Asian School-Seminar Optimization Problems of Complex Systems (OPCS).* 53–57. https://doi.org/10.1109/OPCS.2019.8880248

[13] Pierre Hanjoul and Dominique Peeters. 1987. A facility location problem with clients' preference orderings. *Regional Science and Urban Economics* 17, 3 (1987), 451–473. https://doi.org/10.1016/0166-0462(87)90011-1

[14] Jakob Krarup and Peter Marc Pruzan. 1983. The simple plant location problem: Survey and synthesis. *European Journal of Operational Research* 12, 1 (1983), 36–81. https://doi.org/10.1016/0377-2217(83)90181-9

[15] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama (Eds.). 2019. *Location Science.* Springer. https://doi.org/10.1007/978-3-030-32177-2

[16] Derya C Turkoglu and Mujde E Genevois. 2019. A comparative survey of service facility location problems. *Annals of Operations Research* 292 (2019), 399–468. Issue 1. https://doi.org/10.1007/s10479-019-03385-x.

[17] Igor Vasilyev, Xenia Klimentova, and Maurizio Boccia. 2013. Polyhedral study of simple plant location problem with order. *Oper. Res. Lett.* 41 (2013), 153–158. https://doi.org/10.1016/j.orl.2012.12.006

A Capacitated Mobile Facility Location Problem with Mobile Demand: Recurrent Service Provision to En Route Refugees

Amirreza Pashapour
College of Engineering, Koç University, Istanbul, Turkey
apashapour20@ku.edu.tr

Dilek Günneç
College of Engineering, Özyeğin University, Istanbul Turkey
dilek.gunnec@ozyegin.edu.tr

F. Sibel Salman
College of Engineering, Koç University, Istanbul, Turkey
ssalman@ku.edu.tr

Eda Yücel
College of Engineering, TOBB University of Economics and Technology, Ankara, Turkey
e.yucel@etu.edu.tr

ABSTRACT

In this paper, we help humanitarian organizations provide service via mobile facilities (MFs) to migrating refugees, who attempt to cross international borders. Over a planning horizon, we aim to optimize number and routes and relocations of the MFs over a planning horizon. The problem is represented on a network where several refugee groups relocate in their predetermined paths throughout the periods. To incorporate continuity of service, each refugee group should be served at least once every fixed consecutive periods via capacitated MFs. We aim to minimize the total cost, consisting of fixed, service provision, and MF relocation costs, while ensuring the service continuity requirement. We formulate a mixed integer linear programming (MILP) model for this problem. We develop a matheuristic and an accelerated Benders decomposition algorithm as an exact solution method. The proposed model and solution methods are investigated over instances we extracted from the 2020 Honduras migration crisis.

Keywords: humanitarian logistics; capacitated mobile facility location; mobile demand; en route refugees; Benders decomposition; matheuristic

1 INTRODUCTION

Migration, caused by worldwide economic, political, social, and environmental unfavorable conditions, has become a global phenomenon. It is defined as the movement of people from one place to another with a long-run settling purpose [18]. According to the UN Refugee Agency [31], a refugee is someone who has been forced to leave his or her home country because of violence, famine, or natural disasters. The United Nations High Commissioner for Refugees (UNHCR) reported that about 65.6, 79.5, and 82.4 million individuals were forcibly displaced as refugees during 2017, 2019, and 2020, respectively [30].

Refugees are prone to several health risks. They are highly vulnerable to infectious diseases [32] and are often exposed to traumas due to poor living conditions and forced displacements [8]. According to a survey by Morina et al. [21], post-traumatic stress and anxiety are the most prevalent mental disorders among refugees. Besides psychological problems, refugees also suffer from physical traumas. Comellas et al. [8] emphasizes that the somatic distress associated with functional disabilities warrants more attention in both studies and practice. Clearly, providing medical care, nutrition, and shelter alleviates the difficulties of lengthy and long-lasting displacement for refugees.

In recent years, Mobile Facilities (MFs) have been frequently utilized in providing service for an increasing number of transiting refugees. As an example, we can consider the 2015 refugee crisis, where according to Shortall et al. [25], about 850,000 refugees and asylum seekers moved to Greece in 2015. With the purpose of providing health service, the “Doctors of the World" established the refugee ferry project and provided primary health care on-board a commercial ferry. As another example, Médecins Sans Frontieres (MSF; Doctors Without Borders) opened a mobile clinic on the Serbia-Hungary border and treated almost 100 people each day. This was while about 2,000 people crossed the border every day [13]. These examples underline the significance of providing mobile services such as basic health care and relief item delivery.

In this paper, we focus on the provision of various services including food, medicine, or other relief items to transiting refugees by means of MFs. We aim to support decision making while operating the services efficiently, by optimizing the number, locations, and re-locations of the MFs over time. While doing so, we consider the capacities of the facilities and the service needs as well. We represent the problem on a directed network over several time periods such that the refugee groups entering the network in different periods follow distinct paths defined over the nodes and arcs of the network toward their destination nodes. Meanwhile, the MFs

relocate on the nodes to provide services to the refugee groups at the same node and time period, depending on their service capacities. The service provision for refugee groups should be recurrent and the goal is to serve each refugee group at least once every fixed number of consecutive time periods, that is determined by the service provider, in order to maintain continuity of service over time.

In section 2, we provide a review of the related literature. We define the MCM-FLP-MD formally in section 3 and provide a mixed integer linear programming (MILP) model. In section 4, we develop two alternative solution methods for the problem: a Network Decomposition Matheuristic (NDM) and an accelerated Benders decomposition (BD) approach as an exact solution method. Section 5 presents computational results. Finally, section 6 concludes with remarks and future work directions.

2 LITERATURE REVIEW

Our problem falls into the broad category of Facility Location Problems (FLPs). Such a problem decides where to locate facilities and how to allocate clients while minimizing the costs of serving them [19]. According to Farahani and Hekmatfar [10], FLPs have essentially four components: 1) customers, 2) facilities, 3) a network where customers and facilities are located, and 4) a metric that represents the travel time or distance among customers and facilities. Our focus is mainly on the Mobile Facility Location Problem (MFLP) and Mobile Facility Routing Problem (MFRP), where MFs can re-position on a network. The MFLP has increasingly gained researchers' attention mainly because of its applicability in social outreach activities. Recent surveys of MFLP in health care systems and humanitarian logistics include Afshari and Peng [1], Ahmadi-Javid et al. [2], Song et al. [26], and Nasrabadi et al. [22]. The MFRP was addressed by [14] with the aim of maximizing the served demand via mobile fleets over a time horizon. The authors proposed approximate solution methods for their proposed mixed integer program and showed that although the problem is NP-Hard, it is polynomially solvable for the single facility case. Later on, Halper et al. [15] introduced an integer programming formulation for MFLP and suggested a decomposition algorithm for the formulation followed by local neighborhood search heuristics. Most studies of MFLP aim to minimize the total distance traveled by the facilities, while all demand is served [11]. On the other hand, Bélanger et al. [4] discussed recent optimization models for location, relocation, and dispatching of medical MFs such as ambulances. The authors noted that the equity and fairness metrics have recently become of highest importance in this context. Our problem inherently captures these metrics since we define an identical service frequency τ to cover the needs of all refugees uniformly.

Facility capacities have been ignored in most MFLP studies. Raghavan et al. [23] studied the capacitated version of the MFLP (CMFLP) where facilities may move only once, and clients travel to the facilities. They developed a branch-and-price algorithm and two heuristic algorithms for this problem. Our problem extends this study in terms of multi-period planning and the continuity of service frequency that is explicitly accounted for.

In addition to the mobility of facilities, we focus on the mobility of demand as well. We see that demand mobility has been captured in the flow-interception location problem (FILP) in the literature. In fact, the single-period FILP was initially studied by Hodgson [16, 17] and Berman [6]. The objective of FILP is to find locations for facilities to maximize the coverage of demand flow. Applications of the flow-demand coverage problem lie mostly in urban areas, where providing a service only once is sufficient to cover a customer's requirements, as opposed to our case. Berman [7] studied the FILP for customers who travel in a network, not just for the purpose of receiving service. The author divided the customers into stationary and mobile types and considered the existence of both demand types in their formulations. Zeng et al. [33] introduced an "integer-friendly" generalized formulation for the flow-interception model, which can solve several deterministic flow-interception problems. As an extension, the multi-period FILP was introduced by Sterle et al. [27], where some portable facilities intercept the demand flow, and various objectives such as maximizing the intercepted demand or minimizing the relocation costs are inspected.

Most objective functions focus on minimizing fixed or variable costs, or total distance or travel time, or maximizing total benefit under demand coverage. Also, the mobility of both facilities and demand followed by periodic services has been addressed scarcely in the literature. Among solution techniques, heuristics are prevailing solution methods suggested for the FLP and MFLP; however, we provide an exact solution method in addition to a heuristic solution approach. To the best of our knowledge, the MCM-FLP-MD with periodic service provision is introduced for the first time in this study.

3 PROBLEM DEFINITION

In this section, we provide details of the MCM-FLP-MD by first explaining its key elements, then followed by formulating an MILP model for the problem. This problem is set on a connected graph $G = (V, A)$, where node set V and A indicate locations of interest and the roads connecting them, respectively. Refugee groups follow predetermined paths on the network beginning from a source node and ending at a destination node. Each arc on each path is defined in the following way: it takes only one time period to move along the arc according to a refugee group's transportation mode, i.e., by walking or by a vehicle.

We denote the set of paths by P, which is w.l.o.g. finite. For each path $p \in P$, l_p and n_{pk} represents the number of nodes, and k^{th} node on the path, respectively. For instance, $n_{p,1}$ and n_{p,l_p} denote the origin and destination nodes of path p, respectively. Every period, each refugee group moves from a node to the next node along its path. More than one refugee group may follow the same path by entering the path in different periods.

Refugees typically move in groups, and follow a path that is determined at the beginning of their trip. Refugees who enter the network in the same period and follow the same path are assumed to form one refugee group $r \in R$ in our study. We assume the humanitarian organizations that provide services can predict these paths for planning purposes [12, 28]. We denote the path traversed by some refugee group $r \in R$ by $p_r \in P$, and the time period when

they enter the path by $e_r \in T$, where T is the set of periods on the planning horizon and each time period can stand for a single day. The length of the planning horizon is determined such that all refugee groups will leave the network by period $|T|+1$. Considering the path lengths and time periods when refugee groups enter the network, we note that it is important to determine $|T|$ precisely to avoid unnecessary variables. Accordingly, we use Equation (1) to calculate $|T|$.

$$T = \{1, ..., |T|\}\ , \quad |T| = \max_{r \in R}\{l_{p_r} + e_r - 1\} \tag{1}$$

Let d_r represent the demand level of refugee group $r \in R$, which shows their population size. To guarantee demand satisfaction under limited capacities of MFs, a refugee group can be served simultaneously by multiple MFs at a node. For continuity of service, the whole population of each refugee group must be served at least once and could partially be served several times every consecutive τ periods.

M represents the set of available MFs. Each MF $m \in M$ has a capacity Δ_m, indicating maximum number of refugees that facility m can serve in a single period. Not all MFs need to be used in a solution. Thus, $|M|$ is an upper bound on the number of required MFs and the model decides the number of MFs to be utilized based on service requirements. Recruited MFs can enter the network at any time. The entrance point of an MF is the first node where it should provide service at.

State of being at node $i \in V$ in period $t \in T$ is represented by "(i, t) node-time pair" for both refugees and MFs. The process of providing service via an MF $m \in M$ at node-time pair (i, t) is referred to as a 'service act'. A solution to the problem consists of a list of service acts for each recruited MF. In each time period, Every MF can just show up at one node-time pair and if it provides service at that node, it incurs a service act cost. Indeed, each service act of an MF occurs by the presence of that MF and at least one refugee group at a specific node-time pair. Finally, the objective function of this problem comprises three terms: 1) the total fixed cost of utilizing the MFs, 2) the total operating costs associated with the service acts, and 3) the total transportation cost associated with the relocation of MFs on the network.

3.1 Mathematical Model

We propose the following MILP to formulate the MCM-FLP-MD.

Sets:

V	Set of nodes
P	Set of paths
R	Set of refugee groups entering the network
M	Set of potential mobile facilities
T	Set of time periods

Parameters:

d_r	Population of refugee group $r \in R$
p_r	Path traversed by refugee group $r \in R$ ($p_r \in P$)
e_r	Time period in which refugee group $r \in R$ enters path p_r ($e_r \in T$)
l_p	Number of nodes on path $p \in P$
n_{pk}	k^{th} node on path $p \in P$ where $k = 1, ..., l_p$
Δ_m	Service capacity for mobile facility $m \in M$
f_m	Fixed cost of using mobile facility $m \in M$
o_m	Service act operating cost for mobile facility $m \in M$
c_{ij}	Traveling cost from node $i \in V$ to node $j \in V$
τ	Service frequency level in terms of number of time periods

Decision Variables:

A_{irt}	Fraction of population of refugee group $r \in R$ who are planned to receive service at node $i \in V$ in period $t \in T$
S_{imt}	1, if mobile facility $m \in M$ provides service at node $i \in V$ in period $t \in T$; 0, otherwise
X_{ijmt}	1, if mobile facility $m \in M$ is located at node $i \in V$ in period $t \in T$ and at node $j \in V$ in period $t+1$; 0, otherwise (Since there is no refugee group in the network in $t = \|T\| + 1$, MFs remain at their nodes in between t and $t+1$ where $t = \|T\|$)
Y_m	1, if mobile facility $m \in M$ is used; 0, otherwise

MILP Mathematical Model:

$$\min \sum_{m \in M}(f_m Y_m + \sum_{i \in V}\sum_{t \in T} o_m S_{imt} + \sum_{i \in V}\sum_{j \in V}\sum_{t \in T} c_{ij} X_{ijmt}) \tag{2}$$

subject to

$$\sum_{t'=0}^{\tau-1} A_{(n_{p_r,q+t'+1}),r,(e_r+q+t')} \geq 1 \qquad \forall r \in R, 0 \leq q \leq l_{p_r} - \tau \tag{3}$$

$$\sum_{m \in M} \Delta_m S_{imt} \geq \sum_{r \in R} d_r A_{irt} \qquad \forall i \in V, t \in T \tag{4}$$

$$S_{imt} \leq \sum_{j \in V} X_{ijmt} \qquad \forall i \in V, m \in M, t \in T \tag{5}$$

$$\sum_{j \in V} X_{jimt} \leq \sum_{j \in V} X_{ijm(t+1)} \qquad \forall i \in V, m \in M, t = 1, ..., |T| - 1 \tag{6}$$

$$\sum_{i \in V}\sum_{j \in V} X_{ijmt} \leq Y_m \qquad \forall m \in M, t \in T \tag{7}$$

$$A_{irt} \geq 0 \qquad \forall i \in V, r \in R, t \in T \tag{8}$$

$$S_{imt} \in \{0, 1\} \qquad \forall i \in V, m \in M, t \in T \tag{9}$$

$$X_{ijmt} \in \{0, 1\} \qquad \forall i, j \in V, m \in M, t \in T \tag{10}$$

$$Y_m \in \{0, 1\} \qquad \forall m \in M \tag{11}$$

The objective function (2) minimizes the total costs consisting of the MF establishment costs, service act costs and relocation costs. Constraints (3) guarantee that each refugee group is served at least once, and could partially be served multiple times during consecutive τ periods. Constraints (4) ensure that the total capacity provided by MFs at each node-time pair satisfies the fractional demand of refugee groups who are planned to receive service. Constraints (5) indicate that an MF can provide service at a node-time pair only if it is located there. Constraints (6) define flow conservation of MFs among the nodes of the network from one period to the next. Constraints (7) determine whether an MF is used. Finally, constraints (8)-(11) define the domains of the decision variables.

4 SOLUTION METHODS

In this section, we introduce two solution methods to solve the MCM-FLP-MD. We first develop a Network Decomposition Matheuristic (NDM), which requires relatively short run time compared to

run times of our MILP formulation. Second, we develop an accelerated Benders decomposition (BD) algorithm as an exact solution method. We describe the NDM and the BD in detail in sections 4.1 and 4.2, respectively.

4.1 Network Decomposition Matheuristic

The Network Decomposition Matheuristic (NDM) algorithm aims to manage the complexity associated with variables having multiple indices by decomposing the problem based on paths. Each path $p \in P$ traversed by some refugee groups, corresponds to a smaller network and forms a subproblem for MCM-FLP-MD, referred to as (SP^p). Instead of solving the problem over the entire network, NDM solves $n \leq |P|$ subproblems independently, each having considerably tighter solution space. Results of each subproblem creates a partial solution to the original problem, independent of other subproblems. Aggregation of all solution pieces forms the solution associated with the entire network.

For each subproblem (SP^p), R^p denotes the set of refugee groups that migrate along path p. Also, the set of MFs required for serving R^p on path $p \in P$ is referred to as M^p and the minimum number of these MFs is calculated by Equation (12).

$$|M^p| = \left\lceil \frac{\sum_{r \in R^p} d_r}{\Delta \tau} \right\rceil \tag{12}$$

While solving the SP^p, insufficient $|M^p|$ may lead to infeasible solutions. In such cases, we can simply increment $|M^p|$ by one unit and solve the (SP^p) again until we reach a feasible solution. The planning horizon for each subproblem is kept the same as that of the original problem to facilitate the consolidation of the subproblem solutions. The mathematical model corresponding to subproblem SP^p given below, is a simplified version of the MILP for the original problem where the Y_m variables are excluded. By defining the sets and parameters independently for each path, the problem size is reduced significantly.

New Sets:

V^p	Set of nodes of the network that lie on path $p \in P$
R^p	Set of refugee groups entering path $p \in P$
M^p	Set of recruited mobile facilities to provide service along path $p \in P$

Mathematical Model:

$$\min \sum_{i \in V^p} \sum_{m \in M^p} \sum_{t \in T} o_m S_{imt} + \sum_{i \in V^p} \sum_{j \in V^p} \sum_{m \in M^p} \sum_{t \in T} c_{ij} X_{ijmt} + \sum_{m \in M^p} f_m \tag{13}$$

subject to

$$\sum_{t'=0}^{\tau-1} A_{(n_{p,q+t'+1}),r,(e_r+q+t')} \geq 1 \qquad \forall r \in R^p, 0 \leq q \leq l_p - \tau \tag{14}$$

$$\sum_{m \in M^p} \Delta_m S_{imt} \geq \sum_{r \in R^p} d_r A_{irt} \qquad \forall i \in V^p, t \in T \tag{15}$$

$$S_{imt} \leq \sum_{j \in V^p} X_{ijmt} \qquad \forall i \in V^p, m \in M^p, t \in T \tag{16}$$

$$\sum_{j \in V^p} X_{jimt} \leq \sum_{j \in V^p} X_{ijm(t+1)} \qquad \forall i \in V^p, m \in M^p, t = 1, ..., |T| - 1 \tag{17}$$

$$A_{irt} \geq 0 \qquad \forall i \in V^p, r \in R^p, t \in T \tag{18}$$

$$S_{imt} \in \{0, 1\} \qquad \forall i \in V^p, m \in M^p, t \in T \tag{19}$$

$$X_{ijmt} \in \{0, 1\} \qquad \forall i, j \in V^p, m \in M^p, t \in T \tag{20}$$

The NDM procedure is described in detail in Algorithm 1. In this algorithm, we initially set both the number of MFs and the overall objective value to 0. Then, for each path $p \in P$, we determine the refugee groups traversing path p and calculate the required MFs to serve on path p based on Equation (12). After solving the subproblem SP^p, we update the list of MFs and the objective value of the problem solved up to that point.

Algorithm 1 NDM Framework

1: $N_{MF}^{NDM}, Z^{NDM} \leftarrow 0$. // Total number of MFs to be used in the final solution and final objective function value.
2: **for** $p \in P$ **do**:
3: **if** $p \in \{p_r;\ \forall r \in R\}$ **then**: // Checking if the path p is traversed by refugee groups. If true, we solve SP.
4: $R^p \leftarrow \{r \in R : p_r = p\}$ // Defining R^p.
5: Calculate $|M^p|$ using Equation (12). // Calculating number of MFs to be used in path p.
6: **while** True **do**: // Checking feasibility of the solution of SP^p considering $|M^p|$.
7: $M^p \leftarrow \{N_{MF}^{NDM} + 1, ..., N_{MF}^{NDM} + |M^p|\}$. // Updating the elements of set M^p to differentiate the MFs of subproblems.
8: Solve MILP for the subproblem SP^p and obtain $Z(SP^p)$ if SP^p is feasible.
9: **if** SP^p is infeasible **then**:
10: $|M^p| \leftarrow |M^p| + 1$.
11: **continue**
12: **end if**
13: **break**
14: **end while**
15: $N_{MF}^{NDM} \leftarrow N_{MF}^{NDM} + |M^p|$ // Updating the number of MFs used up to this iteration.
16: $Z^{NDM} \leftarrow Z^{NDM} + Z(SP^p)$ // Updating the costs of the network up to this iteration.
17: **end if**
18: **end for**
Output: N_{MF}^{NDM}, Z^{NDM} // Return the number of MFs and the objective value of the solution of NDM.

4.2 Benders Decomposition Algorithm

The Benders Decomposition (BD) method was introduced in the early 1960s as a partition-based solution strategy for large-scale MILPs [5]. BD is successfully applied in diverse fields [24]. In the field of transportation, Costa [9] presented a review of BD applications on network design problems, where integer and continuous variables are mainly associated with arc selection and commodity flow amounts, respectively. The author indicated that BD outperforms some traditional techniques such as Branch-and-Bound or Lagrangian Relaxation for network design problems because of their special structure.

In BD, the problem is divided into a restricted master problem (RMP) and a linear subproblem (LSP). The RMP consists of constraints that contain pure integer variables. The LSP, is obtained via fixing the values of integer variables based on the solution of the RMP. Iteratively, the solution of the RMP is used in the dual of the LSP, referred to as dual subproblem (DSP) and the solution of the DSP generates Benders feasibility or optimality cuts for the RMP. This procedure is continued until a stopping criterion is met. The LSP, DSP, and RMP models associated with our mathematical model are introduced next.

LSP: (Contains the continuous decision variables A_{irt})

$$\min \quad 0 \tag{21}$$

subject to

$$\sum_{t'=0}^{\tau-1} A_{(n_{p_r,q+t'+1}),r,(e_r+q+t')} \geq 1 \qquad \forall r \in R, 0 \leq q \leq l_{p_r} - \tau \tag{22}$$

$$\sum_{m \in M} \Delta_m \overline{S}^{\theta}_{imt} \geq \sum_{r \in R} d_r A_{irt} \qquad \forall i \in V, t \in T \tag{23}$$

$$A_{irt} \geq 0 \qquad \forall i \in V, r \in R, t \in T \tag{24}$$

In the LSP, $\overline{S}^{\theta}_{imt}$ are defined to be equal to the values of S_{imt} variables of the RMP at iteration θ. Since the A_{irt} do not contribute to the objective function of the problem, the objective function of the LSP is set to 0.

Dual Decision Variables: (Corresponding to const. (22) and (23))

u_{rq} $\forall r \in R, q = 0, ..., l_{p_r} - \tau$
v_{it} $\forall i \in V, t \in T$

DSP: (Dual of the LSP model)

$$\max \sum_{r \in R} \sum_{q=0,...,l_{p_r}-\tau} u_{rq} - \sum_{i \in V} \sum_{m \in M} \sum_{t \in T} \Delta_m \overline{S}^{\theta}_{imt} v_{it} \tag{25}$$

subject to

$$\sum_{max\{0,k-\tau\} \leq q \leq min\{k-1, l_{p_r}-\tau\}} u_{rq} - d_r v_{n_{p_r k}, e_r+k-1} \leq 0 \qquad \forall r \in R,\ k = 1, ..., l_{p_r} \tag{26}$$

$$u_{rq} \geq 0 \qquad \forall r \in R, q = 0, ..., l_{p_r} - \tau \tag{27}$$

$$v_{it} \geq 0 \qquad \forall i \in V, t \in T \tag{28}$$

We refer to the objective function of the DSP at iteration θ as W^{θ}_{DSP}.

RMP: (Consists of pure binary variables and $\eta \geq 0$)

$$\min \sum_{m \in M} f_m Y_m + \sum_{i \in V} \sum_{m \in M} \sum_{t \in T} o_m S_{imt} + \sum_{i \in V} \sum_{j \in V} \sum_{m \in M} \sum_{t \in T} c_{ij} X_{ijmt} + \eta \tag{29}$$

subject to

$$S_{imt} \leq \sum_{j \in V} X_{ijmt} \qquad \forall i \in V, m \in M, t \in T \tag{30}$$

$$\sum_{j \in V} X_{jimt} \leq \sum_{j \in V} X_{ijm(t+1)} \qquad \forall i \in V, m \in M, t = 1, ..., |T| - 1 \tag{31}$$

$$\sum_{i \in V} \sum_{j \in V} X_{ijmt} \leq Y_m \qquad \forall m \in M, t \in T \tag{32}$$

$$S_{imt} \in \{0, 1\} \qquad \forall i \in V, m \in M, t \in T \tag{33}$$

$$X_{ijmt} \in \{0, 1\} \qquad \forall i, j \in V, m \in M, t \in T \tag{34}$$

$$Y_m \in \{0, 1\} \qquad \forall m \in M \tag{35}$$

$$\eta \geq 0 \tag{36}$$

The feasibility and optimality cuts are incorporated into the RMP throughout the iterations according to Equations (37) and (38).

$$0 \geq \sum_{r \in R} \sum_{q=0,...,l_{p_r}-\tau} \overline{u}^{\theta}_{rq} - \sum_{i \in V} \sum_{m \in M} \sum_{t \in T} \Delta_m S_{imt} \overline{v}^{\theta}_{it} \qquad \text{“feasibility cut”} \tag{37}$$

$$\eta \geq \sum_{r \in R} \sum_{q=0,...,l_{p_r}-\tau} \overline{u}^{\theta}_{rq} - \sum_{i \in V} \sum_{m \in M} \sum_{t \in T} \Delta_m S_{imt} \overline{v}^{\theta}_{it} \qquad \text{“optimality cut”} \tag{38}$$

Proposition 4.1. *The optimal value for η is 0.*

Proof. Since the optimal objective value of the LSP is 0, according to the duality theory, optimal objective value of DSP must be 0. This implies that we can directly 0 to the variable η. □

Proposition 4.2. *Starting from any feasible solution, if $W^{\theta}_{DSP} = 0$ at an iteration θ, then the stopping criterion for our BD algorithm is met and the solution of RMP in the corresponding iteration is optimal solution of the original problem.*

Proof. Z^{RMP} provides a lower bound (LB) for the optimal objective value. The term $c^T y + W^{\theta}_{DSP}$ (where y and c^T represent the binary variables (S, X, Y) and their coefficients in the objective function of the original problem, respectively), provides an upper bound (UB). The algorithm stops when $LB = UB$. We showed in Proposition 4.1 that the optimal value for variable η is 0. Letting $\eta = 0$ implies $Z^{RMP} = c^T y$. Therefore, we conclude that $LB = UB$ and the algorithm stops, if $W^{\theta}_{DSP} = 0$ is obtained at any iteration θ. □

4.3 Improvements for the proposed BD algorithm

Although BD benefits a powerful theory, the straightforward application of classical BD is usually slow in convergence. In this section, we applied two refinements on the proposed BD.

(1) **Multi-cut implementation:** At each iteration of the classic BD, a single Benders cut is inserted to the RMP. However, the multi-cut reformulation outperforms the single-cut approach as it strengthens the RMP more quickly [24]. So, we generate multiple Benders cuts by solving DSP several times at a single iteration and insert those cuts simultaneously into the RMP.

(2) **Linear RMP Relaxation:** Solving the RMP is usually time consuming and solving it to optimality in the initial iterations is not necessary. McDaniel and Devine [20] showed that valid Benders cuts can be generated by the solutions to the LP relaxation of RMP. We solve the linear relaxation of the RMP (LR-RMP) at early iterations of the algorithm. By applying this approach, the RMP is enriched with high-quality Benders cuts.

5 COMPUTATIONAL ANALYSIS

We implemented the MILP formulation and solution methods on the real-life migration crisis, took place in Honduras in late 2020, when groups of refugees began migrating from Central America, with the hope of reaching Mexico and the USA, often on foot and in groups known as “caravans". Guatemalan migration officials estimated that about 6,000 migrants, most of them Honduran, were corralled between Chiquimula and the border with Honduras. Another caravan of about 4,000 refugees, mostly Honduran migrants, had camped out near the village of Vado Hondo in Guatemala. The migrants were traveling by a combination of walking, hitchhiking, and bus [3, 29].

In section 4, we proposed two solution methods for the MCM-FLP-MD. In addition to these solution methods, we directly solved the instances via the proposed MILP formulation. We implemented the MILP formulation and the two proposed solution methods in the Python programming environment, Spyder Anaconda IDE 4.1.5 platform and solved them using the Pyomo optimization package and the solver CPLEX 12.10.0.. All experiments are conducted on a workstation with 64-bit operating system, Xenon(R) 2.60 GHz

CPU and 128 GB of RAM (18 cores and 36 processors). Considering instance complexities and solution specifications, 0.01% and 0.02% optimality gaps are allowed for the medium- and large-sized instances. Moreover, all instances are solved with a 6-hour (21600s) time limit. Due to relatively short run times associated with small-sized instances, we applied the accelerated BD procedure only to medium- and large-sized instances.

According to Table 1, the MILP formulation is preferred for small-sized networks with approximately 20 nodes and a time horizon of about two weeks as it achieves optimal solutions quickly. Also, the accelerated BD is preferred for medium-sized networks comprised of approximately 30 nodes and with a time horizon of about 3 weeks. Finally, the NDM is preferred when the instance sizes are large, consisting of about 50 nodes or more and lasting for more than a month. Finally, we observed a 2.6% objective value gap between the NDM and MILP results among all 60 instances.

Instance set	Average run-time			Average Obj. value ratio			Solver mip Gap %	
	NDM	MILP	BD	NDM/MILP	NDM/BD	MILP/BD	MILP	BD
Small	24.8	118.2	-	1.017	-	-	-	-
Medium	152.5	996.4	624.1	1.031	1.031	1	-	-
Large	856.2	19095.2	12919.2	1.027	1.036	1.009	1.55	0.71

Table 1: Overall results corresponding to instance sets

6 CONCLUSION

In this paper, we studied a multi-period capacitated mobile facility location problem with mobile demands (MCM-FLP-MD). This problem aims to provide recurrent humanitarian aid to en route refugee groups during their migration in an effective manner using capacitated Mobile Facilities (MFs). We proposed an MILP formulation for the problem followed by two solution methods: a Network Decomposition Matheuristic (NDM) and an accelerated Benders decomposition (BD) approach as an exact solution method. Our observations indicated that regarding tun times, the MILP formulation, accelerated BD and NDM algorithm are most suitable for solving small, medium, and large-sized instances, respectively.

A future research direction for this problem is to further improve both the MILP model by incorporating suitable valid inequalities, and the NDM algorithm in order to better utilize capacities of MFs in the network. Also, incorporating uncertainties in the network, especially regarding the predetermined paths assigned to the refugees and their displacement patterns is another research direction for which stochastic dynamic programming can be investigated. Constructive heuristics and metaheuristic solution methods may also be applicable for extensions of this problem.

Acknowledgment: This research is supported by TUBITAK [Grant number 119M229].

REFERENCES

[1] H Afshari and Q Peng. 2014. Challenges and solutions for location of healthcare facilities. *Industrial Engineering and Management* 3, 2 (2014), 1–12.

[2] Amir Ahmadi-Javid, Pardis Seyedi, and Siddhartha S Syam. 2017. A survey of healthcare facility location. *Computers & Operations Research* 79 (2017), 223–263.

[3] BBC News. 2021. Migrant caravan: Mexico presses US to reform immigration policies. https://www.bbc.com/news/world-latin-america-55714865, last accessed 06.22.2021.

[4] Valérie Bélanger, A Ruiz, and Patrick Soriano. 2019. Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles. *European Journal of Operational Research* 272, 1 (2019), 1–23.

[5] J. F. Benders. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* 4, 1 (1962), 238–252.

[6] O Berman. 1995. Flow-interception problems. *Facility location: A survey of applications and methods* (1995), 389–426.

[7] O Berman. 1997. Deterministic flow-demand location problems. *Journal of the Operational Research Society* 48, 1 (1997), 75–81.

[8] Ruben Moreno Comellas, Nino Makhashvili, Ivdity Chikovani, Vikram Patel, Martin McKee, Jonathan Bisson, and Bayard Roberts. 2015. Patterns of somatic distress among conflict-affected persons in the Republic of Georgia. *Journal of Psychosomatic Research* 78, 5 (2015), 466–471.

[9] Alysson M Costa. 2005. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research* 32, 6 (2005), 1429–1450.

[10] R.Z. Farahani and M. Hekmatfar. 2009. Facility Location: Concepts, Models, Algorithms and Case Studies. In *Facility Location*. Contributions to Management Science, Physica-Verlag Heidelberg.

[11] Zachary Friggstad and Mohammad R Salavatipour. 2011. Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms (TALG)* 7, 3 (2011), 1–22.

[12] Derek Groen. 2016. Simulating refugee movements: Where would you go? *Procedia Computer Science* 80 (2016), 2251–2255.

[13] Anne Gulland. 2015. The refugee crisis: what care is needed and how can doctors help? *BMJ: British Medical Journal (Online)* 351 (2015).

[14] Russell Halper and Srinivasa Raghavan. 2011. The mobile facility routing problem. *Transportation Science* 45, 3 (2011), 413–434.

[15] Russell Halper, S Raghavan, and Mustafa Sahin. 2015. Local search heuristics for the mobile facility location problem. *Computers & Operations Research* 62 (2015), 210–223.

[16] M John Hodgson. 1981. The location of public facilities intermediate to the journey to work. *European Journal of Operational Research* 6, 2 (1981), 199–204.

[17] M John Hodgson. 1990. A flow-capturing location-allocation model. *Geographical Analysis* 22, 3 (1990), 270–279.

[18] International Organization for Migration. 2020. WORLD MIGRATION REPORT 2020. "https://worldmigrationreport.iom.int/wmr-2020-interactive//", last accessed 06.15.2021.

[19] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha-da Gama. 2019. Introduction to location science. In *Location Science*. Springer, 1–21.

[20] Dale McDaniel and Mike Devine. 1977. A modified Benders' partitioning algorithm for mixed integer programming. *Management Science* 24, 3 (1977), 312–319.

[21] Naser Morina, Aemal Akhtar, Jürgen Barth, and Ulrich Schnyder. 2018. Psychiatric disorders in refugees and internally displaced persons after forced displacement: A systematic review. *Frontiers in Psychiatry* 9 (2018), 433.

[22] Alireza Motallebi Nasrabadi, Mehdi Najafi, and Hossein Zolfagharinia. 2020. Considering short-term and long-term uncertainties in location and capacity planning of public healthcare facilities. *European Journal of Operational Research* 281, 1 (2020), 152–173.

[23] S Raghavan, Mustafa Sahin, and F Sibel Salman. 2019. The capacitated mobile facility location problem. *European Journal of Operational Research* 277, 2 (2019), 507–520.

[24] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. 2017. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259, 3 (2017), 801–817.

[25] Clare K Shortall, Rosanna Glazik, Alvin Sornum, and Ceri Pritchard. 2017. On the ferries: the unmet health care needs of transiting refugees in Greece. *International Health* 9, 5 (2017), 272–280.

[26] Byung Duk Song, Young Dae Ko, James R Morrison, and Hark Hwang. 2013. Capacitated location and allocation models of long-term care facilities. *Industrial Engineering and Management Systems* 12, 3 (2013), 190–197.

[27] Claudio Sterle, Antonio Sforza, and Annunziata Esposito Amideo. 2016. Multi-period location of flow intercepting portable facilities of an intelligent transportation system. *Socio-Economic Planning Sciences* 53 (2016), 4–13.

[28] Diana Suleimenova, David Bell, and Derek Groen. 2017. A generalized simulation development approach for predicting refugee destinations. *Scientific Reports* 7, 1 (2017), 1–13.

[29] The New York Times. 2021. Migrant Caravan, Now in Guatemala, Tests Regional Resolve to Control Migration. https://www.nytimes.com/2021/01/17/world/americas/migrant-caravan-us-biden-guatemala-immigration.html, last accessed 06.22.2021.

[30] UN Refugee Agency. 2021. Refugee Statistics. https://www.unrefugees.org/refugee-facts/statistics/, last accessed 06.25.2021.

[31] UN Refugee Agency. 2021. What is a Refugee? https://www.unrefugees.org/refugee-facts/what-is-a-refugee/, last accessed 06.15.2021.

[32] WHO. 2019. Migrants and refugees at higher risk of developing ill health than host populations, reveals first-ever WHO report on the health of displaced people in Europe. http://bit.ly/2RHpKFa, last accessed 06.17.2021.

[33] Weiping Zeng, Ignacio Castillo, and M John Hodgson. 2010. A generalized model for locating facilities on a network with flow-based demand. *Networks and Spatial Economics* 10, 4 (2010), 579–611.

Location problems with interconnected facilities

Yerlan Kuzbakov[1] and Ivana Ljubić[2]

[1]ESSEC Business School of Paris, Cergy, France, ✉ yerlan.kuzbakov@essec.edu
[2]ESSEC Business School of Paris, Cergy, France, ✉ ivana.ljubic@univie.ac.at

Location problem with interconnected facilities is a cost minimization problem deciding on which facility nodes to open and how to allocate customers to open facilities, under an additional constraint that all open facilities should be interconnected, i.e., neighboring open facilities should be within some radius $r \geq 0$ from each other. Interconnectivity between the facilities is an important feature for modeling communication between sensors [6], or in the context of designing the radio networks [2]. What distinguishes our problem from related ones studied in the literature under the common name of Generalized Steiner Tree-Star problems (see, e.g., [4] and further references therein) is the cost function. There is a fixed cost associated to opening a facility, and a cost for allocating a customer to an open facility. However, we do not incur costs for the edges interconnecting the open facilities. This assumption can be explained by the nature of such connection, informational, not physical, between facilities. We can further require facilities to serve only customers within a radius R.

In this work, we consider two problem variants:

- the Median Problem with Interconnected Facilities (MPIF), and
- the Covering location Problem with Interconnected Facilities (CPIF).

Both problems have been introduced by [1]. For the former problem, we are bound to cover all customers by open facilities, and we search for a subset of facilities to open so that the opening plus allocation cost is minimized. For the latter problem, we incur a penalty for not covering a customer, and the goal is to find a subset of facilities to open so that the sum of facility opening cost together with the penalties for customers that remain uncovered is minimized. A feasible solution to the CPIF is shown in Figure 1.

We introduce new ways to model the MPIF and the CPIF problems by combining non-compact formulations for connectivity, allocation, and coverage constraints. We first provide some theoretical insights concerning the quality of lower bounds of such obtained formulations. We also provide an empirical comparison of these new models based on experiments conducted on a dataset from the literature. Finally, we also conduct a sensitivity analysis discussing the major input properties that make some instances more difficult to solve than others. Our results show that by developing tailored branch-and-cut algorithms, we can outperform off-the-shelf methods provided by the commercial solver CPLEX.

Future research on these problems would be to find under which conditions the problems could be solved in polynomial-time. In particular, what happens if the input network is very sparse (i.e., if it is a tree or a cactus graph), as it is known that without the interconnectivity requirements such algorithms exist for the p-median problem [3], [7]. Similarly, it is an interesting open question to find out whether the MPIF or the CPIF are fixed parameter tractable as it was the case for some of the related network design problems [5].

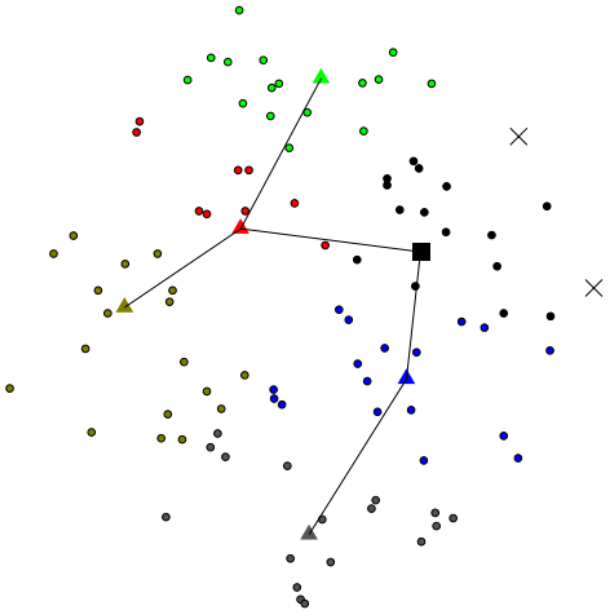


Figure 1: Sample network of interconnected facilities with customers. We illustrate a CPIF solution in which the black square represents the root node, triangles represent open facilities, and crosses stand for uncovered customers. Colored circles represent customers covered by the facility of the same color.

References

[1] M. Cherkesly, M. Landete, and G. Laporte. Median and covering location problems with interconnected facilities. *Computers & Operations Research*, 107:1–18, 2019.

[2] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, and M. Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms (TALG)*, 5(3):1–30, 2009.

[3] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

[4] M. Leitner, I. Ljubić, J.-J. Salazar-González, and M. Sinnl. An algorithmic framework for the exact solution of tree-star problems. *European Journal of Operational Research*, 261(1):54–66, 2017.

[5] D. Rehfeldt and T. Koch. On the exact solution of prize-collecting Steiner tree problems. 2020.

[6] A. Romich, G. Lan, and J. C. Smith. A robust sensor covering and communication problem. *Naval Research Logistics (NRL)*, 62(7):582–594, 2015.

[7] A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19(2):59–64, 1996.

A new IP formulation of the p-center problem and how to make it stronger

Elisabeth Gaar[1] and Markus Sinnl[1,2]

[1]Institute of Production and Logistics Management, Johannes Kepler University Linz, Austria, ✉ elisabeth.gaar@jku.at
[2]JKU Business School, Johannes Kepler University Linz, Linz, Austria, ✉ markus.sinnl@jku.at

Extended Abstract

One of the most fundamental problems in location science is the (vertex) p-center problem (pCP) (see, e.g., Laporte et al. [8]). In this problem we are given customer demand points and potential facility locations. The task is to choose p of these locations to open a facility such that the maximum distance of any customer demand point to its closest open facility is minimized. More formally, given an integer p, a set of customer demand points I with cardinality $|I| = n$, a set of potential facility locations J of cardinality $|J| = m \geq p$ and a distance d_{ij} from a customer demand point i to the potential facility location j for every $i \in I$ and $j \in J$, the task is to find a subset $S \subseteq J$ with cardinality $|S| = p$ of facilities to open such that the maximum distance between a customer demand point and its closest open facility is minimized, i.e., such that $\max_{i \in I} \min_{j \in S}\{d_{ij}\}$ is minimized.

The pCP was first mentioned in 1965 by Hakimi [6] and was shown to be NP-hard by Kariv and Hakimi [7]. However, there are several applications of the pCP, in particular whenever it is critical that at least one open facility can be reached quickly by each customer demand point. This is for example the case for emergency service locations, such as ambulances or fire stations, and also relief actions in humanitarian crisis.

In literature there are various integer programming (IP) formulations for the pCP. The classical textbook formulation (see for example Daskin [4]) has bad linear programming (LP)-relaxation bounds (see, e.g., Snyder and Chen [9]) and scalability issues, as there are many variables and constraints. As a consequence, exact state-of-the-art approaches (see, e.g., Chen and Chen [2] and Contardo et al. [3]) for the pCP exploit the connection to the so-called set cover problem, by iteratively solving SCPs for different input values.

Elloumi et al. [5] introduced a new IP formulation for the pCP in 2004. They proved that the lower bound obtained by the LP-relaxation of their IP formulation is better than the one from the LP-relaxation of the classical IP formulation. Also Çalık and Tansel [1] introduced a new IP formulation in 2013. They also investigated the connection of their IP formulation to the one of [5] and showed that their formulation yields the best LP-relaxation for the pCP so far. Çalık and Tansel also use their IP formulation within an exact solver for the pCP.

We present a novel solution approach for the pCP that uses an IP formulation that can be viewed as an projection from the classical formulation presented in Daskin [4]. This IP is solved with a branch-and-cut, where cuts for customer demand points are iteratively generated. As a consequence this method is suited for large scale instances, because it is not necessary that the complete distance matrix is kept in memory.

Furthermore, we point out how our IP formulation can be strengthened with combinatorial information in order to obtain a much tighter LP-relaxation compared to the LP-relaxation of the classical formulation presented in Daskin [4]. This strengthening procedure utilizes a new way to use lower bound information in order to obtain stronger cuts. We prove that the bound obtained by the LP-relaxation of our strengthened IP formulation has the same quality as the best known bound in literature, which is based on a semi-relaxation. Moreover, we also demonstrate how our formulation is connected to the set cover problem.

In addition to that, we conduct a computational comparison on instances from literature with up to more than 700,000 customers and locations with state-of-the-art solution algorithms.

References

[1] Hatice Çalık and Barbaros C Tansel. Double bound method for solving the p-center location problem. *Computers & Operations Research*, 40(12):2991–2999, 2013.

[2] Doron Chen and Reuven Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646–1655, 2009.

[3] Claudio Contardo, Manuel Iori, and Raphael Kramer. A scalable exact algorithm for the vertex p-center problem. *Computers & Operations Research*, 103:211–220, 2019.

[4] Mark S Daskin. *Network and discrete location: models, algorithms, and applications*. John Wiley & Sons, 2nd edition, 2013.

[5] Sourour Elloumi, Martine Labbé, and Yves Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94, 2004.

[6] S Louis Hakimi. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13:462–475, 1965.

[7] Oded Kariv and S Louis Hakimi. An algorithmic approach to network location problems. i: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.

[8] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, editors. *Location Science*. Springer, 2nd edition, 2019.

[9] Lawrence V Snyder and Zuo-Jun Max Shen. *Fundamentals of supply chain theory*. Wiley Online Library, 2011.

Session Session 6A: public transport

Friday 10 June 2022, 09:30-11:10

Lecture Hall I

Integrated Line Planning and Vehicle Scheduling for Public Transport*

Philine Schiewe
Technical University Kaiserslautern
Kaiserslautern, Germany
p.schiewe@mathematik.uni-kl.de

Moritz Stinzendörfer
Technical University Kaiserslautern
Kaiserslautern, Germany
stinzendoerfer@mathematik.uni-kl.de

ABSTRACT

In public transport planning, the operational costs are mainly determined by the vehicle schedule. However, in the traditionally sequential planning approach, vehicle scheduling is one of the last problems that is considered. We therefore propose an integrated formulation for line planning and vehicle scheduling problem, which brings an appropriate approximation of the operational costs into the first planning stages. We model the integrated problem as a mixed-integer program and propose a heuristic solution approach. Both approaches are tested on close-to real-world data sets from the open source software framework LinTim.

1 INTRODUCTION

With growing urban areas, public transport can play an important role in achieving sustainable mobility by consolidating demand and reducing traffic. For being a viable alternative to individual motorized transport modes, public transport has to be attractive for the passengers, e.g., by offering frequent service and short travel times, and economically viable for the operator. The tasks of finding a public transport supply that is attractive for both passengers and operators, is intricate and comprises various subproblems that are closely interrelated. Some of the most important subproblems are line planning, timetabling and vehicle scheduling, three problems that are traditionally solved sequentially and in that order, see [5, 6]. All three problems are extensively studied, see e.g., [2, 9, 15]. In recent years, the optimization potential arising when several subproblems are considered in an integrated manner has been under research, e.g., in [8, 14].

One important aspect of integration is summarized in the concept of the eigenmodel [16], i.e., to change the order in which the subproblems are considered. In this paper, we combine the idea of the eigenmodel and integrate several subproblems by considering line planning and vehicle scheduling simultaneously in order to minimize the operational costs. Therefore, we construct lines, i.e., paths in the infrastructure network that have to be operated by one vehicle end-to-end, and arrange them into vehicle routes. By refraining from using a fixed line pool, we allow for a larger solution space. Additionally, we provide the possibility to create vehicle schedules for a limited number of vehicles and vehicles with a limited range such as electric vehicles. We provide a mixed-integer programming formulation for a restricted version of the integrated line planning and vehicle scheduling problem and propose a fast heuristic to solve close-to real-world instances. Additionally, our approach can be used for creating line pools to serve as a basis for further planning approaches.

Similar approaches in the literature include the transit route network design problem which often comprises determining routes with corresponding frequencies. Here, many of the approaches are (meta-)heuristics or designed for very specific networks, see [7] for an overview. An integrated model for line planning, timetabling and vehicle scheduling is proposed in [12] but due to its size, it can only be used for very small instances. In [11], a heuristic line pool generation procedure from [4] is adapted to generate lines which allow for cost-efficient vehicle schedules for the case of an undirected public transport network and when no depot is considered. A heuristic sequential approach for first creating a vehicle schedule and then lines is presented in [10] where the goal is to maximize the attractiveness for the passengers.

The remainder of the paper is structured as follows. Section 2 gives an overview on the classical sequential approach to public transport planning. In Section 3, we present our model for the integrated line planning and vehicle routing problem as well as a short analysis. The restricted version with the corresponding mixed-integer program and a heuristic solution approach are presented in Section 4 and experimentally evaluated in Section 5. The paper is concluded in Section 6.

2 SEQUENTIAL PLANNING PROCESS

As input we assume that a *public transport network (PTN)*, i.e., a digraph (V, A), is given. Here, the nodes V represent stations and the arcs A direct connections between them, such as roads or tracks. As we optimize the operational costs of the public transport supply, we assume that the passengers' demand is given and their routes in the PTN are fixed. For a simple path P in (V, A) we denote by $A(P)$ the arcs of P and by $\alpha(P), \omega(P)$ the first and last node of P, respectively. Similarly, we call for arcs $a = (u, v)$ the incident nodes $\alpha(a) = u$ and $\omega(a) = v$.

For the classical *line planning* problem as described, e.g., in [15], lower and upper frequency bounds $f_a^{\min} \leq f_a^{\max}$, $a \in A$, are given which guarantee that passengers can travel on their routes while safety restrictions are respected. The goal is to find a set of lines, i.e., paths in the PTN which adhere to the given bounds.

Definition 2.1. Let a public transport network (V, A) with upper and lower frequency bounds $f_a^{\min} \leq f_a^{\max}$, $a \in A$, and a line pool, i.e., a set of possible lines $\mathcal{L}^0$ be given. The *line planning problem* is to find a subset $\mathcal{L} \subset \mathcal{L}^0$ with frequencies $f(l) \in \mathbb{N}_{>0}$, $l \in \mathcal{L}$, such

*This work is partially funded by DFG under grants SCHO 1140/8-2.

that

$$f_a^{\min} \leq \sum_{l \in \mathcal{L}: a \in A(l)} f(l) \leq f_a^{\max}.$$

We call $(\mathcal{L}, f)$ the corresponding *line concept.*

Obviously, the line pool $\mathcal{L}^0$ has a large influence on the line concept and the quality of the public transport supply, especially if it is too restrictive, see [4]. For the remainder of the paper, we assume that lines have to be simple paths and start and end at terminal stations $\bar{V} \subset V$ but do not impose further restrictions.

During the *timetabling* stage, (periodic) times are assigned to arrivals and departures at each station of each line, i.e., the events are repeated periodically with a fixed period length T. The dependencies between events are modeled as activities, which represent, e.g., vehicles driving or dwelling, and transfers of passengers. Each activity imposes a lower and an upper bound on the difference of the corresponding event times. The resulting problem is the well-known NP-hard periodic event scheduling problem, see [17].

However, for the remainder of the paper, we do not consider headway constraints and assume that transfers between lines have no restricting upper bound. Therefore, a feasible timetable can be constructed easily by considering each line separately.

For a given timetable, a vehicle schedule is constructed with the objective of minimizing the operational costs. We consider a periodic vehicle scheduling problem where all vehicles start from the same depot dep.

Definition 2.2. Let a public transport network (V, A), a line concept $(\mathcal{L}, f)$ and a periodic timetable π with period length T be given. A *periodic vehicle schedule* is a multiset of paths or *(vehicle) routes* $\mathcal{R}$ such that

- each route $r \in \mathcal{R}$ is a concatenation of pairwise disjoint lines $l \in \mathcal{L}$ and
- each line $l \in \mathcal{L}$ is contained in exactly $f(l)$ vehicle routes.

The distance covered by a vehicle route $r \in \mathcal{R}$ is

$$\begin{aligned}\operatorname{dist}(r) = &\sum_{l \in r} \sum_{a \in A(l)} \operatorname{dist}(a) + \operatorname{dist}(\operatorname{dep}, \alpha(r)) + \operatorname{dist}(\omega(r), \operatorname{dep}) \\ &+ \sum_{\substack{l, l' \text{consecutive} \\ \text{lines in } r}} \operatorname{dist}(\omega(l), \alpha(l'))\end{aligned}$$

where $\operatorname{dist}(a)$ is the distance from $\alpha(a)$ to $\omega(a)$. We call

$$\operatorname{dist}(\mathcal{R}) = \sum_{r \in \mathcal{R}} \operatorname{dist}(r)$$

the distance of the vehicle schedule.

Similarly, the duration of a vehicle route $r \in \mathcal{R}$ for timetable π is

$$\begin{aligned}\operatorname{dur}(r, \pi) := &\sum_{l \in r} \sum_{a \in A(l)} \operatorname{dur}(a, l, \pi) \\ &+ \operatorname{dur}(\operatorname{dep}, \alpha(r)) + \operatorname{dur}(\omega(r), \operatorname{dep}) + \operatorname{dur}(\operatorname{dep}) \\ &+ \sum_{\substack{l, l' \text{consecutive} \\ \text{lines in } r}} \operatorname{dur}(\omega(l), \alpha(l'))\end{aligned}$$

where $\operatorname{dur}(a, l, \pi)$ is the duration between the time scheduled for the departure of line l at $\alpha(a)$ and at $\omega(a)$ and $\operatorname{dur}(\omega(l), \alpha(l'))$ the duration of relocating between lines l and l'. Note that the duration for getting from the depot to the first station of r and from the last station of r to the depot does not depend on the timetable. The minimal turn-over time at the depot is represented by dur(dep). We call

$$\operatorname{dur}(\mathcal{R}, \pi) = \sum_{r \in \mathcal{R}} \operatorname{dur}(r, \pi)$$

the duration of the vehicle schedule.

For each route $r \in \mathcal{R}$, the number of vehicles needed to operate it is

$$\operatorname{veh}(r, \pi) = \left\lceil \frac{\operatorname{dur}(r, \pi)}{T} \right\rceil.$$

The total number of vehicles needed to operate $\mathcal{R}$ is

$$\operatorname{veh}(\mathcal{R}, \pi) = \sum_{r \in \mathcal{R}} \operatorname{veh}(r, \pi).$$

For parameters $(\lambda, \mu, \kappa) \in \mathbb{R}^3_{\geq 0}$, we define the *costs* of vehicle schedule $\mathcal{R}$ for timetable π as

$$\operatorname{cost}(\mathcal{R}, \pi) := \lambda \cdot \operatorname{dist}(\mathcal{R}) + \mu \cdot \operatorname{dur}(\mathcal{R}, \pi) + \kappa \cdot \operatorname{veh}(\mathcal{R}, \pi).$$

In the basic definition, there are no restrictions on the vehicle routes. However, restricting the duration of a vehicle route might be important, especially if electric vehicles are considered. In this case, restricting the duration of a route according to the battery capacity and choosing dur(dep) such that the battery can be reloaded guarantees that the vehicle schedule can be operated by electric vehicles.

3 MODELING THE LINE PLANNING AND VEHICLE SCHEDULING PROBLEM

As in the sequential planning process a vehicle schedule is constructed for a given line plan and timetable, we have to adapt our notation for defining the integrated problem.

Definition 3.1. Let a public transport network (V, A) with minimal and maximal frequency $f_a^{\min} \leq f_a^{\max}$, $a \in A$, arc duration $\operatorname{dur}(a)$, $a \in A$, minimal turn-over time dur(dep) at the depot and set of terminal stations $\bar{V} \subset V$ as well as a maximal line duration K and maximal number of routes R be given. The *line planning and vehicle scheduling problem* (LVP) is to find a multiset of simple paths $\mathcal{R}$, i.e., vehicle routes, such that

- for all arcs $a \in A$
$$f_a^{\min} \leq |\{r \in \mathcal{R} \colon a \in A(r)\}| \leq f_a^{\max},$$
i.e., each arc a is contained in at least $f_a^{\min}$ and at most $f_a^{\max}$ vehicle routes,
- for all routes $r \in \mathcal{R}$
$$\begin{aligned}\operatorname{dur}(r) := &\sum_{a \in A(r)} \operatorname{dur}(a) + \operatorname{dur}(\operatorname{dep}) \\ &+ \operatorname{dur}(\operatorname{dep}, \alpha(r)) + \operatorname{dur}(\omega(r), \operatorname{dep}) \\ &\leq K,\end{aligned}$$
i.e., the duration of each route r does not exceed K,
- the number of vehicle routes $|\mathcal{R}| \leq R$,
- $\alpha(r), \omega(r) \in \bar{V}$, i.e., the first and the last node of each vehicle route r are contained in the set of terminal stations and
- for parameter set $(\lambda, \mu, \kappa) \in \mathbb{R}^3_{\geq 0}$ the approximated costs
$$\operatorname{cost}(\mathcal{R}) := \lambda \cdot \operatorname{dist}(\mathcal{R}) + \mu \cdot \sum_{r \in \mathcal{R}} \operatorname{dur}(r) + \kappa \cdot \sum_{r \in \mathcal{R}} \left\lceil \frac{\operatorname{dur}(r)}{T} \right\rceil$$

are minimized.

The corresponding set of lines $\mathcal{L}$ consists of the paths from $\mathcal{R}$ where the multiplicity of $r \in \mathcal{R}$ corresponds to the frequency $f(r)$.

Note that for a feasible solution of (LVP) the vehicle schedule $\mathcal{R}$ and the line concept $(\mathcal{L}, f)$ are feasible by construction. As each vehicle route consists of only one line, we do not have to consider relocating between lines.

Unfortunately, (LVP) is NP-hard even when no restrictions on K and R are imposed.

Theorem 3.2. (LVP) *is NP-hard, even if* $K = R = \infty$ *and* $\bar{V} = V$.

Proof. In this setting, (LVP) is equivalent to finding a costs-minimal line concept from the set of all simple paths. Setting $\lambda = \kappa = 0$ and $\text{dur}(\text{dep}, v) = \text{dur}(v, \text{dep}) = 0$ for all $v \in V$ leads to the same cost structure as in [4] where this problem is shown to be NP-hard. □

The maximal route duration K and the maximal number of routes R can be used to influence the structure of the resulting line concept and vehicle schedule. R restricts the number of vehicle routes and therefore the number of lines such that there are not too many - possible very short - lines which would be undesirable from a passengers' point of view. K restricts the duration of a vehicle route, which is beneficial from a robustness viewpoint as long vehicle routes tend to propagate delays.

Together, they can also be used to bound the number of operated vehicles.

Lemma 3.3. *Let* π *be a feasible timetable with* $\text{dur}(a) \geq \text{dur}(a, l, \pi)$ *for all* $a \in A(l)$, $l \in \mathcal{L}$. *Then*

$$R \cdot \left\lceil \frac{K}{T} \right\rceil \geq |\mathcal{R}| \cdot \left\lceil \frac{K}{T} \right\rceil \geq \sum_{r \in \mathcal{R}} \left\lceil \frac{\text{dur}(r)}{T} \right\rceil \geq \text{veh}(\mathcal{R}, \pi).$$

Proof. The first inequality follows directly from $|\mathcal{R}| \leq R$, the second from $\text{dur}(r) \leq K$ and the last from $\text{dur}(a) \geq \text{dur}(a, l, \pi)$ for all $a \in A(l)$, $l \in \mathcal{L}$. □

Note that we can choose $\text{dur}(a) \geq \text{dur}(a, l, \pi)$ a priori when the construction of the bounds for timetabling is known. Additionally, we can bound the costs $\text{cost}(\mathcal{R}, \pi)$ of the vehicle schedule for a feasible timetable π.

Lemma 3.4. *Let* π *be a feasible timetable with* $\text{dur}(a) \geq \text{dur}(a, l, \pi)$ *for all* $a \in A(l)$, $l \in \mathcal{L}$. *Then*

$$\text{cost}(\mathcal{R}) \geq \text{cost}(\mathcal{R}, \pi).$$

Proof. This follows directly from Lemma 3.3, as $\text{dist}(\mathcal{R})$ is independent of the timetable and $\text{dur}(a) \geq \text{dur}(a, l, \pi)$ for all $a \in A(l)$, $l \in \mathcal{L}$. □

4 SOLVING THE RESTRICTED LINE PLANNING AND VEHICLE SCHEDULING PROBLEM

To solve the integrated line planning and vehicle scheduling problem, we consider the following restriction (rLVP): For each arc $a \in A$, we suppose that an arc frequency $f(a) \in \{f_a^{\min}, \dots, f_a^{\max}\}$ is given and we have to find a solution to (LVP) such that

- for all arcs $a \in A$

$$f(a) = |\{r \in \mathcal{R} \colon a \in r\}|,$$

i.e., each arc a is contained in exactly $f(a)$ vehicle routes and
- for parameter set $(\lambda, \mu, \kappa) \in \mathbb{R}^3_{\geq 0}$ the approximated costs

$$\bar{\text{cost}}(\mathcal{R}) := \lambda \cdot \text{dist}(\mathcal{R}) + \mu \cdot \sum_{r \in \mathcal{R}} \text{dur}(r) + \kappa \cdot |\mathcal{R}| \cdot \left\lceil \frac{K}{T} \right\rceil$$

are minimized.

From Lemma 3.3, we know that the optimal objective value of (rLVP) is an upper bound on the optimal objective value of (LVP).

Corollary 4.1. *Let* $\mathcal{R}$ *be an optimal solution of* (LVP) *and* $\bar{\mathcal{R}}$ *an optimal solution of* (rLVP). *Then*

$$\bar{\text{cost}}(\bar{\mathcal{R}}) \geq \text{cost}(\bar{\mathcal{R}}) \geq \text{cost}(\mathcal{R}).$$

Proof. The first inequality follows directly from Lemma 3.3 while the second inequality follows as $\bar{\mathcal{R}}$ is a feasible solution of (LVP). □

4.1 Graph Construction for Vehicle Routing Formulation

We can model (rLVP) as a slightly modified capacitated vehicle routing problem on the following digraph $\tilde{G} = (\tilde{V}, \tilde{A})$. The idea is that the nodes $\tilde{V}$ represent the arcs of PTN (V, A). By setting the demand of each node in $\tilde{V}$ to the duration of the arc in A and the capacity of the vehicle routing problem to $K - \text{dur}(\text{dep})$, we get a direct correspondence between the vehicle routes in both graphs.

An example of the construction is given in Figure 1.

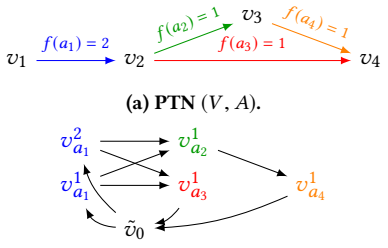
We set

$$\tilde{V} := \{v_a^i : a \in A, i \in \{1, \dots, f(a)\}\} \cup \{\tilde{v}_0\},$$

i.e., we create $|f(a)|$ nodes for each arc $a \in A$ in the public transport network and an artificial depot node $\tilde{v}_0$.

To indicate the corresponding arc $a = (v, w) \in A$ of the created nodes $\tilde{v} \in \tilde{V}$ in the vehicle routing graph, we use $\tilde{v}' := a$ if $\tilde{v} = v_a^i$ for $i \in \{1, \dots, f(a)\}$.

For each node $v \in V$ and each incoming arc $a \in \delta^-(v)$ and outgoing arc $b \in \delta^+(v)$ in PTN (V, A) we create arcs (v_a^i, v_b^j), $i \in \{1, \dots, f(a)\}, j \in \{1, \dots, f(b)\}$, in the vehicle routing graph $\tilde{G}$.



(a) PTN (V, A).

(b) Vehicle routing graph $\tilde{G} = (\tilde{V}, \tilde{A})$.

Figure 1: Transformation of the public transport network (V, A) **to the vehicle routing graph** $\tilde{G}$. **Terminal stations are** $\bar{V} = \{v_1, v_4\}$.

To ensure that each line begins and ends at a terminal station, we have arcs from the artificial depot node $\tilde{v}_0$ to each node $\tilde{v}$ whose corresponding left station $\alpha(\tilde{v}')$ is a terminal station and the other way round if the corresponding right station $\omega(\tilde{v}')$ is a terminal station. Formalized we have

$$\begin{aligned}\tilde{A} :=& \{(v_a^i, v_b^j): \quad a, b \in A, i \in \{1, ..., f(a)\}, j \in \{1, ..., f(b)\},\\ & \omega(a) = v = \alpha(b) \text{ for } v \in V\}\\ \cup\ & \{(\tilde{v}_0, v_a^i): \quad a \in A, i \in \{1, ..., f(a)\}, \alpha(a) = v \text{ for } v \in \bar{V}\}\\ \cup\ & \{(v_a^i, \tilde{v}_0): \quad a \in A, i \in \{1, ..., f(a)\}, \omega(a) = v \text{ for } v \in \bar{V}\}.\end{aligned}$$

The demand of each node $\tilde{v} \in \tilde{V}$ is defined as $d(\tilde{v}) := \text{dur}(a)$ for $\tilde{v}' = a$, whereas the capacity of the vehicles is set to $C := K - \text{dur}(\text{dep})$ and the maximal number of vehicles is set to R.

For a feasible solution to the vehicle routing problem in $\tilde{G}$ we know

- there are at most R tours,
- the demand of all nodes in a tour does not exceed K and
- all nodes $\tilde{V}$ are covered by exactly one tour.

We can translate such a tour $(\tilde{v}_0, \tilde{v}_1, ..., \tilde{v}_n, \tilde{v}_0)$ in $\tilde{G}$ back to a not necessarily simple path $(\tilde{v}_1', ..., \tilde{v}_n')$ in (V, A). By construction, the resulting set of paths covers all arcs according to their frequencies $f(a)$, there are at most R paths and the duration of each path including the turn-over time does not exceed K.

Before we consider how simple paths can be constructed, we define the costs of arcs $\tilde{A}$ to correspond to cost.

$$\text{cost}(v_a^i, v_b^j) = \lambda \cdot \text{dist}(a) + \mu \cdot \text{dur}(a),$$

if $v_a^i \neq \tilde{v}_0 \neq v_b^j$,

$$\text{cost}(\tilde{v}_0, v_b^j) = \lambda \cdot \text{dist}(\text{dep}, v) + \mu \cdot (\text{dur}(\text{dep}, v) + \text{dur}(\text{dep})) + \kappa \cdot \left\lceil \frac{K}{T} \right\rceil,$$

if $\alpha(b) = v$ and

$$\text{cost}(v_a^j, \tilde{v}_0) = \lambda \cdot \text{dist}(w, \text{dep}) + \mu \cdot \text{dur}(w, \text{dep}),$$

if $\omega(a) = w$.

4.2 MIP Formulation for Simple Lines

We formulate the mixed-integer program as a modified capacitated vehicle routing problem in (1)-(13).

The first part up to constraints (7) is equal to the formulation of the capacitated vehicle routing problem as in [18].

The variable $x_{\tilde{a}}$ indicates if the corresponding arc $\tilde{a} \in \tilde{A}$ is used and $u_{\tilde{v}}$ describes the summed up demands of the nodes on the corresponding tour starting at $\tilde{v}_0$ up to $\tilde{v} \in \tilde{V}$. Constraints (6) - (7) ensure that the capacity C is not exceeded. Constraints (4)-(5) ensure that no more than R vehicles are used.

By construction, we can have multiple nodes in the transformed vehicle routing graph corresponding to the same arc in the public transport network. As a consequence, it is possible to obtain non-simple paths after translating the vehicle routing solution back to a line concept, i.e., multiple identical arcs on the same line.

Therefore we adapt the previously mentioned capacity constraints to ensure that all lines are simple paths.

$$\min \sum_{\tilde{a} \in \tilde{A}} x_{\tilde{a}} \cdot \text{cost}(\tilde{a}) \tag{1}$$

$$s.t. \sum_{\tilde{a} \in \delta^-(\tilde{v})} x_{\tilde{a}} = 1 \qquad \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \tag{2}$$

$$\sum_{\tilde{a} \in \delta^+(\tilde{v})} x_{\tilde{a}} = 1 \qquad \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \tag{3}$$

$$\sum_{\tilde{a} \in \delta^-(\tilde{v}_0)} x_{\tilde{a}} \leq R \tag{4}$$

$$\sum_{\tilde{a} \in \delta^+(\tilde{v}_0)} x_{\tilde{a}} \leq R \tag{5}$$

$$d(\tilde{v}) \leq u_{\tilde{v}} \leq C \qquad \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \tag{6}$$

$$u_{\tilde{v}} + d(\tilde{w}) - u_{\tilde{w}} \leq (1 - x_{(\tilde{v}, \tilde{w})}) \cdot (C + d(\tilde{v})) \qquad (\tilde{v}, \tilde{w}) \in \tilde{A} \tag{7}$$

$$u_{p, \tilde{v}}^a = 1 \qquad p \in A, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\}, p = \tilde{v}' \tag{8}$$

$$u_{p, \tilde{v}}^a - u_{p, \tilde{w}}^a \leq 1 - x_{(\tilde{v}, \tilde{w})} \qquad p \in A, (\tilde{v}, \tilde{w}) \in \tilde{A}, \tilde{v} \neq \tilde{v}_0 \neq \tilde{w} \tag{9}$$

$$u_{\tilde{w}', \tilde{v}}^a \leq 1 - x_{(\tilde{v}, \tilde{w})} \qquad (\tilde{v}, \tilde{w}) \in \tilde{A}, \tilde{v} \neq \tilde{v}_0 \neq \tilde{w} \tag{10}$$

$$x_{\tilde{a}} \in \{0, 1\} \qquad \tilde{a} \in \tilde{A} \tag{11}$$

$$u_{\tilde{v}} \geq 0 \qquad \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \tag{12}$$

$$0 \leq u_{p, \tilde{v}}^a \leq 1 \qquad p \in A, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \tag{13}$$

We introduce the variable $u_{p, \tilde{v}}^a$ that indicates if the tour that contains node $\tilde{v} \in \tilde{V}$ also contains a node $\tilde{w} \in \tilde{V}$ with $\tilde{w}' = p \in A$, i.e., the arc $p \in A$ of the public transport network is already covered by the vehicle routing tour containing node $\tilde{v}$. Obviously, $u_{p, \tilde{v}}^a = 1$ if $p = \tilde{v}'$ (see constraints (8)).

If we use arc $(\tilde{v}, \tilde{w})$, i.e., $x_{(\tilde{v}, \tilde{w})} = 1$, we copy the value of $u_{p, \tilde{v}}^a$ to $u_{p, \tilde{w}}^a$ for all $p \in A$ in (9). Additionally, this tour may not have covered the public transport network arc $\tilde{w}' \in A$ before node $\tilde{w} \in \tilde{V}$, i.e., $u_{\tilde{w}', \tilde{v}}^a = 0$ and is ensured in constraints (10).

4.3 MIP Formulation for Elementary Lines

In a similar way, we can ensure that we only obtain elementary lines after translating the vehicle routing solution back to a line concept, i.e., we get lines with no repeating nodes.

Therefore, we use nearly the same constraints as (8) - (10) to exclude repeating source nodes (see constraints (14) - (17)) and target nodes (see constraints (18) - (21)) in the lines translated back from the vehicle routing tours.

Again, we have the variable $u_{p, \tilde{v}}^\alpha$ that indicates if the tour that contains node $\tilde{v} \in \tilde{V}$ also contains a node $\tilde{w} \in \tilde{V}$ with $\alpha(\tilde{w}') = p \in V$, i.e., the node $p \in V$ of the public transport network is already covered (as source node) by the vehicle routing tour containing node $\tilde{v}$. The same applies for the variables $u_{p, \tilde{v}}^\omega$ and the target nodes. By this construction, it is possible that a line begins and ends at the same node.

Note that if we ensure elementary lines, we do not have to ensure simple lines.

$$u^{\alpha}_{p,\tilde{v}} = 1 \qquad p \in V, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\}, p = \alpha(\tilde{v}') \quad (14)$$

$$u^{\alpha}_{p,\tilde{v}} - u^{\alpha}_{p,\tilde{w}} \leq 1 - x_{(\tilde{v},\tilde{w})} \qquad p \in V, (\tilde{v},\tilde{w}) \in \tilde{A}, \tilde{v} \neq \tilde{v}_0 \neq \tilde{w} \quad (15)$$

$$u^{\alpha}_{\alpha(\tilde{w}'),\tilde{v}} \leq 1 - x_{(\tilde{v},\tilde{w})} \qquad (\tilde{v},\tilde{w}) \in \tilde{A}, \tilde{v} \neq \tilde{v}_0 \neq \tilde{w} \quad (16)$$

$$0 \leq u^{\alpha}_{p,\tilde{v}} \leq 1 \qquad p \in V, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \quad (17)$$

$$u^{\omega}_{p,\tilde{v}} = 1 \qquad p \in V, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\}, p = \omega(\tilde{v}') \quad (18)$$

$$u^{\omega}_{p,\tilde{v}} - u^{\omega}_{p,\tilde{w}} \leq 1 - x_{(\tilde{v},\tilde{w})} \qquad p \in V, (\tilde{v},\tilde{w}) \in \tilde{A}, \tilde{v} \neq \tilde{v}_0 \neq \tilde{w} \quad (19)$$

$$u^{\omega}_{\omega(\tilde{w}'),\tilde{v}} \leq 1 - x_{(\tilde{v},\tilde{w})} \qquad (\tilde{v},\tilde{w}) \in \tilde{A}, \tilde{w} \neq \tilde{v}_0 \neq \tilde{w} \quad (20)$$

$$0 \leq u^{\omega}_{p,\tilde{v}} \leq 1 \qquad p \in V, \tilde{v} \in \tilde{V} \setminus \{\tilde{v}_0\} \quad (21)$$

4.4 Heuristic Solution Approach

In addition to solving the MIP directly, we can use modifications of know heuristics for the capacitated vehicle routing problem to solve (rLVP). In particular, we tested a modification of the savings algorithm by Clarke and Wright [3].

The algorithm is initialized with $|\tilde{V} - 1|$ tours $(\tilde{v}_0, \tilde{v}, \tilde{v}_0)$ for all $\tilde{v} \in \tilde{V}$. After that, the saving

$$s(\tilde{v}_i, \tilde{v}_j) = \text{cost}(\tilde{v}_i, \tilde{v}_0) + \text{cost}(\tilde{v}_0, \tilde{v}_j) - \text{cost}(\tilde{v}_i, \tilde{v}_j)$$

is calculated for all $\tilde{v}_i, \tilde{v}_j \in \tilde{V}$ with $\tilde{v}_i \neq \tilde{v}_0 \neq \tilde{v}_j$ and sorted in non-increasing order.
Now, the first unused saving $s(\tilde{v}_i, \tilde{v}_j)$ is taken and the tours T_i, T_j corresponding to node $\tilde{v}_i$ and $\tilde{v}_j$ are merged by removing arcs $(\tilde{v}_i, \tilde{v}_0), (\tilde{v}_0, \tilde{v}_j)$ and adding $(\tilde{v}_i, \tilde{v}_j)$, if the following conditions are fulfilled:

- $T_i \neq T_j$
- there exists arc $(\tilde{v}_i, \tilde{v}_0)$ in T_i and $(\tilde{v}_0, \tilde{v}_j)$ in T_j
- the summed up demand of both tours T_i and T_j does not exceed the capacity C.

Subsequently, the next unused saving is taken and this step is repeated until R tours are left.

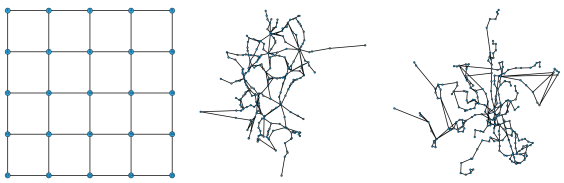
As in the MIP, by adding further conditions we can ensure simple and elementary lines, respectively. To guarantee the former, we only merge two tours T_i, T_j, if the corresponding public transport network arcs of the T_i are not the same as those in T_j.

In a similar way we proceed with the target and source nodes of the corresponding public transport network arcs of tour T_i and T_j to ensure elementary lines.

In some cases, the algorithm may terminate even though there are still more than R tours. This is due to the significantly increased number of conditions for merging two tours.

5 EXPERIMENTAL EVALUATION

We test the two solution approaches for (rLVP) on three data sets from the open source software framework LinTim [13], `grid`, `long-distance` and `goettingen`, see Figure 2, and compare them to the traditional sequential solution approach. The data sets represent an artificial benchmark instance, the long-distance train network in Germany and the bus network in Göttingen, respectively.



(a) `grid`. (b) `long-distance`. (c) `goettingen`.

Figure 2: Public transport networks for various data sets.

Table 1: Instance size and mean solver time (in seconds) and gap of the MIP formulation as well as the mean run time the heuristic approach in seconds. Note that for the MIP solution approach a time limit of 60 minutes is applied.

	PTN		Heu.	MIP	
Data Set	$\|V\|$	$\|A\|$	Time	Time	Gap
`grid`	25	80	0	3600	100%
`long-distance`	250	652	2	3600	99%
`goettingen`	257	548	12	3600	66%

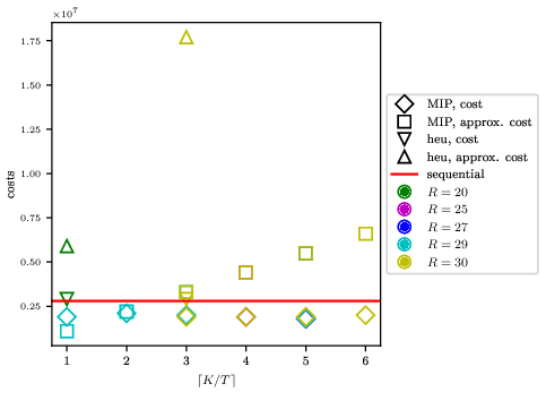
For each data set, we evaluate the approximated costs $\bar{\text{cost}}(\mathcal{R})$ and the actual costs $\text{cost}(\mathcal{R}, \pi)$ for various settings of K and R for the MIP formulation and the heuristic solution approach of (rLVP) for simple lines. For the MIP formulation, we use Gurobi 8.1.1 [1] and report the best solution found within a time limit of 60 minutes. These solutions are compared to the traditional approach of sequentially finding a line plan for a given pool, a timetable and a vehicle schedule.

Note that the runtime for the heuristic is considerably smaller than for the MIP-formulation, especially on the largest data set `goettingen`. The mean runtimes for all settings of K and R are reported in Table 1.

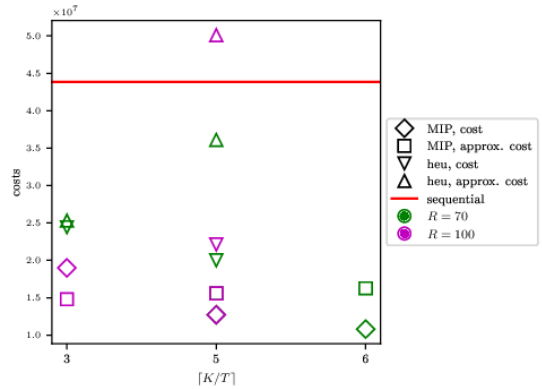
The results are depicted in Figure 3. We make the following observations:

- By using an integrated approach to line planning and vehicle scheduling, it is possible to find solutions that are much cheaper than by using the sequential solution approach, even when the duration of a vehicle route is restricted. For data sets `goettingen`, `grid` and `long-distance`, the costs can be reduced by up to 18%, 36% and 75%, respectively.
- For smaller K, i.e., for shorter vehicle routes, more routes R have to be allowed to find feasible solutions.
- With the heuristic approach, it is not possible to find feasible solutions for all given combinations of K and R.
- For larger K, the error of using $\bar{\text{cost}}(\mathcal{R})$ instead of $\text{cost}(\mathcal{R}, \pi)$ increases.
- While the costs $\text{cost}(\mathcal{R}, \pi)$ do increase slightly for smaller, i.e., more restrictive K, the difference is much smaller than suggest by $\bar{\text{cost}}(\mathcal{R})$.

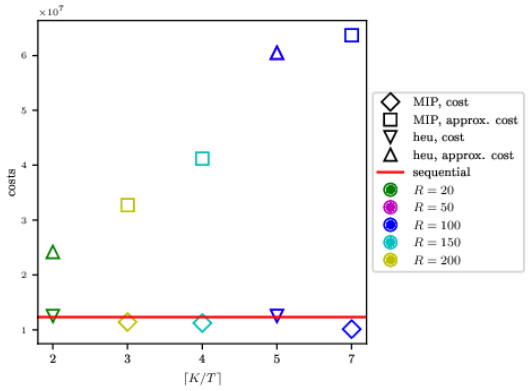
For data set `grid`, we additionally compared using simple paths for the vehicle route to restricting the routes to elementary paths. Here, the costs of the heuristic solutions increased by about 8%



(a) `grid`.



(b) `long-distance`.



(c) `goettingen`.

Figure 3: Costs for the different solution approaches for various data sets.

compared to the simple paths. Additionally, there are more infeasible combinations of K and R when the routes are restricted further.

However, for some applications it might be necessary to restrict the set of lines to elementary instead of simple lines.

6 CONCLUSION AND OUTLOOK

In this paper, we modeled the integrated line planning and vehicle scheduling problem and proposed a solution approach for fixed arc frequencies. While the heuristic solution approach is especially fast and therefore can be used for larger data sets, the MIP-based solution approach outperforms the classical sequential solution approach even when restricting the duration of vehicle routes.

To better incorporate the passengers' perspective, it is also possible to use (rLVP) to generate a *line pool* instead of a line concept by choosing higher values for $f(a)$, $a \in A$. This allows for creating a larger set of potential lines which can be operated within the duration restriction K from which a line concept can be chosen separately. Evaluating these line pools in comparison to the approach from [4] and in combination with other integrated solution approaches such as integrated line planning, timetabling and passenger routing may lead to interesting new solution approaches.

REFERENCES

[1] 2019. Gurobi Optimizer. http://www.gurobi.com/. Gurobi Optimizer Version 8.1.1, Houston, Texas: Gurobi Optimization, Inc.
[2] S. Bunte and N. Kliewer. 2009. An overview on vehicle scheduling models. *Public Transport* 1, 4 (2009), 299–317.
[3] G. Clarke and J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research* 12, 4 (1964), 568–581.
[4] P. Gattermann, J. Harbering, and A. Schöbel. 2017. Line pool generation. *Public Transport* 9, 1-2 (2017), 7–32.
[5] V. Guihaire and J. Hao. 2008. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice* 42, 10 (2008), 1251–1273.
[6] D. Huisman, L. Kroon, R. Lentink, and M. Vromans. 2005. Operations research in passenger railway transportation. *Statistica Neerlandica* 59, 4 (2005), 467–497.
[7] K. Kepaptsoglou and M. Karlaftis. 2009. Transit route network design problem. *Journal of transportation engineering* 135, 8 (2009), 491–505.
[8] R. N. van Lieshout. 2021. Integrated Periodic Timetabling and Vehicle Circulation Scheduling. *Transportation Science* 55, 3 (2021), 768–790.
[9] R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. 2011. Railway track allocation: models and methods. *OR spectrum* 33, 4 (2011), 843–883.
[10] M. Michaelis and A. Schöbel. 2009. Integrating Line Planning, Timetabling, and Vehicle Scheduling: A customer-oriented approach. *Public Transport* 1, 3 (2009), 211–232.
[11] J. Pätzold, A. Schiewe, P. Schiewe, and A. Schöbel. 2017. Look-Ahead Approaches for Integrated Planning in Public Transportation. In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017) (OpenAccess Series in Informatics (OASIcs))*, Vol. 59. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 17:1–17:16.
[12] J. Pätzold, A. Schiewe, and A. Schöbel. 2018. Cost-Minimal Public Transport Planning. In *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018) (OpenAccess Series in Informatics (OASIcs))*, Vol. 65. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 8:1–8:22.
[13] A. Schiewe, S. Albert, P. Schiewe, A. Schöbel, and F. Spühler. 2020. *LinTim: An integrated environment for mathematical public transport optimization. Documentation for version 2020.12.* Technical Report. TU Kaiserslautern. https://nbn-resolving.org/urn:nbn:de:hbz:386-kluedo-62025
[14] P. Schiewe. 2020. *Integrated Optimization in Public Transport Planning.* Optimization and Its Applications, Vol. 160. Springer. https://doi.org/10.1007/978-3-030-46270-3
[15] A. Schöbel. 2012. Line planning in public transportation: models and methods. *OR spectrum* 34, 3 (2012), 491–510.
[16] A. Schöbel. 2017. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies* 74 (2017), 348–365.
[17] P. Serafini and W. Ukovich. 1989. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics* 2, 4 (1989), 550–581.
[18] P. Toth and D. Vigo. 2002. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics* 123, 1-3 (2002), 487–512.

The zone tariff design problem with multiple counting

Reena Urban[1] and Anita Schöbel[2]

[1]Department of Mathematics, TU Kaiserslautern, Kaiserslautern, Germany, ✉ urban@mathematik.uni-kl.de
[2]Department of Mathematics, TU Kaiserslautern, and Fraunhofer-Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany, ✉ schoebel@mathematik.uni-kl.de

Public transport planning includes several planning stages, one of which is fare planning. Fare planning (or tariff design) deals with the question which fare strategy to choose (e.g., a distance- or zone-based one with or without particularities) and how to configure it. A fare strategy together with specific prices is called a fare structure. Tariff design concerns not only the financial requirements of the public transport operator, but also the passenger satisfaction. A fare structure needs to be comprehensible and should be perceived as fair [2]. Our focus lies on the design of zone tariffs with a counting zones pricing because this fare strategy is very popular in practice [1,2]. We formulate and analyze multiple problem variations of the zone tariff design problem.

For fare planning, we assume that a public transport network (PTN) is given. This is an undirected, connected graph with stations V and edges E between the stations along which traveling is possible. A fare structure then assigns a price to each path in the PTN. In a zone tariff with counting zones pricing, the stations in the PTN are partitioned into zones, and the price of a path depends on the number of zones that are traversed on this path.

We distinguish four versions of the zone tariff design problem. First, it has to be specified how to count zones. *Multiple counting* means that a zone is counted each time that it is traversed, whereas *single counting* means that the number of different zones that are traversed is counted. Secondly, we distinguish between two types of zonings, namely *connected* and *arbitrary* zonings, i.e., whether the subgraph induced by the stations of a zone needs to be connected or not.

There are many possible objectives like maximizing the demand, the user benefit or the income of the public transport operator. We consider an objective function based on reference prices. For each origin-destination (OD) pair (v, w) in the PTN, a reference price p_{vw} is given, which can be the price of a former fare structure when the fare strategy is changed or it can be a price that is considered as fair or otherwise preferable. The goal is to determine new prices π_{vw} that fit as good as possible to the reference prices, i.e., we minimize the sum of absolute deviations $\sum_{(v,w)} C_{vw}|\pi_{vw} - p_{vw}|$, where C_{vw} denotes the number of passengers traveling from v to w. The objective function ensures that the new prices do not deviate too much in any direction, meaning that the income of the operator does not decrease too much and that the prices for the passengers do not increase tremendously compared to the former fare structure.

The zone tariff design problem which we consider thus is the following: Given a PTN (V, E), the number of passengers C_{vw} which travel between stations v and w on a fixed path W_{vw} and reference prices p_{vw}, determine a zoning, i.e., a partition of the stations into zones, and a price list such that the sum of absolute deviations from the reference prices is minimized in the resulting zone tariff. Considering the distinctions mentioned above, we receive the following four versions of the zone tariff design problem:

	multiple counting	single counting
arbitrary zoning	ZD-MA	ZD-SA [2]
connected zoning	ZD-MC	ZD-SC [2]

We show the following results about the four problems:

- For all four problems it holds: The version with an upper bound on the number of allowed zones is a relaxation of the version with a fixed number of zones.

- ZD-MA is a relaxation of ZD-MC, and ZD-SA is a relaxation of ZD-SC, i.e., the version with arbitrary zoning is a relaxation of the version with connected zoning.
- If the number of zones is bounded, the optimal objective value for ZD-MA can be strictly smaller than for ZD-MC. The same holds for ZD-SA and ZD-SC.
- If the number of zones is unbounded, then ZD-MA and ZD-MC yield the same optimal objective value. Even if the number of zones is unbounded, the optimal objective value for ZD-SA can be strictly smaller than for ZD-SC.

Based on the mixed-integer programming (MIP) formulation by [2], we provide a first MIP formulation for ZD-MC and ZD-MA. As parameters it needs the PTN (V, E), the set of OD-pairs $D \subseteq V \times V$ with the numbers of passengers C_{vw} and fixed paths W_{vw} for $(v, w) \in D$, a maximum number of allowed zones N, the maximum number of stations on a path K and a sufficiently large M. The MIP formulation is as follows:

$$\begin{aligned}
\min \quad & \textstyle\sum_{(v,w)\in D} C_{vw}|\pi_{vw} - p_{vw}| \\
\text{s.t.} \quad & \sum_{z=1}^{N} x_{vz} = 1 && v \in V, && (1)\\
& \{\text{Connectivity constraints from [2]}\} && && (2)\\
& x_{vz} - x_{wz} \leq c_{vw} && \{v,w\} \in E,\ z \in \{1,\dots,N\}, && (3)\\
& c_{vw} \leq 2 - x_{vz} - x_{wz} && \{v,w\} \in E,\ z \in \{1,\dots,N\}, && (4)\\
& \sum_{k=1}^{K} d^k_{vw} = 1 && (v,w) \in D, && (5)\\
& 1 + \sum_{e\in W_{vw}} c_e = \sum_{k=1}^{K} k \cdot d^k_{vw} && (v,w) \in D, && (6)\\
& \pi_{vw} \leq p_k + (1 - d^k_{vw}) \cdot M && (v,w) \in D,\ k \in \{1,\dots,K\}, && (7)\\
& p_k \leq \pi_{vw} + (1 - d^k_{vw}) \cdot M && (v,w) \in D,\ k \in \{1,\dots,K\}, && (8)\\
& x_{vz} \in \{0,1\} && v \in V,\ z \in \{1,\dots,N\}, && (9)\\
& c_e,\ d^k_{vw} \in \{0,1\} && e \in E,\ (v,w) \in D,\ k \in \{1,\dots,K\}, && (10)\\
& p_k,\ \pi_{vw} \in \mathbb{R}_{\geq 0} && k \in \{1,\dots,K\},\ (v,w) \in D. && (11)
\end{aligned}$$

This is a MIP formulation of ZD-MC when including the connectivity constraints (2), and a MIP formulation of ZD-MA when dropping (2). Each station is assigned to exactly one zone (1). If Constraints (2) are included, the zoning is connected. In order to compute the number of zones traversed on a path, it needs to be determined whether an edge $\{v, w\}$ crosses a zone border, i.e., whether v and w are in different zones (3,4). Then the number of traversed zones on the path W_{vw} is determined for each OD-pair (v, w). Finally, the new price π_{vw} for the OD-pair (v, w) is set to the price p_k of k zones if and only the v-w-path traverses k zones (7,8). Further constraints for the no-stopover and no-elongation property, introduced in [3], can be included.

References

[1] Horst W. Hamacher and Anita Schöbel. Design of Zone Tariff Systems in Public Transportation. *Operations Research*, 52(6):897–908, December 2004.

[2] Benjamin Otto and Nils Boysen. Zone-based tariff design in public transportation networks. *Networks*, 69(4):349–366, 2017.

[3] Anita Schöbel and Reena Urban. Cheapest Paths in Public Transport: Properties and Algorithms. In Dennis Huisman and Christos D. Zaroliagis, editors, *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*, volume 85 of *OpenAccess Series in Informatics (OASIcs)*, pages 13:1–13:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Designing Railway Networks based on Timetables

Tim Sander[1]

[1]Prof. für Verkehrsströmungslehre, TU Dresden, Dresden, Germany, ✉ tim.sander@tu-dresden.de

During the last decades, many middle-European railway companies changed the philosophy of their long-term infrastructure design ([8], [3]). Instead of new railway lines and extensions of existing tracks based on demand estimations, the development is based on a long-term timetable. The Swiss federal railways SBB were the first who designed such a nationwide, long-term timetable within their Bahn-2000 development program. All further infrastructure expansions were afterwards planned according to the needs of the timetable. The goal was to connect cities as fast as necessary, not as fast as possible. The immense success of the new timetable prompted other countries in Europe to follow the swiss example. In Germany, the third and final draft for a nationwide timetable concept called the Deutschlandtakt has been published in 2020 [6].

With this shift of perspective, the traditional planning process of railway and public transport operations has been fundamentally changed. The old process contains three different planning levels (from long-term to short-term planning): the strategical, the tactical and the operational level. The strategic level includes network design and line planning, followed by the timetabling process at the tactical level. The operational level comprises both vehicle and crew scheduling as well as crew rostering. This process is very well supported by different optimization models. Many researchers and companies have also applied various levels of integration, most often by combining crew and vehicle scheduling or timetabling and vehicle scheduling. However, there exists a gap in the literature regarding models that deal with the recent approach of timetable-based network design.

Many approaches to network design use an adaptation of the network-design problem, made popular by [4]. It has been used for a variety of different applications, including transportation planning in general and railway network design in particular, for example in [7], where the network design problem for railway infrastructure is introduced. It is used to design cost-optimal but timetable-independent networks based on a complete multigraph, where each arc includes different stages of capacity extensions. There has also been an interest in freight-focussed models which combine Network Design and capacity extensions, most recently in [5]. However, these models also lack the consideration of an input timetable.

When considering papers that deal with network design and timing or timetabling, one comes across time-expanded networks, which are used in a variety of papers. An overview is given in [1] which focuses on service network design problems. Apart from using time-expanded networks, there have been approaches to continuous-time modelling by specifying time points not as nodes in the network, but as variables [2].

This abstract presents an expansion of the very well-known Fixed Charge Network Design Problem. Based on an input timetable which consists of a set of trains, each having an origin node, a destination node, and fixed time bounds, a cost-optimal railway network is designed. The railway infrastructure is modelled in a macroscopical way, where the nodes in the graph symbolize stations or interlockings while the arcs represent train line tracks. The underlying graph is defined as a MultiGraph with bidirectional arcs, so multiple arcs can be constructed between two nodes. In this way, lines with one or more tracks can be modelled. By assigning train-type-dependant travel times to each arc and using minimal headway times, the capacity on the railway lines is realistically estimated. The minimal headway times depend on both the running order and the direction of trains.

The optimization model has been split into two steps to assure a decent performance. The first one, which focuses on the network design, chooses a set of arcs, which allow each train to find a feasible path while respecting the time bounds set by the timetable. Both building costs and travel times are minimized by the objective function. This first stage uses a worst-case capacity measure which assumes that the longest headway time occurring anywhere in the data has to be respected between all trains. This underestimation of the capacity is used to ensure that the solution contains enough parallel tracks

to accommodate all trains, which in turn leaves enough flexibility for the second stage. The network resulting in the first stage can be interpreted as an upper bound for the second step of the optimization.

Based on the result of the first step, the second stage further refines the network while focussing on timetabling. Therefore, continuous-time variables are introduced for every train and every line it is running on. The routing of the first step is fixed in a way that a train can still switch between parallel tracks of the same line, but re-routing a train via different lines is not possible anymore. The use of variables for both departure and arrival times allows to respect minimal headway times and to estimate the capacity in a much more precise way than the worst-case capacity measure of the first optimization step. This combined with the possibility to reroute trains between parallel tracks makes it possible to further reduce the number of tracks needed in the network. The result is a cost-minimal network and a macroscopical timetable.

The model is implemented in python and solved with Gurobi 9.1.2. Test instances have been generated from the publicly available timetable drafts of the Deutschlandtakt. For the largest test case, which comprises 92 trains in a network with 110 nodes and 149 possible train lines, an optimal solution for the first stage can be calculated in about 11 minutes. The second stage needs significantly more time to solve, after 1 hour the optimality gap could be reduced to 3,7 %.

Further improvements and extensions to the model will include the consideration of uncertainty in the input timetable, the usage of decomposition or heuristic solution approaches to reduce calculation times and the inclusion of different railway specific features such as buffer times, connections between several trains and basic station capacity measures.

References

[1] Natashia Boland et al. “The price of discretizing time: a study in service network design”. In: *EURO Journal on Transportation and Logistics* 8.2 (2019), pp. 195–216. ISSN: 21924376. DOI: 10.1007/s13676-018-0119-x.

[2] Amin Hosseininasab. “The Continuous Service Network Design Problem”. Master Thesis. Waterloo, Ontario, Canada: University of Waterloo, 2015.

[3] Christian Kräuchi et al. *Mehr Zug für die Schweiz die Bahn-2000-Story*. Zürich: AS-Verl., 2004. ISBN: 9783909111060. URL: http://slubdd.de/katalog?TN_libero_mab213735277.

[4] T. L. Magnanti and R. T. Wong. “Network Design and Transportation Planning: Models and Algorithms”. In: *Transportation Science* 18.1 (1984), pp. 1–55. ISSN: 0041-1655. DOI: 10.1287/trsc.18.1.1.

[5] Francisca Rosell and Esteve Codina. “A model that assesses proposals for infrastructure improvement and capacity expansion on a mixed railway network”. In: *Transportation Research Procedia* 47 (2020), pp. 441–448. ISSN: 23521465. DOI: 10.1016/j.trpro.2020.03.119.

[6] SMA und Partner AG. *Software and consultancy support for Capacity Simulation and Optimisation of Commuter Train System*. June 30, 2020. URL: https://assets.ctfassets.net/scbs508bajse/aY44eEzQg6MJ5JvOL3Dxx/f1430927d606cd8719982b8ae87bf3f4/Dokumentation_zum_3._Gutachterentwurf_des_Zielfahrplans_Deutschlandtakt.pdf (visited on 11/18/2021).

[7] Jacob Christian Spönemann. “Network design for railway infrastructure by means of linear programming”. PhD thesis. Aachen: Hochschulbibliothek der Rheinisch-Westfälischen Technischen Hochschule Aachen, 2013.

[8] Werner Weigand. “Langfristfahrplan und Kapazitätsuntersuchungen”. In: *Deine Bahn* 5 (2012), pp. 28–31.

Tariff Zone Planning for Public Transport Companies

Sven Müller[1], Knut Haase[2], and Lorena Reyes-Rubiano[1]

[1]Chair of Operations Management, Otto von Guericke University Magdeburg, Magdeburg, Germany, ✉ {sven.mueller;lreyes}@ovgu.de
[2]Institute for Transport Economics, University of Hamburg, Hamburg, Germany, ✉ knut.haase@uni-hamburg.de

Public transport companies are under pressure to reduce operational costs and increase the number of passengers and revenue. The design of the tariff system has been proven valuable to impact revenues in public transportation. There exist different tariff systems in public transport [3]. The most widely accepted public transport tariff system is the counting zones tariff system. The literature on combining zoning and fare problems is scarce. The study of Hamacher and Schöbel [1] and Otto and Boysen [2] are two examples of the literature that consider a counting zones tariff system. Otto and Boysen can optimally solve problems with up to 5 zones, 50 customers (demand points), and 10 stops. While Hamacher and Schöbel use heuristics to solve instances up to 400 stops. We contribute to this literature by a new approach yielding a counting zones tariff system that maximizes the revenue (R). This paper presents a new mixed-integer programming (MIP) model and a MIP-based heuristic method. Our approach allows specific spatial patterns of the tariff zones (rings and stripes, for example).

The revenue maximizing tariff zone problem (RMTZP) consists of partitioning the service area of the public transport service provider into zones and to find a price per zone (price system) such that R is maximized. The proposed methods can optimally solve instances of sizes of more than 120 stops. Moreover, the methods are flexible enough to enforce the counting zones tariff system to any spatial pattern. The results show that enforcing tariff zones to a specif spatial pattern reduces R and CPU time.

The contributions of this paper are:

- Design a counting zones tariff system that maximizes R.
- Formal description of the price and districting (zoning) problem.
- Development of a MIP-based heuristic method.
- Inclusion of constraints to enforce a desired spatial pattern.

1 Modeling

Our approach relies on the assumptions (i) that passengers always choose the time-shortest path and (ii) that feasible values for the price per zone are denumerable. We have developed a MIP model **P1** to solve RMTZP. One innovative feature of our modeling approach is that we employ dual graph theory to ensure zone contiguity and desired spatial patterns. Our MIP model maximizes R in the service area for a given price system $p \in \mathcal{P}$ ("price per zone"). Following, we summarize the variables and main constraints of RMTZP:

Decision variables F

X_{ijt} = 1, if $t = 1, \ldots, T_{ij}$ tariff zones are visited on the shortest path from $i \in \mathcal{I}$ to $j \in \mathcal{I}$ (0, otherwise), with T_{ij} as the maximum number of tariff zones visited along the shortest path from i to j

Y_{nm} = 1, if a tariff zone border is established along border arc $(n, m) \in \mathcal{B}$ (0, otherwise)

W_{nm} = artificial flow along the border arc $(n, m) \in \mathcal{B}$

Constraints

- The number of tariff zones visited from i to j is equal to 1 plus the number of tariff zone borders visited along the shortest path from node i to j.

- Flow constraints that indicate flow out minus flow in along a border arc $(n, m) \in \mathcal{B}$ must equal to b_n at border node n. Values of b_n depend on the desired spatial pattern of the tariff zone. The flow conservation constraints ensure contiguous tariff zones.

Relaxation of RMTZP: The size of the RMTZP is mainly influenced by the number of O-D tuples. For this reason, we consider a subset, $\mathcal{C}$ of all O-D tuples. Set $\mathcal{C}$ contains $\gamma \cdot \mid \mathcal{I} \mid$ of O-D tuples with highest R. We propose a MIP-based heuristic to find $\mathcal{C}$. Then, the model **P1** is solved by considering O-D tuples i-$j \in \mathcal{C}$ instead of all $i \in \mathcal{I}$ and $j \in \mathcal{I}$.

Fare problem: The zone problem **P1** depends on a given price system $p \in \mathcal{P}$. The trip price under a counting zone tariff system depends on the visited zones along the trip and the price per zone. Let $r_{ijt}(\pi_{p_t})$ represent R on the shortest path from $i \in \mathcal{I}$ to $j \in \mathcal{I}$ given price system $p \in \mathcal{P}$, with π_{p_t} as the price per zone when visiting t zones under price system $\mathcal{P}$. The total R in **P1** is given by:

$$\textbf{Maximize } R(p) \;=\; \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \sum_{t=1}^{T_{ij}} r_{ijt}(\pi_{p_t}) \cdot X_{ijt} \tag{1}$$

2 Computational experiments

We generate a set of artificial instances representing a city. Public transport travel-time and hence demand and R follow a random distribution. We solve 12 problem sets with 10 instances each to optimality using GAMS/CPLEX and compare the MIP model against the MIP-based heuristic, considering the constraints to enforce tariff zones to have a ring pattern. For instance, on average, our heuristic underestimates optimal R by 2.54% but is faster by 42.97%. Results demonstrate that solutions with a lower R are obtained when constraints are included to enforce a specific spatial pattern than when there is no specific pattern. Solutions without any spatial pattern have a R of 0.43% higher than solutions with a ring pattern, for example. The CPU time for the solutions with no spatial pattern is 85.12% higher than solutions with ring patterns.

3 Conclusion

We investigate how to design a counting zones tariff system to maximize the revenues of public transportation service companies. We design a MIP model and MIP-based heuristic method to design an optimal counting zones tariff system and solve the fare problem. Our approach basis on dual and primal graph properties to deal with contiguity between tariff zones and enforce the tariff system to a desired spatial pattern. The proposed approaches can optimally solve instances up to $\mid \mathcal{I} \mid$=120 in a reasonable time. Moreover, the methods are flexible enough to enforce the counting zones tariff system to various spatial patterns. The results show that enforcing tariff zones to a desired spatial pattern reduces R and CPU time. Currently, we are working on testing our MIP-based heuristic approach with real data from the San Francisco Bay Area with $\mid \mathcal{I} \mid$=1,415 districts. We are particularly interested in studying how to enforce tariff zones to follow a desired pattern. In addition, we plan to analyze a price system with discounts.

References

[1] H.W. Hamacher and A. Schöbel. Design of zone tariff systems in public transportation. *Operations Research*, 52(6):897–908, 2004.

[2] Benjamin Otto and Nils Boysen. Zone-based tariff design in public transportation networks. *Networks*, 69(4):349–366, 2017.

[3] Anita Schöbel and Reena Urban. Cheapest paths in public transport: Properties and algorithms. In *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Session Session 6B: communication networks 2

Friday 10 June 2022, 09:30-11:10

Lecture Hall III

The roll-out of new mobile technologies as a timing game

Paolo Zappalà[1,2], Amal Benhamiche[1], Matthieu Chardy[1], Francesco De-Pellegrini[2], and Rosa Figueiredo[2]

[1]Orange Innovation, 92320 Chatillon, France, ✉ name.surname@orange.com
[2]LIA, Avignon Université, 84029 Avignon, France, ✉ name.surname@univ-avignon.fr

When adopting a novel mobile technology, a mobile network operator faces the dilemma of determining which is the best time to install the next generation equipment onto the existing infrastructure. In a strategic context, the optimal deployment time is the best response to competitors' actions, subject to normative and material constraints and to the customer's adoption curve. To capture the main features of the problem, we formulate in this paper a timing game in discrete time for a two-player setting. Under mild assumptions on the time scale at which operators decide on the installation, we provide a methodology to obtain a subgame-perfect equilibrium for the resulting extensive form game.

The proposed model describes two telecommunication operators, or Service Providers (SP), whose objective is to optimize their deployment strategy as a pair of actions. First, each operator chooses the subsidy to offer to customers from a discrete set S_i of possible subsidy budgets. Afterwards, they schedule when and where to deploy the new technology on their own *sites*. Operators act on a discrete time horizon. The timing game is described by $N = \{1, 2\}$, the set of players, S_i, the set of possible budgets allowed for subsidies by player $i \in N$ (chosen at time $t = 0$), $T = \{1, ..., |T|\}$, i.e., the set of time-intervals over which operators act to install the new technology and, finally, $\mathcal{A}$, the set of sites where to deploy the new technology.

We introduce $z_{i,a,t} \in \{0, 1\}$, a binary variable indicating if the new technology is installed on site a by operator i at time $t \geq 1$. We call $t_{i,a}$ the time at which operator i installs it on site a. The operators' schedule is bounded by some constraints:

- *Logistic constraints:* the operator i can invest on a limited number Z_i of sites at each time $t \geq 1$. Thus for every player it holds $\sum_{a \in \mathcal{A}}(z_{i,a,t+1} - z_{i,a,t}) \leq Z_i$;
- *Regulator constraints:* before every time t at least $R(t)$ sites have to support the new technology. Thus for all players and for all $t \geq 1$ it holds $\sum_{a \in \mathcal{A}} z_{i,a,t} \geq R(t)$.

Furthermore, we assume that players can observe, at time t, the history of actions taken by the other player for $t' < t$, and react accordingly; the choice of the opponent's subsidy is observable by both players at time $t = 0$.

A convenient model that complies to such assumption is that of a *game in extensive form* [2], whose mathematical model is based on a *game tree*. Every combination of chosen strategies $\{(\mathbf{s}_i, \mathbf{z}_i), i = 1, 2\}$ leading to a leaf of such tree is called an *outcome*. The *utility function* evaluates such outcomes for every operator; it increases with their market share and it decreases with the costs incurred for the technology upgrades and subsidies. The higher the value of u_i, the higher the value a player assigns to an outcome.

Definition 1 (Service providers (SP) game)**.** *The service providers game $\langle N, S_1, S_2, \mathcal{A}, T, u\rangle$ is an extensive form game with two players $N = \{1, 2\}$ competing over set of sites $\mathcal{A}$ in which:*

- *at the root vertex, i.e., at $t = 0$, both players choose independently the subsidies $s_1 \in S_1$ and $s_2 \in S_2$;*
- *the players act in sequence at every round $t \geq 1$, starting from player 1. At every step they can decide on which sites $A_{1t} \subseteq \mathcal{A}$ and $A_{2t} \subseteq \mathcal{A}$ install the new technology, given the constraints;*
- *after $|T|$ rounds the game ends and the actions chosen at each round are evaluated for every player i by the utility functions $u_i : S_1 \times S_2 \times (\mathcal{A} \times T)^2 \to \mathbb{R}$.*

The solution i.e., an equilibrium of the game, is determined in two steps. First, at $t = 0$ the operators pick a subsidy at the same time. The second operator does not know what the first operator has played, and vice versa. Then, in the second part (all subtrees rooted at $t = 1$), they get to know what the other operator has chosen and decide one after another if installing on a site or not at each time step, starting by player 1. Let us denote with $\Gamma(s_1, s_2)$ the subgame which starts at $t = 1$, given that the first operator has chosen $s_1 \in S_1$ and the second operator $s_2 \in S_2$. If no player has the same payoffs at two

different outcomes, such game has a unique *subgame perfect equilibrium* [2] $\sigma(s_1, s_2) \in (\mathcal{A} \times T)^2$, i.e., a Nash equilibrium [3] for every subgame (which is represented by a subtree). We define the matrix M which maps a couple of choices for the subsidies to the utility of such outcome $(s_1, s_2) \to \sigma(s_1, s_2) \to M(s_1, s_2) = u(s_1, s_2, \sigma(s_1, s_2))$. Matrix M defines a game, which has at least one Nash equilibrium [3]. Operators pick the optimal combination of strategies within the set of such Nash equilibria.

Definition 2 (SP game equilibrium)**.** *Given a SP game $\langle N, S_i, \mathcal{A}, T, u\rangle$ and its correspondent matrix $M : (s_1, s_2) \mapsto u(s_1, s_2, \sigma(s_1, s_2))$, with (s_1, s_2) chosen at time $t = 0$ and $\sigma(s_1, s_2) \in (\mathcal{A} \times T)^2$ the optimal installation times chosen at times $t \geq 1$, we say $(\overline{s}_1, \overline{s}_2) \in S_1 \times S_2$ is an equilibrium if for all $s_1 \in S_1$ and $s_2 \in S_2$ we have: $M_1(\overline{s}_1, \overline{s}_2) \geq M_1(s_1, \overline{s}_2)$, $M_2(\overline{s}_1, \overline{s}_2) \geq M_2(\overline{s}_1, s_2)$.*

Note that the choice of the order of the players can be arbitrary. Indeed, in reality the players act simultaneously, so there is no natural order to be followed. Moreover, the solution does not change significantly if the order of the players is inverted. Operators play in turns with very small difference of time between one and another turn: postponing an action of one interval does not change the outcome significantly. This intuition is formalised in Proposition 1.

Proposition 1 (Order insensitivity)**.** *Given two games $\Gamma = \{z_{nt}, n = \{1, 2\}\}$ and $\Gamma' = \{z'_{n't}, n' = \{2, 1\}\}$ in which players act in inverted order, if there is for every $t \in T$ and for every subsequence $z_{nt} \in \Gamma$ an action $\overline{z}_{it}$ that can allow a player $i \in N$ to postpone their action without changing significantly their utility (within an interval $\epsilon > 0$):*

$$|u_i(\dots, z_{it}, \dots, z_{i,t+1}, \dots) - u_i(\dots, \overline{z}_{it}, \dots, z_{i,t}, \dots)| < \epsilon,$$

then given two subgame perfect equilibria $\sigma \in \Gamma$, $\sigma' \in \Gamma'$ we have $|u_i(\sigma) - u_i(\sigma')| < \epsilon \ \forall i$.

Hereafter we describe an example of utility function $\mathbf{u}$, which describes the dynamics on a model with only one site $|\mathcal{A}| = 1$. We fix some assumptions: 1) every customer on the site decides to switch to the new technology at a given time t and stick to the choice till the end of the horizon [1]; 2) the quantity of customers switching at every time is known a priori by both operators according to a given dynamics; 3) every customer has a preference over the two operators which they subscribe for; if their preferred operator does not offer the technology at time t they subscribe to the other operator if it offers it, otherwise they wait for one of them to offer it; 4) an operator can admit only a limited amount of customers per time interval; 5) once subscribed, customers remain with the chosen operator till $t = |T|$.

Operators choose a pair of subsidies $(s_1, s_2) \in S_1 \times S_2$; those influence the potential market of customers willing to switch to the new technology, which varies in time. For every $t \in T$, such customers are identified by the parameter $\{y_t\}_{t \in T}$, which is subject to the condition that $\sum_t y_t = 1$, i.e. that all the customers eventually switch to the new technology. In order to capture these customers the operators have to install the new technology. We suppose that operators need $\tau > 0$ intervals of time to fill it. This constraint is due to the fact that customers do not discover the technology all at once, but little by little.

Let $\alpha_1(t)$ and $\alpha_2(t)$ the number of customers acquired at time $t \in T$ with the new technology by operators 1 and 2, respectively. Since the customers, once acquired, are kept until the end of the horizon the final market share is $\alpha(|T|)$. On the other hand, there exist some costs related to the installation of the new technology and to subsidies. We denote the installation costs $c \in (0, 1)$: they are discounted by a factor $e^{-\gamma t}$ which accounts for the depreciation since installation time t and lower maintenance costs over the period. The utilities for the players are thus $u_i(s_1, s_2, t_1, t_2) = \alpha_i(|T|) - c \cdot e^{-\gamma t_i} - s_i$, where c and γ are parameters known a priori. Here, subsidies s_i are fixed costs, thus they do not depend on the quantity of customers acquired. Such utility function satisfies the conditions of Proposition 1.

References

[1] Adrien Cambier, Matthieu Chardy, Rosa Figueiredo, Adam Ouorou, and Michael Poss. Optimizing the investments in mobile networks and subscriber migrations for a telecommunication operator. *Networks*, 77(4):495–519, 2021.

[2] Harold William Kuhn and Albert William Tucker. *Contributions to the Theory of Games*, volume 2. Princeton University Press, 1953.

[3] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

A framework for routing and spectrum assignment in optical networks, driven by combinatorial properties

Pedro H. Fernandes da Silva
LIMOS, Clermont Auvergne Institute of Technology, Clermont-Ferrand and IMT Atlantique, Brest, France
pedrohenrique.phfs@gmail.com

Hervé Kerivin
LIMOS, Clermont Auvergne Institute of Technology Clermont Auvergne University, Clermont-Ferrand, France
herve.kerivin@limos.fr

Juan Pablo Nant
LIMOS, Clermont Auvergne Institute of Technology Clermont Auvergne University, Clermont-Ferrand, France
jpnant@gmail.com

Annegret K. Wagler
LIMOS, Clermont Auvergne Institute of Technology Clermont Auvergne University, Clermont-Ferrand, France
annegret.wagler@limos.fr

ABSTRACT

The routing and spectrum assignment problem is an NP-hard problem that has received increasing attention during the last years. The majority of existing models for the problem uses edge-path formulations where variables are associated with all possible routing paths so that the number of variables grows exponentially with the size of the instance. To bypass this difficulty, precomputed subsets of all possible paths per demand are typically used, which cannot guarantee optimality of the solutions in general. Our contribution is to provide a framework for the use of edge-path formulations to minimize the spectrum width of a solution. For that, we select an appropriate subset of paths to operate on with the help of combinatorial properties in such a way that optimality of the solution can be guaranteed. Computational results indicate that our approach is indeed promising to solve the routing and spectrum assignment problem.

1 INTRODUCTION

Optical networks represent a crucial infrastructure for our information society and use light as a communication medium between sending and receiving nodes. For over two decades, *Wavelength-Division Multiplexing (WDM)* has been the most popular technology used in fiber-optic communications. WDM combines multiple wavelengths to simultaneously transport signals over a single optical fiber, but has to select the wavelengths from a rather coarse fixed grid of frequencies specified by the International Telecommunication Union (ITU) and leads to an inefficient use of spectral resources. In response to the sustained growth of data traffic volumes in communication networks, so-called flexgrid optical networks have been introduced to enhance the spectrum efficiency and enlarge the network capacity. In such networks, the frequency spectrum of an optical fiber is divided into narrow frequency slots and any sequence of consecutive slots can form a channel on optical fibers to create an optical connection, called *lightpath*, and thus enables capacity gain by allocating minimum required bandwidth [8].

The *Routing and Spectrum Assignment (RSA)* problem consists of establishing the lightpaths for a set of traffic demands, given as sending and receiving nodes and frequency slot numbers. Since lightpaths are determined by a route and a channel, the RSA problem involves finding a route and assigning a channel of frequency slots for each demand. To comply with ITU recommendation, the following constraints need to be respected:

- *slot continuity*: the slots remain the same on all the links of a route;
- *slot contiguity*: the slots allocated to a demand must be contiguous;
- *non-overlapping slot*: on each link, a slot can be allocated to at most one demand.

More precisely, we are given an optical network $G = (V, E)$ with edge length l_e for all $e \in E$, an optical spectrum $S = \{1, \ldots, \bar{s}\}$, and a set $\mathcal{D}$ of demands between pairs o_k, d_k of nodes in G specifying the maximum length $\bar{l}_k$ of a route and the number w_k of required slots. The routing selects, for each demand $k = (o_k, d_k, \bar{l}_k, w_k) \in \mathcal{D}$, an (o_k, d_k)-path P_k of length at most $\bar{l}_k$ as route from o_k to d_k through G. The spectrum assignment consists of selecting, for each $k \in \mathcal{D}$, a channel $S_k \subseteq S$ of w_k consecutive frequency slots that satisfies the three above constraints. We denote a routing of $\mathcal{D}$ by $\mathcal{P} = \{P_k : k \in \mathcal{D}\}$, and a spectrum assignment by $\mathcal{S} = \{S_k : k \in \mathcal{D}\}$ so that any pair $(\mathcal{P}, \mathcal{S})$ is a solution to the RSA problem.

In addition, the selected set of lightpaths is supposed to minimize a chosen objective function, e.g. minimize the number of edges in the routing paths P_k [19], minimize the number of edges from the network used to route the demands [17], or minimize the spectrum width (and, thus, the width of the subspectrum of S used for the spectrum assignment) [2].

The RSA problem has been shown to be NP-hard [3, 18]. In fact, if all routes are already known or uniquely determined (e.g. if the optical network is a tree), then the RSA problem reduces to the spectrum assignment and only consists of determining the demand's channels. It is NP-complete to decide whether there is a feasible spectrum assignment within a given optical spectrum, even if the optical network is a path, see e.g. [16]. This makes the RSA problem much harder than the WDM problem which is polynomially solvable on paths, see e.g. [5].

To solve the RSA problem, various approaches have been studied in the literature, based on different Integer Linear Programming (ILP) models. Few models use *edge-node formulations* which are

compact in terms of the number of variables and constraints, see e.g. [2, 17, 19] and [1] for an overview, but have the disadvantage that the routing is rather involved. As noticed in [7], the models from [2, 17, 19] are incomplete as their feasible region is a superset of all feasible solutions to the RSA problem. The first complete edge-node formulation presented in [7] exactly encodes the feasible solutions, but requires an exponential number of constraints.

The majority of the existing models uses an *edge-path formulation* where for each demand, variables are associated with all possible routes for this demand, leading to an exponential number of variables issued from the total number of all feasible paths between origin-destination pairs in the network, which grows exponentially with the size of the network. To bypass the exponential number of variables, edge-path formulations with a restricted precomputed subset of all possible paths per demand have been studied, e.g. in [10, 12, 17, 19], see [19] for an overview. However, such formulations cannot guarantee optimality of the solutions in general (as only a subset of paths is considered and, thus, a restricted problem is solved). In order to find optimal solutions to the RSA problem w.r.t. any objective function with the help of an edge-path formulation, all possible paths have to be taken into account. As the explicit models are far too big for computation, it is in order to apply column-generation methods. However, computational results from [11, 13, 15] show that the size of the instances that can be solved that way is rather limited[1].

Our goal is to compute the minimum spectrum width which has turned out to be particularly difficult, see e.g. [2, 7]. For that, we provide a framework for the use of edge-path formulations that selects an appropriate subset of paths to operate on with the help of combinatorial properties in such a way, that it is neither necessary to enumerate all possible routing paths nor to apply generic column generation techniques, but that optimality of the solution can still be guaranteed.

The idea is to iterate two major steps: on the one hand, a min-cost multi-commodity flow computes a lower bound on the minimum spectrum width and provides us with routing paths, on the other hand, solving an edge-path formulation using the already generated subset of routing paths provides a solution and an upper bound on the minimum spectrum width. As long as there is a gap between the lower and upper bounds, we add constraints to forbid (partial or full) routings that cause the use of a larger spectrum than the current lower bound, and iterate both major steps until the lower bound equals the upper bound and, thus, an optimal solution has been found (or infeasibility of the instance has been detected).

In the following, we present details on all steps involved in the framework in Section 2, we provide computational results in Section 3, and close with some concluding remarks and lines of future research.

[1] An exception is an edge-path formulation from [4] that seems to be scalable to real-size instances by using column-generation methods. However, the authors of [4] consider an asymmetric version of the RSA problem where each link of the optical network is composed by two optical fibers to be used to transmit signals in one direction only. This makes the spectrum assignment easier (as less restrictions have to be taken into account), but is not used very often in practice by network operators as that way it is not possible to use the full spectral resources of the optical links.

2 FRAMEWORK TO COMPUTE THE MINIMUM SPECTRUM WIDTH

In order to compute the minimum spectrum width, we adopt a reinterpretation of the spectrum assignment as interval coloring of the edge intersection graph $I(\mathcal{P})$ of the routing $\mathcal{P}$ [9]. Each path $P_k \in \mathcal{P}$ becomes a node of $I(\mathcal{P})$, two nodes are joined by an edge if their corresponding paths in G are in conflict as they share an edge. An interval coloring in $I(\mathcal{P})$ corresponds to the spectrum assignment: assign a frequency interval S_k of w_k consecutive frequency slots (*slot contiguity*) to every node k and, thus, to every path P_k (*slot continuity*) such that the intervals of adjacent nodes are disjoint (*non-overlapping slots*).

Let $\mathbf{w} \in \mathbf{Z}_+^{|\mathcal{D}|}$ be the vector whose entries w_k are the slot requirements associated with the demands $k \in \mathcal{D}$. The interval chromatic number $\chi_I(I(\mathcal{P}), \mathbf{w})$ is the smallest size of a spectrum such that $I(\mathcal{P})$ weighted with w_k for each path P_k has a proper interval coloring. Given G and $\mathcal{D}$, the minimum spectrum width of any solution to the RSA problem thus equals

$$\chi_I(G, \mathcal{D}) = \min\{\chi_I(I(\mathcal{P}), \mathbf{w}) : \mathcal{P} \in \mathcal{R}\}$$

where $\mathcal{R}$ denotes the set of all possible routings of the demands $\mathcal{D}$ in G. Our goal is to compute $\chi_I(G, \mathcal{D})$ which has turned out to be particularly difficult, see e.g. [2, 7].

Lower bounds on $\chi_I(G, \mathcal{D})$. Consider the following two lower bounds of $\chi_I(G, \mathcal{D})$ that are exclusively related to the routing aspect of the problem (not yet taking the spectrum assignment into account).

We denote by $\ell(G, \mathcal{D})$ the minimum number of slots that need to be installed on all edges of the optical network G to allow a routing of all demands in $\mathcal{D}$. The value $\ell(G, \mathcal{D})$ corresponds to the maximum edge load $w(\mathcal{P}) = \max\{\sum_{P_k \ni e} w_k : e \in E\}$ in the most balanced routing $\mathcal{P}$, i.e., to the minimum maximum edge load, taken over all possible routings: we call

$$\ell(G, \mathcal{D}) = \min\{w(\mathcal{P}) : \mathcal{P} \in \mathcal{R}\}$$

the *load bound*. Due to the non-overlapping slot condition, all channels S_k of paths routed along a same edge of G need to be disjoint, thus, $\ell(G, \mathcal{D})$ is a lower bound of $\chi_I(G, \mathcal{D})$.

We further consider the weighted clique number $\omega(I(\mathcal{P}), \mathbf{w})$ of the edge intersection graph $I(\mathcal{P})$ of the routing $\mathcal{P}$ (that is the maximum weight of a clique, a set of pairwise adjacent nodes, in $I(\mathcal{P})$, taking the node weights $\mathbf{w}$ into account). We denote by

$$\omega(G, \mathcal{D}) = \min\{\omega(I(\mathcal{P}), \mathbf{w}) : \mathcal{P} \in \mathcal{R}\}$$

the *clique bound*, i.e., the minimum over all maximum weighted cliques in $I(\mathcal{P})$, taken over all possible routings $\mathcal{P}$. On the one hand, all paths in a routing $\mathcal{P}$ passing through a same edge e of G are mutually in conflict and form a clique in $I(\mathcal{P})$, which shows that $\ell(G, \mathcal{D})$ is a lower bound of $\omega(G, \mathcal{D})$. On the other hand, all channels S_k of paths P_k forcing a clique in $I(\mathcal{P})$ need to be disjoint due to the non-overlapping slot condition such that $\omega(I(\mathcal{P}), \mathbf{w}) \leq \chi_I(I(\mathcal{P}), \mathbf{w})$ holds for any $I(\mathcal{P})$ and, thus, $\omega(G, \mathcal{D})$ is a lower bound of $\chi_I(G, \mathcal{D})$:

$$\ell(G, \mathcal{D}) \leq \omega(G, \mathcal{D}) \leq \chi_I(G, \mathcal{D}). \tag{1}$$

There are instances of the RSA problem where there is a gap between any two parameters from this chain, see Exp. 2.1.

Example 2.1. Consider the following instance of the RSA problem with the optical network G shown in Fig. 1 and the following set $\mathcal{D}$ of demands:

k	$o_k \to d_k$	$\bar{l}_k$	w_k	path P_k	channel S_k
1	$a \to c$	3	1	$a \to b \to c$	3
2	$c \to e$	3	2	$c \to b \to d \to e$	1 2
3	$e \to f$	3	2	$e \to d \to f$	3 4
4	$f \to g$	3	2	$f \to d \to g$	1 2
5	$g \to h$	3	2	$g \to d \to h$	3 4
6	$h \to a$	3	2	$h \to d \to b \to a$	5 6

As the network G is a tree, there is a unique routing $\mathcal{P}$, as indicated in the table above.

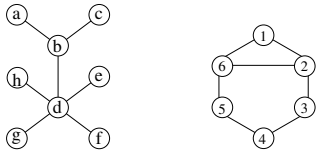


Figure 1: A network G and $I(\mathcal{P})$ of the routing.

Since the load of all edges incident to node d equals 4, $\ell(G, \mathcal{D}) = 4$ follows. The edge intersection graph $I(\mathcal{P})$ of the routing is also shown in Fig. 1. The nodes 1, 2, 6 form a clique of weight 5, hence we have $\omega(G, \mathcal{D}) = 5$. Any interval coloring of $I(\mathcal{P})$ needs at least 6 colors, as indicated in the table above. Hence, there is a gap between any two parameters from the chain (1).

We call a clique Q in $I(\mathcal{P})$ a *non-edge clique* if the routing paths composing Q pairwise intersect, but do not all meet in a same edge (like the clique formed by nodes 1, 2, 6 in the above example). Only non-edge cliques can cause a gap between $\ell(G, \mathcal{D})$ and $\omega(G, \mathcal{D})$.

A graph is *superperfect* if and only if the weighted clique number and the interval chromatic number coincide for all possible non-negative integral node weights, see e.g. [6]. Only non-superperfect subgraphs of $I(\mathcal{P})$ can cause $\omega(I(\mathcal{P}), \mathbf{w}) < \chi_I(I(\mathcal{P}), \mathbf{w})$ for the given weight $\mathbf{w}$ (as the 5-hole formed by nodes 2, 3, 4, 5, 6 in the above example), and, thus a gap between $\omega(G, \mathcal{D})$ and $\chi_I(G, \mathcal{D})$.

We here restrict to the search for cliques as it has been shown in [9] that there are too many different non-superperfect subgraphs that may occur in $I(\mathcal{P})$ and thus, it is questionable whether the time spent for their analysis is a gain for the overal running time.

Multi-commodity flows (MCF). We use multi-commodity flows in an auxiliary network G_f constructed from G to handle the lower bound and to determine routings $\mathcal{P}$.

We denote by $G_f = (V, A)$ the directed graph obtained from the optical network $G = (V, E)$ by replacing every edge $e = uv$ of G by a pair of oppositely-directed arcs $a = (u, v), \bar{a} = (v, u)$. We say that $a = (u, v)$ is the arc *outgoing* from u and *incoming* to v and denote by $\delta^-(v)$ the set of arcs incoming to v and by $\delta^+(v)$ the set of arcs outgoing from v.

Each demand $k \in \mathcal{D}$ corresponds to a commodity f_k with source $o_k \in V$ and sink $d_k \in V$ in G_f. The ILP to determine the minimum number *cap* of slots needed on all edges of G to allow the routing of all demands $k \in \mathcal{D}$ is as follows:

$$
\begin{array}{ll}
\min\ cap & \\
\sum_{a \in \delta^+(o_k)} f_k(a) & = 1\ \forall k \\
\sum_{a \in \delta^-(o_k)} f_k(a) & = 0\ \forall k \\
\sum_{a \in \delta^-(d_k)} f_k(a) & = 1\ \forall k \\
\sum_{a \in \delta^+(d_k)} f_k(a) & = 0\ \forall k \\
\sum_{a \in \delta^-(v)} f_k(a) - \sum_{a \in \delta^+(v)} f_k(a) & = 0\ \forall k, \forall v \neq o_k, d_k \\
\sum_{a \in \delta^-(v)} f_k(a) & \leq 1\ \forall k, \forall v \neq o_k, d_k \\
\sum_{a \in \delta^-(v)} l_a f_k(a) & \leq \bar{l}_k\ \forall k \\
\sum_{k \in \mathcal{D}} w_k f_k(a) + \sum_{k \in \mathcal{D}} w_k f_k(\bar{a}) & \leq cap\ \forall \text{ pairs } a, \bar{a} \in A \\
f_k(a) & \in \{0, 1\} \forall k, a
\end{array}
\tag{2}
$$

As the sum of flow values on each pair of oppositely-directed arcs $a = (u, v), \bar{a} = (v, u)$ (and, thus, on each edge $e = uv$ of G) is bounded by *cap* and the value of *cap* is minimized, ILP (2) indeed computes the load bound by

$$\ell(G, \mathcal{D}) = cap.$$

This happens in the initial run of the multi-commodity flow. If $\ell(G, \mathcal{D}) > \bar{s}$, the considered instance $(G, \bar{s}, \mathcal{D})$ is infeasible. Otherwise, the computed multi-commodity flow provides us with a routing $\mathcal{P}$ which allows us to continue with solving an edge-path formulation.

In later runs of the multi-commodity flow, we have to take forbidden cliques Q (of weight $w(Q) > \ell(G, \mathcal{D})$) and forbidden routings (to never consider a same routing twice) into account. For that, ILP (2) is enhanced by the following constraints: forbidden routing constraints associated with a routing $\mathcal{P}$

$$\sum_{k \in \mathcal{D}} \sum_{a \in A^k_{\mathcal{P}}} f_k(a) \leq \sum_{k \in \mathcal{D}} |A^k_{\mathcal{P}}| - 1 \tag{3}$$

where $A^k_{\mathcal{P}}$ denotes the subset of arcs with $f_k(a) > 0$ in $\mathcal{P}$ and forbidden clique constraints associated with a clique Q

$$\sum_{k \in Q} \sum_{a \in A^k_Q} f_k(a) \leq \sum_{k \in Q} |A^k_Q| - 1 \tag{4}$$

where A^k_Q is the subset of arcs a corresponding to edges in G where two paths from Q meet.

The objective function value *cap* computed by ILP (2) enhanced by constraints (3) and (4) does not necessarily equal the load bound $\ell(G, \mathcal{D})$ anymore, but corresponds to the maximal edge load of a most-balanced routing, taking forbidden cliques and forbidden previous routings into account.

That way, it is possible to increase the lower bound towards a match with the current upper bound, coming from the spectrum width of the best solution found so far.

An edge-path formulation (EPF). For the framework to compute $\chi_I(G, \mathcal{D})$, any edge-path formulation to solve the RSA problem can be used. Here we make use of the edge-path formulation related to the novel edge-node model from [7], i.e., we will adopt the way to encode the spectrum assignment from [7], but will simplify the routing as follows.

Let $\bar{\mathcal{P}}$ be the set of currently-considered routing paths, partitioned into $\bar{\mathcal{P}} = \bar{\mathcal{P}}_1 \cup \ldots \cup \bar{\mathcal{P}}_{|\mathcal{D}|}$ where $\bar{\mathcal{P}}_k = \{P^1_k, \ldots, P^{m_k}_k\}$ denotes the subset of routing paths currently available for demand

$k \in \mathcal{D}$. To find a routing $\mathcal{P}$, we use path selection variables

$$y_k^i = \begin{cases} 1 & \text{if path } P_k^i \in \bar{\mathcal{P}}_k \text{ is selected for } \mathcal{P}, \\ 0 & \text{otherwise,} \end{cases}$$

and have to ensure that one path per demand is taken.

For the spectrum assignment, we adopt the following sets of binary variables from [7]: For each demand $k \in \mathcal{D}$ and each edge $e \in E$, variable $x_e^k \in \{0,1\}$ indicates whether or not demand k is routed through edge e. For each demand $k \in \mathcal{D}$ and each slot $s \in S$, variable $z_s^k \in \{0,1\}$ encodes the fact of whether or not the slot s is the last slot of the channel assigned to demand k. For each demand $k \in \mathcal{D}$, each slot $s \in S$ and each edge $e \in E$, variable $t_e^{sk} \in \{0,1\}$ indicates whether or not demand k uses the slot s on edge E. Moreover, a variable $s_{max} \in S$ is used to express the maximum slot used.

Let b_{low} be the current lower bound and b_{up} be the current upper bound on $\chi_I(G,\mathcal{D})$, then the edge-path formulation based on the edge-node model from [7] reads as minimum violation problem:

$$\begin{array}{ll}
\min |\mathcal{D}| s_{max} & + \sum_{b_{low}<s<b_{up}, k\in\mathcal{D}} z_k^s \\
\sum_{P_k^i \in \bar{\mathcal{P}}_k} y_k^i & = 1 \ \forall k \\
\sum_{P_k^i \in \bar{\mathcal{P}}_k, e \in P_k^i} y_k^i & = x_k^e \ \forall k, e \\
\sum_{1\le s<w_k} z_k^s & = 0 \ \forall k \\
\sum_{w_k \le s<b_{up}} z_k^s & = 1 \ \forall k \\
\sum_{1\le j\le w_k} z_k^{s+j} + x_k^e & \le t_k^{e,s} + 1 \ \forall k, e, s \in \{1,\dots,b_{up}-1\} \\
\sum_{1\le s<b_{up}} t_k^{e,s} & = w_k x_k^e \ \forall k, e \\
\sum_{k\in\mathcal{D}} t_k^{e,s} & \le 1 \forall e, s \in \{1,\dots,b_{up}-1\} \\
\sum_{w_k \le s<b_{up}} s z_k^s & \le s_{max} \ \forall k \\
s_{max} & \le b_{up} - 1 \\
y_k^i, x_k^e, z_k^s, t_k^{e,s} & \in \{0,1\}
\end{array} \tag{5}$$

The objective function ensures that a span-minimal solution is found and that the use of frequency slots within $\{b_{low}+1,\dots,b_{up}-1\}$ is penalized. The path selection constraints ensure that one path per demand is selected, the remaining constraints are adopted from [7] where it was shown that they correctly encode a solution (when all demands have to be served).

Note that for the first run of the edge-path formulation, the intitial values are $b_{low} = \ell(G,\mathcal{D})$ and $b_{up} = \bar{s}+1$ so that we operate on the full spectrum $\{1,\dots,\bar{s}\}$; each subset $\bar{\mathcal{P}}_k$ contains exactly one path, obtained from the initial routing $\mathcal{P}$. If the first run of the edge-path formulation does not result in a solution, we have to go back to the multi-commodity flow to find another routing. If a solution $(\mathcal{P},\mathcal{S})$ with span s_{max} has been found, we proceed as follows: if s_{max} equals b_{low}, then $(\mathcal{P},\mathcal{S})$ is clearly optimal; otherwise, we have $b_{low} < s_{max} < b_{up}$, update $b_{up} = s_{max}$ and keep $(\mathcal{P},\mathcal{S})$ as currently best solution.

To analyze the solution $(\mathcal{P},\mathcal{S})$ in terms of cliques of weight greater than b_{low}, we proceed as follows. Determine from $(\mathcal{P},\mathcal{S})$ the subset $\mathcal{D}_c \subset \mathcal{D}$ of critical demands k whose channel $S_k \in \mathcal{S}$ uses frequency slots within $\{b_{low}+1,\dots,b_{up}-1\}$. Critical demands k may be contained in a clique Q of weight $w(Q) > b_{low}$, and this clique Q must be contained in the closed neighborhood $N[k] = N(k)\cup\{k\}$ of k in the edge intersection graph $I(\mathcal{P})$ of the routing $\mathcal{P}$. Hence, for each critical demand $k \in \mathcal{D}_c$, we construct the subgraph H_k of $I(\mathcal{P})$ induced by $N[k]$, enumerate in H_k all cliques Q of weight $w(Q) > b_{low}$ and include them in a set $\mathcal{Q}$ of forbidden cliques as triples $(\mathcal{P}_Q, E_Q, w(Q))$ with $\mathcal{P}_Q = \{P_k \in \mathcal{P} : k \in Q\}$ and E_Q subset of edges of G where paths from $\mathcal{P}_Q$ meet.

In later runs of the edge-path formulation, the use of the spectrum $\{1,\dots,b_{up}-1\}$ ensures that every new solution improves the current upper bound. In addition, we have to take forbidden cliques Q (of weight $w(Q) > b_{low}$) and forbidden routings (to never consider a same routing twice) into account. For that, (5) is enhanced by forbidden (partial or full) routing constraints: for a forbidden clique or a forbidden routing $\mathcal{P}'$,

$$\sum_{P_k^i \in \mathcal{P}'} y_k^i \le |\mathcal{P}'| - 1 \tag{6}$$

ensures that not all paths $P_k^i \in \mathcal{P}'$ can be selected together again. Note that if $\mathcal{P}'$ corresponds to a clique, then all routings containing this subset of paths are forbidden, to exclude all routings $\mathcal{P}$ with $\omega(I(\mathcal{P}),\mathbf{w}) > b_{low}$. If in later runs of the edge-path formulation, the current lower bound b_{low} is larger than the weight of a forbidden clique Q, then the clique Q has to be reallowed to operate on the whole set of routings with b_{low} as lower bound. For that, an intermediate value b_q indicating the weight of the lightest forbidden clique will be used.

Framework to compute $\chi_I(G,\mathcal{D})$. Here, we summarize the results from the previous sections to formulate a framework to compute $\chi_I(G,\mathcal{D})$.

Input: We take as input an instance $(G,\bar{s},\mathcal{D})$.
Output: The output will be a solution $(\mathcal{P}^*,\mathcal{S}^*)$ with span $\chi_I(G,\mathcal{D})$ or a certificate for infeasibility.
Initialization: We initialize

- an upper bound by $b_{up} = \bar{s}+1$, a clique bound by $b_q = \bar{s}+1$,
- a set of previously used routings by $\mathcal{F} = \emptyset$, and a set of critical non-edge cliques by $\mathcal{Q} = \emptyset$.

We construct the auxiliary network G_f from G and compute in G_f a multi-commodity flow f using ILP (2) with the objective to minimize *cap*.

If no feasible solution has been found then

- return "instance infeasible (due to transmission reach)"

Else (flow f with capacity *cap* has been found):

- if $cap > \bar{s}$ return "instance infeasible (as $\ell(G,\mathcal{D}) > \bar{s}$)"
- else set lower bound $b_{low} = cap$
  determine from f the according routing $\mathcal{P}_f$
  initialize a set of considered paths by $\bar{\mathcal{P}} = \mathcal{P}_f$.

Edge-Path Formulation (EPF): Launch the edge-path formulation (5) with $\bar{\mathcal{P}}$ as set of paths, enhanced by forbidden (partial or full) routing constraints (6) for all $\mathcal{P}' \in \mathcal{Q}\cup\mathcal{F}$ as a minimum violation problem where the objective is to minimize the span of the solution and penalties.

If no feasible solution has been found:

- Let $\mathcal{F} := \mathcal{F}\cup\{\mathcal{P}_f\}$ and continue with MCF.

Else (i.e. a solution $(\mathcal{P},\mathcal{S})$ with span s_{max} has been found):

- If $s_{max} = b_{low}$, then return $(\mathcal{P},\mathcal{S})$ as optimal solution.

- Else update $b_{up} = s_{max}$ and keep $(\mathcal{P}, \mathcal{S})$ as currently best solution.
  Determine from $(\mathcal{P}, \mathcal{S})$ the subset $\mathcal{D}_c \subset \mathcal{D}$ of critical demands k whose channel $S_k \in \mathcal{S}$ uses frequency slots within $\{b_{low}+1, \ldots, b_{up}-1\}$. For each critical demand $k \in \mathcal{D}_c$:
  – Construct the subgraph H_k of $I(\mathcal{P})$ induced by $N[k]$, find in H_k all cliques Q of weight $w(Q) > b_{low}$.
  – Include them in $\mathcal{Q}$; if $w(Q) < b_q$ then update $b_q = w(Q)$.
  Let $\mathcal{F} = \mathcal{F} \cup \{\mathcal{P}_f, \mathcal{P}\}$ and continue with MCF.

Multi-Commodity Flow (MCF): Compute a multi-commodity flow f minimizing *cap* using ILP (2), enhanced by forbidden routing constraints (3) associated with $\mathcal{P} \in \mathcal{F}$ and forbidden clique constraints (4) associated with $Q \in \mathcal{Q}$. If no flow has been found:

- if $b_q = b_{up} = \bar{s} + 1$, return "instance infeasible ($\chi_I(G, \mathcal{D}) > \bar{s}$)"
- else if $b_{up} \leq b_q, \bar{s}$, return $(\mathcal{P}^*, \mathcal{S}^*)$ as optimal solution
- else (i.e. we have $b_{low} < b_q < b_{up} \leq \bar{s}$): remove from $\mathcal{Q}$ all cliques Q of weight $w(Q) = b_q$, update b_q to $\min(w(Q) : Q \in \mathcal{Q})$ or to $\bar{s}+1$ if $\mathcal{Q} = \emptyset$, continue with MCF.

Else (i.e. a flow f with capacity *cap* has been found):

- if $b_q = b_{up} = \bar{s} + 1 \leq cap$, return "instance infeasible ($\chi_I(G, \mathcal{D}) > \bar{s}$)"
- else if ($b_q = b_{up} \leq \bar{s}$ and $b_{up} \leq cap$) or ($b_{up} < b_q$ and $b_{up} \leq cap$), return $(\mathcal{P}^*, \mathcal{S}^*)$ as optimal solution
- else (we make some updates and continue):
  if ($cap < b_q$ and $cap < b_{up}$), set $b_{low} = cap$,
  if ($b_q \leq cap$ and $b_q < b_{up}$), set $b_{low} = b_q$, remove from $\mathcal{Q}$ all cliques Q of weight $w(Q) = b_q$,
  update b_q to $\min\{w(Q) : Q \in \mathcal{Q}\}$ or to $\bar{s}+1$ if $\mathcal{Q} = \emptyset$,
  determine from f the routing $\mathcal{P}_f$, add the paths from $\mathcal{P}_f$ to $\bar{\mathcal{P}}$ and continue with EPF.

With the help of the above given arguments and a case analysis of the possible situations after running the multi-commodity flow, we can show:

Theorem 2.2. *Given an instance $(G, \bar{s}, \mathcal{D})$ of the RSA problem, the above described framework correctly computes a solution $(\mathcal{P}^*, \mathcal{S}^*)$ with span $\chi_I(G, \mathcal{D})$ or certifies infeasibility.*

3 COMPUTATIONAL RESULTS

In this section, we present some preliminary computational results to evaluate the computational performances achieved with the herein proposed framework for the use of edge-path formulations in comparison with the related edge-node formulation from [7] (i.e., where both formulations use the same way to encode the spectrum assignment).

For that, we use two different sets of test instances. On the one hand, we use artificially constructed test instances $(G, \bar{s}, \mathcal{D})$ with networks having up to 14 nodes and only few demands, some of them having the property that $\ell(G, \mathcal{D}) < \chi_I(G, \mathcal{D})$, some being infeasible due to $\bar{s} < \chi_I(G, \mathcal{D})$, indicated by "inf." in Table 1.

On the other hand, three network topologies from the literature are investigated: Spain, NSF and German [14, 17]. The Spain topology has 21 nodes, 35 edges and 50 slots; the NSF topology has 14 nodes, 21 edges and 60 slots; the German topology has 17 nodes, 25 edges and 60 slots. For each network topology, three sets of randomly generated demands are evaluated. Each considered demand requires either 3, 5 or 6 slots and supports, respectively, 100 Gb/s (3000 km reach), 200 Gb/s (1500 km reach), or 400 Gb/s (600 km reach). The computational results are listed in Table 2.

Network	$\bar{s}$	$\ell(G, \mathcal{D})$	$\chi_I(G, \mathcal{D})$	#	FWK (ms)	ENF (ms)
Test net 1	5	3	4	2	58	95
Test net 2	5	-	inf.	0	4	15
Test net 2	8	6	6	2	36	137
Test net 3	16	11	13	3	368	1182
Test net 4	20	11	16	9	1247	17738
Test net 5	16	12	14	4	454	1425
Test net 6	12	8	10	3	227	233
Test net 7	7	-	inf.	0	7	151
Test net 7	8	-	inf.	1	50	211
Test net 7	9	8	9	2	61	251
Test net 7	10	7	9	3	245	654

Table 1: Comparison between the two approaches on artificially constructed test instances.

Network	$\bar{s}$	$\lvert\mathcal{D}\rvert$	$\ell(G, \mathcal{D})$	$\chi_I(G, \mathcal{D})$	#	FWK (ms)	ENF
Spain	50	10	4	4	2	20523	n.t.
Spain	50	20	7	7	9	647694	n.t.
Spain	50	30	8			n.t.	n.t.
German	60	10	12	12	4	241565	n.t.
German	60	20	23	23	2	190286	n.t.
German	60	30	32			n.t.	n.t.
NSF	60	30	32	32	2	100851	n.t.
NSF	60	60	38	38	2	60650	n.t.
NSF	60	90	50	50	2	2135706	n.t.

Table 2: Comparison between the two approaches on networks from the literature.

All experiments are performed using the state-of-the-art MIP solver CPLEX 12.10 on the high performance platform available at LIMOS. For each instance, the lower bound $\ell(G, \mathcal{D})$ and the minimum spectrum width $\chi_I(G, \mathcal{D})$ are given, followed by the number # of iterations needed by the framework FWK. For both formulations, the herein proposed framework (FWK) and the related edge-node formulation (ENF), the total time in milliseconds required for the optimization is displayed. A closer analysis of the overall running time spent by our framework FKW shows moreover that the percentage of the time spent solving the subproblems with the edge-path formulation increases with the number of demands (up to 99 % for the last instance in Table 2).

A time limit of 100 hours was imposed in each run. The absence of results for some instances indicates that the computation could not terminate within this time limit[2], indicated by "n.t." in Table 2.

We clearly see that, for all artificially constructed test instances, the computation time of the herein proposed framework is certainly smaller than for the related edge-node formulation from [7]. Moreover, the herein proposed framework could solve substantially more instances to optimality within the time limit than the related edge-node formulation, see Table 2.

[2]Unfortunately, the high performance platform does not return any information (e.g. on the objective function value of the best solution found) when the process exceeds the time limit so that no information about the remaining gap can be provided.

4 CONCLUDING REMARKS

In this paper, we studied the routing and spectrum assignment problem. The majority of existing models for the problem uses edge-path formulations where variables are associated with all possible routing paths so that the number of variables grows exponentially with the size of the instance. Therefore, either precomputed subsets of all possible paths per demand are used (which cannot guarantee optimality of the solutions) or column-generation methods have to be applied (as the explicit models are far too big for computation). However, computational results show that the size of the instances that can be solved to optimality that way is rather limited, see e.g. [2, 7, 11, 13, 15].

Our contribution is to provide a framework for the use of edge-path formulations to minimize the spectrum width of a solution. For that, we select an appropriate subset of paths to operate on with the help of combinatorial properties in such a way that optimality of the solution can be guaranteed due to a match of a lower bound (derived from the edge load of the routings) and an upper bound (coming from the span of the best solution found so far).

First computational results suggest that the herein proposed framework for the use of edge-path formulations is competitive in comparison with the related edge-node formulation from [7] (i.e., where both formulations use the same way to encode the spectrum assignment).

Our future work includes comparing different edge-path formulations from the literature (or edge-path formulations derived from edge-node formulations) to see which one behaves best in the context of our framework.

Moreover, there are different directions to further improve the current framework. On the one hand, we observe that two demands $k, k' \in \mathcal{D}$ with the same origin-destination pair operate on the same set of routing paths $\bar{\mathcal{P}}_k = \bar{\mathcal{P}}_{k'}$. If, in addition, $w_k = w_{k'}$ holds, then both the routes and the channels assigned to k and k' can be exchanged while keeping the same physical solution in the network. With an increasing number of demands, this effect causes a large number of symmetric solutions so that applying symmetry breaking techniques seems to be advantageous.

On the other hand, the cliques $Q \in \mathcal{Q}$ are used to prevent that all routings containing them are explored during the process. We note that all forbidden clique contraints (4) for MCF and (6) for EPF are redundant if they are associated with cliques $Q \in \mathcal{Q}$ properly contained in another clique $Q' \in \mathcal{Q}$. This observation shall be used to reduce the number of redundant constraints in order to speed up the computation.

Finally, our future work includes proposing similar frameworks to handle the RSA problem w.r.t. other objectives like minimizing the number of edges in routing paths, the lengths of the routing paths, or the number of edges from the network used to route the demands.

ACKNOWLEDGEMENT

This work was supported by the French National Research Agency grant ANR-17-CE25-0006, project FLEXOPTIM, and the STIC Am-Sud project 22-STIC-08. In addition, the stay of Juan Pablo Nant at the University Clermont Auvergne was supported by the WOW! CAP 20-25 funding program.

REFERENCES

[1] F. Bertero, M. Bianchetti, and J. Marenco. 2018. Integer programming models for the routing and spectrum allocation problem. *TOP* 26, 3 (2018), 465–488.
[2] Anliang Cai, Gangxiang Shen, Limei Peng, and Moshe Zukerman. 2013. Novel node-arc model and multiiteration heuristics for static routing and spectrum assignment in elastic optical networks. *Journal of Lightwave Technology* 31, 21 (2013), 3402–3413.
[3] Konstantinos Christodoulopoulos, Ioannis Tomkos, and Emmanuel A Varvarigos. 2011. Elastic bandwidth allocation in flexible OFDM-based optical networks. *Journal of Lightwave Technology* 29, 9 (2011), 1354–1366.
[4] Julian Enoch and Brigitte Jaumard. 2018. Towards optimal and scalable solutionfor routing and spectrum allocation. *Electronic Notes in Discrete Mathematics* 64 (2018), 335–344.
[5] Mahmoud Fayez, Iyad Katib, George N Rouskas, and HM Faheem. 2015. Spectrum assignment in mesh elastic optical networks. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–6.
[6] Martin Charles Golumbic. 2004. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier.
[7] Youssouf Hadhbi, Hervé Kerivin, and Annegret Wagler. 2019. A novel integer linear programming model for routing and spectrum assignment in optical networks. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 127–134.
[8] Masahiko Jinno, Hidehiko Takara, Bartlomiej Kozicki, Yukio Tsukishima, Yoshiaki Sone, and Shinji Matsuoka. 2009. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *IEEE communications magazine* 47, 11 (2009), 66–73.
[9] Hervé Kerivin and Annegret Wagler. 2020. On superperfection of edge intersection graphs of paths. In *Graphs and Combinatorial Optimization: from Theory to Applications*, C. Gentile et al. (Ed.). AIRO Springer Series 5, 79–91.
[10] Mirosław Klinkowski, João Pedro, Davide Careglio, Michał Pióro, João Pires, Paulo Monteiro, and Josep Solé-Pareta. 2010. An overview of routing methods in optical burst switching networks. *Optical Switching and Networking* 7, 2 (2010), 41–53.
[11] Mirosław Klinkowski, Michał Pióro, Mateusz Żotkiewicz, Krzysztof Walkowiak, Marc Ruiz, and Luis Velasco. 2015. Spectrum allocation problem in elastic optical networks-a branch-and-price approach. In *2015 17th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 1–5.
[12] Miroslaw Klinkowski and Krzysztof Walkowiak. 2011. Routing and spectrum assignment in spectrum sliced elastic optical path network. *IEEE Communications Letters* 15, 8 (2011), 884–886.
[13] Mirosław Klinkowski and Krzysztof Walkowiak. 2015. A simulated annealing heuristic for a branch and price-based routing and spectrum allocation algorithm in elastic optical networks. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 290–299.
[14] Pablo Pavon-Marino, Siamak Azodolmolky, Ramon Aparicio-Pardo, Belen Garcia-Manrubia, Yvan Pointurier, Marianna Angelou, Josep Sole-Pareta, Joan Garcia-Haro, and Ioannis Tomkos. 2009. Offline impairment aware RWA algorithms for cross-layer planning of optical networks. *Journal of Lightwave Technology* 27, 12 (2009), 1763âĂŞ1775.
[15] Marc Ruiz, Michał Pióro, Mateusz Żotkiewicz, Mirosław Klinkowski, and Luis Velasco. 2013. Column generation algorithm for RSA problems in flexgrid optical networks. *Photonic network communications* 26, 2-3 (2013), 53–64.
[16] Shahrzad Shirazipourazad, Chenyang Zhou, Zahra Derakhshandeh, and Arunabha Sen. 2013. On routing and spectrum allocation in spectrum-sliced optical networks. In *2013 Proceedings IEEE INFOCOM*. IEEE, 385–389.
[17] Luis Velasco, Miroslaw Klinkowski, Marc Ruiz, and Jaume Comellas. 2012. Modeling the routing and spectrum allocation problem for flexgrid optical networks. *Photonic Network Communications* 24, 3 (2012), 177–186.
[18] Yang Wang, Xiaojun Cao, and Yi Pan. 2011. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *2011 Proceedings Ieee Infocom*. IEEE, 1503–1511.
[19] Mateusz Żotkiewicz, Michal Pióro, Marc Ruiz, Miroslaw Klinkowski, and Luis Velasco. 2013. Optimization models for flexgrid elastic optical networks. In *2013 15th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 1–4.

A branch-and-cut algorithm for the availability-aware VNF placement problem in virtualized networks

Rafael Colares
Orange Innovation
Châtillon, France
rafael.colares@gmail.com

Amal Benhamiche
Orange Innovation
Châtillon, France
amal.benhamiche@orange.com

Yannick Carlinet
Orange Innovation
Châtillon, France
yannick.carlinet@orange.com

Nancy Perrot
Orange Innovation
Châtillon, France
nancy.perrot@orange.com

ABSTRACT

In this work, we address the problem of optimally placing Virtual Network Functions throughout a 5G network so that a given set of Service Function Chains can achieve high levels of end-to-end availability. We tackle this problem from a combinatorial perspective and propose a probabilistic approach to evaluate the real end-to-end availability of a service. This generates a non-linear character to the problem which is then linearized to derive an original integer programming formulation for it. We also introduce new families of valid inequalities reinforcing the proposed formulation. Based on these inequalities, we derive an efficient branch-and-cut algorithm.

KEYWORDS

Network reliability, combinatorial optimization, valid inequalities

1 INTRODUCTION

Network Function Virtualization (NFV) is one of the key enabler technologies for tackling the challenges of the upcoming 5G use-cases requirements. These high-level-requirement use-cases include autonomous vehicles, smart factories, smart cities, e-health, for instance. With virtualization, Network Functions gain the ability to be run as applications in Virtual Machines (VMs) or containers on off-the-shelf hardware. This allows higher scalability, more flexibility and reduces network management costs.

Virtual Network Functions (VNF) however are more prone to errors and failures when compared to purpose-built hardware [10, 14, 16]. Indeed, a major challenge for NFV is to ensure high availability levels for its services. The service availability refers to its probability of being operational when required and is defined as the ratio between its expected uptime and total time values (see [2]). A service in NFV-based networks – also called a Service Function Chain (SFC) – is an origin-destination traffic demand composed of a set of VNFs that must be visited in a given order along its route.

In this sense, an SFC is available if and only if all its VNFs can be properly processed. Strict Service Level Agreements (SLAs) impose that SFCs should be highly available (in some cases, for more than 99.999% of the time, which roughly translates to 25.9 s downtime per month [11, 15]). Then, backup VNFs must be placed on the network so that the services can still be ensured even if some network nodes fail.

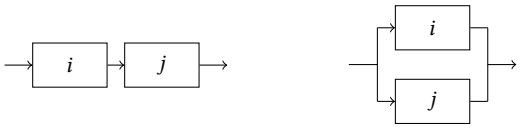
Most of the literature related to SFC's resilience is devoted to securing the network against single-node failures. In this case, it suffices to find two node-disjoint routes (a nominal and a backup) for each SFC. In [4], nominal SFC routes are known in advance and the goal is to find a minimum cost backup VNF placement. In [13], the authors impose a single route for each SFC and focus on minimizing the number of SFCs affected by a single node failure.

A few papers deal with multi-node failures. In [8, 9], heuristic methods treating the multi-node failure scenario are presented. These heuristics (i) construct a VNF assignment for each SFC based on node's remaining resources, (ii) reinforce the assignments by sequentially adding backup VNFs until the availability requirements are met, and (iii) routing is then done through k-shortest paths computations. An interesting exact approach is presented in [5], where service availabilities are computed using a probabilistic approach. However, each service is supposed to request only one VNF which reduces the end-to-end availability computation complexity. In [16], an in-between approach is considered where the goal is to protect the network against single-node failures while knowing that the failure of a node may impact other nodes (failure events are not independent and are related to the network topology structure).

In this paper, we further explore the service availability definition considered in [5] with the purpose of providing a mathematical model that optimizes VNF placements while formally taking into account the SFCs' availability. This allows us to ensure the required SLAs within a multi-node failure scenario. The paper is organized as follows. The problem is formally defined in Section 2 and its computational complexity is briefly discussed in Section 3. In Section 4, we propose an original ILP formulation for the problem which is then reinforced with valid inequalities in Section 5. To verify the efficiency of the proposed inequalities, Section 6 is devoted to the description of a branch-and-cut framework based on such inequalities and Section 7 presents the preliminary computational results obtained with this approach.

2 PROBLEM DEFINITION

Let $G = (V, A)$ be a directed, loopless, connected graph. Each node $v \in V$ has a capacity $C_v \in \mathbb{R}^+$, and an availability $0 < a_v < 1$, (*i.e.*,



(a) Serial composition. **(b) Parallel composition.**

Figure 1: Basic compositions of components

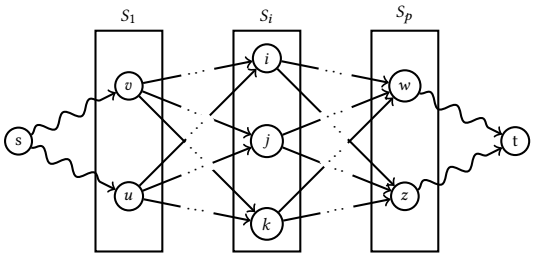


Figure 2: Illustration of a VNF assignment and the possible SFC routes induced by it.

a risk of $1 - a_v$ of being down). Moreover, let $\mathcal{F}$ be the set of VNF types, where each VNF $f \in \mathcal{F}$ has a resource consumption $r^f \in \mathbb{R}^+$, and a placement cost $c_v^f \in \mathbb{R}^+$ for each node $v \in V$. Finally, let K be the set of SFC demands, where each demand $k \in K$ is defined by (i) an origin $o_k \in V$ and a destination $d_k \in V$, (ii) a bandwidth $b_k \in \mathbb{R}^+$, (iii) a required availability $A^k \in [0, 1]$, and (iv) an ordered set of distinct VNFs $F^k \subseteq \mathcal{F}$ that should be visited.

While different VNFs should be placed in series along the SFC route (see Figure 1a), redundant VNFs must be placed in parallel (see Figure 1b) so that whenever one fails, the SFC can be rerouted through one of the redundancies. In serial composition, all subcomponents need to be operational for the system to be available. For a parallel composition, however, it suffices that one subcomponent is operational [12]. It follows that given two subcomponents i and j with respective availabilities a_i and a_j, the availability induced by their serial (resp. parallel) composition is $a_i a_j$ (resp. $1 - \big[(1 - a_i)(1 - a_j)\big]$). Based on these remarks, we next describe the VNF assignment of an SFC as well as the availability it induces.

Given an SFC requiring the visit of p distinct VNFs, let $\mathcal{S} = \{S_1, \ldots, S_p\}$ denote a VNF assignment for such SFC, where each nonempty subset $S_i \subseteq V$, for $i = 1, \ldots, p$, represents the set of nodes where its i-th VNF can be processed. Figure 2 illustrates such assignment. The subset $S_i \in \mathcal{S}$ is also called the i-th *section* of an SFC. From now on, let $I^k = \{1, \ldots, |F^k|\}$ denote the set of sections associated with SFC k. The probability that the i-th VNF of the considered SFC can be properly processed (*i.e.* the availability of its i-th section) is the probability that at least one of the nodes in S_i is operational. This probability, denoted by $a(S_i)$, is hence defined by

$$a(S_i) = 1 - \prod_{v \in S_i} \Big(1 - a_v\Big). \tag{1}$$

For an SFC to be operational, all its VNFs must be properly processed. The end-to-end SFC availability induced by a VNF assignment $\mathcal{S}$ is denoted by $A(\mathcal{S})$ and given by

$$A(\mathcal{S}) = \prod_{S \in \mathcal{S}} a(S) = \prod_{S \in \mathcal{S}} \left(1 - \prod_{v \in S} \Big(1 - a_v\Big)\right). \tag{2}$$

A VNF assignment $\mathcal{S}$ is said to satisfy the availability requirements of an SFC k if and only if $A(\mathcal{S}) \geq A^k$.

Given a VNF assignment $\mathcal{S}^k$ for each $k \in K$, let $\mathcal{G} = \{\mathcal{S}^k : k \in K\}$ denote the associated *global VNF assignment*. A global VNF assignment $\mathcal{G}$ is said to be feasible if the availability requirements of each SFC $k \in K$ is satisfied and the amount of resources consumed on any given node $v \in V$ is at most the node's capacity C_v, that is,

$$\sum_{k \in K, i \in I^k : v \in S_i^k} b_k r^{f(i,k)} \leq C_v \qquad \forall v \in V,$$

where $f(i, k)$ is a function mapping the i-th element of F^k, that is, the i-th VNF of SFC k.

Notice that a node $v \in V$ can only process a given VNF $f \in \mathcal{F}$ if f is *placed* on such node. Therefore, given a global VNF assignement $\mathcal{G}$, let $V_f(\mathcal{G}) \subseteq V$ denote, for any $f \in \mathcal{F}$, the set of nodes where f must be placed, that is,

$$V_f(\mathcal{G}) = \bigcup_{k \in K} \bigcup_{\substack{i \in I^k : \\ f(i,k) = f}} S_i^k.$$

The cost induced by a global VNF assignment $\mathcal{G}$ is therefore

$$\sum_{f \in \mathcal{F}} \sum_{v \in V_f(\mathcal{G})} c_v^f,$$

and our goal is to find the feasible global VNF assignment inducing the minimum cost.

3 COMPUTATIONAL COMPLEXITY

Even without taking into account the availability restrictions, the problem is already NP-Hard. Indeed, if each SFC requires the same and only one VNF, then the problem reduces to choosing a minimum cost subset of nodes $S \subseteq V$ where the VNF should be placed such that S is able to treat all SFCs. This is equivalent to a Variable Cost and Size Bin-Packing Problem[1] (see [6, 7]) where bins and items correspond to nodes and SFCs, respectively.

Furthermore, dealing with availability restrictions on their own is also an NP-Hard problem. Indeed, if node capacities are said to be unlimited and there is only one SFC (*i.e.*, $K = \{k\}$) requiring a single VNF to be considered, then the problem reduces to finding a minimum cost subset of nodes $S \subseteq V$ where the VNF should be placed such that the SFC required availability is achieved, *i.e.*, $a(S) \geq A^k$. This is clearly equivalent to a (Nonlinear) Knapsack Problem (see [3]) where each item corresponds to a node in V.

4 ILP FORMULATION

In this section, a natural formulation for the problem is described. The binary variables x_{vik} indicate whether or not the i-th VNF of SFC k can be processed on node v (*i.e.*, if $x_{vik} = 1$ then $v \in S_i^k$, otherwise $v \notin S_i^k$). For each node $v \in V$ and VNF $f \in \mathcal{F}$, the variable y_v^f indicates whether or not VNF f is placed on node v.

$$\min \sum_{v \in V} \sum_{f \in \mathcal{F}} c_v^f y_v^f \tag{3}$$

subject to

[1]The Variable Cost and Size Bin-Packing Problem is a generalization from the well-known Bin-Packing Problem where each bin has its own capacity and cost.

$$\sum_{v \in V} x_{vik} \geq 1 \qquad \forall k \in K, i \in I^k, \quad (4)$$

$$x_{vik} \leq y_v^{f(i,k)} \qquad \forall k \in K, i \in I^k, v \in V, \quad (5)$$

$$\sum_{k \in K} \sum_{i \in I^k} b_k r^{f(i,k)} x_{vik} \leq C_v \qquad \forall v \in V, \quad (6)$$

$$\prod_{i \in I^k} \left(1 - \prod_{v \in V} \left(1 - a_v x_{vik}\right)\right) \geq A^k \qquad \forall k \in K, \quad (7)$$

$$x_{vik} \in \{0,1\} \qquad \forall v \in V, k \in K, i \in I^k, \quad (8)$$

$$y_v^f \in \{0,1\} \qquad \forall v \in V, f \in \mathcal{F}. \quad (9)$$

The objective function (3) evaluates the total VNF placement cost. The assignment constraints (4) ensure that at least one VNF should be assigned to each section of each SFC. The VNF placement constraints (5) impose that a VNF can only be assigned to an SFC if it is already placed. The capacity constraints (6) guarantee that the capacity of each node is not exceeded. Availability constraints (7) force the SFCs' Service Level Agreement to be respected. Finally, constraints (8) and (9) settle the domains of variables.

The presence of constraints (7) however clearly makes the proposed formulation non-linear. Such non-linearity is difficult to be handled by traditional commercial solvers (*e.g.*, CPLEX, Gurobi). Thereby we next propose a linear reformulation of such constraints. For this, consider the following set of inequalities.

$$\sum_{(v,i) \in (V \times I^k) \setminus \mathcal{S}} x_{vik} \geq 1 \quad \forall k \in K, \mathcal{S} \subseteq V \times I^k : A(\mathcal{S}) < A^k. \quad (10)$$

Proposition 4.1. *An integer solution $\bar{x}$ satisfies inequalities* (7) *if and only if it satisfies inequalities* (10).

Proof. Let $\mathcal{G} = \{\mathcal{S}^k : k \in K\}$ denotes the global VNF assignment associated with solution $\bar{x}$, where $\mathcal{S}^k = \{(v,i) : \bar{x}_{vik} = 1\}$. Let $\bar{x}$ be an integer solution satisfying inequalities (7). Then, $A(\mathcal{S}^k) \geq A^k$ for any $k \in K$. We next show that inequalities (10) are all satisfied by $\bar{x}$. For this, suppose there exists $k' \in K$ and $\mathcal{S}' \subseteq V \times I^{k'}$ for which

$$\sum_{(v,i) \in (V \times I^{k'}) \setminus \mathcal{S}'} \bar{x}_{vik'} = 0.$$

By definition, $\mathcal{S}^k \subseteq \mathcal{S}'$ and hence $A(\mathcal{S}') \geq A(\mathcal{S}^k) \geq A^k$, which concludes the first part of the proof. To finish the proof, suppose now that $\bar{x}$ does not satisfy inequalities (7). Then, there exists $k' \in K$ for which $A(\mathcal{S}^{k'}) < A^{k'}$. It follows that, by definition, inequality (10) associated with k' and $\mathcal{S}^{k'}$ is violated. □

Proposition 4.1 shows that inequalities (10) are sufficient for ensuring the required availabilities and hence, we can now replace the non-linear inequalities (7) for (10). The resulting formulation – defined by (3)-(6),(8)-(10) – is linear but also non-compact since it requires an exponential number of constraints. A standard approach is hence to relax such constraints and append them when violated. For this, one needs to deal with the separation problem associated with inequalities (10). Recall that for a family of valid inequalities $\mathcal{I}$, the separation problem for $\mathcal{I}$ consists of either finding an inequality in $\mathcal{I}$ violated by a given vector $(\bar{x}, \bar{y})$ or proving that $(\bar{x}, \bar{y})$ satisfies all the inequalities in $\mathcal{I}$. For inequalities (10), the associated separation problem amounts to solve a (non-linear) Knapsack Problem (*c.f.* Section 3), that is, an NP-Hard problem.

Proposition 4.2. *The separation problem for inequalities* (10) *can be solved in linear time when vector $(\bar{x}, \bar{y})$ is integer.*

Proof. Notice that every component of vector $(\bar{x}, \bar{y})$ is binary. For each $k \in K$, let $\mathcal{S}^k = \{(v,i) : \bar{x}_{vik} = 1\}$. There exists an inequality in (10) violated by $(\bar{x}, \bar{y})$ if and only if there exists $k \in K$ such that $A(\mathcal{S}^k) < A^k$, which can be checked in linear time. □

As a consequence, we propose to solve the associated separation problem heuristically[2] whenever the solution is fractional and exactly otherwise. Nevertheless, such an approach leads to performance issues and the reasons are twofold:

(1) The initially relaxed constraints are the only ones enforcing the assignment of backup VNFs (which directly impacts the objective function). Since their separation problem is only solved exactly on integer solutions, the dual bound convergence is significantly slowed down.
(2) Even if inequalities (10) well-define the availability requirements, they are not strong inequalities. Indeed, the inclusion of a violated inequality (10) imposes that one non-assigned VNF should be in the solution. Such information is quite vague and hence the inclusion of a huge number of inequalities is required to obtain a feasible solution.

The next section is thus dedicated to the reinforcement of the studied formulation through the investigation of valid inequalities.

5 FORMULATION STRENGTHENING

As briefly discussed in the previous section, a major challenge consists of providing good bounds on the number of VNFs required to secure a given SFC without having to appeal to the linearized availability constraints (10). In order to derive such bounds, let us consider the following related combinatorial problem.

Given a set of nodes V and an SFC composed of p VNF types, find the VNF assignment $\mathcal{S}^*$ that induces the highest possible availability for such SFC while placing exactly $n \in \mathbb{N}$ VNFs ($n \geq p$) over the node-set V. We next show that this combinatorial problem may be solved in polynomial time and we use it to derive valid inequalities reinforcing the previously considered formulation.

Claim 1. *If $\mathcal{S} = \{S_1, \ldots, S_p\}$ is an optimal assignment, then each subset S_i is composed of the $|S_i|$ most available nodes.*

Claim 2. *If $\mathcal{S} = \{S_1, \ldots, S_p\}$ is an optimal assignment, then every assignment built from a permutation of sets $S_1, \ldots, S_p$ is also optimal.*

Proof. The availability function $A(\mathcal{S})$ defined by (2) is commutative over the elements of $\mathcal{S}$. □

Claim 3. *If $\mathcal{S} = \{S_1, \ldots, S_p\}$ is an optimal assignment, then the difference between the number of nodes in any two sets S_i and S_j within $\mathcal{S}$ is at most one, that is, $|S_i| - |S_j| \leq 1$.*

Proof. The proof is done by contradiction. Suppose that $\mathcal{S} = \{S_1, \ldots, S_p\}$ is an optimal assignment where there exists i and j for which $|S_i| - |S_j| \geq 2$. From Claim 1, S_i and S_j are composed of the

[2] Our heuristic procedure greedily constructs a VNF assignment $\mathcal{S}$ for each SFC k by iteratively picking the pair (v,i) that maximizes $\bar{x}_{vik}$ and minimizes $A(\mathcal{S} \cup (v,i))$.

most available nodes and hence $S_j \subset S_i$. Let $u_i \in S_i$ be the least available node in S_i. By definition, $u_i \notin S_j$.

Consider the VNF assignment $\mathcal{S}^* = \{S_1^*, \ldots, S_p^*\}$, where $S_k^* = S_k$ for any $k \in \{1, \ldots, m\} \setminus \{i, j\}$, $S_j^* = S_j \cup u_i$ and $S_i^* = S_i \setminus u_i$. Since u_i is the least available node in S_i, we have $a(S_i) > a(S_i^*) > a(S_j^*) > a(S_j)$. Moreover,

$$a(S_i^*) = 1 - \frac{\prod_{v \in S_i}(1 - a_v)}{1 - a_{u_i}} = \frac{a(S_i) - a_{u_i}}{1 - a_{u_i}},$$

and

$$a(S_j^*) = 1 - \left[\left(\prod_{v \in S_j}(1 - a_v)\right)(1 - a_{u_i})\right] = a(S_j) + a_{u_i}(1 - a(S_j)).$$

Next we show that $A(\mathcal{S}^*) > A(\mathcal{S})$, a contradiction since $\mathcal{S}$ is optimal. By definition,

$$A(\mathcal{S}^*) = A(\mathcal{S})\frac{a(S_j^*)a(S_i^*)}{a(S_j)a(S_i)} = A(\mathcal{S})\left(1 + \frac{a_{u_i}(a(S_i) - a(S_j^*))}{a(S_j)a(S_i)(1 - a_{u_i})}\right).$$

Since $a(S_i) > a(S_j^*)$, we have $A(\mathcal{S}^*) > A(\mathcal{S})$. □

As a result of Claims 1, 2 and 3, a simple greedy algorithm – see Algorithm 1 – solves the previously presented combinatorial problem.

Algorithm 1: Computation of VNF assignment $\mathcal{S}^*$ inducing the highest possible availability with exactly n VNFs

Let $S_i^* = \emptyset$ for $i = 1, \ldots, p$;
for $j = 0, \ldots, n-1$ **do**
  $i \leftarrow j \bmod p + 1$;
  Add the most available node in $V \setminus S_i^*$ to S_i^*;
end
Return $\mathcal{S}^* = \{S_1^*, \ldots, S_p^*\}$;

5.1 Valid inequalities

Remark 1. *If $\mathcal{S}$ is a VNF assignment such that $A(\mathcal{S}) \geq B$, then $A(\mathcal{S}') \geq B$ for any $\mathcal{S}' \subseteq \mathcal{S}$, since from equations* (1) *and* (2) *we have that $A(\mathcal{S}') \geq A(\mathcal{S})$.*

We next combine the results obtained from Remark 1 and Algorithm 1 in order to derive reinforcing valid inequalities. For this, let $\eta(U, p, B) \in \mathbb{N}$ denote the minimum number of VNFs required to be installed within node-set $U \subseteq V$ so that an SFC composed of p sections can meet the availability requirement B. Notice that $\eta(U, p, B)$ can be easily computed in polynomial time with the help of Algorithm 1. From the definition of $\eta(U, p, B)$, the following inequalities are obviously valid.

$$\sum_{i \in I^k}\sum_{v \in V} x_{vik} \geq \eta(V, |I^k|, A^k) \qquad \forall k \in K.$$

Such inequalities provide a lower bound on the number of VNFs required to be assigned to a given SFC. This can be further extended using Remark 1, which gives rise to the following *Chain Cover* inequalities.

$$\sum_{i \in Q}\sum_{v \in V} x_{vik} \geq \eta(V, |Q|, A^k) \qquad \forall k \in K, Q \subseteq I^k. \tag{11}$$

Proposition 5.1. *The Chain Cover inequalities* (11) *are valid.*

Proof. From Remark 1, the VNF assignment associated with sections in Q must induce an availability of at least A^k. Thus, inequalities (11) are clearly valid from the definition of $\eta(V, |Q|, A^k)$. □

Chain Cover inequalities provide important information concerning the minimum number of VNFs to be assigned over the sections of an SFC. However, such a number is constructed following the principles of Algorithm 1, that is, only the most available nodes in V are actually taken into account. If for any reason (*e.g.*, elevated cost, insufficient capacity, etc) some of these nodes are banned from being used, such lower bound might increase. We next focus on this case to derive a new family of valid inequalities. For this, consider the following *Node Cover* inequalities.

$$\sum_{v \in V \setminus U} \eta(U, 1, A^k)x_{vik} + \sum_{v \in U} x_{vik} \geq \eta(U, 1, A^k) \qquad \forall k \in K, i \in I^k, U \subseteq V. \tag{12}$$

Proposition 5.2. *The Node Cover inequalities* (12) *are valid.*

Proof. Let S denote the set of nodes where the i-th VNF of SFC k can be processed for an arbitrary feasible solution. If there is a node $v \in S$ such that $v \in V \setminus U$, then the inequality is clearly satisfied. Hence, suppose $S \subseteq U$. In this case, we have that $|S| \geq \eta(U, 1, A^k)$ from Remark 1 and thus the inequality is also verified. □

Notice that inequalities (12) might become quite loose when there exists a node $v \in V \setminus U$ that can process the i-th VNF of SFC k. For this reason, we next propose a lifted version of such inequalities. For this, let $\beta(v', V, B)$ denote the minimum number of backup VNF replicas that need to be assigned in parallel to v' within the node-set V so that a certain availability requirement B is reached. Remark that such number can be easily obtained by slightly modifying the computation of $\eta(V, 1, B)$. Indeed, it suffices to oblige node v' to be part of the VNF assignment in Algorithm 1. Moreover, let $\bar{V}(v)$ denote the subset of nodes in V that are at most as available as v, that is,

$$\bar{V}(v) = \Big\{u \in V : a(u) \leq a(v)\Big\}. \tag{13}$$

The *Lifted Node Cover* inequalities are defined as follows.

$$\sum_{v \in V \setminus U} c_v x_{vik} + \sum_{v \in U} x_{vik} \geq \eta(U, 1, A^k) \;\; \forall k \in K, i \in I^k, U \subseteq V, \tag{14}$$

where $c_v = \max\Big(\eta(U, 1, A^k) - \beta(v, U \cup \bar{V}(v), A^k), 1\Big)$.

Proposition 5.3. *Lifted Node Cover inequalities* (14) *are valid.*

Proof. Let S denote the set of nodes where the i-th VNF of SFC k can be processed for an arbitrary feasible solution. That is, $S = \{v \in V : x_{vik} = 1\}$. If $S \subseteq U$, then the inequality is satisfied since inequalities (12) are valid. Hence, let us focus on the case where $S \nsubseteq U$. Let v' denote the most available node in $S \setminus U$. By definition, $S \subseteq U \cup \bar{V}(v')$ and hence $|S \setminus v'| \geq \beta(v', U \cup \bar{V}(v'), A^k)$. Since $c_{v'} \geq \eta(U, 1, A^k) - \beta(v, U \cup U_v, A^k)$ and every other coefficient is at least 1, the inequality is verified. □

Chain Cover inequalities (11) deal with the non-linearity of the availability function arising from the chaining of VNFs. Node Cover inequalities (12) treat the availability non-linearity appearing rather from the parallel assignment of VNFs. These two ideas are next combined to form a single large family of valid inequalities. The Generalized Cover inequalities are defined as follows.

$$\sum_{i\in Q}\sum_{v\in V\setminus U}\eta(U,|Q|,A^k)x_{vik}+\sum_{i\in Q}\sum_{v\in U}x_{vik}\geq\eta(U,|Q|,A^k)$$
$$\forall k\in K, Q\subseteq I^k, U\subseteq V. \quad (15)$$

Proposition 5.4. *Generalized Cover inequalities* (15) *are valid.*

Proof. Let $\mathcal{S}=\{S_1,\ldots,S_{|I^k|}\}$ denote the VNF assignment of SFC k for an arbitrary feasible solution. That is, $S_i=\{v\in V: x_{vik}=1\}$, for any $i\in I^k$. If there exists a node $v\in S_i$, for any $i\in Q$, such that $v\in V\setminus U$, then the inequality is clearly satisfied. Hence, suppose $S_i\subseteq U$, for every $i\in Q$. In this case, at least $\eta(U,|Q|,A^k)$ VNFs must be assigned to SFC k and hence $\sum_{i\in Q}\sum_{v\in U}x_{vik}\geq\eta(U,|Q|,A^k)$. The inequality is thus verified. □

Next, consider the following *Section Failure* inequalities.

$$\sum_{v\in V}\Big(\log\left(1-a_v\right)\Big)x_{vik}\leq\log\left(1-A^k\right)\qquad\forall k\in K, i\in I^k. \quad (16)$$

Proposition 5.5. *The Section Failure inequalities* (16) *are valid.*

Proof. Let $\mathcal{S}=\{S_1,\ldots,S_{|I^k|}\}$ denote a feasible VNF assignment for an SFC $k\in K$, that is, $A(\mathcal{S})\geq A^k$. It follows directly from Remark 1 that $a(S_i)\geq A^k$ for any $i\in I^k$, *i.e.*,

$$\prod_{v\in S_i}\left(1-a_v\right)\leq 1-A^k\qquad\forall i\in I^k. \quad (17)$$

Notice that if inequalities (17) hold, then

$$\log\left(\prod_{v\in S_i}\left(1-a_v\right)\right)\leq\log\left(1-A^k\right)\qquad\forall i\in I^k,$$

also hold. Using the fundamental property of logarithms that states $\log(ab)=\log(a)+\log(b)$, such inequalities can be rewritten as

$$\sum_{v\in S_i}\Big(\log\left(1-a_v\right)\Big)\leq\log\left(1-A^k\right)\qquad\forall i\in I^k.$$

Since by definition $x_{vik}=1$ for $v\in S_i$ and $x_{vik}=0$ for $v\in V\setminus S_i$, inequalities (16) are hence valid. □

It is worth noting that if one searches for Cover inequalities (see [1]) associated with Section Failure inequalities (16), the inequalities obtained form a subset of the linearized availability constraints (10).

So far, all the introduced valid inequalities have dealt with the availability restrictions. We next propose a family of valid inequalities reinforcing the capacity constraints (6). The *Stronger Capacity* inequalities are defined as follows.

$$\sum_{k\in K, i\in I^k: f(i,k)=f}b_k r^f x_{vik}\leq C_v y_v^f\qquad\forall v\in V, f\in F. \quad (18)$$

Proposition 5.6. *Stronger Capacity inequalities* (18) *are valid.*

Proof. If a VNF is placed on node v, then the amount of resources it consumes must be at most the node's capacity. □

6 BRANCH-AND-CUT FRAMEWORK

We next describe the branch-and-cut framework we have developed based on the results obtained from Section 5. In this framework we consider the linearized formulation (3)-(6),(8)-(10) presented in Section 4 reinforced with Chain Cover inequalities (11), Lifted Node Cover inequalities (14), Section Failure inequalities (16) and Stronger Capacity inequalities (18).

As stated in Section 4, the separation problem for the linearized availability constraints (10) is solved exactly whenever an integer solution is found. This allows us to guarantee the feasibility of the solution provided by the end of the optimization procedure.

Since the Section Failure inequalities (16) and the Stronger Capacity inequalities (18) appear in polynomial numbers, storing them in a pool and checking, by enumeration, whether they all are satisfied remains an efficient way of handling them. For the Chain Cover inequalities (11) and Lifted Node Cover inequalities (14) we next focus on their separation problems.

Proposition 6.1. *The separation problem for the Chain Cover inequalities* (11) *can be solved in polynomial time.*

Proof. For a given SFC $k\in K$, the right-hand side of the inequality depends only on the cardinality of subset $Q\subseteq I^K$. Therefore, one can compute $\eta(V,|Q|,A^k)$, for $|Q|=1,\ldots,|I^k|$, in polynomial time. Additionally, for each chosen cardinality of Q, the left-hand side can be easily minimized by choosing the sections $i\in I^k$ with the smallest values of $\sum_{v\in V}x_{vik}$. □

Solving the separation problem for Lifted Node Cover inequalities (14) is less trivial since the value of the right-hand side depends not only on the cardinality of subset $U\subseteq V$ but also on its composition. For this reason, we consider the following sub-family of inequalities (14) that can be separated in polynomial time by simple enumeration:

$$\sum_{v\in V\setminus\bar{V}(u)}c_v x_{vik}+\sum_{v\in\bar{V}(u)}x_{vik}\geq\eta(\bar{V}(u),1,A^k), \quad (19)$$

for any $k\in K, i\in I^k, u\in V$, where $\bar{V}(v)$ is defined as in (13) and

$$c_v=\max\left(\eta(\bar{V}(u),1,A^k)-\beta(v,\bar{V}(v),A^k),1\right).$$

At each node of the branch-and-cut tree, the separation routines are called following a hierarchical order defined by their computational complexity – from the simplest to the hardest – and once a violated inequality is found, the node is re-optimized with the additional cut. All cuts are globally valid. A branching operation is performed once no separation routine can find a violated inequality.

7 COMPUTATIONAL RESULTS

In order to evaluate the efficiency of our approach, this section provides some preliminary results on the computational performances obtained over a small set of randomly generated instances. Two availability scenarios – denoted by *Constant* and *Variable* – were examined on a small network containing 15 nodes. In *Constant* case, all nodes are considered to have the same fixed availability which is set to 0.90. On *Variable*, each node has an availability taken at random between 0.90 and 0.99. For each availability scenario, five sets of $|K|\in\{10,20,30,40\}$ SFCs were randomly generated, where

Table 1: Computational results comparison

Instance		opt		time (s)		gap (%)	
$\|K\|$	Type	(I)	(R)	(I)	(R)	(I)	(R)
10	C	4/5	5/5	0.07	0.04	3.94	0
20	C	3/5	5/5	2.9	1.12	4.35	0
30	C	3/5	5/5	65.5	16.9	15.9	0
40	C	1/5	2/5	2236	540.6	9.48	1.35
10	V	5/5	5/5	3.3	0.2	0	0
20	V	4/5	5/5	95.7	65.8	0.72	0
30	V	0/5	2/5	-	1776	13.3	7.70
40	V	0/5	0/5	-	-	25.3	14.9

each SFC is required to visit between 1 and 5 VNFs chosen at random from a set of 8 VNF types. Moreover, each SFC has a required availability between 0.9999 and 0.999999, which is in accordance with specifications in [15].

Table 1 summarizes the computational results obtained using the initial linearized formulation (3)-(6), (8)-(10) – columns (I) – and the reinforced formulation featured in our branch-and-cut framework – columns (R). All our computational experiments were implemented in C++ and performed using the state-of-the-art MIP solver CPLEX 12.10 on a computer equipped with a 1.60 GHz Intel Core i5-8265U processor and 16 Gb RAM. A time limit of one hour was imposed in each run. For each instance size and type (C for Constant and V for Variable), the number of instances that could be solved to optimality within the time limit is displayed under column opt. Column time (s) provides the average time in seconds required to achieve optimality. Column gap (%) displays the average remaining gap for the instances that could not be solved within time limit.

For the Constant scenario, out of the 20 tested instances, only 11 could be solved to optimality within the time limit by the initial formulation. Our branch-and-cut approach allowed us to solve up to 17 instances to optimality in less than one hour. Indeed, up to 30 demands, all instances could be optimally solved. Moreover, the remaining instances were left with a relatively small gap. For the Variable scenario, the performance of the initial formulation was even worse. Only 9 out of the 20 tested instances could be solved to optimality and none of them had more than 20 demands. With our branch-and-cut framework, 3 more instances could be optimally solved. Besides, the average remaining gaps for the unsolved instances were considerably reduced.

The gain of performance observed with our branch-and-cut approach is largely due to its capability of rejecting unfeasible solutions earlier in the optimization. Indeed, the average number of linearized availability constraints added by the exact separation problem (and hence later in the optimization) dropped from 1440 to 13 in the Constant case and from 1608 to 748 in the Variable case. In addition, considering only the instances that could be solved with both approaches, the branch-and-cut framework was on average 2.92 times faster, and the average number of nodes that were required to be explored in the enumeration tree to prove optimality went down from 148.3 thousand to 38 thousand in the Constant case and from 55.6 thousand to 7.5 thousand in the Variable case.

8 FINAL REMARKS AND NEXT STEPS

In this work, we have studied the problem of optimally placing VNFs throughout a given network so that a set of SFCs can achieve their required availability. The non-linearity inherent to the definition of the end-to-end availability of an SFC represents a major challenge in such problem. We have proposed an original ILP formulation that solves the addressed optimization problem. Such ILP was then reinforced through the investigation of valid inequalities and preliminary computational results testify in favor of their efficiency. Even with the improvements proposed in this paper, our approach can only solve instances of limited size. With this in mind, the use of heuristics can help improve the primal bounds faster and hence speed up the convergence towards the optimal solution. The next steps may also include the consideration of SFCs' routing aspect explicitly into the formulation so that maximum SFC delay constraints and link capacity constraints can be imposed in order to treat a more generalized problem.

ACKNOWLEDGMENTS

This work is supported by the french Agence Nationale de la Recherche (ANR), Project MAESTRO-5G ANR-18-CE25-0012.

REFERENCES

[1] Egon Balas. 1975. Facets of the knapsack polytope. *Mathematical programming* 8, 1 (1975), 146–164.
[2] Eric Bauer and Randee Adams. 2012. *Reliability and availability of cloud computing*. John Wiley & Sons.
[3] Kurt M Bretthauer and Bala Shetty. 2002. The nonlinear knapsack problem–algorithms and applications. *European Journal of Operational Research* 138, 3 (2002), 459–472.
[4] Yannick Carlinet, Nancy Perrot, and Anderson Alves-Tzitas. 2019. Minimum-Cost Virtual Network Function Resilience. In *INOC 2019*.
[5] Marco Casazza, Pierre Fouilhoux, Mathieu Bouet, and Stefano Secci. 2017. Securing virtual network function placement with high availability guarantees. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 1–9.
[6] Isabel Correia, Luís Gouveia, and Francisco Saldanha-da Gama. 2008. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research* 35, 6 (2008), 2103–2113.
[7] Teodor Gabriel Crainic, Guido Perboli, Walter Rei, and Roberto Tadei. 2011. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research* 38, 11 (2011), 1474–1482.
[8] Weiran Ding, Hongfang Yu, and Shouxi Luo. 2017. Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme. In *2017 IEEE international conference on communications (ICC)*. IEEE, 1–6.
[9] Jingyuan Fan, Zilong Ye, Chaowen Guan, Xiujiao Gao, Kui Ren, and Chunming Qiao. 2015. GREP: Guaranteeing reliability with enhanced protection in NFV. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. 13–18.
[10] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. 2015. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine* 53, 2 (2015), 90–97.
[11] Bo Han, Vijay Gopalakrishnan, Gnanavelkandan Kathirvel, and Aman Shaikh. 2017. On the resiliency of virtual network functions. *IEEE Communications Magazine* 55, 7 (2017), 152–157.
[12] N Isg. 2016. Network functions virtualisation (nfv); reliability; report on models and features for end-to-end reliability. *ETSI GS NFV-REL* 1 (2016), v1.
[13] Purnima Murali Mohan and Mohan Gurusamy. 2019. Resilient VNF placement for service chain embedding in diversified 5G network slices. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.
[14] Justine Sherry, Peter Xiang Gao, Soumya Basu, Aurojit Panda, Arvind Krishnamurthy, Christian Maciocco, Maziar Manesh, João Martins, Sylvia Ratnasamy, Luigi Rizzo, et al. 2015. Rollback-recovery for middleboxes. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 227–240.
[15] 3GPP TS 22.261 version 15.7.0 Release 15. 2019. Service requirements for next generation new services and markets. (2019).
[16] Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang. 2019. Towards carrier-grade service provisioning in nfv. In *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 130–137.

Max-Min Optimization of Controller Placements vs. Min-Max Optimization of Attacks on Nodes in Service Networks

Artur Tomaszewski
Warsaw University of Technology
Warsaw, Poland
a.tomaszewski@tele.pw.edu.pl

Michał Pióro
Warsaw University of Technology
Warsaw, Poland
m.pioro@tele.pw.edu.pl

Mariusz Mycek
Warsaw University of Technology
Warsaw, Poland
m.mycek@tele.pw.edu.pl

ABSTRACT

The paper deals with two complementary optimization problems related to the resilience of communication networks against targeted node attacks, where the proper functioning of the network requires that the nodes are connected to the so called controllers that are placed in selected node locations – a node that looses such a connection in the result of an attack is considered lost. These two problems can be used to optimize (maximize in the case of a network operator and minimize in the case of an attacker) the resilience in question when the interaction between these two parties is considered within the framework of game theory. The presented formulations and their solution algorithms are original. The efficiency of the algorithms is illustrated for a medium size network by means of a numerical example.

1 INTRODUCTION

We consider a network that offers some kind of service in a given set of network locations. Each location houses a service node that actually provides the service, and might also house a controller node (controller in short). The service node needs a controller to operate; for that it may use either the local controller that is placed in the same location or, if the location does not house a controller, a remote controller in some other location. In the latter case the service node must communicate with the controller node using some network path. That is why the locations are interconnected with transport links.

Such a setting is directly applicable, in particular, to software defined networks (SDN) [2, 3]. Note however that it may also arise in a number of other contexts. In the ICT area it may also apply to content delivery networks (CDN): delivery nodes, which deliver content to the user, correspond to the service nodes, and origin nodes, which are the primary sources of the original content, correspond to the control nodes (storage node, which are responsible for storing copies of original data, may correspond either to the service nodes or to the control nodes). One may find applications of the considered model in other areas as well, would it be utilities, manufacturing, logistics, sales, or medicine. In those areas the service nodes and the control nodes might correspond, respectively, to power dispatch stations and power plants, factories and transportation hubs, dispatch centers or warehouses and factories, sale points or supermarkets and warehouses, clinics or testing points and medical laboratories, etc. Depending on the context, those nodes are interconnected with different kinds of utility and transportation networks, and their inaccessibility may result not only from attacks, but, e.g., from technical malfunctioning and natural disasters.

We aim at protecting the network against targeted node attacks. The attack targets a set of selected locations making both service nodes and controller nodes (if any) at those locations unavailable. Moreover, the attack makes unavailable the transport links that are terminated at the attacked locations, potentially disconnecting the network graph into a number of (connected) components. After the attack, the service node will still provide service (and will be called a surviving (service) node) only if its location has not been attacked and the component it belongs to still contains at least one location with a controller node.

In the paper we consider two complementary optimization problems for the so described network layout. The first of them consists in finding a placement of a given number of controllers that maximizes the number of nodes that survive the worst case attack from a given, in general non-compact, list of attacks. The complementary problem, in turn, is to find an attack targeted at a given number of locations that minimizes the number of surviving nodes for any placement from a given list (also in general non-compact) of controller placements. We note here that although related problems have been widely researched in the literature (mainly in the context of SDN, see [1, 5–7] and references therein), the two problem formulations and algorithms for solving them presented below are original.

2 NOTATION AND PROBLEM DESCRIPTION

2.1 Notation

First, we model the service network by means of a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of nodes $\mathcal{V} = \{1, 2, \ldots, V\}$ represents network locations, and $\mathcal{E}$ ($\mathcal{E} \subseteq \{\mathcal{X} \subseteq \mathcal{V} : |\mathcal{X}| = 2\}$) is the set of transport links represented by unordered nodes pairs that interconnect the locations; for each $e \in \mathcal{E}$, let $\alpha(e), \beta(e) \in \mathcal{V}$ denote the end nodes of link e, and for each $v \in \mathcal{V}$, let $\delta(v) = \{e \in \mathcal{E} : v \in \{\alpha(e), \beta(e)\}\}$ denote the set of links incident with node v.

Next, we assume that the network is equipped with controllers and the set of (allowable) controller placements is denoted by $\mathcal{S}$. Each placement $s \in \mathcal{S}$ is characterized by the set $\mathcal{V}(s)$ ($\mathcal{V}(s) \subseteq \mathcal{V}$) where the controllers are actually placed (apart form the service nodes). A typical example of the set of placements $\mathcal{S}$ is the set of all M-node placements (where $0 < M \leq V$), i.e., the set of all placements s with $|\mathcal{V}(s)| = M$; such a set will be denoted by $\mathcal{S}(M)$.

Then, we consider a set $\mathcal{A}$ of attacks targeted at networks nodes. Each attack $a \in \mathcal{A}$, is characterized by the set of the attacked locations $\mathcal{V}(a)$ ($\mathcal{V}(a) \subseteq \mathcal{V}$), which defines the set $C(a)$ of (non-empty) connected components into which the network graph $\mathcal{G}$

Table 1: Summary of notation.

$\mathcal{V}, \mathcal{E}$	sets of nodes and links ($V = \|\mathcal{V}\|, E = \|\mathcal{E}\|$)
$\alpha(e), \beta(e)$	end nodes of link $e \in \mathcal{E}$
$\delta(v)$	set of links incident with node $v \in \mathcal{V}$
$\mathcal{S}$	set of allowable controller placements
$\mathcal{V}(s)$	set of controller nodes locations in placement $s \in \mathcal{S}$
$\mathcal{A}$	set of expected attacks
$\mathcal{V}(a)$	set of nodes affected by attack $a \in \mathcal{A}$
$\mathcal{C}(a)$	set (family) of components induced by attack $a \in \mathcal{A}$
$\mathcal{V}(c)$	set of nodes of component $c \in \mathcal{C}(a)$
$\mathcal{S}(M)$	set of all placements composed of M controllers
$\mathcal{A}(K)$	set of all K-node attacks
$V(s, a)$	number of nodes that survive attack $a \in \mathcal{A}$ when placement $s \in \mathcal{S}$ is assumed
$a(s)$	the worst attack in $\mathcal{A}$ for a given placement $s \in \mathcal{S}$
$s(a)$	the best placement in $\mathcal{S}$ for a given attack $a \in \mathcal{A}$
$\mathbb{R}^+$	set of nonnegative real numbers

is split as the result of attack a. For each component $c \in \mathcal{C}(a)$, $\mathcal{V}(c)$ will denote the set of its nodes. A typical example of the set of attacks $\mathcal{A}$ is the set of all K-node attacks, i.e., the set of all attacks a with $|\mathcal{V}(a)| = K$, for a given integer parameter K (where $0 < K < V$); such a set will be denoted by $\mathcal{A}(K)$.

Finally, we assume that as a result of an attack a in each (directly attacked) location in $\mathcal{V}(a)$ its service node and the controller (if any) become out of service. Moreover, all service nodes in those components in $\mathcal{C}(a)$ that do not contain any controller also stop working. In effect, the service nodes that are still operational after the attack (called the *surviving nodes*) are precisely those nodes that belong to the components in $\mathcal{C}(a)$ that contain a controller.

The basic *resilience (to attack) measure* considered in this paper is the number of nodes, denoted by $V(s, a)$ ($s \in \mathcal{S}, a \in \mathcal{A}$), that survive a given attack a in the network equipped with controllers deployed according to placement s. For such a measure, we can introduce the notions of the worst attack and the best controller placement. Taking the operator's point of view, the *worst attack* with respect to a given placement $s \in \mathcal{S}$ (denoted by $a(s)$) is defined as any attack a in $\mathcal{A}$ that minimizes the number of surviving nodes $V(s, a)$, i.e., $V(s, a(s)) = \min_{a \in \mathcal{A}} V(s, a)$. Symmetrically, the *best placement* with respect to a given attack $a \in \mathcal{A}$ (denoted by $s(a)$) is defined as any controller placement s in $\mathcal{S}$ that maximizes the value of $V(s, a)$, i.e., $V(s(a), a) = \max_{s \in \mathcal{S}} V(s, a)$. (Certainly, there can be multiple worst attacks and multiple best placements.)

2.2 Problem description

Since the network operator is interested in maximizing the value of $V(s, a)$ while the attacker seeks to minimize it, both sides need to consider some kind of optimization approaches for finding controller placements (the operator) and for constructing attacks (the attacker). In this paper we introduce a mathematical model aimed at solving optimization problems related to these issues.

The optimization problems we consider stem from the assumption that the set of possible controller placements $\mathcal{S}$ and the set of possible attacks $\mathcal{A}$ are known to both the operator and the attacker, and each of them is trying to find a solution that is most effective in the case of the worst attacks (the operator) and the best placements (the attacker). Hence, it is natural to consider the following two problems.

Controller Placement Optimization Problem (CPOP): Find a placement s^* whose resilience measure observed for its worst attack, i.e., $V(s^*, a(s^*))$, is the maximum over set $\mathcal{S}$:

$$V(s^*, a(s^*)) = \max_{s \in \mathcal{S}} V(s, a(s)) = \max_{s \in \mathcal{S}} \min_{a \in \mathcal{A}} V(s, a). \quad (1)$$

Each placement s^* that solves problem (1) will be called the best placement for a given set of attacks $\mathcal{A}$. Clearly, such a placement s^* guarantees that the number of surviving nodes is equal at least to Y^* for any attack in $\mathcal{A}$, where Y^* is the maximum number with this property.

Node Attack Optimization Problem (NAOP): Find an attack a^* whose resilience measure observed for its best placement, i.e., $V(s(a^*), a^*)$, is the minimum over set $\mathcal{A}$:

$$V(s(a^*), a^*) = \min_{a \in \mathcal{A}} V(s(a), a) = \min_{a \in \mathcal{A}} \max_{s \in \mathcal{S}} V(s, a). \quad (2)$$

Each attack a^* that solves problem (2) will be called the worst attack for a given set of placements $\mathcal{S}$. Thus, attack a^* guarantees that the number of surviving nodes is equal at most to Z^* for any placement in $\mathcal{S}$, where Z^* is the minimum number with this property.

3 OPTIMIZATION PROBLEMS

In this section we will present integer programming (IP) formulations of the two basic optimization problems (CPOP and NAOP) described in Section 2.2, with an intention to be able to solve them using commercial IP solvers.

3.1 Max-min controller placement optimization problem (CPOP)

In the formulation of CPOP presented below we assume that $\mathcal{S} = \mathcal{S}(M)$ (i.e., we consider the set of all M-node placements) and $\mathcal{A}$ is an arbitrary set of attacks. This means that we consider the problem of finding an M-node controller placement that maximizes the number of nodes surviving its worst attack from set $\mathcal{A}$.

Let s_v ($v \in \mathcal{V}$) be a binary variable that equals 1 if, and only if, a network controller is placed at location v. (Each vector $s = (s_v)_{v \in \mathcal{V}}$ specifies placement s with $\mathcal{V}(s) = \{v \in \mathcal{V} : s_v = 1\}$.) And for all $a \in \mathcal{A}$, $v \in \mathcal{V}$, let y_v^a be a binary variable that equals 1 if, and only if, service node at location v survives attack a. The formulation (abbreviated by $\mathbb{P}[M, \mathcal{A}]$) is as follows:

$$\mathbb{P}[M, \mathcal{A}] : \max \; Y \quad (3a)$$

$$\sum_{v \in \mathcal{V}} s_v = M \quad (3b)$$

$$y_v^a = 0 \qquad a \in \mathcal{A},\ v \in \mathcal{V}(a) \quad (3c)$$

$$\sum_{v \in \mathcal{V}(c)} y_v^a \le |\mathcal{V}(c)| \sum_{v \in \mathcal{V}(c)} s_v \qquad a \in \mathcal{A},\ c \in \mathcal{C}(a) \quad (3d)$$

$$Y \le \sum_{v \in \mathcal{V}} y_v^a \qquad a \in \mathcal{A} \quad (3e)$$

$$s_v, y_v^a \in \{0, 1\} \qquad a \in \mathcal{A},\ v \in \mathcal{V} \quad (3f)$$

$$Y \in \mathbb{R}^+. \quad (3g)$$

Constraint (3b) sets the number of deployed controllers to M. Then, constraints (3c) explicitly force variables y_v^a with node v directly destroyed by attack a to be equal to 0 (these nodes do not survive after attack a wherever the controllers are placed).

Constraints (3d), in turn, imply that when a component c induced by attack a does not contain any controller ($\sum_{v \in \mathcal{V}(c)} s_v = 0$) then

all nodes in c do not survive and hence the corresponding variables y_v^a are explicitly set to 0 (because $\sum_{v\in\mathcal{V}(c)} y_v^a$ is forced to be equal to 0). Otherwise, when c contains at least one controller, then the right-hand side of (3d) is greater than or equal to the number of elements in component c and hence it allows all y_v^a with v in c to be greater than 0 (but not greater than 1 since these are binary variables). Now we note that for any fixed vector of controller placement variables $s = (s_v)_{v\in\mathcal{V}}$, optimization objective (3a) and constraints (3e) will force the value of variable Y to be equal to the actual number of surviving nodes after at least one attack a for which this number is minimal over $\mathcal{A}$, because for such an attack the values of those variables y_v^a that are not explicitly set to 0 will reach their maximum, i.e., 1.

Hence, when variables s are optimized, the final value Y^* of the objective function will be equal to the maximum, over all placements in $\mathcal{S}(M)$, of the number of surviving nodes (i.e., the total number of nodes appearing in the components containing one or more controllers) when the worst attack is assumed for each of the considered placements.

In summary, the maximum value Y^* of objective (3a) is equal to

$$V(s^*, a(s^*)) = \max_{s\in\mathcal{S}(M)} \min_{a\in\mathcal{A}} V(s,a), \tag{4}$$

where s^* denotes an arbitrary optimal placement resulting from (3); this means that any optimal s^* is one of the best placements in $\mathcal{S}(M)$ with respect to the set of attacks $\mathcal{A}$.

3.2 Min-max node attack optimization problem (NAOP)

In the formulation of NAOP presented below we assume that $\mathcal{A} = \mathcal{A}(K)$ (i.e., we consider the set of all K-node attacks) and $\mathcal{S}$ is an arbitrary set of placements. This means that we consider the problem of finding a K-node attack that minimizes the number of surviving nodes when the best placement in the set $\mathcal{S}$ with respect to this attack is considered.

In the formulation (abbreviated by $\mathbb{A}[K,\mathcal{S}]$), for each $v\in\mathcal{V}$, a_v is a binary variable equal to 1 if, and only if, node v is attacked. (Each vector $a = (a_v)_{v\in\mathcal{V}}$ specifies attack a with $\mathcal{V}(a) = \{v\in\mathcal{V} : a_v = 1\}$.) Next, for each $e\in\mathcal{E}$, t_e is a binary variable that equals 1 if, and only if, link e is not available as a result of the attack (i.e., one or both of its end-nodes are attacked). Finally, for each $s\in\mathcal{S}$ and $v\in\mathcal{V}$, z_v^s is a binary variable equal to 1 if, and only if, node v survives the constructed attack when controller placement s is assumed. The formulation uses the fact that if after the attack a node can still provide service, then every node in its neighborhood (i.e., in a location interconnected with it by a transport link) can also provide service unless its location was directly attacked, and is as follows:

$$\mathbb{A}[K,\mathcal{S}]: \min\ Z \tag{5a}$$
$$\textstyle\sum_{v\in\mathcal{V}} a_v = K \tag{5b}$$
$$t_e \geq a_v \qquad v\in\mathcal{V}, e\in\delta(v) \tag{5c}$$
$$t_e \leq a_{\alpha(e)} + a_{\beta(e)} \qquad e\in\mathcal{E} \tag{5d}$$
$$z_v^s \leq 1 - a_v \qquad s\in\mathcal{S}, v\in\mathcal{V} \tag{5e}$$
$$z_v^s \geq 1 - a_v \qquad s\in\mathcal{S}, v\in\mathcal{V}(s) \tag{5f}$$
$$z_{\alpha(e)}^s \geq z_{\beta(e)}^s - t_e \qquad s\in\mathcal{S}, e\in\mathcal{E} \tag{5g}$$
$$z_{\beta(e)}^s \geq z_{\alpha(e)}^s - t_e \qquad s\in\mathcal{S}, e\in\mathcal{E} \tag{5h}$$
$$Z \geq \textstyle\sum_{v\in\mathcal{V}} z_v^s \qquad s\in\mathcal{S} \tag{5i}$$
$$a_v, t_e, z_v^s \in \{0,1\} \qquad s\in\mathcal{S}, v\in\mathcal{V}, e\in\mathcal{E} \tag{5j}$$
$$Z\in\mathbb{R}^+. \tag{5k}$$

Constraint (5b) sets the number of attacked nodes to K. Then, constraints (5c) and (5d) force, as required, the value of each binary variable t_e to be equal to 0 (which means that link e is available after attack a) if, and only if, both end nodes of e are not directly attacked. Next, constraints (5e) set z_v^s to 0 (which means that node v does not survive attack a if node v is attacked directly, whatever placement is selected. Constraints (5f), in turn, set z_v^s to 1 (which means that node v survives attack a when placement s is assumed) if node v is not directly attacked and its location contains a controller.

The next two sets of constraints, (5g) and (5h), make sure that if link e is available after attack a (i.e., when $t_e = 0$), then its end nodes either simultaneously survive or are simultaneously out of service. This property assures that all nodes in any component $c\in C(a)$ (i.e., in any component c resulting from the constructed attack a) have the same values of z_v^s (for any fixed placement s):

$$z_v^s = z_w^s, \quad v,w\in c, c\in C(a), s\in\mathcal{S}. \tag{6}$$

Moreover, for any given placement $s\in\mathcal{S}$, if a location $v\in\mathcal{V}(c)$ contains a controller then the inequality in (5f) sets z_v^s to 1 since, by definition, this location is not attacked. In this case the equalities in (6) imply that the values of z_v^s are set to 1 for all $v\in\mathcal{V}(c)$, i.e., all nodes in c survive the attack, as required. In effect, for any $s\in\mathcal{S}$, all these nodes are counted in the summation on the right hand side of inequality (5i).

On the other hand, when component $c\in C(a)$ does not contain any controller from placement s, then in the feasible solutions of the considered formulation all values of z_v^s, $v\in\mathcal{V}(c)$, are simultaneously equal either to 0 or to 1. This, however, is not an issue. To see this consider an optimal solution of (5) and let $\mathcal{S}^*$ denote the set of all placements in $\mathcal{S}$ for which the inequalities in (5i) are binding, i.e., $Z^* = \sum_{v\in\mathcal{V}} z_v^s$ if, and only if, $s\in\mathcal{S}^*$, where Z^* is the optimal value of Z. Denoting the optimized attack by a^*, we can rewrite the equalities in question as follows

$$Z^* = \textstyle\sum_{v\in\mathcal{V}} z_v^s = \sum_{c\in C(a^*)} \sum_{v\in\mathcal{V}(c)} z_v^s, \quad s\in\mathcal{S}^* \tag{7}$$

(because $\mathcal{V} = \mathcal{V}(a^*)\cup\bigcup_{c\in C(a^*)}\mathcal{V}(c)$ and $z_v^s = 0$ for $v\in\mathcal{V}(a^*)$, i.e., when $a_v = 1$). This shows that for each $s\in\mathcal{S}^*$, the sum $\sum_{v\in\mathcal{V}(c)} z_v^s$ must be equal to 0 for all components $c\in C(a^*)$ that do not contain a controller from placement s. Otherwise, Z^* would not be minimal since if any of such sums were greater than 0 then setting it to 0, which is allowed by the constraints, would result in a feasible solution with $Z < Z^*$. (Note that this argumentation reveals that in fact constraints (5e) are redundant.)

In summary, the minimum value Z^* of objective (5a) is equal to

$$V(s(a^*), a^*) = \min_{a\in\mathcal{A}(K)} \max_{s\in\mathcal{S}} V(s,a), \tag{8}$$

where a^* denotes an arbitrary optimal attack resulting from (5), that is one of the worst attacks in $\mathcal{A}(K)$ for the assumed set of placements $\mathcal{S}$.

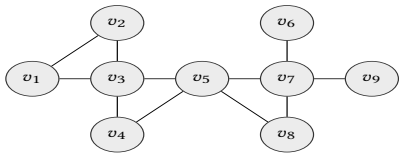


Figure 1: Sample 9-node and 11-link network.

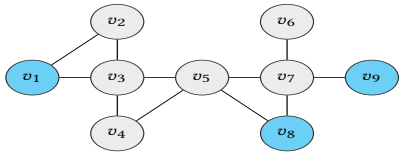


Figure 2: One of 32 best placements (with $\mathcal{V}(s) = \{1, 8, 9\}$).

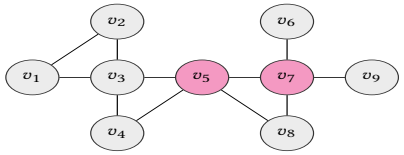


Figure 3: One of 2 worst attacks (with $\mathcal{V}(a) = \{5, 7\}$).

3.3 An example

We will now characterize solutions of formulations $\mathbb{P}[M, \mathcal{A}(K)]$ and $\mathbb{A}[K, \mathcal{S}(M)]$ for the network depicted in Figure 1 and the 3-node placements ($M = 3$) and the 2-node attacks ($K = 2$).

Then, in the set $\mathcal{A}(2)$ there are 2 worst attacks (out of all $\binom{9}{2} = 36$ attacks in $\mathcal{A}(2)$) with respect to $\mathcal{S}(3)$. Each of them guarantees that at most 6 nodes will survive whatever 3-node placement is selected. These two attacks, a^1 and a^2, have the sets of attacked nodes equal to $\mathcal{V}(a^1) = \{3, 7\}$ and $\mathcal{V}(a^2) = \{5, 7\}$; the second of them is shown in Figure 3.

It turns out that in the set $\mathcal{S}(3)$ there are 32 best placements (out of all $\binom{9}{3} = 84$ placements in $\mathcal{S}(3)$) with respect to the set $\mathcal{A}(2)$ of all 2-node attacks. Each of them guarantees that at least 4 nodes will survive whatever 2-node attack is selected. We do not list all placements because there are too many of them; instead we show one of these placements (with the set of controller nodes $\{1, 8, 9\}$) in Figure 2.

Now let us assume that the attacker decides to use one of the two worst attacks to attack the network. Knowing that, the operator will select one of the best placements with respect to the set $\mathcal{A} = \{a^1, a^2\}$ and deploy it. Actually, there are four such best placements, s^1, s^2, s^3, s^4 (with $\mathcal{V}(s^1) = \{1, 8, 9\}, \mathcal{V}(s^2) = \{2, 8, 9\}, \mathcal{V}(s^3) = \{1, 6, 8\}, \mathcal{V}(s^4) = \{2, 6, 8\}$, all of them belonging to the set of 32 best placements with respect to $\mathcal{A}(2)$), and each of them guarantees that at least 6 nodes will survive any of the two considered attacks. Note that this value is substantially better than 4, i.e., the number of surviving nodes guaranteed by the best placement with respect to the full set of the 2-node attacks. Moreover, there is no single 2-node attack ensuring that less than 6 nodes will survive if any of the placements in the set $\mathcal{S} = \{s^1, s^2, s^3, s^4\}$ is deployed.

4 SOLVING NON-COMPACT VERSIONS OF CPOP AND NAOP

Note that both formulations $\mathbb{P}[M, \mathcal{A}]$ and $\mathbb{A}[K, \mathcal{S}]$ are in general non-compact as the number of attacks in the set $\mathcal{A}$ appearing in the first formulation, and the number of placements in the set $\mathcal{S}$ appearing in the second formulation may grow exponentially with the number of nodes V. When this is the case, CPOP can be approached using an attack generation procedure (provided $\mathcal{A}$ can be characterized in a tractable way) while NAOP can be approached using a controller placement generation procedure (provided $\mathcal{S}$ can be characterized in a tractable way).

Such non-compactness can appear for $\mathcal{A} = \mathcal{A}(K)$ (for example when $V = 2K$), and for $\mathcal{S} = \mathcal{S}(M)$ (for example when $V = 2M$). Therefore, below we present an algorithm for solving formulation $\mathbb{P}[M, \mathcal{A}(K)]$ (Section 4.1) and a similar algorithm for solving formulation $\mathbb{A}[K, \mathcal{S}(M)]$ (Section 4.2).

4.1 Solving $\mathbb{P}[M, \mathcal{A}(K)]$ by attack generation

In the algorithm, the list of attacks $\mathcal{A}$ is iteratively extended by means of solving consecutive formulations of NAOP of the form $\mathbb{A}[K, \{s^*\}]$ for a particular placement s^* (NAOP is called the *pricing problem* in this context), and at each iteration, for the current list $\mathcal{A}$, an optimal placement s^* is found by means of solving formulation $\mathbb{P}[M, \mathcal{A}]$ (CPOP is called the *master problem* in this context). In effect, in each iteration we find out whether there exists an attack $a \in \mathcal{A}(K) \setminus \mathcal{A}$ such that when a is added to the current list of attacks $\mathcal{A}$, then the number of surviving nodes after attack a is smaller than the maximum number of surviving nodes guaranteed for any attack in $\mathcal{A}$ (achieved for the current optimal placement). If this is the case, we re-optimize s^* and continue.

A1: Algorithm for controller placement optimization by means of attack generation

Step 0: Generate a random M-node controller placement s^*; $\mathcal{A} := \emptyset$, $Y^* := V$.
(Comment: for $\mathcal{A} = \emptyset$ any placement in $\mathcal{S}(M)$ solves $\mathcal{P}[M, \mathcal{A}]$ giving $Y^* = V$.)
Step 1: Solve $\mathbb{A}[K, \{s^*\}]$ to get the worst attack a^* (assuring Z^* surviving nodes) with respect to placement s^*. If $Z^* \geq Y^*$ then go to Step 3.
Step 2: $\mathcal{A} := \mathcal{A} \cup \{a^*\}$. Solve $\mathbb{P}[M, \mathcal{A}]$ to get the best placement s^* (assuring at least Y^* surviving nodes) with respect to set $\mathcal{A}$. Go to Step 1.
(Comment: Y^* is equal to
$V(s^*, a(s^*)) = \max_{s \in \mathcal{S}(M)} \min_{a \in \mathcal{A}} V(s, a)$.)
Step 3: Stop: current placement s^* is an optimal solution of $\mathbb{P}[M, \mathcal{A}(K)]$, that is
$Y^* = V(s^*, a(s^*)) = \max_{s \in \mathcal{S}(M)} \min_{a \in \mathcal{A}(K)} V(s, a)$.

4.2 Solving $\mathbb{A}[K, \mathcal{S}(M)]$ by controller placement generation

In the algorithm, the list of placements $\mathcal{S}$ is iteratively extended by means of solving consecutive formulations of CPOP of the form $\mathbb{P}[M, \{a^*\}]$ for a particular attack a^* (CPOP is called the *pricing*

problem in this context), and in each iteration, for the current list $\mathcal{S}$, an optimal attack a^* is found by means of solving formulation $\mathbb{A}[K, \mathcal{S}]$ (NAOP is called the *master problem* in this context). In effect, in each iteration we find out whether there exists a placement $s \in \mathcal{S}(M) \setminus \mathcal{S}$ such that when s is added to the current list of placements $\mathcal{S}$, then the number of surviving nodes for s is greater than the minimum number of surviving nodes guaranteed for any placement in $\mathcal{S}$ (achieved for the current optimal attack). If this is the case, we re-optimize a^* and continue.

A2: Algorithm for attack optimization by means of controller placement generation

Step 0: Generate a random K-node attack a^*; $\mathcal{S} := \emptyset$, $Z^* := 0$ (Comment: for $\mathcal{S} = \emptyset$ any attack in $\mathcal{A}(K)$ solves $\mathbb{A}[K, \mathcal{S}]$ giving $Z^* = 0$.)

Step 1: Solve $\mathbb{P}[M, \{a^*\}]$ to get the best placement s^* (assuring Y^* surviving nodes) with respect to attack a^*. If $Y^* \leq Z^*$ then go to Step 3.

Step 2: $\mathcal{S} := \mathcal{S} \cup \{s^*\}$. Solve $\mathbb{A}[K, \mathcal{S}]$ to get the worst attack a^* (assuring at most Z^* surviving nodes) with respect to set $\mathcal{S}$. Go to Step 1. (Comment: Z^* is equal to $V(s(a^*), a^*) = \min_{a \in \mathcal{A}(K)} \max_{s \in \mathcal{S}} V(s, a)$.)

Step 3: Stop: current attack a^* is an optimal solution of $\mathbb{A}(\mathcal{S}(M))$, that is $Z^* = V(s(a^*), a^*) = \min_{a \in \mathcal{A}(K)} \max_{s \in \mathcal{S}(M)} V(s, a)$.

5 NUMERICAL EXPERIMENTS

Below we will illustrate the performance of the two algorithms introduced in the previous section. For this purpose we expressed the formulated problems and the algorithms as models and procedures in the AMPL language. We ran the computations on a standard laptop using the AMPL runtime and the CPLEX MIP solver. In our experiment we set the number of controller nodes M to 6 and the number of attacked locations K to 4.

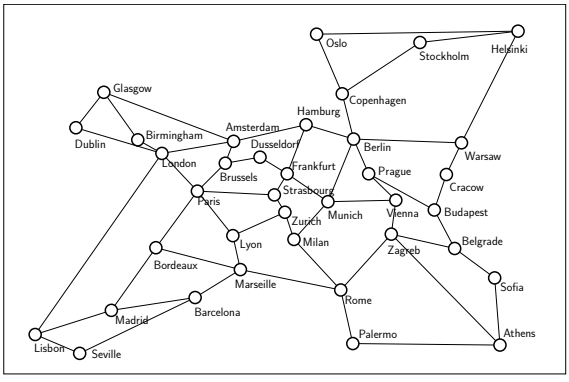


Figure 4: The *cost266* network instance.

The results of applying algorithm A1 to CPOP are summarized in Table 2. In the second row, a medium size network instance *cost66* available in SNDlib [4], whose graph is composed of $V = 37$ nodes and $E = 57$ links, is considered. The optimal objective function value $Y^* = 29$ shows that the optimal placement is capable of assuring that at least 29 nodes out of $V = 37$ nodes will survive any 4-node attack. The optimal solution of $\mathbb{P}[6, \mathcal{A}(4)]$ was reached in 29 iterations of A1, which took 16 seconds of the total computation time on a standard laptop. Most of this time (15 seconds) was spent in Step 1 on solving NAOP. It is remarkable that while the total number of different 4-node attacks equals 66, 045, we needed to generate only 29 of them to get the optimal solution of the considered problem.

The third row of Table 2 shows analogous results for *coronet conus* [8], a network instance substantially larger than *cost66*. This time the optimal placement protects at least $Y^* = 64$ nodes out of 99 nodes for any 4-node attack, and only 72 attacks (out of 3, 764, 376 possible 4-node attacks) need to be generated (which takes only 63 seconds).

Table 2: Results of optimal controller placement.

V	E	M	K	Y^*	$\lvert\mathcal{A}\rvert$	T($\mathbb{P}[6, \mathcal{A}]$)	T($\mathbb{A}[4, \{s^*\}]$)
37	57	6	4	29	29	1 sec.	15 sec.
75	99	6	4	64	72	3 sec.	63 sec.

The results of applying algorithm A2 to NAOP for *cost266* are summarized in Table 3. The optimal objective function value $Z^* = 33$ indicates that the optimal attack is capable of guaranteeing that (only) 4 nodes out of $V = 37$ nodes will be damaged if any of the 6-node controller placements can be deployed. The optimal solution of $\mathbb{A}[4, \mathcal{S}(6)]$ was reached in 40 iterations of A2, which took 1102 seconds in total, and, similarly as for A1, virtually the entire computation time was spent on solving the NAOP problem (this time in Step 2). Again, it is worth noticing that, while the number of different 6-node placements equals 2, 324, 784, we needed to consider only 40 of them to get the optimal solution of $\mathbb{A}[4, \mathcal{S}(6)]$. Table 3 shows no results for *coronet conus* because for this network it took too much computational time to run A2 on the laptop.

Table 3: Results of attack optimization.

V	E	M	K	Z^*	$\lvert\mathcal{S}\rvert$	T($\mathbb{P}[6, \{a^*\}]$)	T($\mathbb{A}[4, \mathcal{S}]$)
37	57	6	4	33	40	1 sec.	1101 sec.

In conclusion, A1 solves the controller placement problem CPOP very quickly, much faster than A2 solves the node attack optimization problem NAOP. So in the case of large networks the efficiency of A2 (which is determined by the efficiency of solving formulation $[K, \mathcal{S}]$) needs to be improved. Fortunately, the limited number of iterations required by A2 will help to achieve this goal.

6 AN ALTERNATIVE RESILIENCE MEASURE

The resilience (to attacks) measure assumed in the previous sections expresses the number of nodes surviving a given attack a. In this section we consider another important measure of this kind, namely the number of surviving (unordered) node-pairs $\{v, w\}$, i.e., the pairs for which both nodes belong to the same component $c \in C(a)$ and this component contains a controller. Note that such a measure is able to account for the traffic relations affected by an attack.

In order to extend the introduced optimization model to the new measure, we need to specify formulations for the counterparts of the two basic problems presented in Section 3. In fact, in the case of CPOP a modified formulation (denoted by $\mathbb{P}'[M, \mathcal{A}]$) is straightforward and merely replaces constraints (3d) with

$$\sum_{v \in \mathcal{V}(c)} y_v^a \leq \binom{|\mathcal{V}(c)|}{2} \sum_{v \in \mathcal{V}(c)} x_v \qquad a \in \mathcal{A},\ c \in C(a) \qquad (9)$$

in the $\mathbb{P}[M, \mathcal{A}]$ formulation (3).

However, in the case of NAOP, modification of formulation $\mathbb{A}[K, \mathcal{S}]$ is not that obvious, and is achieved as described below.

In the modification, apart from variables $a = (a_v)_{a \in \mathcal{V}}, t = (t_e)_{e \in \mathcal{V}}, z = (z_v^s)_{s \in \mathcal{S}, v \in \mathcal{V}}$ and Z, the following additional binary variables are used. For all $v, w \in \mathcal{V}$, let y_{vw} be a binary variable equal to 1 if, and only if, at least one path between nodes v and w composed of links not affected by the constructed attack is available. And for all $s \in \mathcal{S}, v, w \in \mathcal{V}$ and $v < w$, let X_{vw}^s be a binary variable equal to 1 if, and only if, service relation $\{v, w\}$ still provides service after the attack, i.e., there still exists a path between nodes v and w and both v and w are still connected to a controller in placement s (we notice that if the path exists and one of the nodes is connected to a controller, the other node is also connected to a controller). Using these variables the considered modification of the NAOP formulation is as follows:

$$\mathbb{A}'[K, \mathcal{S}] : \min\ Z \qquad (10a)$$

$$\sum_{v \in \mathcal{V}} a_v = K \qquad (10b)$$

$$t_e \geq a_v \qquad v \in \mathcal{V}, e \in \delta(v) \qquad (10c)$$

$$t_e \leq a_{\alpha(e)} + a_{\beta(e)} \qquad e \in \mathcal{E} \qquad (10d)$$

$$z_v^s \leq 1 - a_v \qquad s \in \mathcal{S}, v \in \mathcal{V} \qquad (10e)$$

$$z_v^s \geq 1 - a_v \qquad s \in \mathcal{S}, v \in \mathcal{V}(s) \qquad (10f)$$

$$z_{\alpha(e)}^s \geq z_{\beta(e)}^s - t_e \qquad s \in \mathcal{S}, e \in \mathcal{E} \qquad (10g)$$

$$z_{\beta(e)}^s \geq z_{\alpha(e)}^s - t_e \qquad s \in \mathcal{S}, e \in \mathcal{E} \qquad (10h)$$

$$y_{vw} = y_{wv} \qquad v, w \in \mathcal{V} \qquad (10i)$$

$$y_{vv} = 1 - a_v \qquad v \in \mathcal{V} \qquad (10j)$$

$$y_{v\alpha(e)} \geq y_{v\beta(e)} - t_e \qquad v \in \mathcal{V}(s), e \in \mathcal{E} \qquad (10k)$$

$$y_{v\beta(e)} \geq y_{v\alpha(e)} - t_e \qquad v \in \mathcal{V}(s), e \in \mathcal{E} \qquad (10l)$$

$$X_{vw}^s \geq y_{vw} + z_v^s - 1 \qquad s \in \mathcal{S}, v, w \in \mathcal{V} : v < w \qquad (10m)$$

$$Z \geq \sum_{v,w \in \mathcal{V}: v<w} X_{vw}^s \qquad (10n)$$

$$a_v, t_e, z_v^s \in \{0, 1\} \qquad s \in \mathcal{S}, v \in \mathcal{V}, e \in \mathcal{E} \qquad (10o)$$

$$y_{vw} \in \{0, 1\} \qquad s \in \mathcal{S}, v, w \in \mathcal{V} \qquad (10p)$$

$$X_{vw}^s \in \{0, 1\} \qquad s \in \mathcal{S}, v, w \in \mathcal{V} : v < w \qquad (10q)$$

$$Z \in \mathbb{R}^+. \qquad (10r)$$

In the formulation, constraints (10b)-(10h) imposed on variables a, t and z are the same as constraints (5b)-(5h) in formulation (5). Additional constraints (10i)-(10l), in turn, force that for each node v and each link e unaffected by the constructed attack, node v is either connected (by a path composed of unaffected links) to both ends of the link or is not connected to any of them (i.e., $y_{v\alpha(e)} = y_{v\beta(e)}$ when $t_e = 0$).

Clearly, a necessary and sufficient condition for a pair of nodes $\{v, w\}$ to be in service for a given placement s is that these nodes are connected by means of a path (not necessarily elementary) of surviving links containing an unaffected controller, that is if, and only if, both y_{vw} and z_v^s are equal to 1. To express this condition, constraints (10m) that force variable X_{vw}^s to be equal to 1 only when $y_{vw} = z_v^s = 1$ is introduced.

Finally, for the reasons similar to those used for formulation (5), constraint (10n) together with objective (10a) assure (by setting appropriate values in z_v^s, y_{vw} and X_{vw}^s to 0 when needed) the proper value of the objective function for optimal solutions of the considered formulation.

Clearly, algorithms A1 and A2 formulated in Sections (4.1) and (4.2) remain unchanged when used for the so modified versions of CPOP and NAOP, provided that in the max-min and min-max quantities, respectively, defined at the end of Section 2, the value of $V(s, a)$ expresses the number of surviving node-pairs instead of the number of surviving nodes.

7 FINAL REMARKS

The two problems studied in this paper become important when both the network operator and the attacker are trying to optimize their decisions about, respectively, controller placement and attack selection. In such a case, the presented problems can be of value when the interaction between the two parties is considered within the framework of game theory.

Regarding the direct extensions of the material presented in this paper, we plan to improve the efficiency of the NAOP formulation and thanks to that extend the numerical studies to large networks (e.g., with 100 nodes), also taking into account the alternative resilience measure considered in Section 6.

Finally, let us emphasize that the presented optimization model can be applied to systems other than SDN, mentioned in Section 1.

ACKNOWLEDGEMENTS: This work was supported by the POB Research Centre Cybersecurity and Data Science of Warsaw University of Technology within the Excellence Initiative Program - Research University ID-UB (grant no. CyberiADa/1/2020/W13) and by the National Science Centre, Poland (grant no. 2017/25/B/ST7/02313). At the same time, we thank the anonymous reviewers whose comments helped us improve the presentation.

REFERENCES

[1] Eusebi Calle, David Martínez, Mariusz Mycek, and Michał Pióro. 2021. Resilient backup controller placement in distributed SDN under critical targeted attacks. *Int. J. of Critical Infrastructure Protection* 33 (2021), 12–260.

[2] Tamal Das, Vignesh Sridharan, and Mohan Gurusamy. 2020. A survey on Controller Placement problem in SDN. *IEEE Communications Surveys and Tutorials* 22, 1 (2020), 472–503.

[3] Tao Hu, Zehua Guo, Peng Yi, Thar Baker, and Julong Lan. 2018. Multi-controller Based Software-Defined Networking: A Survey. *IEEE Access* 6 (2018), 15980–15996.

[4] Sebastian Orlowski, Michal Pióro, Artur Tomaszewski, and Roland Wessäly. 2010. SNDlib 1.0 – Survivable network design library. *Networks: An International Journal* 55 (2010), 276–286.

[5] Michał Pióro, Mariusz Mycek, and Artur Tomaszewski. 2021. Network protection against node attacks based on probabilistic availability measures. *IEEE Trans. on Network and Service Management* 18, 3 (2021), 2742–2763.

[6] Dorabella Santos, Amaro de Sousa, Carmen Mas-Machuca, and Jacek Rak. 2021. Assessment of Connectivity-Based Resilience to Attacks Against Multiple Nodes in SDNs. *IEEE Access* 9 (2021), 58266–58286.

[7] Dorabella Santos, Amaro de Sousa, and Paulo Monteiro. 2018. Compact Models for Critical Node Detection in Telecommunication Networks. *Electronic Notes in Discrete Mathematics* 64 (2018), 325–334.

[8] Jane M. Simmons. 2014. *Optical Network Design and Planning (2nd edition).* Springer, Switzerland.

Session Session 6C: location and routing

Friday 10 June 2022, 09:30-11:10

Lecture Hall IV

Robust Alternative Fuel Refueling Station Location Problem with Routing under Decision-Dependent Flow Uncertainty

Özlem Mahmutoğulları[1] and Hande Yaman[2]

[1]ORSTAT, Faculty of Economics and Business, KU Leuven, 3000 Leuven, Belgium, ✉ ozlem.mahmutoullar@kuleuven.be
[2]ORSTAT, Faculty of Economics and Business, KU Leuven, 3000 Leuven, Belgium , ✉ hande.yaman@kuleuven.be

Abstract

The refueling station location problem with routing (RSLP-R) is defined as a maximal coverage problem that locates alternative fuel refueling stations (AFSs) on a road network to maximize the refueled alternative fuel vehicle flows by considering the limited range of vehicles and the willingness of drivers to deviate from their paths for refueling. In this study, we consider the RSLP-R under decision-dependent polyhedral flow uncertainty. We model the flow uncertainty set using a hybrid model that comprises a hose model and individual flow bounds. To take into account the fact that vehicle flows are affected by AFS deployment decisions in their neighborhoods, we incorporate the decision-dependency notion into the flow uncertainty set. We propose two linear mixed integer programming formulations and a Benders reformulation. We confirm the effectiveness of the reformulation in solving larger instances based on the road network of Belgium and also assess the value of incorporating uncertainty and decision-dependency into the problem.
Keywords: Alternative fuel vehicles, Location, Routing, Robust optimization, Hose model, Decision-dependent uncertainty, Benders reformulation

1 Introduction

Transportation is heavily dependent on fossil fuels, especially petroleum-based products. This strong dependency has two main drawbacks; consuming fossil fuels results in greenhouse gas emissions and fossil fuels have limited reserves that may be depleted in the near future. Using alternative fuels is one of the solutions to deal with the problems caused by fossil fuel consumption. In recent years, there has been a substantial increase in the promotion of vehicles that need alternative fuels (electricity, hydrogen, natural gas, etc.) to break the transportation sector's reliance on fossil fuels. The lack of alternative fuel station (AFS) infrastructure and the rather limited range of alternative fuel vehicles (AFVs) are two significant obstacles that are slowing down the introduction of AFVs and, as a result, the wide adoption of these vehicles by drivers. In this regard, the refueling station location problem (RSLP) has recently started to be studied in the literature. In the RSLP, the stations can be located on the drivers' predetermined paths, which are usually the shortest paths. Since the drivers may sometimes tolerate deviating from their paths to refuel their vehicles, the refueling station location problem with routing (RSLP-R) extends the RSLP and determines the locations of stations and routes of drivers simultaneously.

It is likely to observe uncertainties in the flows because the rollout of AFVs and the development of the AFS network are still at their initial stages. Moreover, the relationship between the lack of AFS network design and the low level adoption of AFVs is regarded as a "chicken-egg-problem" in the literature because the statistical data shows that the insufficient number of AFSs causes a poor incentive for drivers to use AFVs and vice versa. It is thus important to consider that the availability of AFSs in the neighborhood affects the proliferation of AFVs during the development of infrastructure. Hence, we incorporate robustness and decision-dependency into the RSLP-R.

In this study, we introduce the robust RSLP-R under decision-dependent polyhedral vehicle flow uncertainty. We derive mathematical programming formulations and propose a Benders reformulation and a branch-and-cut algorithm. We perform the following computational experiments: We first compare the

performances of the proposed mathematical models and the Benders reformulation. Then, we investigate the changes in station locations and total covered flows when the optimal solutions of the deterministic, robust and decision-dependent robust problems are employed. We also analyze the changes under different parameter settings. We observe that recognizing the uncertainty in flows and the decision-dependency of uncertain flow realizations may lead to significant gains in the total AFV flows covered.

2 Problem Definition and Solution Methods

The RSLP-R is defined on a road network and an AFV demand is defined by the origin node of the flow, the destination node of the flow, the vehicle flow volume, the range of vehicle, and the total distance tolerated by drivers. The RSLP-R aims to maximize the total amount of AFV flows that can be refueled by locating a predetermined number of AFSs on a network by considering the willingness of drivers to deviate from their shortest paths to refuel their vehicles as well as the limited range of the vehicles. We use the deterministic problem introduced by [1]. We introduce our uncertainty set using the hybrid model ([2]; [4]). The hybrid model comprises a hose model for the aggregate flow bounds of nodes and an interval model for the lower and upper bounds of individual flows. We define the hybrid uncertainty set of the vehicle flows under the impact of station location decisions. We suppose that, when a new station is opened, vehicle flows in the neighborhood increase because the drivers will be more willing to use AFVs if there are AFSs nearby. We investigate the case where the aggregate and individual bounds are affine functions of the location decisions. Since introducing decision-dependent flow bounds results in bilinear terms in the objective function, for linearization, we use two different ways leading to an aggregated and a disaggregated formulation. As the problem size grows, we encounter difficulties in solving the aggregated and disaggregated models and thus we propose a Benders reformulation based on the disaggregated model. We solve this formulation using a branch-and-cut algorithm. The separation, which is exact and polynomial, is done by inspection.

3 Computational Results

We use four data sets to perform our computational experiments. The first one is a commonly used data set in the RSLP literature. We generated the other data sets based on the road network of Belgium. The sets differ in the number of nodes, edges, and origin-destination pairs. The nominal flow volumes are computed using the gravity model and the decision-dependency parameters are chosen using a similar way to that presented by [3]. We first evaluate the performances of the branch-and-cut algorithms to solve the Benders reformulation and the aggregated and disaggregated formulations. We observe that the Benders reformulation outperforms the other formulations. Then, we compare the station location decisions obtained by solving the deterministic, robust (without decision-dependency) and decision-dependent robust problems. We assess the importance of considering only uncertainty and uncertainty and decision-dependency simultaneously. In these experiments, we also examine the effect of different parameter settings, e.g., range, tolerance, and uncertainty level, on the results. Under all settings, we highlight the gain of incorporating uncertainty and decision-dependency into strategic-level decisions.

References

[1] Arslan, O., Karaşan, O. E., Mahjoub, A. R., & Yaman, H. (2019). A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. *Transportation Science*, 53 (4), 1107–1125.

[2] Altın, A., Yaman, H., & Pınar, M. C¸ . (2011). A hybrid polyhedral uncertainty model for the robust network loading problem. *In Performance models and risk management in communications systems* (pp. 157–172). Springer.

[3] Basciftci, B., Ahmed, S., & Shen, S. (2021). Distributionally robust facility location problem under decision-dependent stochastic demand. *European Journal of Operational Research*, 292 (2), 548–561.

[4] Meraklı, M., & Yaman, H. (2016). Robust intermodal hub location under polyhedral demand uncertainty. *Transportation Research Part B: Methodological*, 86, 66–85.

Optimization problems in graphs with locational uncertainty

Michael Poss[1], Jérémy Omer[2], and Marin Bougeret[1]

[1]LIRMM, University of Montpellier, CNRS, France ✉ {marin.bougeret,michael.poss}@lirmm.fr
[2]IRMAR, INSA de Rennes, Rennes, France, ✉ jeremy.omer@insa-rennes.fr

1 Introduction

Research in combinatorial optimization has provided efficient algorithms to solve a large variety of complex discrete decision problems, providing exact or near-optimal solutions in reasonable amounts of time. The applications are countless, ranging from logistics (network design, facility location, ...) to scheduling, including even important data science applications such as clustering. Many of these applications amount to select a subset of edges of a graph $G = (V, E)$ among a family of feasible subsets $\mathcal{F}$ and that minimizes its total weight. Among those, we focus on spatial graphs on a given metric space $(\mathcal{M}, d)$, where each vertex i is assigned a position $u_i \in \mathcal{M}$ and the cost of set $F \in \mathcal{F}$ is given by $\sum_{\{i,j\}\in F} d(u_i, u_j)$, leading to the combinatorial optimization problem

$$\min_{F\in\mathcal{F}} \sum_{\{i,j\}\in F} d(u_i, u_j). \tag{1}$$

Problem (1) encompasses many applications, such as network design, facility location, and clustering. These are typically subject to data uncertainty, be it because of the duration of the decision process, measurement errors, or simply lack of information.

One successful framework that has emerged to address uncertainty is robust optimization [2], modeling the uncertain parameters with convex sets, such as polytopes, or finite sets of points, among which combinatorial robust optimization focuses on discrete robust optimization problems [4]. We enter this framework by considering the model where the positions of the vertices are subject to uncertainty, therefore impacting the distances among the vertices. The resulting problem thus seeks to find the feasible subgraph that minimizes its worst-case sum of distances. Formally, we introduce for each vertex $i \in V$ the set of possible locations as the uncertainty set $\mathcal{U}_i \subseteq \mathcal{M}$. Using the notations $u = (u_1, \ldots, u_{|V|})$ and $\mathcal{U} = \times_{i\in V}\mathcal{U}_i$, the general problem considered in this paper can be cast as

$$\min_{F\in\mathcal{F}} \max_{u\in\mathcal{U}} \sum_{\{i,j\}\in F} d(u_i, u_j). \tag{2}$$

2 Literature

Traditionally, robust optimization problems with an objective function that is concave in the uncertain parameters are reformulated as monolithic models using conic duality [2]. These techniques do not readily extend to function $d(u_i, u_j)$ as the latter is non-concave in general. Actually, for Euclidean metric spaces based on the vector space $\mathbb{R}^\ell, \ell \in \mathbb{Z}^+$, $d(u_i, u_j) = \|u_i - u_j\|_2$ is convex in u_i and u_j. Function $\|u_i - u_j\|_2$ is closely related to the second-order cone (SOC) constraints considered by [5] for robust problems with polyhedral uncertainty sets. The authors of [5] linearize such robust SOC constraints by introducing adjustable variables, turning the problem into an adjustable robust optimization problem.

A second work closely related to (2) is [3], which relies on computational geometry techniques to provide constant-factor approximation algorithms in the special case where $\mathcal{F}$ contains all Hamiltonian cycles of G. They propose in particular to solve a deterministic counterpart of (2) where the uncertain distances are replaced by the maximum pairwise distances $d_{ij}^{max} = \max_{u_i\in\mathcal{U}_i, u_j\in\mathcal{U}_j} d(u_i, u_j)$, for each $(i, j) \in V^2, i \neq j$.

3 Contributions

Our contributions can be summarized as follows:

- We prove that problem (2) is $\mathcal{NP}$-hard even when $\mathcal{F}$ consists of all $s-t$ paths and $(\mathcal{M}, d)$ is the one-dimensional Euclidean metric space or when $\mathcal{F}$ consists of all spanning trees of G. These results illustrate how the nature of problem (2) fundamentally differs from the classical min-max robust problem with cost uncertainty, which is known to be polynomially solvable whenever the costs lie in independent uncertainty sets [1].
- We provide a general cutting-plane algorithm for problem (2) that relies on integer programming formulations for $\mathcal{F}$. We further show that the separation problem $c(F) = \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in F} d(u_i, u_j)$ is $\mathcal{NP}$-hard and provide two algorithms for computing $c(F)$. One is based on integer programming formulations while the other one relies on a dynamic programming algorithm that involves the threewidth of F.
- We extend the approximation algorithm based on d^{max} to general sets $\mathcal{F}$ and metric spaces different from the Euclidean one. We study in depth the resulting approximation ratios, which depend on the structure of $\mathcal{F}$ and $(\mathcal{M}, d)$.
- We provide a dynamic programming algorithm for the special case where $\mathcal{F}$ consists of all $s-t$ paths, which is turned into a fully-polynomial time approximation scheme by rounding data appropriately.
- We compare numerically the exact cutting plane algorithm with the approximation algorithm that relies on d^{max}. The benchmark is composed of two families of instances. The first family includes Steiner tree instances that illustrate subway network design. The second one is composed of strategic facility location instances. The former application relies on two-dimensional Euclidean metric spaces so we can further include the affine decision rule reformulation from [5] to the comparison.

References

[1] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.*, 197(2):427–438, 2009.

[2] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[3] Gui Citovsky, Tyler Mayer, and Joseph S. B. Mitchell. TSP With Locational Uncertainty: The Adversarial Model. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[4] Panos Kouvelis and Gang Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 2013.

[5] Jianzhe Zhen, Frans JCT de Ruiter, Ernst Roos, and Dick den Hertog. Robust optimization for models with uncertain second-order cone and semidefinite programming constraints. *INFORMS Journal on Computing*, 2021.

Bifactor Approximation for Location Routing with Vehicle and Facility Capacitie

Oscar F. Carrasco Heine[1], Antonia Demleitner[2], and Jannik Matuschke[3]

[1]Research Centre for Operations Management, KU Leuven, Belgium, ✉ felipe.carrasco@kuleuven.be
[2]valantic Supply Chain Excellence GmbH, Germany, ✉ antonia.demleitner@sce.valantic.com
[3]Research Centre for Operations Management, KU Leuven, Belgium, ✉ jannik.matuschke@kuleuven.be

Introduction

It has long been observed that basing location decisions on oversimplified cost models can lead to inferior solutions with costs significantly exceeding those of an optimal solution [6, 10, 9]. This motivates the study of *Capacitated Location Routing (CLR)*, an integrated approach that combines Facility Location with a Vehicle Routing problem: Given a finite set F of possible facility locations, the task is to determine a subset of these facilities that is to be opened and to plan tours originating from the open facilities in order to supply clients and satisfy their demand. These tours have to respect capacity constraints for vehicles (the total demand served by a single tour must be less than or equal to $\bar{u}$) and facilities (the total demand served by a facility $w \in F$ must be less than or equal to $u(w)$). All of this should be done at minimum total cost, which is the sum of opening costs for the facilities and the routing cost, i.e., the total length of all tours.

By combining Facility Location and Vehicle Routing, each of which is an *NP*-hard problem in its own right, Location Routing constitutes a computationally challenging problem. Numerous heuristic and exponential-time exact approaches have been proposed in literature; see [7] and [4] for recent surveys. In this paper, we study *approximation algorithms* for Location Routing, that is, algorithms that come with a proven worst-case guarantee both on their running time and the quality of the produced solution, measured in terms of deviation from the cost of an optimal solution. In particular, we provide a *bifactor* approximation algorithm, which computes solutions in which the capacity of any facility can be exceeded by at most a small, adjustable factor while the cost is within a constant factor of the optimal solution. Moreover, our algorithm allows us to solve instances of considerably larger size than those considered in the literature so far. Previously, constant-factor approximation guarantees have only been known for variants of the problem in which facilities are uncapacitated [8, 5] or for soft-capacitated variants, where facility capacities can be extended arbitrarily at linear cost [2, 3].

Algorithm

The algorithm and the accompanying analysis prove the following result:

Theorem 1. *There is an algorithm that, given $\varepsilon \in (0, 1]$, computes in polynomial time a solution to a given instance of CLR such that each tour originates from an open facility, the demand of every client is served entirely by the tours visiting it, the total demand served by any tour is at most the vehicle capacity, the load at any facility $w \in F$ is no more than $u(w)+\varepsilon\bar{u}$, and the total cost is no more than $(4+\frac{2\alpha}{\varepsilon})\,\mathrm{OPT}$, where $\alpha \geq 1$ is the approximation factor of an algorithm for Capacitated Facility Location (CFL).*

The algorithm consists of three steps. In the first step, a minimum spanning tree for a modified instance is partitioned into clusters such that each cluster contains clients with a total demand of at most $\bar{\bar{u}}$ and every cluster with demand less than $\bar{\bar{u}}/2$ contains a facility. The clustering technique is based on a procedure for relieving overloaded subtrees by [1] and [8]. This minimum spanning tree constitutes a lower bound to the optimal CLR solution. In the second step, the clusters are assigned to open facilities via a rounding procedure for an assignment LP. Using the fact that most clusters have large aggregated demand, it is shown that solutions to a certain CFL instance, derived from the original CLR instance, induce feasible solutions of bounded cost for this LP (similarly to the aforementioned spanning tree, the cost of this CFL solution is a lower bound to the optimal CLR solution). The rounding procedure might

allocate one additional cluster per facility, thus resulting in an additional demand of at most $\bar{\bar{u}}$ at any open facility. Finally, each cluster is converted into a tour by adding an edge to its assigned facility an using the classic doubling-and-shortcutting technique for TSP.

Our theoretical results can be seen as a natural next step towards an approximation algorithm strictly respecting hard capacities. However, as we also show with an appropriate class of example instances, such an algorithm would require the use of new and stronger lower bounds on the optimal solution value, beyond the known spanning tree and CFL lower bound.

Variations and Computational Experiments

Complementing our theoretical results, we devise several heuristic modifications to the algorithm that improve its practical performance. In particular, by replacing the aforementioned load-balancing linear program by an integer program of moderate size, we show how our framework can be used to obtain solutions that strictly respect the original facility capacities while still achieving a comparably low cost. We analyze the empirical performance of the algorithm with and without heuristic improvements in an extensive computational study on different sets of benchmark instances from literature and additional newly generated instances. These new instances include up to 10000 customers, which is considerably larger than what has been previously solved in the literature. This is not only a display of the scalability of our method, but these instances could also result in a new benchmark set for the discipline.

Clearly outperforming its theoretical worst-case guarantee, the algorithm is capable of computing near-optimal solutions that either slightly exceed facility capacities (when using the original polynomial-time variant) or strictly respect these capacities (when using the integer program) in a fraction of the running time necessary for exact or other heuristic approaches.

References

[1] Charles J Alpert, Andrew B Kahng, Bao Liu, Ion I Mandoiu, and Alexander Z Zelikovsky. Minimum buffered routing with bounded capacitive load for slew rate and reliability control. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(3):241–253, 2003.

[2] Xujin Chen and Bo Chen. Approximation algorithms for soft-capacitated facility location in capacitated network design. *Algorithmica*, 53(3):263–297, 2009.

[3] Xujin Chen and Bo Chen. Cost-effective designs of fault-tolerant access networks in communication systems. *Networks: An International Journal*, 53(4):382–391, 2009.

[4] Michael Drexl and Michael Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308, 2015.

[5] Tobias Harks, Felix G König, and Jannik Matuschke. Approximation algorithms for capacitated location routing. *Transportation Science*, 47(1):3–22, 2013.

[6] F.E. Maranzana. On the location of supply points to minimize transport costs. *Journal of the Operational Research Society*, 15(3):261–270, 1964.

[7] Caroline Prodhon and Christian Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17, 2014.

[8] R Ravi and Amitabh Sinha. Approximation algorithms for problems combining facility location and network design. *Operations Research*, 54(1):73–81, 2006.

[9] Said Salhi and Graham K Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[10] MHJ Webb. Cost functions in the location of depots for multiple-delivery journeys. *Journal of the Operational Research Society*, 19(3):311–320, 1968.

A Sample Average Approximation based heuristic for the stochastic Production Routing Problem

Andreas Geiger[1]

[1]Institute for Operations Research, Universität Hamburg, Hamburg, Germany, ✉ andreas.geiger@uni-hamburg.de

Integrated planning systems can be used to better coordinate processes and better align decisions within supply chains. A system that integrates the three key processes production, inventory and distribution is called the Production Routing Problem (PRP) [3].

In this work, a stochastic PRP (SPRP) is considered, which includes the production and distribution of a single product from a production plant to multiple retailers using capacitated vehicles in a discrete and finite time horizon. Since demand is a critical information for decision making and is only known approximately by forecasts in most cases, the uncertainty is represented by demand scenarios.
The PRP solely considering demand uncertainty has only been addressed three times. [2] are the first to deal with demand uncertainty and introduce the SPRP under demand uncertainty in a two-stage decision process and rolling horizon framework for the multistage SPRP. [4] and [5] tackle the SPRP allowing backlog and solve a similar two-stage decision process using different Sample Average Approximation (SAA) methods and extend the general SAA with several heuristic approaches. Within these three articles, setup and routing decisions are made on the first stage and quantity decisions can be adjusted after demand realisation. This might lead to retailer visits, where retailers have little or no demand and unnecessary costs may occur. Therefore, a new assumption is made - routes can be adjusted at short notice in the second stage.

As the new two-stage stochastic optimization model can only be solved for very small instances in reasonable computing time, a heuristic approach is presented. The approach is based on a common method used in the literature. As in [1] and [10] the PRP is decomposed into a production and distribution problem (PrDP) and into a vehicle routing problem (VRP). The PrDP is solved with approximated routing costs and the identified delivery quantites and planned retailer visits can be transferred to the VRP to determine better routing costs. The process is repeated multiple times, until there is no change in the production decisions or another termination criterion is met. The heuristic can be transferred into an SAA approach. Several small scenario sets are solved using a two-stage stochastic PrDP and a common VRP solution method. The first stage decision provided by the smaller scenario sets are evaluated by a larger scenario set. Then, the first stage decisions resulting in the lowest total cost are chosen. First computational results show significant improvements of the heuristic-based SAA framwork compared to the expected-value problem of the SPRP for a moderate number of periods. For solving the VRP a simple Sweep-based heuristic [8] is used. Instances of five or ten retailers and up to seven periods can be improved 1% to 3.75% on average. As a second approach, the heuristic will be applied to an adjustable SAA framework as described in [4]. To find promising variables, which can be partially fixed for further replications and/or larger scenario sets, decisions' weights are calculated according to their objective value. By this, a bigger scenario space can be examined and the solution might be improved.
To further investigate the influence of the VRP on the solution, two methods are compared. A GRASP-based metaheuristic according to [9] and a Sweep-based heuristic. First results show slight improvements using the GRASP-based metaheuristic over the Sweep-based heuristic. But the GRASP-based method increases the computation time significantly. Since a VRP must be solved for each scenario and period, an intensive metaheuristic may not be appropriate for larger instances and a simple heuristic like the Sweep-based heuristic might be more applicable.

Due to the ability to adjust routing decisions at short notice, these findings could be used to get closer to real-world applications, as noted in [7] or [6]. Also the solution approaches could be used to investigate,

if there is a crucial difference in the solution quality and therefore in the overal costs, considering the routing decision in the first or second stage.

References

[1] N. Absi, C. Archetti, S. Dauzère-Pérès, and D. Feillet. A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, 49(4):784–795, 2015.

[2] Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. Benders decomposition for production routing under demand uncertainty. *Operations Research*, 63(4):851–867, 2015.

[3] Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, 55:141–152, 2015.

[4] Agostinho Agra, Cristina Requejo, and Filipe Rodrigues. An adjustable sample average approximation algorithm for the stochastic production-inventory-routing problem. *Networks*, 72(1):5–24, 2018.

[5] Agostinho Agra, Cristina Requejo, and Filipe Rodrigues. A hybrid heuristic for a stochastic production-inventory-routing problem. *Electronic Notes in Discrete Mathematics*, 64:345–354, 2018.

[6] Manuel Díaz-Madroñero, David Peidro, and Josefa Mula. A review of tactical optimization models for integrated production and transport routing planning decisions. *Computers & Industrial Engineering*, 88:518–535, 2015.

[7] Josefa Mula, David Peidro, Manuel Díaz-Madroñero, and Eduardo Vicens. Mathematical programming models for supply chain production and transport planning. *European Journal of Operational Research*, 204(3):377–390, 2010.

[8] Zahrul Jannat Peya, M. A., Tanzima Sultana, and M. M. Distance based sweep nearest algorithm to solve capacitated vehicle routing problem. *International Journal of Advanced Computer Science and Applications*, 10(10), 2019.

[9] Christian Prins. A grasp × evolutionary local search hybrid for the vehicle routing problem. In Francisco Babtista Pereira and Jorge Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, volume 161 of *Studies in Computational Intelligence*, pages 35–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[10] Yuzhuo Qiu, Liang Wang, Xiaoling Xu, Xuanjing Fang, and Panos M. Pardalos. A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, 66:311–318, 2018.