Kostenminimierung in Multi-Interface Drahtlosnetzwerken

von
Sébastien Auroux

Masterarbeit in Mathematik

vorgelegt der

Fakultät für Mathematik, Informatik und
Naturwissenschaften der Rheinisch-Westfälischen
Technischen Hochschule Aachen

im September 2012

angefertigt am Lehrstuhl II für Mathematik

Betreuer: Prof. Arie M. C. A. Koster

Zweitgutachter: Prof. Berthold Vöcking

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

§1 Einleitung

Drahtlosnetzwerke sind fester Bestandteil unserer Umgebung. Bereits im Jahr 1968 plante man an der Universität von Hawaii die sich auf verschiedenen Inseln befindenen Standorte der Universität mit Hilfe von drahtloser Funktechnologie mit dem Zentralrechner der Universität auf der Insel Oahu zu verbinden. Dieses Vorhaben wurde schließlich in Form des ALOHAnet (siehe Abramson [1]) im Jahre 1971 erfolgreich umgesetzt.

Bis zur kommerziellen Verbreitung dieser Technologie dauerte es jedoch noch einige Jahre. In den späten achtziger Jahren kamen jedoch schließlich erste Funkkarten auf den freien Markt. Zu Anfang waren diese noch sehr teuer, was sich aber mit den kommenden Jahren änderte. Die Drahtlostechnologie wurde daraufhin zunehmend fester Bestandteil von gewerblich verkauften Endgeräte, insbesondere bei mobilen Notebooks und später auch Smartphones, und verbreitete sich so immer mehr mit einer großen Geschwindigkeit. Bereits im Jahr 2005 wurden beispielsweise in der Europäischen Union mehr Notebooks als Desktop-Rechner verkauft, meist mit eingebautem WLAN-Chip. Heute besitzt ein großer Teil der westlichen Bevölkerung mit WLAN und Bluetooth ausgestattete Notebooks und Smartphones und öffentliche und kommerzielle WLAN-Zugangspunkte zur Internet-Anbindung sind an vielen Orten verfügbar und ermöglichen den Zugriff auf das weltweite Datennetz.

Da mobile Endgeräte in Drahtlosnetzwerken in der Regel ohne feste Energiequelle auskommen müssen, ist Energie in solchen Netzwerken die entscheidende Ressource, um Endgeräte möglichst lange in Betrieb zu halten. Diese Ressource muss somit effizient genutzt werden. Mobile Endgeräte nutzen zudem meist mehrere verschiedene Schnittstellen zur Kommunikation (wie etwa WLAN, Bluetooth oder GSM). Jede verfügbare Schnittstelle verbraucht im aktivierten Zustand Energie und sollte deshalb nur aktiviert sein, wenn dies tatsächlich notwendig ist.

Das Optimierungsproblem *Kostenminimierung in Multi-Interface Drahtlosnetzwerken* (englisch: *cost minimization in wireless and multi-interface networks*, kurz: *CMI*) beschäftigt sich damit, eine Verteilung aktivierter Schnittstellen für jedes Endgerät eines gegebenen Netzwerks zu finden, so dass alle erforderlichen Verbindungen bestehen und der globale Energieverbrauch minimal ist. Diese Masterarbeit untersucht CMI, nach Wissen des Autors und des Betreuers erstmalig, mit Methoden der *ganzzahligen linearen Optimierung* und zeigt die dadurch erzielten Ergebnisse auf.

Ganzzahlige Optimierungsprobleme spielen in der angewandten Mathematik eine zentrale Rolle. Neben CMI lassen sich viele alltägliche Probleme, wie Produktions- oder Routenplanungen, in Form von (ganzzahligen) linearen Optimierungsproblemen beschreiben. Solange eine nicht ganzzahlige Lösung ausreicht, sind diese Probleme effizient (siehe Kapitel 2.1.2) lösbar (siehe SCHRIJVER [32]). Ist jedoch eine (teilweise) ganzzahlige Lösung gefordert, so ist dies im Allgemeinen nicht mehr der Fall. Bekannte Vertreter solcher *ganzzahligen linearen Programme* (englisch: *integer linear program*, kurz: *ILP*) sind zum Beispiel das *Handelsreisendenproblem*, das *Rucksackproblem* oder das *Cliquenproblem* (siehe GAREY [14] und COOK [7]). Die *ganzzahlige lineare Optimierung* beschäftigt sich damit, auch diese Probleme in der Praxis zu lösen.

Nach der in Kapitel 2 gegebenen Einführung in die für die Thematik wichtigen Grundlagen, stellen wir in Kapitel 3 zunächst eine mathematische Beschreibung von CMI vor und präsentieren bisher für CMI bekannte Resultate. Im Anschluss wird eine Beschreibung von CMI als ganzzahliges lineares Programm vorgestellt und die formale Äquivalenz der Beschreibungen gezeigt.

In Kapitel 4 untersuchen wir im Folgenden den durch die ILP-Beschreibung von CMI induzierten Polyeder mit den Methoden der Polyedertheorie, mit dem Ziel, Resultate zur schnelleren Berechnung von CMI-Instanzen in der Praxis zu erzielen. Mit dem Gedanken an die praxistaugliche Lösbarkeit von CMI werden dann in Kapitel 5 Algorithmen vorgestellt und evaluiert, welche mit schneller Rechenzeit Näherungslösungen für CMI ermitteln.

Die Masterarbeit schließt mit der Präsentation einer umfassenden Rechenstudie in Kapitel 6, im Verlauf derer die praktische Berechenbarkeit anhand diverser CMI-Testinstanzen dargelegt wird. Des Weiteren wird die Güte der durch die Algorithmen aus Kapitel 5 ermittelten Näherungslösungen geprüft und dokumentiert. Den Abschluss bildet eine Zusammenfassung der Ergebnisse und ein Ausblick auf weitere mögliche Ansätze zur Verbesserung der Berechenbarkeit optimaler Lösungen für CMI.

§2 Notationen und Grundlagen

Dieses Kapitel gibt eine Zusammenfassung der in dieser Arbeit benutzten grundlegenden Definitionen und Bezeichnungen aus der Komplexitätstheorie, der Graphentheorie und der ganzzahligen linearen Optimierung.

2.1 Komplexitätstheorie

2.1.1 Komplexität von Problemen und Algorithmen

Probleme sind *Aufgaben* oder *Fragestellungen*, für die es eine schematische, strukturelle Lösung zu finden gilt. In dieser Arbeit beschränken wir uns dabei auf *Optimierungsprobleme*. Dies sind Probleme, bei denen das Minimum oder Maximum einer Zielfunktion, abhängig von den Eingabeparametern, bestimmt werden soll.

Die Komplexität eines Problems ist ein Maß dafür, wie schwierig es ist, das Problem zu lösen und wird in der Regel anhand der Komplexität der Algorithmen gemessen, welche jenes Problems lösen. Ein (deterministischer) *Algorithmus* ist eine aus endlich vielen Schritten bestehende, eindeutige Handlungsvorschrift zur Erzeugung einer *Ausgabe* anhand einer *Eingabe*. Ein Algorithmus *löst* ein Problem, wenn er für jede Probleminstanz dieses Problems eine Lösung für das Problem zurückgibt.

Die Laufzeit eines Algorithmus wird als Funktion in Abhängigkeit von der Eingabelänge n angegeben. Hierbei wird außerdem das *uniforme Kostenmaß* der *Registermaschine* (*Random Access Machine*, kurz: RAM, beschrieben von Cook und Reckhow [9]) zu Grunde gelegt, welches es erlaubt grundlegende Rechenoperationen, wie etwa die Addition oder Multiplikation von Zahlen, in konstanter Rechenzeit durchzuführen.

(2.1) Definition (Laufzeit eines Algorithmus)
Die *worst-case-Laufzeit* (im Folgenden auch einfach nur *Laufzeit*) $t_A(n), n \in \mathbb{N}$ eines Algorithmus A ist definiert als die maximale Laufzeit auf Eingaben der Länge n.

Oft reicht es aus statt der exakten Laufzeit eines Algorithmus eine obere Schranke für dessen Laufzeit anzugeben. Wir führen hierzu die *Landau Notation* ein.

(2.2) Definition (Landau-Notation)
Sei $f : \mathbb{N} \to \mathbb{R}$ eine Funktion, dann ist

$$O(f) := \{g : \mathbb{N} \to \mathbb{R} \mid \exists c \in \mathbb{R}_+ \, \exists N \in \mathbb{N} \, \forall n \geq N : |g(n)| \leq c|f(n)| \}$$

die Klasse aller Funktionen von $\mathbb{N} \to \mathbb{R}$, die mindestens so *gut* sind wie f. Ist die Laufzeit eines Algorithmus A in $O(f)$, so schreibt man informell einfach $t_a(n) = O(f)$ oder sagt A ist in $O(f)$.

(2.3) Definition (Polynomialzeitalgorithmus)
Die Laufzeit eines Algorithmus A ist *polynomiell beschränkt*, falls

$$\exists \, \alpha \in \mathbb{N} : t_A(n) = O(n^\alpha).$$

Ein solcher Algorithmus A heißt *Polynomialzeitalgorithmus*.

Um Probleme danach zu klassifizieren, wie schwierig sie zu lösen sind, ordnet man sie *Komplexitätsklassen* zu. Dabei ist die Komplexität eines Algorithmus für ein Problem maßgeblich dafür, welche Komplexität ein Problem höchstens haben kann. Nichtsdestotrotz sei erwähnt, dass es auch theoretische, also von Algorithmen losgelöste, Aussagen gibt, welche die Komplexität eines Problems nach oben oder nach unten beschränken.

(2.4) Definition (Komplexitätsklassen)
Die folgenden grundlegenden Komplexitätsklassen, sind für diese Arbeit von Bedeutung:

Klasse	Probleme, für welche...
P	ein Polynomialzeitalgorithmus existiert.
NP	eine gegebene Lösung in Polynomialzeit verifiziert werden kann.
EXPTIME	ein Lösungsalgorithmus mit Laufzeit in $O(2^{p(n)})$ existiert, wobei p ein Polynom ist.

Tabelle 1: Grundlegende Komplexitätsklassen

Offenbar gilt P $\subseteq$ NP $\subseteq$ EXPTIME. Cook [8] hat gezeigt, dass P $\subsetneq$ EXPTIME. Der Zusammenhang zwischen P und NP ist jedoch ein bekanntes, offenes Problem in der Informatik (siehe Cook [7]). Im Allgemeinen wird mit der Hypothese P $\neq$ NP gearbeitet.

(2.5) Definition (Polynomielle Reduktion)
Seien P_1, P_2 zwei Probleme und x eine Eingabe für P_1. Dann heißt P_1 *polynomiell reduzierbar* auf P_2, falls mit polynomieller Laufzeit invertierbar aus x eine Eingabe x^* für P_2 und aus y^*, der Lösung für P_2 zur Eingabe x^*, eine Lösung y für P_1 berechnet werden kann, so dass gilt:

$$y \text{ ist Lösung für } P_1 \text{ zur Eingabe } x \quad \Leftrightarrow \quad y^* \text{ ist Lösung für } P_2 \text{ zur Eingabe } x^*.$$

Man schreibt $P_1 \leq_p P_2$.

(2.6) Lemma
Seien P_1, P_2 zwei Probleme und $P_1 \leq_p P_2$, so gilt:

(1) $P_2 \in \mathrm{P} \quad \Rightarrow \quad P_1 \in \mathrm{P}$,

(2) $P_2 \in \mathrm{NP} \quad \Rightarrow \quad P_1 \in \mathrm{NP}$.

(2.7) Definition (NP-Vollständigkeit)
Ein Problem P_2 heißt *NP-schwer*, falls sich jedes Problem $P_1 \in \mathrm{NP}$ auf P_2 polynomiell reduzieren lässt, d.h.

$$P_1 \leq_p P_2 \quad \forall\ P_1 \in \mathrm{NP}.$$

Ist zusätzlich $P_2 \in \mathrm{NP}$, so heißt P_2 *NP-vollständig*. Die Klasse aller NP-vollständigen Probleme bezeichnet man mit **NPC**.

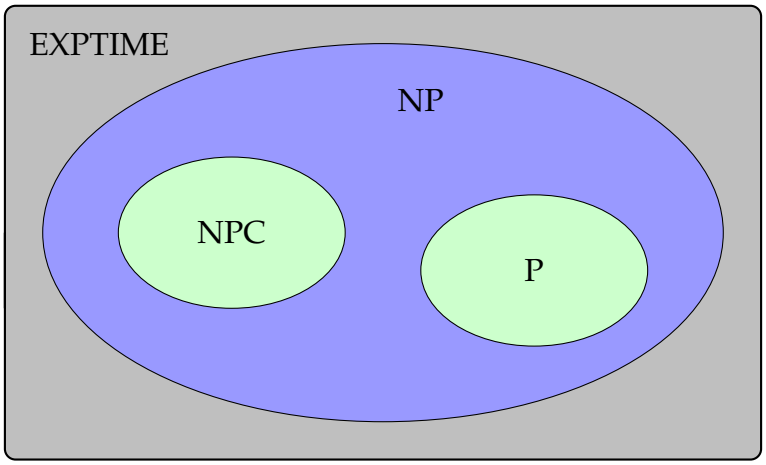

Abbildung 1: Die Komplexitätsklassen P, NP, NPC, EXPTIME

2.1.2 Approximationsalgorithmen

Probleme aus P werden allgemein als gut lösbar angesehen. Für Probleme aus NP gilt dies jedoch nicht. Oft bedient man sich daher Algorithmen, welche in polynomieller Laufzeit eine gültige, aber meist nicht optimale, Lösung für solche Probleme bestimmen. Solch ein Algorithmus heißt *Approximationsalgorithmus*.

(2.8) Definition (Approximationsverhältnis)
Sei A ein Approximationsalgorithmus für ein Problem, F der von A erzielte Wert und F^* der optimale Wert der Zielfunktion. Eine Funktion $f(n), \mathbb{N} \to \mathbb{N}$, ist das *Approximationsverhältnis* von A, falls gilt:

$$\left| \frac{F}{F^*} \right| \leq f(n) \quad \forall\, n \in \mathbb{N}.$$

Ein Algorithmus A mit Approximationsverhältnis $f(n)$ heißt $f(n)$-*Approximationsalgorithmus* und ein Problem, für das ein $f(n)$-Approximationsalgorithmus existiert heißt $f(n)$-approximierbar.

Die Klasse der Probleme, für die ein $f(n)$-Approximationsalgorithmus mit konstantem Approximationsverhältnis $f(n) = \alpha$ für alle $n \in \mathbb{N}$ für ein $\alpha \in \mathbb{R}$ existiert, bezeichnet man mit **APX**.

(2.9) Definition (PTAS)
Ist A(ϵ) ein polynomieller Algorithmus, der von einem Parameter ϵ so abhängt, dass A(ϵ) ein ϵ-Approximationsalgorithmus für jedes $\epsilon > 0$ ist, dann heißt A(ϵ) *polynomial time approximation scheme* (kurz: PTAS).

Probleme, für welche ein PTAS existiert, lassen sich also beliebig gut approximieren. Die Klasse dieser Probleme wird ebenso mit **PTAS** bezeichnet.

Eine *PTAS-Reduktion* ist eine polynomielle Reduktion eines Problems P_1 auf ein Problem P_2, welche die PTAS Eigenschaft von P_1 beibehält, d.h.

$$P_1 \in \text{PTAS} \quad \Rightarrow \quad P_2 \in \text{PTAS}.$$

Man schreibt hierfür $P_1 \leq_{PTAS} P_2$. Ein Problem P_2 heißt *APX-schwer*, falls sich jedes Problem $P_1 \in$ APX auf P_2 PTAS-reduzieren lässt. Ist zusätzlich $P_2 \in$ APX, so heißt P_2 *APX-vollständig*.

Nach Jansen und Margraf [21] gilt unter der Voraussetzung P $\neq$ NP die Inklusionskette

$$\mathrm{P} \subsetneq \mathrm{PTAS} \subsetneq \mathrm{APX} \subsetneq \mathrm{NP}.$$

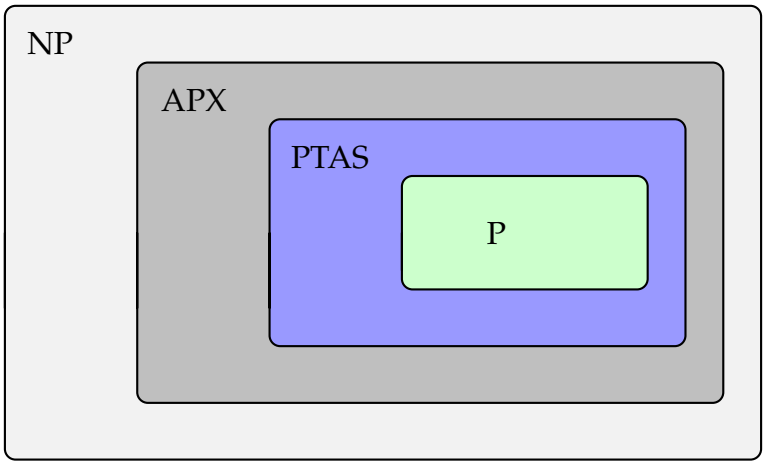

Abbildung 2: Die Komplexitätsklassen P, PTAS, APX, NP

2.2 Notationen aus der Graphentheorie

Die hier eingeführten Begriffe wurden zum Großteil aus Volkmann [34] übernommen.

(2.10) Definition (Graph)
Sei V eine beliebige, endliche Menge und $E \subseteq \{\, \{v,w\} \mid v,w \in V,\, v \neq w \,\}$. Dann ist $G := G(V,E)$ ein (endlicher, einfacher und ungerichteter) *Graph*.

Für einen gegebenen Graphen $G(V,E)$ bezeichnet $V = V(G)$ ist die *Knotenmenge* von G und die Elemente von V heißen *Knoten*. Analog ist $E = E(G)$ die *Kantenmenge* von G und die Elemente von E heißen *Kanten*. Gilt für die Kantenmenge eines Graphen G die Gleichheit $E = \{\{v,w\} \mid v,w \in V,\, v \neq w\}$, so heißt G *vollständig*. Im Weiteren seien für einen Graphen G außerdem $n = n(G) := |V|$ und $m = m(G) := |E|$.

Ist $\{v,w\} \in E$ so heißen v und w *Endpunkte* von $\{v,w\}$. Sind zwei Knoten v und w durch eine Kante verbunden, d.h. $\{v,w\} \in E$, so sind v und w *benachbart* oder auch *adjazent*, ebenso sagt man die Kante $\{v,w\}$ *inzidiert* mit den Ecken v und w, oder v und w sind durch die Kante $\{v,w\}$ *verbunden*. Inzidieren zwei verschiedene Kanten mit einer gemeinsamen Ecke, so sind die Kanten *inzident*.

Ist $V' \subseteq V$ und $E' \subseteq E$ so, dass $\bigcup_{\{v,w\}\in E'}\{v,w\} \subseteq V'$, so bezeichnet $G' = G'(V', E')$ einen *Teilgraph* von G und man schreibt $G' \subseteq G$. Ist $E' \subseteq E$, so ist $G[E'] = G(V', E')$ mit $V' = \bigcup_{\{v,w\}\in E'}\{v,w\}$ der von E' *erzeugte Teilgraph*. Ist umgekehrt $V' \subseteq V$ und $E' \subseteq E$ mit $E' = \{\, \{v,w\} \in E \mid v,w \in V' \,\}$, so bezeichnet $G[V'] = G(V', E')$ den von V' *induzierten Teilgraphen*.

Für $k \in \mathbb{N}$ ist ein *Weg* der Länge k eine Folge von Knoten $(v_1, v_2, \ldots, v_k)$, so dass $\{v_i, v_{i+1}\} \in E$ für alle $i = 1, \ldots, k-1$. Gilt außerdem $v_i \neq v_j$ für alle $i, j = 1, \ldots, k$, $i \neq j$, so spricht man von einem *Pfad*. Ist ein Pfad $(v_1, v_2, \ldots, v_k)$, $k \geq 3$ geschlossen, d.h. $v_1 = v_k$, so nennt man $(v_1, v_2, \ldots, v_k)$ einen *Kreis* der Länge k. Ein zusammenhängender Graph ohne Kreise heißt *Baum*.

(2.11) Definition (Zusammenhang)
Ein Graph G heißt *zusammenhängend*, wenn zwischen zwei beliebigen Knoten $v_i, v_j \in V$ mindestens ein Weg existiert. Eine maximal zusammenhängende Teilmenge $V' \subseteq V$ heißt *Zusammenhangskomponente* von G, d.h. $G[V']$ ist zusammenhängend, jedoch ist $G[V' \cup \{w\}]$ nicht zusammenhängend für ein beliebiges $w \in V \backslash V'$.

(2.12) Definition (Grad)
Sei G ein Graph und $v \in V$. Dann bezeichnet

$$d(v) = d(v, G) := |\{\, \{v,w\} \in E \mid w \in V \,\}|$$

die Anzahl der mit v inzidierenden Kanten, auch genannt *Grad* von v. Der maximale Grad eines Graphen $\Delta(G)$ ist der maximale Grad eines Knotens in G.

Die *Nachbarschaft* $N(v) = N(v, G)$ eines Knoten $v \in V$ ist definiert als

$$N(v) := \{w \in V \mid \{v,w\} \in E\}.$$

Weiter ist $N[v] = N[v, G] := N(v) \cup \{v\}$ die abgeschlossene Nachbarschaft von v.

2.3 Ganzzahlige Programmierung

2.3.1 Polyedertheorie

(2.13) Definition (Polyeder)
Eine Teilmenge $P \subseteq \mathbb{R}^n$ beschrieben durch eine endliche Menge linearer Ungleichungen, d.h.

$$P = \{\, x \in \mathbb{R}^n \mid Ax \leq b \,\}, \quad A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m,$$

heißt *Polyeder*. Im Folgenden bezeichnen A_i für $i \in 1,\ldots,m$ die Zeilen, a_j für $j \in 1,\ldots,n$ die Spalten und $a_{i,j}$ für $i \in 1,\ldots,m$, $j \in 1,\ldots,n$ die einzelnen Einträge der (Koeffizienten-)Matrix A. A_I, $I \subseteq \{1,\ldots,m\}$ ist die Matrix, welche nur aus den Zeilen A_i, $i \in I$, besteht.

Ein beschränkter Polyeder heißt auch *Polytop*. Auf Grund der Konstruktion ist ein Polyeder P stets *konvex*, d.h.

$$\forall\ x,y \in P: \quad \{\, \lambda x + (1-\lambda)y \mid \lambda \in (0,1) \,\} \subseteq P.$$

Ein Punkt des Polyeders $x \in P$ heißt *Extremalpunkt* von P, falls x nicht als endliche Konvexkombination anderer Punkte in P, d.h. als

$$x = \sum_{i=1}^{k} \lambda_i x_i, \quad \lambda_i \in \mathbb{R}_+,\ \sum_{i=1}^{k} \lambda_i = 1,\ x_i \in P \setminus \{x\},\ i \in \{1,\ldots,k\}$$

dargestellt werden kann. Im geometrischen Sinne entsprechen die Extremalpunkte eines Polyeders gerade den Ecken der Polyederdarstellung.

(2.14) Definition (Formulierung)
Ein Polyeder $P \subseteq \mathbb{R}^n$ ist eine *Formulierung* für eine Menge $X \subseteq \mathbb{Z}^n$, wenn

$$X = P \cap \mathbb{Z}^n.$$

Sind P_1, P_2 zwei Formulierungen für eine Menge X, so heißt P_1 *besser* als P_2 genau dann wenn $P_1 \subsetneq P_2$.

(2.15) Definition (Konvexe Hülle)
Sei $X \subseteq \mathbb{R}^n$, dann ist die *konvexe Hülle* von X, $conv(X)$, definiert als

$$conv(X) := \left\{ x \in \mathbb{R}^n \;\middle|\; x = \sum_{i=1}^{k} \lambda_i x_i,\ \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \in \mathbb{R}_+,\ x_i \in X,\ i \in \{1,\ldots,k\} \right\}.$$

$conv(X)$ ist die kleinste konvexe Menge, welche X enthält. Ist die Menge X endlich, so ist $conv(X)$ ein Polyeder (siehe Weyl [35]) und damit die *beste* oder auch *ideale* Formulierung für eine Menge X.

(2.16) Definition (Affine Unabhängigkeit)
Eine endliche Anzahl von Vektoren $x_1,\ldots,x_k \in \mathbb{R}^n$ heißt *affin unabhängig*, wenn für $\lambda_1,\ldots,\lambda_k \in \mathbb{R}$ gilt:

$$\sum_{i=1}^{k} \lambda_i x_i = 0 \text{ und } \sum_{i=1}^{k} \lambda_i = 0 \quad \Rightarrow \quad \lambda_i = 0\ \forall\ i \in \{1,\ldots,k\}.$$

Das folgende Lemma stellt den Zusammenhang zwischen affiner Unabhängigkeit und der aus der linearen Algebra bekannten linearen Unabhängigkeit (siehe Beutelspacher [4]) her.

(2.17) Lemma
Die folgenden Aussagen sind äquivalent:

(i) $x_1, \ldots, x_k \in \mathbb{R}^n$ sind affin unabhängig,

(ii) $x_2 - x_1, \ldots, x_k - x_1 \in \mathbb{R}^n$ sind linear unabhängig.

Beweis:

$(i) \Rightarrow (ii)$: Angenommen $x_2 - x_1, \ldots, x_k - x_1$ sind linear abhängig obwohl $x_1, \ldots, x_k$ affin unabhängig sind. Dann gibt es $(\lambda_2, \ldots, \lambda_k) \neq (0, \ldots, 0)$ mit $\sum_{i=2}^{k} \lambda_i (x_i - x_1) = 0$. Setze $\lambda_1 = -\sum_{i=2}^{k} \lambda_i$, dann gilt:

$$\sum_{i=1}^{k} \lambda_i x_i = \sum_{i=1}^{k} \lambda_i (x_i - x_1 + x_1) = \sum_{i=2}^{k} \lambda_i (x_i - x_1) + \sum_{i=2}^{k} \lambda_i x_1 + \lambda_1 x_1 = 0,$$

was die affine Abhängigkeit von $x_1, \ldots, x_k$ bedeuten würde, im Widerspruch zur Voraussetzung.

$(ii) \Rightarrow (i)$: Angenommen $x_2 - x_1, \ldots, x_k - x_1$ sind linear unabhängig und $x_1, \ldots, x_k$ sind nicht affin unabhängig, d.h. es gibt $(\lambda_1, \ldots, \lambda_k) \neq (0, \ldots, 0)$ mit $\sum_{i=1}^{k} \lambda_i = 0$ und $\sum_{i=1}^{k} \lambda_i x_i = 0$. Dann gilt

$$\sum_{i=2}^{k} \lambda_i (x_i - x_1) = \sum_{i=1}^{k} \lambda_i (x_i - x_1) = \Big(- \sum_{i=1}^{k} \lambda_i \Big) x_1 = 0,$$

was im Widerspruch steht zur linearen Abhängigkeit der Vektoren $x_2 - x_1, \ldots, x_k - x_1$. □

Die *Dimension* $\dim(P)$ eines Polyeders $P \subseteq \mathbb{R}^n$ ist die maximale Anzahl an affin unabhängigen Vektoren in P minus 1. Wir legen fest, dass $\dim(\emptyset) = -1$ ist. Ist $\dim(P) = n$ so heißt P *volldimensional*. Sind P_1, P_2 Polyeder mit $P_1 \subseteq P_2$, dann gilt $\dim(P_1) \leq \dim(P_2)$.

Eine weitere Möglichkeit der Bestimmung von $dim(P)$ liefert der folgende Satz aus Bachem und Grötschel [3]:

(2.18) Satz
Sei $P = \{ x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0 \} \neq \emptyset$ ein Polyeder. Dann gilt

$$\dim(P) = n - \text{Rang}(A_{eq(P)}),$$

wobei

$$eq(P) := \{ i \in \{1, \ldots, m\} \mid A_i x = b_i \ \forall\, x \in P \}$$

die *Gleichheitsmenge* von P ist.

(2.19) Definition (Gültige Ungleichungen und Dominanz)
Eine Ungleichung $\pi x \leq \pi_0$, $\pi \in \mathbb{R}^n$, $\pi_0 \in \mathbb{R}$ ist eine *gültige Ungleichung* für $X \subseteq \mathbb{R}^n$, wenn

$$\pi x \leq \pi_0 \quad \forall\, x \in X.$$

Seien $\pi x \leq \pi_0$ und $\mu x \leq \mu_0$ zwei gültige Ungleichungen für X, so *dominiert* Ungleichung $\pi x \leq \pi_0$ die Ungleichung $\mu x \leq \mu_0$, falls ein $\lambda > 0$ existiert, so dass $\lambda\mu \leq \pi$, $\pi_0 \leq \lambda\mu_0$ und $(\pi, \pi_0) \neq (\lambda\mu, \lambda\mu_0)$.

(2.20) Definition (Seitenflächen und Facetten)
Sei $P \subseteq \mathbb{R}^n$ ein Polyeder und $\pi x \leq \pi_0$ eine gültige Ungleichung für P. Dann heißt die Menge

$$S = \{ x \in P \mid \pi x = \pi_0 \}$$

Seitenfläche von P. Ist $S \neq \emptyset$ und $S \neq P$, so ist S eine *nicht-triviale* oder auch *echte* Seitenfläche von P. Wenn $S \neq \emptyset$, dann *fördert* oder *definiert* $\pi x \leq \pi_0$ die Seitenfläche S. Jede Seitenfläche S von P ist wiederum ein Polyeder, da man S schreiben kann als

$$S = \{ x \in \mathbb{R} \mid Ax \leq b,\, \pi x \leq \pi_0,\, \pi x \geq \pi_0 \}.$$

Die Seitenfläche S_I von P, *induziert* durch $I \subseteq \{1, \ldots, m\}$ ist definiert als

$$S_I := \left\{ x \in P \middle| \left(\sum_{i \in I} A_i \right) x = \sum_{i \in I} b_i \right\}.$$

Ist S eine echte Seitenfläche von P, so heißt S *Facette* von P, falls S in keiner echten Seitenfläche von P strikt enthalten ist.

Eine weitere Charakterisierung von Facetten bietet der folgende Satz aus Schrijver [32]:

(2.21) Satz
Ist P ein Polyeder und S eine Seitenfläche von P, dann ist S genau dann eine Facette von P, falls gilt:

$$dim(S) = dim(P) - 1.$$

2.3.2 Lineare Optimierung

Ein *lineares Optimierungsproblem* (oder auch *lineares Programm*, kurz: LP) ist ein Optimierungsproblem mit linearer *Zielfunktion* und linearen *Nebenbedingungen*. Im Folgenden beschränken wir uns auf Minimierungsprobleme, da sich jedes Maximierungsproblem durch Multiplikation der Zielfunktion mit -1 in ein Minimierungsproblem überführen lässt. Genauso besitzt jede Nebenbedingung die Form $\pi x \leq \pi_0$ mit $\pi, \pi_0 \in \mathbb{R}$ oder lässt sich durch Multiplikation mit -1 in diese überführen. Weiterhin lässt sich jede Variable $x \in \mathbb{R}$ durch zwei Variablen $x_+, x_- \in \mathbb{R}_+$ ersetzen. Eine lineares Optimierungsproblem hat damit die allgemeine Form

$$\begin{array}{lrl} \min & c^T x & \\ \text{s.d.} & Ax & \leq b \\ & x & \geq 0 \end{array}$$

wobei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$. Ein Vektor $x \in \mathbb{R}^n$ ist eine *zulässige Lösung* für ein LP, falls x alle Nebenbedingungen des LP erfüllt. Die Menge der zulässigen Lösungen ist

$$P_{LP} := \{\, x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0 \,\}.$$

Offenbar ist P_{LP} ein Polyeder. Eine zulässige Lösung x^* mit $c^T x^* \leq c^T x \;\forall\, x \in P_{LP}$ heißt *optimale Lösung* für ein LP. Ist $P_{LP} = \emptyset$, dann heißt das LP *unzulässig* und *zulässig* wenn $P_{LP} \neq \emptyset$. Gibt es für jedes $x \in P_{LP}$ ein $\bar{x} \in P_{LP}$, so dass $c^T \bar{x} < c^T x$, dann heißt das LP *unbeschränkt*, sonst *beschränkt*.

Ist ein LP beschränkt und zulässig, so ist jede optimale Lösung x^* stets ein Extremalpunkt des Polyeders P_{LP}, wie durch Dantzig [12] gezeigt wurde. Die Lösung eines linearen Programms lässt sich in polynomieller Zeit ermitteln. Der erste Algorithmus, welcher dies sicherstellte, war das 1979 vom russischen Mathematiker Leonid Khachiyan weiterentwickelte Ellipsoid-Verfahren (siehe Khachiyan [23]). Die am häufigsten verwendete Methode zur Lösung von LPs geht jedoch auf das 1947 von Dantzig [12] entwickelte Simplex-Verfahren zurück. Das Simplex-Verfahren hat zwar in der Theorie eine exponentielle Laufzeit, weist jedoch in der Praxis meist eine bessere Laufzeit auf als andere Verfahren. Geometrisch ist das Simplex-Verfahren als eine Abfolge sich bezüglich der Zielfunktion sukzessiv verbessernder Extremalpunkte eines Polyeder anzusehen.

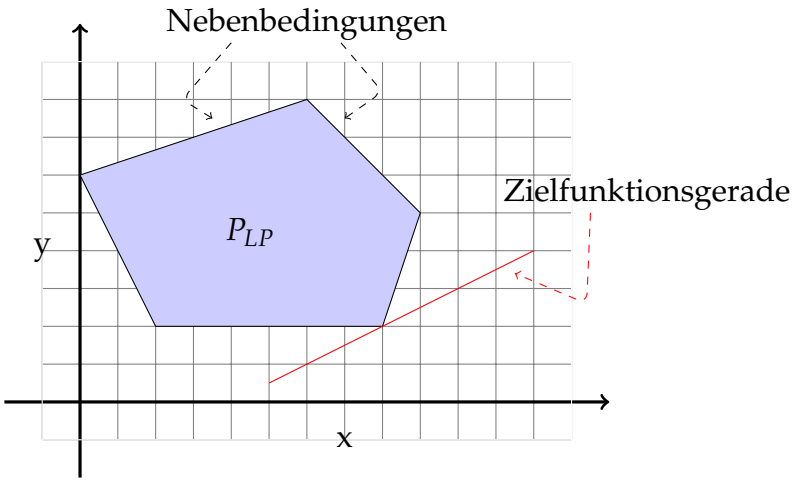

Abbildung 3: Darstellung eines LP im $\mathbb{R}^2$

2.3.3 Ganzzahlige lineare Programmierung

Ein *ganzzahliges lineares Optimierungsproblem* (oder auch *ganzzahliges lineares Programm*, englisch: *integer linear programm*, kurz: ILP) hat grundsätzlich die gleiche Form wie ein LP, nur dass eine zusätzliche, nicht-lineare Nebenbedingung die Ganzzahligkeit aller Variablen fordert. Es ergibt sich als Standardform für ILPs:

$$\begin{array}{lll} \min & c^T x & \\ \text{s.d.} & Ax & \leq b \\ & x & \in \mathbb{Z}_+. \end{array}$$

Das LP, welches man durch Weglassen der Ganzzahligkeitsbedingung eines ILP erhält, wird als *LP-Relaxierung* des ILP (kurz: RP) bezeichnet. Ist x^* eine ganzzahlige optimale Lösung für die Relaxierung eines ILP, so ist x^* auch eine optimale Lösung für das ILP. Im Folgenden bezeichnet

$$P_{RP} := \{\, x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0 \,\}$$

die Menge der zulässigen Punkte der Relaxierung eines ILP,

$$L_{ILP} := P_{RP} \cap \mathbb{Z}^n$$

die Lösungsmenge eines ILP und

$$P_{ILP} := conv(L_{ILP})$$

die konvexe Hülle der zulässigen ganzzahligen Lösungen oder auch *ganzzahlige Hülle* des ILP. P_{RP} ist somit eine Formulierung für L_{ILP}, die wir auch als die *natürliche*

Formulierung bezeichnen, und P_{ILP} ist die ideale Formulierung für L_{ILP}.

Ist P_{RP} bereits *ganzzahlig*, d.h. jeder Extremalpunkt von P_{RP} ist ganzzahlig, so lässt sich ein ILP mit den Methoden für allgemeine lineare Programme in polynomieller Zeit lösen, da die optimale Lösung der Relaxierung ein Extremalpunkt von P_{RP} ist, welcher in diesem Fall die Ganzzahligkeitbedingung des ILP erfüllt. Wichtig ist in diesem Zusammenhang ein Begriff aus der linearen Algebra:

(2.22) Definition (Totale Unimodularität)
Sei $A \in \mathbb{Z}^{m \times n}$ eine Matrix. Dann heißt A *total unimodular*, falls für jede quadratische Untermatrix B von A gilt:

$$det(B) \in \{-1, 0, 1\}.$$

Eine weitere Charakterisierung der totalen Unimodularität liefert der folgende Satz.

(2.23) Satz (Ghouila-Houri [16])
Sei $A \in \mathbb{Z}^{m \times n}$ eine Matrix. Dann ist A genau dann total unimodular, wenn es für jede Auswahl an Spalten $J \subseteq \{1, \ldots, n\}$ eine Partition $J = J_1 \cup J_2,\ J_1 \cap J_2 = \emptyset$ gibt, so dass für die Differenz der Summen der Spaltenvektoren aus J_1 und J_2 gilt:

$$\sum_{j \in J_1} a_j - \sum_{j \in J_2} a_j \in \{-1, 0, 1\}^m.$$

Weiter gilt:

(2.24) Satz (Hoffman und Kruskal [20])
Sei $A \in \mathbb{Z}^{m \times n}$ eine Matrix. Dann ist A genau dann total unimodular, wenn der Polyeder

$$P = \{\, x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0 \,\}$$

ganzzahlig ist, d.h. nur ganzzahlige Extremalpunkte besitzt.

Dies ist jedoch in der Regel nicht der Fall und das Lösen eines ILP ist im Allgemeinen nicht einfach. Wie Schrijver [32] zeigt, ist das Problem aus komplexitätstheoretischer Sicht NP-vollständig. Zur Lösung eines ILPs eignet sich allgemein unter Anderem das von Land und Doig [29] sowie von Dakin [10] entwickelte *Branch&Bound*-Verfahren.

Beim Branch&Bound-Verfahren wird zu einem gegebenen ILP zunächst die LP-Relaxierung gelöst. Ist diese bereits ganzzahlig, so ist die Lösung x^* auch für das

ILP zulässig und dieses ist gelöst. Anderenfalls wird das Problem durch Hinzufügen von Nebenbedingungen in endlich viele disjunkte Subprobleme unterteilt, wobei die zusätzlichen Nebenbedingungen sowohl die Subprobleme unterscheiden als auch den fraktionalen Extremalpunkt abschneiden. Im klassischen Branch&Bound-Verfahren reicht eine Fallunterscheidung nach einer in der erhaltenen Lösung nicht ganzzahligen Variable x_i aus. Es werden zwei Teilprobleme konstruiert, die jeweils eine zusätzliche Nebenbedingung enthalten:

$$x_i \leq \lfloor x_i^* \rfloor \quad \text{bzw.} \quad x_i \geq \lfloor x_i^* \rfloor + 1.$$

Für die Subprobleme wird iterativ wie für das Ausgangsproblem verfahren. Diese fortlaufenden Verzweigungen lassen sich als Baum darstellen, dem sogenannten *Branch&Bound-Baum*. Ist die Lösung eines Teilproblems ganzzahlig, so stellt deren Lösungswert eine *globale obere Schranke* (kurz: *GOS*) für das Ausgangsproblem dar. Andererseits ist jeder Lösungswert einer nicht ganzzahligen Lösung eines Teilproblems eine *lokale untere Schranke*. Ist diese für ein Teilproblem bereits größer oder gleich der bisher kleinsten gefundenen GOS, so muss für dieses Teilproblem nicht weiter verfahren werden. Ebenso werden unzulässige Teilprobleme nicht weiter betrachtet.

Ist schließlich kein Teilproblem mehr zu betrachten, so stellt die kleinste gefundene GOS die optimale Lösung des Anfangsproblems dar.

Wenn man das Hinzufügen zusätzlicher Nebenbedingungen im Branch&Bound-Verfahren als sukzessives Stärken der Formulierung hin zur konvexen Hülle (der Teilprobleme) betrachtet, führt dies zum *Branch&Cut*-Verfahren, einer Verfeinerung des Branch&Bound-Verfahrens.

Im Branch&Bound-Verfahren ist die Geschwindigkeit dieser Stärkung zunächst nicht klar und kann in schlechten Fällen in nur sehr kleinen Schritten voranschreiten. Im Branch&Cut-Verfahren wird daher versucht, schon mit einer möglichst guten Formulierung zu starten. Hierbei sucht man zusätzliche Nebenbedingungen, die möglichst viele zulässigen nicht-ganzzahligen Punkte abschneiden, jedoch alle zulässigen ganzzahligen Punkte beibehalten. Solche gültigen Ungleichen werden auch als *Schnittebenen* oder *Schnitte* (englisch: *cut*) bezeichnet. Es existieren Verfahren mit denen sich gegebene Formulierungen sukzessiv verbessern lassen.

2.3.4 Chvátal-Gomory Schnitte

Eine wichtige Methoden zur Erzeugung von Schnittebenen ist die *Chvátal-Gomory Prozedur* (siehe Gomory [18]), deren Entwicklung entscheidend zu der Leistungsfähigkeit heutiger Software zur Lösung von ILPs beigetragen hat. Die Methode nutzt

die Ganzzahligkeitsbedingung der Variablen aus und besteht aus drei Schritten. Sei hierzu ein ILP gegeben und sei $P = \{ x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0 \}$ eine Formulierung für L_{ILP}.

(2.25) Chvátal-Gomory Prozedur:

(1) Ist $u \in \mathbb{R}^m_{\geq 0}$ ein Gewichtsvektor, dann ist die Ungleichung

$$u^T Ax \;=\; \sum_{j=1}^{n} \Big((u^T a_j)\, x_j \Big) \;\leq\; u^T b$$

gültig für P, da $u \geq 0$ und $Ax \leq b$.

(2) Die Ungleichung

$$\sum_{j=1}^{n} \Big(\Big\lfloor u^T a_j \Big\rfloor\, x_j \Big) \;\leq\; u^T b$$

ist gültig für P, da $x \geq 0$.

(3) Die Ungleichung

$$\sum_{j=1}^{n} \Big(\Big\lfloor u^T a_j \Big\rfloor\, x_j \Big) \;\leq\; \Big\lfloor u^T b \Big\rfloor$$

ist gültig für P_{ILP}, da $x \in \mathbb{Z}^n$.

Die Ungleichung aus (3) ist ein sogenannter *Chvátal-Gomory Schnitt*. Definiere nun eine Folge von Formulierungen mittels:

$$\begin{aligned} P^{(0)} &:= P_{RP} \\ P^{(k+1)} &:= \bigcap_{u \geq 0} \Big\{ x \in P^{(k)} \;\Big|\; \sum_{j=1}^{n} \Big(\Big\lfloor u^T a_j \Big\rfloor\, x_j \Big) \;\leq\; \Big\lfloor u^T b \Big\rfloor \Big\}, \quad k \geq 0. \end{aligned}$$

Dann ist $P^{(k)}$ der k-te *Chvátal-Gomory Abschluss* von P_{RP}. Die im k-ten Chvátal-Gomory Abschluss erzeugten Chvátal-Gomory Schnitte heißen Chvátal-Gomory Schnitte *k-ter Ordnung*. Für jedes $k \in \mathbb{N}_0$ ist $P^{(k)}$ wieder ein Polyeder (siehe Schrijver [32]). Tatsächlich gilt:

(2.26) Satz (Chvátal [6] und Schrijver [31])
Sei P ein Polyeder. Dann kann jede gültige Ungleichung für $X = P \cap \mathbb{Z}^n$ durch endlich viele Wiederholungen der Chvátal-Gomory Prozedur erzeugt werden. Das heißt, es existiert ein $k \in \mathbb{N}_0$ mit

$$P_{ILP} = P^{(k)} \subseteq P^{(k-1)} \subseteq \ldots \subseteq P^{(1)} \subseteq P^{(0)} = P_{RP}.$$

Das kleinste $k \in \mathbb{N}_0$ mit $P_{ILP} = P^{(k)}$ heißt auch *Chvátal-Rang* von P.

Die zu betrachtende Menge von Chvátal-Gomory Schnitten lässt sich weiter eingrenzen. Es ist hinreichend nur solche Chvàtal-Gomory-Schnitte zu betrachten, die nicht durch andere Chvàtal-Gomory-Schnitte dominiert werden. KUTSCHKA [28] beweist hierzu folgedes Lemma.

(2.27) Lemma
Zu jedem Chvátal-Gomory Schnitt mit Gewichtvektor $u \in \mathbb{R}_+^m$ existiert ein Chvátal-Gomory Schnitt mit Gewichtsvektor $u' \in [0,1)^m$, welcher gleich ersterem ist oder diesen dominiert.

Beweis:

Setze $u' := u - \lfloor u \rfloor$. Dann gilt folgende Abschätzung:

$$\begin{aligned} \lfloor u^T A \rfloor x &= \lfloor (u - \lfloor u \rfloor + \lfloor u \rfloor)^T A \rfloor x = \lfloor (u - \lfloor u \rfloor)^T A \rfloor x + \left\lfloor \lfloor u \rfloor^T A \right\rfloor x \\ &= \lfloor u'^T A \rfloor x + \lfloor u \rfloor^T A x \leq \lfloor u'^T b \rfloor + \lfloor u \rfloor^T b \\ &= \lfloor (u - \lfloor u \rfloor)^T b \rfloor + \left\lfloor \lfloor u \rfloor^T b \right\rfloor \leq \lfloor u^T b \rfloor \end{aligned}$$

für alle $x \in X$. Also wird die Ungleichung $\lfloor u^T A \rfloor \leq \lfloor u^T b \rfloor$ durch die Ungleichung $\lfloor u'^T A \rfloor \leq \lfloor u'^T b \rfloor$ dominiert oder beide Ungleichungen sind gleich. Es genügt also Chvátal-Gomory Schnitte mit Gewichtsvektor $u' \in [0,1)^m$ zu betrachten. □

Einen Chvátal-Gomory Schnitt mit $u \in \{0, \frac{1}{2}\}^m$ nennt man auch $\{0, \frac{1}{2}\}$-*Schnitt*. Für den Fall beschränkter Variablen in einem ILP, d.h. zusätzlich zu $x \geq 0$ existiert ein $ub \in \mathbb{Z}^n$ mit $x \leq ub$, haben GENTILE ET AL. [15] gezeigt, dass man ebenso jede gültige Ungleichung für $X = P \cap \mathbb{Z}^n$ durch endlich viele Wiederholungen der Chvátal-Gomory Prozedur erzeugen kann, wenn für jeden Abschluss lediglich Gewichtsvektoren $u \in \{0, \frac{1}{2}\}^m$ verwendet werden.

Der Vollständigkeit halber sei erwähnt, dass sich obige Ausführungen auch auf den gemischt-ganzzahligen Fall verallgemeinern lassen. Im Kontext dieser Arbeit reicht es jedoch aus den rein ganzzahligen Fall zu betrachten.

§3 Kostenminimierung in Multi-Interface Drahtlosnetzwerken

In der modernen Welt sind Telekommunikationsnetzwerke allgegenwertig. Viele der teilnehmenden Endgeräte besitzen viele verschiedene Schnittstellen (wie etwa W-LAN oder Bluetooth) über die eine Kommunikation untereinander möglich ist. Um eine Verbindung zwischen zwei Endgeräten herzustellen, ist es notwendig, dass eine Schnittstelle existiert, welche an beiden Endgeräten aktiviert ist. Welche Schnittstelle zur Verbindung zwischen zwei Endgeräten in der Praxis genutzt wird, hängt von vielen Faktoren ab, wie zum Beispiel Verfügbarkeit, Reichweite oder auch Energieverbrauch durch die Nutzung einer Schnittstelle. Da mobile Endgeräte über keine feste Energiequelle verfügen, muss auf letzteren Punkt ein besonderes Augenmerk gelegt werden. Ein möglichst geringer Energieverbrauch ist somit essentiell, um ein Netzwerk möglichst lange aufrecht zu erhalten. Dies führt auf natürliche Weise zu einem Optimierungsproblem:

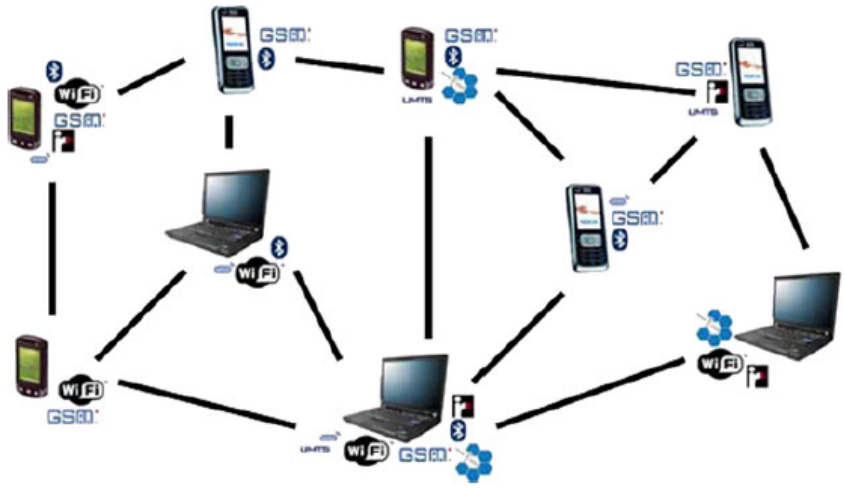

Abbildung 4: CMI in der Praxis (übernommen aus D'Angelo et al. [11])

Gegeben sei ein Netzwerk von mobilen Endgeräten mit $k \in \mathbb{N}$ im Netzwerk verfügbaren Schnittstellen und einer Menge an geforderten Verbindungen zwischen den Endgeräten (vgl. Abbildung 3). Wir reduzieren den Kontext zum Begriff der Mobilität somit auf das Problem der Minimierung des Energieverbrauchs, eine Veränderung der geforderten Kommunikationsverbindungen im Laufe der Zeit betrachten wir nicht.

Die Optimierungsfrage lautet:

An welchen Endgeräten innerhalb des Netzwerks müssen welche Schnittstellen aktiviert werden, um alle geforderten Verbindungen herzustellen und gleichzeitig den gesamten Energieverbrauch innerhalb des Netzwerks zu minimieren?

Dieses Problem ist bekannt als *Kostenminimierung in Multi-Interface Drahtlosnetzwerken* (englisch: *Cost Minization in Multi-Interface Wireless Networks*, kurz: CMI, siehe Koster und Muñoz [26]).

3.1 Mathematische Beschreibung von CMI

Gegeben sei ein Graph $G = (V, E)$, wobei $V = \{1, \ldots, n\}$ die mobilen Endgeräte repräsentiert und die Kanten in E die erforderlichen Verbindungen innerhalb des Netzwerks. Weiter ist mit $I = \{1, \ldots, k\}$, $k \in \mathbb{N}$ die Menge der im Netzwerk verfügbaren Schnittstellen gegeben. Jedem $v \in V$ weist die Funktion $W : V \to 2^I$, $v \mapsto I_v \subseteq I$ die Menge der in v verfügbaren Schnittstellen zu und gesucht ist eine Funktion $W_A : V \to 2^I$, $v \mapsto A_v \subseteq I_v$, die die Menge der in v aktivierten Schnittstellen beschreibt. Eine Verbindung zwischen zwei Endgeräten $v, w \in V$ ist somit genau dann realisiert, wenn $A_v \cap A_w \neq \emptyset$. Schließlich weist eine Funktion $c : I \to \mathbb{N}$, $i \mapsto c(i)$ jeder Schnittstelle $i \in I$ ihre im aktivierten Zustand benötigten Energiekosten $c(i) > 0$ zu. Zusammengefasst ergibt sich folgende Beschreibung von CMI:

Eingabe:	$G = (V, E), k \in \mathbb{N}, W : V \to 2^I, v \mapsto I_v \subseteq I, c : I \to \mathbb{N}, i \mapsto c(i).$
Gültige Lösung:	$W_A : V \to 2^I, v \mapsto A_v \subseteq I_v$ so, dass $$A_v \cap A_w \neq \emptyset \; \forall \, \{v, w\} \in E.$$
Ziel:	Finden einer gültigen Lösung W_A^*, welche unter allen gültigen Lösungen die Gesamtkosten im Netzwerk $$c(W_A) := \sum_{v \in V} \sum_{i \in A_v} c(i)$$ minimiert.

Abbildung 5: Mathematische Beschreibung von CMI

Offenbar hat CMI genau dann eine Lösung, wenn $I_v \cap I_w \neq \emptyset$ für alle $\{v, w\} \in E$. Ob dies der Fall ist, lässt sich mit Laufzeit in $O(|E|)$ überprüfen, indem man alle Kanten des Eingabegraphen auf diese Bedingung hin überprüft. Ohne Beschränkung der Allgemeinheit wird demnach die Existenz einer Lösung im Folgenden vorausgesetzt.

3.2 Bekannte Resultate für CMI

Es ist klar, dass man die optimale Lösung für CMI im Falle $k = 1$ erhält, indem man die einzige vorhandene Schnittstelle in allen Knoten $v, w \in V$ mit $\{v, w\} \in E$ aktiviert. Dies lässt sich mit Laufzeit in $O(|E|)$ bewerkstelligen. Klasing et al. [24] beweisen darüber hinaus, dass CMI für $k = 2$ in $O(|V|^3)$ lösbar ist. Somit ist CMI für $k \leq 2$ in Polynomialzeit lösbar. Jedoch ist CMI APX-hart für $k \geq 3$, wie die Autoren außerdem zeigen. Weiterhin wird ein $(k-1)$-Approximationsalgorithmus vorgestellt und gezeigt, dass CMI, falls $\Delta(G)$ bekannt ist, $\Delta(G)$-approximierbar und APX-hart für $\Delta(G) \geq 5$ ist.

D'Angelo et al. [11] zeigen, dass CMI NP-hart ist für $\Delta(G) \geq 5$ und $k \geq 16$. Darüberhinaus ist CMI in $O((1+b)\ln\Delta)$ approximierbar, falls es für den Graphen $G(V, E)$ eine in polynomieller Zeit berechenbare *b-beschränkte Zugehörigkeitsfunktion* gibt. Eine b-beschränkte Zugehörigkeitsfunktion ist eine Funktion $Own : E \to V$, $\{v, w\} \mapsto u \in \{v, w\}$, so dass

$$\{\, \{v, w\} \in E \mid Own(\{v, w\}) = u \,\} \leq b \quad \forall\, u \in V.$$

Die Autoren zeigen des Weiteren, dass CMI nicht besser als um einen Faktor $\eta\ln\Delta$, für eine bestimmte Konstante η, approximierbar ist. Die Autoren geben außerdem einfache Approximationsalgorithmen mit Approximationsverhältnis $1 + (k-1)\frac{c_{max}}{c_{min}}$ bzw. $\Delta(G)\frac{c_{max}}{c_{min}}$ an. Hierbei ist

$$c_{max} := \max_{i \in I} c(i)$$

und respektive

$$c_{min} := \min_{i \in I} c(i).$$

Schließlich präsentieren sowohl Klasing et al. [24] als auch D'Angelo et al. [11] weitere Resultate für bestimmte Klassen von Graphen. Ist etwa der Eingabegraph ein Baum, so ist CMI in $O(n)$ lösbar, unabhängig von der Anzahl der Schnittstellen $k \in \mathbb{N}$. Ebenso ist CMI in $O(n^2)$ lösbar, falls der Eingabegraph vollständig ist.

In Kapitel 6 werden einige dieser Approximationsalgorithmen noch detaillierter beschrieben und die Güte der von ihnen gelieferten Resultate werden innerhalb einer Rechenstudie geprüft.

3.3 Ganzzahliges lineares Modell

In diesem Kapitel wird eine Modellierung von CMI als ganzzahliges lineares Programm vorgestellt.

Zur Modellierung als ILP definieren wir binäre Variablen $x_{v,i} \in \{0,1\}$ für $v \in V$, $i \in I_v$ und $z_{\{v,w\},i} \in \{0,1\}$ für $\{v,w\} \in E$, $i \in I_v \cap I_w$ mit den Entsprechungen

$$x_{v,i} \stackrel{\wedge}{=} \begin{cases} 1 & i \in A_v, \\ 0 & sonst, \end{cases}$$

$$z_{\{v,w\},i} \stackrel{\wedge}{=} \begin{cases} 1 & i \in A_v \cap A_w, \\ 0 & \text{sonst.} \end{cases}$$

Ist $z_{\{v,w\},i} = 1$, so sagen wir im Folgenden auch, dass die Knoten v und w über die Schnittstelle *i kommunizieren*. Die Minimierung der Gesamtkosten im Netzwerk entspricht damit der Minimierung des Audrucks

$$c(W_A) := \sum_{v \in V} \sum_{i \in I_v} c_i x_{v,i}.$$

Nun ist es notwendig, die Anforderungen an die Lösungen von CMI in Form von Nebenbedingungen zu modellieren. Folgende Nebenbedingungen stellen sicher, dass es für jede Kante mindestens eine Schnittstelle gibt, über die die Endknoten kommunizieren können.

$$\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} \geq 1 \qquad \forall \{v,w\} \in E. \tag{1}$$

Um sicher zu stellen, dass zwei Knoten nur über eine Schnittstelle kommunizieren, wenn diese in beiden Knoten aktiviert ist formulieren wir die Nebenbedingungen

$$\begin{aligned} z_{\{v,w\},i} &\leq x_{v,i} \\ z_{\{v,w\},i} &\leq x_{w,i} \end{aligned} \qquad \forall \{v,w\} \in E, i \in I_v \cap I_w. \tag{2}$$

Ebenso scheint es zunächst sinnvoll, Nebenbedingungen hinzuzufügen, welche sicherstellen, dass eine Kommunikation über eine Schnittstelle, die an beiden Endknoten aktiviert ist, stattfindet:

$$x_{v,i} + x_{w,i} \leq z_{\{v,w\},i} + 1$$

Diese Nebenbedingungen hätten jedoch lediglich Auswirkungen auf die Variablen $z_{\{v,w\},i}$ mit $\{v,w\} \in E,\ i \in I_v \cap I_w$, welche für die Zielfunktion und im übertragenen Sinne für CMI keine Bedeutung haben. Da die für CMI notwendige Konnektivität bereits durch die Nebenbedingungen (1) und (2) gesichert wird, kann auf diese Nebenbedingungen also zur Vereinfachung der ILP-Beschreibung verzichtet werden. Beachte: Die zu Anfang des Unterkapitels dargestellte Entsprechung der Variablen $z_{\{v,w\},i}$ mit $\{v,w\} \in E,\ i \in I_v \cap I_w$ ist somit nicht mehr gültig, da für $i \in A_v \cap A_w$ nicht zwangsweise gilt $z_{\{v,w\},i} = 1$. Ist jedoch eine Verbindung zwischen zwei Knoten realisiert, so existiert mindestens ein $i \in I_v \cap I_w$ mit $z_{\{v,w\},i} = 1$.

Insgesamt ergibt sich damit als ganzzahliges lineares Programm:

$$
\begin{array}{lllll}
\text{(CMI ILP)} & \min & \displaystyle\sum_{v \in V} \sum_{i \in I_v} c_i x_{v,i} & & \\
& \text{s.d.} & \displaystyle -\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} & \leq -1 & \forall\, \{v,w\} \in E \\
& & z_{\{v,w\},i} - x_{v,i} & \leq 0 & \forall\, \{v,w\} \in E,\ i \in I_v \cap I_w \\
& & z_{\{v,w\},i} - x_{w,i} & \leq 0 & \\
& & x_{v,i} & \in \{0,1\} & \forall\, v \in V,\ i \in I_v \\
& & z_{\{v,w\},i} & \in \{0,1\} & \forall\, \{v,w\} \in E,\ i \in I_v \cap I_w
\end{array}
$$

Abbildung 6: CMI als ILP in Standardform

Die Ungleichungen aus der ILP-Beschreibung bezeichnen wir im Folgenden auch als *Modellungleichungen*.

3.3.1 Formale Äquivalenz der Beschreibungen

Im Folgenden zeigen wir die formale Äquivalenz zwischen CMI und (CMI ILP):

(3.1) Satz
Jede Lösung von CMI induziert eine Lösung für (CMI ILP) mit gleichem Lösungswert und umgekehrt.

Beweis:

„⇒“: Sei $W_A : V \to 2^I$, $v \mapsto A_v \subseteq I_v$ eine zulässige Lösung für CMI. Dann gilt per Definition $A_v \cap A_w \neq \emptyset$ für alle $\{v, w\} \in E$. Wähle nun (x, z) so, dass

$$x_{v,i} = \begin{cases} 1 & , i \in A_v, \\ 0 & , \text{sonst}, \end{cases}$$

$$z_{\{v,w\},i} = \begin{cases} 1 & , i \in A_v \cap A_w, \\ 0 & , \text{sonst}. \end{cases}$$

Offenbar erfüllt (x, z) die genannte Bedingung und alle Nebenbedingungen von (CMI ILP), ist also eine zulässige Lösung. Des Weiteren ist

$$c(W_A) = \sum_{v \in V} \sum_{i \in A_v} c(i) = \sum_{v \in V} \sum_{i \in I_v} c_i x_{v,i} \,.$$

Also besitzen W_A und (x, z) den gleichen Lösungswert.

„⇐“: Sei nun (x, z) eine zulässige Lösung für (CMI ILP). Dann existiert für jede Kante $\{v, w\} \in E$ ein $i \in I$ mit

$$z_{\{v,w\},i} = 1, \quad x_{v,i} = 1, \quad x_{w,i} = 1. \tag{3}$$

Definiere nun $W_A : V \to 2^I$, $v \mapsto A_v$ mit

$$A_v = \{i \mid x_{v,i} = 1\} \quad \forall\, v \in V.$$

Da $x_{v,i}$, $v \in V$ nur definiert ist falls $i \in I_v$ gilt $A_v \subseteq I_v \;\forall\, v \in V$. Wegen (3) gilt zudem $A_v \cap A_w \neq \emptyset$ für alle $\{v, w\} \in E$. Es folgt direkt, dass W_A eine zulässige Lösung für CMI ist. Des Weiteren ist

$$\sum_{v \in V} \sum_{i \in I_v} c_i x_{v,i} = \sum_{v \in V} \sum_{i \in A_v} c(i) = c(W_A) \,.$$

Also haben (x, z) und W_A den gleichen Lösungswert. □

Es ergibt sich also unmittelbar aus jeder Lösung für (CMI ILP) auch eine Lösung für CMI mit gleichem Lösungswert. Insbesondere ist eine optimale Lösung für (CMI ILP) auch eine optimale Lösung für CMI. Wegen der Äquivalenz beider Probleme bezeichnen wir im Folgenden (CMI ILP) einfach nur noch als CMI.

3.3.2 CMI mit zwei verfügbaren Interfaces

Wie bereits in Kapitel 3.2 erwähnt, ist bekannt, dass CMI für $k = 2$ in Polynomialzeit gelöst werden kann. Dies lässt sich anhand der ILP-Beschreibung verifizieren:

(3.2) Satz
Die Koeffizientenmatrix A einer CMI Instanz mit $I = \{I_1, I_2\}$ ist total unimodular.

Beweis:

Für den Beweis nutzen wir Satz 2.23. Sei also $J \subseteq \{1, \ldots, n\}$ eine Auswahl an Spaltenindizes von A. Wir partitionieren $J = J_1 \cup J_2$, indem wir J_i die zu den Variablen x_{v,I_i} bzw. $z_{\{v,w\},I_i}$ gehörigen Spaltenindizes aus J zuweisen.

Nun gilt:

- Für die zu den Nebenbedingungen $\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} \geq 1$ gehörigen Zeilen i ist

$$\sum_{j \in J_1} a_{i,j} - \sum_{j \in J_2} a_{i,j} \;=\; \begin{cases} -1 & , I_v \cap I_w = \{I_1\}, \\ 1 & , I_v \cap I_w = \{I_2\}, \\ 0 & , I_v \cap I_w = \{I_1, I_2\}. \end{cases}$$

- Für die zu den Nebenbedingungen $z_{\{v,w\},i} \leq x_{v,i}$, $z_{\{v,w\},i} \leq x_{w,i}$ gehörigen Zeilen i ist

$$\sum_{j \in J_1} a_{i,j} - \sum_{j \in J_2} a_{i,j} \;=\; 0\,.$$

- Für die zu den Nebenbedingungen $x_{v,i} \leq 1$, $z_{\{v,w\},i} \leq 1$ gehörigen Zeilen i ist

$$\sum_{j \in J_1} a_{i,j} - \sum_{j \in J_2} a_{i,j} \;=\; \begin{cases} 1 & , i = I_1, \\ -1 & , i = I_2. \end{cases}$$

Es gilt also

$$\sum_{j \in J_1} a_j - \sum_{j \in J_2} a_j \in \{-1, 0, 1\}^m.$$

und A ist total unimodular. □

Als direkte Folgerung ist die natürliche Formulierung einer CMI Instanz mit $k = 2$ bereits ganzzahlig und kann somit in Polynomialzeit mit Methoden für allgemeine LPs gelöst werden.

§ 4 Polyhedrale Studie des CMI-Polyeders

Vor dem Hintergrund der Beschreibung von CMI als ILP betrachten wir in diesem Kapitel die ganzzahlige Hülle

$$P_{CMI} := conv(\, P_{RCMI} \cap \mathbb{Z}^N \,)$$

von CMI, wobei

$$P_{RCMI} := \Big\{ (x,z) \in [0,1]^N \mid \sum_{i \in I_v \cap I_w} z_{\{v,w\},i} \geq 1 \;\forall \{v,w\} \in E; \\ z_{\{v,w\},i} \leq x_{v,i},\, z_{\{v,w\},i} \leq x_{w,i} \;\forall \{v,w\} \in E, i \in I_v \cap I_w \Big\},$$

die natürliche Formulierung ist und

$$N := \sum_{v \in V} |I_v| \;+\; \sum_{\{v,w\} \in E} |I_v \cap I_w|$$

der Anzahl der Variablen im ILP entspricht.

Wie später in Kapitel 6 aufgezeigt wird, ist ein standardisiertes Branch&Bound-Verfahren nicht bei allen CMI-Instanzen ausreichend, um in angemessener Zeit die optimale Lösung zu erhalten. Wir betrachten somit den Polyeder P_{CMI} mit dem Ziel gültige Ungleichungen oder sogar Facetten zu beschrieben.

4.1 Dimension des Polyeders

Ein wichtiger Schritt bei der Untersuchung eines Polyeders ist die Bestimmung der Dimension. Zur Angabe und des Beweises der Dimension von P_{CMI} benötigen wir folgendes Lemma.

(4.1) Lemma
Sei $P = \{\, x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0 \,\} \neq \emptyset$ ein Polyeder und $S \subseteq eq(P)$. Dann gilt:

$$\dim(P) \leq n - \text{Rang}(A_S).$$

Beweis:

Sei $S \subseteq eq(P)$. Dann ist $\text{Rang}(A_{eq(P)}) \geq \text{Rang}(A_S)$ und mit Satz 2.18 gilt:

$$\dim(P) = n - \text{Rang}(A_{eq(P)}) \leq n - \text{Rang}(A_S).$$

□

Seien nun

$$\begin{aligned}
E_i^1 &:= \Big\{ \{v,w\} \in E \,|\, I_v \cap I_w = \{i\} \Big\} && \forall\ i \in I,\\
E^1 &:= \bigcup_{i=1}^{k} E_i^1 = \Big\{ \{v,w\} \in E \,|\, |I_v \cap I_w| = 1 \Big\}, &&\\
I_v^1 &:= \big\{\ i \in I_v \,|\, v \in V(G[E_i^1])\ \big\} && \forall\ v \in V,\\
I_v^* &:= I_v \setminus I_v^1 && \forall\ v \in V.
\end{aligned}$$

Wir kommen zum Dimensionsbeweis von P_{CMI}.

(4.2) Satz (Dimension von P_{CMI})
Es gilt

$$\dim(P_{CMI}) = \sum_{v \in V} |I_v^*| \ + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w|.$$

Beweis:

Betrachten wir zunächst die Modellungleichungen, für welche Gleichheit gilt auf ganz P_{RCMI}, also diejenigen, welche durch die Zeilen der Matrix $A_{eq(P)}$ induziert werden, wobei A die Koeffizientenmatrix von CMI ist. Die Menge E^1 enthält genau die Kanten, deren Endknoten genau eine gemeinsame Schnittstelle besitzen. Es ist also für jede zulässige Lösung notwendig, dass diese genutzt wird. Somit gilt für jedes (x,z) in P_{RCMI}

$$z_{\{v,w\},i} = 1 \quad \forall\ \{v,w\} \in E_i^1,\ i \in I. \tag{4}$$

Es folgt unmittelbar:

- $x_{v,i} = 1$ $\qquad \forall\ v \in V,\ i \in I_v^1,$ (5)
- $z_{\{v,w\},i} = x_{v,i},\ z_{\{v,w\},i} = x_{w,i}$ $\qquad \forall\ \{v,w\} \in E_i^1,\ i \in I,$ (6)
- $\displaystyle\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} = z_{\{v,w\},i} = 1$ $\qquad \forall\ \{v,w\} \in E_i^1\ i \in I.$ (7)

Sei nun A' die Matrix, welche aus den zu (4)-(7) gehörigen Zeilen der Koeffizientenmatrix A besteht. Offenbar lassen sich die zu (6) und (7) gehörigen Zeilenvektoren als Linearkombinationen der zu (4) und (5) gehörigen Zeilenvektoren darstellen,

während letztere linear unabhängige Einheitsvektoren sind. Die Anzahl der Gleichungen aus (4) entspricht genau der Kardinalität der Menge E^1, während die Anzahl der Gleichungen aus (5) für jede Schnittstelle $i \in I$ der Anzahl an Knoten in dem von E_i^1 erzeugten Teilgraphen von G entspricht. Es gilt also:

$$\text{Rang}(A') = |E^1| + \sum_{v \in V} |I_v^1|$$

und es folgt mit Lemma 4.1:

$$\begin{aligned} \dim(P_{RCMI}) &\leq \sum_{v \in V} |I_v| + \sum_{\{v,w\} \in E} |I_v \cap I_w| - |E^1| - \sum_{v \in V} |I_v^1| \\ &= \sum_{v \in V} |I_v^*| + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w|. \end{aligned} \tag{8}$$

Betrachte nun folgende Vektoren aus P_{CMI}:

- $(x,z)_1 := (1,\ldots,1)$
- $\forall\ \{\overline{v},\overline{w}\} \in E \setminus E^1, j \in I_{\overline{v}} \cap I_{\overline{w}}$:

  $(x,z)_{\{\overline{v},\overline{w}\},j} := (x,z)$ mit $x_{v,i} = 1 \quad \forall\ v \in V, i \in I$

  und $z_{\{v,w\},i} = \begin{cases} 0 & ,i = j \wedge \{v,w\} = \{\overline{v},\overline{w}\}, \\ 1 & ,\text{sonst.} \end{cases}$
- $\forall\ \overline{v} \in V,\ j \in I_{\overline{v}}^*$:

  $(x,z)_{\overline{v},j} := (x,z)$ mit $x_{v,i} = \begin{cases} 0 & ,i = j \wedge v = \overline{v}, \\ 1 & ,\text{sonst,} \end{cases}$

  und $z_{\{v,w\},i} = \begin{cases} 0 & ,i = j \wedge \overline{v} \in \{v,w\}, \\ 1 & ,\text{sonst.} \end{cases}$

Nun gilt mit Lemma 2.17:

$$\begin{aligned} & \begin{pmatrix} (x,z)_1,\ (x,z)_{\{v,w\},i} \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w, \\ (x,z)_{v,i} \text{ für alle } v \in V,\ j \in I_v^* \quad \text{affin unabhängig} \end{pmatrix} \\ \Leftrightarrow \quad & \begin{pmatrix} -\Big((x,z)_{\{v,w\},i} - (x,z)_1\Big) \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w, \\ -\Big((x,z)_{v,i} - (x,z)_1\Big) \text{ für alle } v \in V,\ j \in I_v^* \quad \text{linear unabhängig.} \end{pmatrix} \end{aligned}$$

Betrachte nun die Matrix bestehend aus letzteren Vektoren. Die Vektoren

$$-\Big((x,z)_{\{v,w\},i} - (x,z)_1\Big) \text{ für alle } \{v,w\} \in E \setminus E^1, i \in I_v \cap I_w$$

sind bereits unterschiedliche Einheitsvektoren. Subtrahiert man von den Vektoren

$$-\Big((x,z)_{u,j} - (x,z)_1\Big) \text{ für alle } u \in V,\ j \in I_u^*$$

jeweils die Vektoren

$$-\Big((x,z)_{\{v,w\},i} - (x,z)_1\Big) \text{ mit } i = j \wedge (u \in \{v,w\}),$$

so erhält man jeweils weitere wiederum verschiedene Einheitsvektoren. Somit hat die Matrix vollen Rang. Es folgt die lineare Unabhängigkeit der Vektoren und damit sind die Vektoren

$$(x,z)_1,\ (x,z)_{\{v,w\},i} \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w,\ (x,z)_{v,i} \text{ für alle } v \in V,\ j \in I_v^*,$$

affin unabhängig. Dies sind

$$1 + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w| + \sum_{v \in V} |I_v^*|$$

affin unabhängige Vektoren, also gilt:

$$\dim(P_{CMI}) \geq \sum_{v \in V} |I_v^*| + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w|.$$

Zusammen mit (8) folgt nun wegen $P_{CMI} \subseteq P_{RCMI}$:

$$\begin{aligned} \sum_{v \in V} |I_v^*| + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w| &\leq \dim(P_{CMI}) \leq \dim(P_{RCMI}) \\ &\leq \sum_{v \in V} |I_v^*| + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w|, \end{aligned}$$

also schließlich:

$$\dim(P_{CMI}) = \sum_{v \in V} |I_v^*| + \sum_{\{v,w\} \in E \setminus E^1} |I_v \cap I_w|.$$

□

4.2 Allgemeine Form facettendefinierender Ungleichungen

Die allgemeine Form einer gültigen Ungleichung für P_{CMI} ist

$$\sum_{v\in V}\sum_{i\in I_v}\alpha_{v,i}\,x_{v,i}+\sum_{\{v,w\}\in E}\sum_{i\in I_v\cap I_w}\beta_{\{v,w\},i}\,z_{\{v,w\},i} \quad\geq\quad \gamma,$$

mit $\alpha_{v,i}\in\mathbb{R}$ für $v\in V$, $i\in I_v$ und $\beta_{\{v,w\},i}\in\mathbb{R}$ für $\{v,w\}\in E$, $i\in I_v\cap I_w$ und $\gamma\in\mathbb{R}$.

Da man die *fixierten* Variablen, also diejenigen, welche durch die Gleichungen aus (4)-(7) bereits festgelegt sind, dabei außer Acht lassen kann, ergibt sich somit

$$\sum_{v\in V}\sum_{i\in I_v^*}\alpha_{v,i}x_{v,i}+\sum_{\{v,w\}\in E\setminus E^1}\sum_{i\in I_v\cap I_w}\beta_{\{v,w\},i}z_{\{v,w\},i} \quad\geq\quad \gamma.$$

Wir leiten folgende Aussage ab:

(4.3) Satz
Für eine facettendefinierende Ungleichung von P_{CMI}

$$\sum_{v\in V}\sum_{i\in I_v^*}\alpha_{v,i}\;x_{v,i}+\sum_{\{v,w\}\in E\setminus E^1}\sum_{i\in I_v\cap I_w}\beta_{\{v,w\},i}\;z_{\{v,w\},i} \quad\geq\quad \gamma \tag{9}$$

gilt genau eine der folgenden Aussagen:

(i) $\alpha_{v,i}\geq 0\;\forall\; v\in V,\, i\in I_v^*$.

(ii) Die Ungleichung lautet $x_{v,i}\leq 1$ mit $v\in V, i\in I_v^*$, bis auf Vielfache oder Addition bzw. Subtraktion impliziter Gleichungen.

Beweis:

Seien $v_0\in V$, $i_0\in I_v^*$ fest gewählt und sei

$$F:=\left\{(x,z)\in P_{CMI}\,|\,\sum_{v\in V}\sum_{i\in I_v^*}\alpha_{v,i}\,x_{v,i}+\sum_{\{v,w\}\in E\setminus E^1}\sum_{i\in I_v\cap I_w}\beta_{\{v,w\},i}\,z_{\{v,w\},i}=\gamma\right\}$$

die von (9) geförderte Facette.

Angenommen $x_{v_0,i_0}=1$ für alle $(x,z)\in F$. Dann wäre $F\subseteq\{(x,z)\in P_{CMI}\,|\,x_{v_0,i_0}=1\}$. $\{(x,z)\in P_{CMI}\,|\,x_{v_0,i_0}=1\}$ ist eine echte Seitenfläche von P_{CMI}, da beispielsweise (x^+,z^+) mit

$$x^+_{v,i}\qquad=\begin{cases}0 & ,v=v_0,\,i=i_0,\\ 1 & ,\text{ sonst,}\end{cases}$$

$$z^{+}_{\{v,w\},i} \quad = 1 \quad \forall \{v,w\} \in E,\, i \in I_v \cap I_w$$

ein Element von P_{CMI}, aber nicht in $\{\, (x,z) \in P_{CMI} \,|\, x_{v_0,i_0} = 1 \,\}$ enthalten ist. Also gilt $F = \{\, (x,z) \in P_{CMI} \,|\, x_{v_0,i_0} = 1 \,\}$, da F laut Voraussetzung eine Facette ist und per Definition nicht strikt in einer anderen echten Seitenfläche enthalten sein kann. Damit ist F definiert durch $x_{v_0,i_0} \leq 1$ und es gilt (ii).

Nehme nun an, es existiert ein $(\overline{x}, \overline{z}) \in F$ mit $\overline{x}_{v,i} < 1$. Dann ist (x^*, z^*) mit

$$x^*_{v,i} \qquad = \begin{cases} 1 & , v = v_0,\, i = i_0 \\ \overline{x}_{v,i} & , \text{sonst} \end{cases}$$

$$z^*_{\{v,w\},i} \quad = \overline{z}_{\{v,w\},i} \quad \forall \{v,w\} \in E,\, i \in I_v \cap I_w$$

ein Element von P_{CMI}. Angenommen $\alpha_{v_0,i_0} < 0$, dann gilt:

$$\sum_{v \in V} \sum_{i \in I^*_v} \alpha_{v,i}\, x^*_{v,i} + \sum_{\{v,w\} \in E \setminus E^1} \sum_{i \in I_v \cap I_w} \beta_{\{v,w\},i}\, z^*_{\{v,w\},i} \quad < \quad \gamma,$$

im Widerspruch dazu, dass (9) eine gültige Ungleichung für P_{CMI} ist. Also gilt $\alpha_{v_0,i_0} \geq 0$ und damit (i). □

4.3 Facettendefinierende Modellungleichungen

Die Kenntnis der Dimension eines Polyeders ist entscheidend für die Bestimmung von facettendefinierenden Ungleichungen. In diesem Unterkapitel nutzen wir die Kenntnis der Dimension des Polyeders und stellen heraus, welche der Modellungleichungen von CMI eine Facette von P_{CMI} definieren.

(4.4) Satz
Die CMI-Modellungleichungen

$$x_{v,i} \leq 1 \quad , v \in V$$

sind genau dann facettendefinierend, wenn $i \in I^*_v$.

Beweis:

Sei $x_{v_0,i_0} \leq 1$ eine Modellungleichung mit $v_0 \in V$, $i_0 \in I^*_{v_0}$. Betrachte die von $x_{v_0,i_0} \leq 1$ geförderte Seitenfläche $S = \{(x,z) \in P_{CMI} \,|\, x_{v_0,i_0} = 1\}$. Dann enthält S

die Vektoren $(x,z)_1$ und $(x,z)_{\{v,w\},i}$ für alle $\{v,w\} \in E \setminus E^1, i \in I_v \cap I_w$, sowie die Vektoren $(x,z)_{v,i}$ für alle $v \in V$, $i \in I_v^*$, bis auf $(x,z)_{v_0,i_0}$. Dies sind $\dim(P_{CMI})$ viele affin unabhängige Vektoren. Da $(x,z)_{v_0,i_0} \notin S$, ist insbesondere $S \neq P_{CMI}$. Also ist $\dim(S) = \dim(P_{CMI}) - 1$ und $x_{v_0,i_0} \leq 1$ definiert laut Satz 2.21 eine Facette von P_{CMI}.

Sei nun $x_{v,i} \leq 1$ eine Modellungleichung mit $v \in V(G[E_i^1])$ und sei $S = \{(x,z) \in P_{CMI} \,|\, x_{v,i} = 1\}$ die von $x_{v,i} \leq 1$ geförderte Seitenfläche. Dann gilt jedoch nach Kapitel 4.1, dass $x_{v,i} \leq 1$ für alle $(x,z) \in P_{CMI}$, also ist $S = P_{CMI}$ keine Facette von P_{CMI}. □

(4.5) Satz
Die CMI-Modellungleichungen

$$z_{\{v,w\},i} \leq x_{v,i},\ z_{\{v,w\},i} \leq x_{w,i} \quad \text{für} \quad \{v,w\} \in E,\ i \in I_v \cap I_w$$

sind genau dann facettendefinierend, wenn $\{v,w\} \in E \setminus E_i^1$.

Beweis:

Auf Grund der Symmetrie der Ungleichungen beschränken wir uns auf den Beweis für die Nebenbedingungen $z_{\{v,w\},i} \leq x_{v,i}$.

Sei also $z_{\{v_0,w_0\},i_0} \leq x_{v_0,i_0}$ eine Modellungleichung mit $\{v_0,w_0\} \in E \setminus E_{i_0}^1$, $i_0 \in I$. Betrachte die geförderte Seitenfläche $S = \{(x,z) \in P_{CMI} \,|\, z_{\{v_0,w_0\},i_0} = x_{v_0,i_0}\}$. Dann enthält S die Vektoren $(x,z)_1$ und $(x,z)_{v,i}$ für alle $v \in V$, $i \in I_v^*$, sowie die Vektoren $(x,z)_{\{v,w\},i}$ für alle $\{v,w\} \in E \setminus E^1, i \in I_v \cap I_w$, bis auf den Vektor $(x,z)_{\{v_0,w_0\},i_0}$. Insbesondere ist daher $S \neq P_{CMI}$. Dies sind $\dim(P_{CMI})$ viele affin unabhängige Vektoren, also ist $\dim(S) = \dim(P_{CMI}) - 1$ und $z_{\{v_0,w_0\},i_0} \leq x_{v_0,i_0}$ definiert laut Satz 2.21 eine Facette von P_{CMI}.

Sei nun $z_{\{v,w\},i} \leq x_{v,i}$ eine Modellungleichung mit $\{v,w\} \in E_i^1$. Dann gilt nach Kapitel 4.1, dass $z_{\{v,w\},i} = x_{v,i} = 1$ für alle $(x,z) \in P_{CMI}$, also ist $S = P_{CMI}$ keine Facette von P_{CMI}. □

(4.6) Satz
Die CMI-Modellungleichungen

$$\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} \geq 1 \quad , \{v,w\} \in E$$

sind genau dann facettendefinierend, wenn $\{v,w\} \in E \setminus E^1$.

Beweis:

Sei $\sum_{i \in I_{v_0} \cap I_{w_0}} z_{\{v_0,w_0\},i} \geq 1$ eine Modellungleichung mit $\{v_0, w_0\} \in E \setminus E^1$. Betrachte die geförderte Seitenfläche $S = \{(x,z) \in P_{CMI} \mid \sum_{i \in I_{v_0} \cap I_{w_0}} z_{\{v_0,w_0\},i} = 1\}$. Wegen $(x,z)_1 \notin S$ ist $S \neq P_{CMI}$. Wähle nun $i_0, i_1 \neq i_0 \in I_{v_0} \cap I_{w_0}$ und betrachte folgende Vektoren aus S:

- $\forall\ \{\overline{v}, \overline{w}\} \in E \setminus E^1, j \in I_{\overline{v}} \cap I_{\overline{w}}$:
  $(x,z)^*_{\{\overline{v},\overline{w}\},j} := (x,z)$ mit $x_{v,i} = 1 \quad \forall\ v \in V, i \in I$

  und $z_{\{v,w\},i} = \begin{cases} & ,(i = j \wedge \{v,w\} = \{\overline{v},\overline{w}\} \neq \{v_0,w_0\}) \\ 0 & \vee (\{\overline{v},\overline{w}\} \neq \{v_0,w_0\} = \{v,w\} \wedge i \neq i_0) \\ & \vee (\{\overline{v},\overline{w}\} = \{v_0,w_0\} = \{v,w\} \wedge i \neq j), \\ 1 & ,\text{sonst.} \end{cases}$
- $\forall\ \overline{v} \in V,\ j \in I^*_{\overline{v}}$:
  $(x,z)^*_{\overline{v},j} := (x,z)$ mit $x_{v,i} = \begin{cases} 0 & ,i = j \wedge v = \overline{v}, \\ 1 & ,\text{sonst,} \end{cases}$

  und $z_{\{v,w\},i} = \begin{cases} & ,(i = j \wedge \overline{v} \in \{v,w\}) \\ & \vee (i_0 \neq i \wedge \{v,w\} = \{v_0,w_0\} \\ 0 & \quad \wedge (j \neq i_0 \vee \overline{v} \notin \{v_0,w_0\})) \\ & \vee (i_1 \neq i \wedge \{v,w\} = \{v_0,w_0\} \\ & \quad \wedge j = i_0 \wedge \overline{v} \in \{v_0,w_0\}), \\ 1 & ,\text{sonst.} \end{cases}$

Nun gilt mit Lemma 2.17:

$$\begin{pmatrix} (x,z)^*_{\{v,w\},i} \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w, \\ (x,z)^*_{v,i} \text{ für alle } v \in V,\ i \in I^*_v \quad \text{affin unabhängig} \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} -\left((x,z)^*_{\{v,w\},i} - (x,z)^*_{\{v_0,w_0\},i_0}\right) \text{ für alle} \\ \quad (\{v,w\},i) \in \left(E \setminus E^1 \times (I_v \cap I_w)\right) \setminus (\{v_0,w_0\},i_0), \\ -\left((x,z)^*_{v,i} - (x,z)^*_{\{v_0,w_0\},i_0}\right) \text{ für alle } v \in V,\ i \in I^*_v \quad \text{linear unabhängig.} \end{pmatrix}$$

Betrachte nun die Matrix bestehend aus letzteren Vektoren. Die Vektoren

$$-\left((x,z)^*_{\{v,w\},i} - (x,z)^*_{\{v_0,w_0\},i_0}\right) \text{ für alle } \{v,w\} \in E \setminus (E^1 \cup \{v_0,w_0\}),\ i \in I_v \cap I_w$$

haben nur Nulleinträge bis auf $z_{\{v,w\},i} = 1$. Die Vektoren

$$-\Big((x,z)^*_{\{v_0,w_0\},i} - (x,z)^*_{\{v_0,w_0\},i_0}\Big) \text{ für alle } i \in (I_v \cap I_w) \setminus \{i_0\}$$

haben wiederum nur Nulleinträge bis auf $z_{\{v_0,w_0\},i} = 1$ und $z_{\{v_0,w_0\},i_0} = -1$. Subtrahiert man von den Vektoren

$$-\Big((x,z)^*_{u,j} - (x,z)^*_{\{v_0,w_0\},i_0}\Big) \text{ für alle } u \in V,\ j \in I^*_u$$

jeweils die Vektoren

$$-\Big((x,z)^*_{\{v,w\},i} - (x,z)^*_{\{v_0,w_0\},i_0}\Big) \text{ mit } i = j \wedge (u \in \{v,w\}) \wedge \{v,w\} \neq \{v_0,w_0\}$$

und von den Vektoren

$$-\Big((x,z)^*_{v_0,i_0} - (x,z)^*_{\{v_0,w_0\},i_0}\Big), \quad -\Big((x,z)^*_{w_0,i_0} - (x,z)^*_{\{v_0,w_0\},i_0}\Big)$$

zusätzlich den Vektor $-\big((x,z)^*_{\{v_0,w_0\},i_1} - (x,z)^*_{\{v_0,w_0\},i_0}\big)$, so erhält man Vektoren, die nur Nulleinträge haben, bis auf $x_{u,j} = 1$. Somit hat die Matrix vollen Rang und die Vektoren sind linear unabhängig. Damit hat man also $\dim(P_{CMI})$ viele affin unabhängige Vektoren, wegen $S \neq P_{CMI}$ gilt $\dim(S) = \dim(P_{CMI}) - 1$ und die Ungleichung $\sum_{i \in I_{v_0} \cap I_{w_0}} z_{\{v_0,w_0\},i} \geq 1$ definiert laut Satz 2.21 eine Facette von P_{CMI}.

Sei nun

$$\sum_{i \in I_v \cap I_w} z_{\{v,w\},i} \geq 1$$

eine Modellungleichung mit $\{v,w\} \in E^1$ und

$$S = \{(x,z) \in P_{CMI} \mid \sum_{i \in I_v \cap I_w} z_{\{v,w\},i} =\geq 1\}$$

die von ihr geförderte Seitenfläche. Dann gilt jedoch nach Kapitel 4.1 die Gleichheit $S = P_{CMI}$ und S ist keine Facette von P_{CMI}. □

(4.7) Satz
Die CMI-Modellungleichungen

$$z_{\{v,w\},i} \geq 0 \quad , \{v,w\} \in E,\ i \in I_v \cap I_w$$

sind genau dann facettendefinierend, wenn $|I_v \cap I_w| \geq 3$.

Beweis:

Sei $z_{\{v_0,w_0\},i_0} \geq 0$ eine Modellungleichung mit $\{v_0,w_0\} \in E$, $i_0 \in I_{v_0} \cap I_{w_0}$, $|I_{v_0} \cap I_{w_0}| \geq 3$. Betrachte die von $z_{\{v_0,w_0\},i_0} \geq 0$ geförderte Seitenfläche $S = \{(x,z) \in P_{CMI} \,|\, z_{\{v_0,w_0\},i_0} = 0\}$ und folgende Vektoren aus S:

- $\forall\ \{\overline{v},\overline{w}\} \in E \setminus E^1, j \in I_{\overline{v}} \cap I_{\overline{w}}$:
  $(x,z)^+_{\{\overline{v},\overline{w}\},j} := (x,z)$ mit $x_{v,i} = 1 \quad \forall\ v \in V, i \in I$
  $$\text{und} \quad z_{\{v,w\},i} = \begin{cases} 0 & ,(i = j \wedge \{v,w\} = \{\overline{v},\overline{w}\}) \\ & \vee\ (i = i_0 \wedge \{v,w\} = \{v_0,w_0\}), \\ 1 & ,\text{sonst.} \end{cases}$$
- $\forall\ \overline{v} \in V,\ j \in I^*_{\overline{v}}$:
  $(x,z)^+_{\overline{v},j} := (x,z)$ mit
  $$x_{v,i} = \begin{cases} 0 & ,i = j \wedge v = \overline{v}, \\ 1 & ,\text{sonst,} \end{cases}$$
  $$\text{und} \quad z_{\{v,w\},i} = \begin{cases} 0 & ,(i = j \wedge \overline{v} \in \{v,w\}) \\ & \vee\ (i = i_0 \wedge \{v,w\} = \{v_0,w_0\}), \\ 1 & ,\text{sonst.} \end{cases}$$

Nun gilt mit Lemma 2.17:

$$\begin{pmatrix} (x,z)^+_{\{v,w\},i} \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w, \\ (x,z)^+_{v,i} \text{ für alle } v \in V,\ i \in I^*_v \quad \text{affin unabhängig} \end{pmatrix}$$
$$\Leftrightarrow \begin{pmatrix} -\left((x,z)^+_{\{v,w\},i} - (x,z)^+_{\{v_0,w_0\},i_0}\right) \text{ für alle} \\ \quad (\{v,w\},i) \in \left(E \setminus E^1 \times (I_v \cap I_w)\right) \setminus (\{v_0,w_0\},i_0), \\ -\left((x,z)^+_{v,i} - (x,z)^+_{\{v_0,w_0\},i_0}\right) \text{ für alle } v \in V,\ i \in I^*_v \quad \text{linear unabhängig.} \end{pmatrix}$$

Betrachte nun die Matrix bestehend aus letzteren Vektoren. Die Vektoren

$$-\left((x,z)^+_{\{v,w\},i} - (x,z)^+_{\{v_0,w_0\},i_0}\right) \text{ für alle } \{v,w\} \in E \setminus E^1,\ i \in I_v \cap I_w$$

haben nur Nulleinträge bis auf $z_{\{v,w\},i} = 1$. Subtrahiert man von den Vektoren

$$-\left((x,z)^+_{u,j} - (x,z)^+_{\{v_0,w_0\},i_0}\right) \text{ für alle } u \in V,\ j \in I^*_u$$

jeweils die Vektoren

$$-\Big((x,z)^{+}_{\{v,w\},i} - (x,z)^{+}_{\{v_0,w_0\},i_0}\Big) \text{ mit } i = j \wedge (u \in \{v,w\}),$$

so erhält Vektoren, die nur Nulleinträge haben bis auf $x_{u,j} = 1$. Somit hat die Matrix vollen Rang und die Vektoren sind linear unabhängig. Damit hat man $\dim(P_{CMI})$ affin unabhängige Vektoren. Wegen $(x,z)_1 \notin S$ ist $S \neq P_{CMI}$ und es gilt $\dim(S) = \dim(P_{CMI}) - 1$. Also definiert die Ungleichung $z_{\{v_0,w_0\},i_0} \geq 0$ laut Satz 2.21 eine Facette von P_{CMI}.

Sei nun $z_{\{v,w\},i} \geq 0$ eine Modellungleichung mit $\{v,w\} \in E$, $i \in I_v \cap I_w$, $|I_v \cap I_w| < 3$. Nach der Lösbarkeitsbedingung des Problems ist $|I_v \cap I_w| \in \{1,2\}$.

Ist $I_v \cap I_w = \{i\}$, so ist $\{v_0,w_0\} \in E^1$ und damit ist $z_{\{v,w\},i} = 1 \neq 0$ für alle $(x,z) \in P_{CMI}$, nach Kapitel 4.1.

Ist $|I_v \cap I_w| = 2$, so ist $I_v \cap I_w = \{I_1, I_2\}$ und es gilt für alle $(x,z) \in P_{CMI}$:

$$z_{\{v,w\},I_1} + z_{\{v,w\},I_2} \geq 1 \quad \Leftrightarrow \quad z_{\{v,w\},I_1} \geq 1 - z_{\{v,w\},I_2}.$$

Wegen $z_{\{v,w\},I_2} \leq 1$ folgt daraus $z_{\{v,w\},I_1} \geq 0$ (analog folgt $z_{\{v,w\},I_2} \geq 0$). Die beiden Ungleichungen sind somit redundant.

In beiden Fällen ist definiert also $z_{\{v,w\},i} \geq 0$ keine Facette von P_{CMI}. □

4.4 Facettendefinierende $\{0, \frac{1}{2}\}$-Schnitte erster Ordnung

Mit dem Ziel weitere Klassen facettendefinierender Ungleichung für P_{CMI} zu finden, wurden für einige Testinstanzen die facettendefinierenden Ungleichungen der ganzzahligen Hülle mit Hilfe der Software PORTA (POlyhedron Representation Transformation Algorithm) von Christof und Löbel [5] bestimmt.

Die resultierenden Ungleichungen waren jedoch so vielfältig, dass es uns nicht möglich war, einzelne Ungleichungen zu Klassen zu verallgemeinern. Es fiel aber auf, dass sich die gefundenen, von den Modellungleichungen abweichenden, Ungleichungen häufig als $\{0, \frac{1}{2}\}$-Schnitte erster Ordnung, also als $\{0, \frac{1}{2}\}$-Schnitte der Modellungleichungen, darstellen ließen.

Dieser Beobachtung folgend wurden weitere Testrechnungen unter Verwendung der in Kapitel 6 eingeführten Testgruppen durchgeführt. Dass das Hinzufügen von $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung zum Lösungspolyeder zur ganzzahligen Lösung der

linearen Relaxierung einer Instanz nicht ausreichend ist, wird dabei anhand vieler Instanzen gezeigt. Die detaillierten Ergebnisse werden in Kapitel 6.2 vorgestellt.

Im Folgenden stellen wir anhand einer einfachen Testinstanz ein Beispiel einer facettendefinierenden Ungleichung vor, welche kein $\{0, \frac{1}{2}\}$-Schnitt erster Ordnung ist. Die Instanz entstammt einer von den in Kapitel 6 eingeführten Testgruppen unabhängigen Testreihe mit 10000 zufälligen Graphen mit 5-7 Konten und 5 Schnittstellen. Die Instanz ist eine von 26 Instanzen aus dieser Testreihe, welche durch das Hinzufügen von $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung nicht ganzzahlig gelöst werden konnte.

Gegeben sei folgende CMI-Instanz:

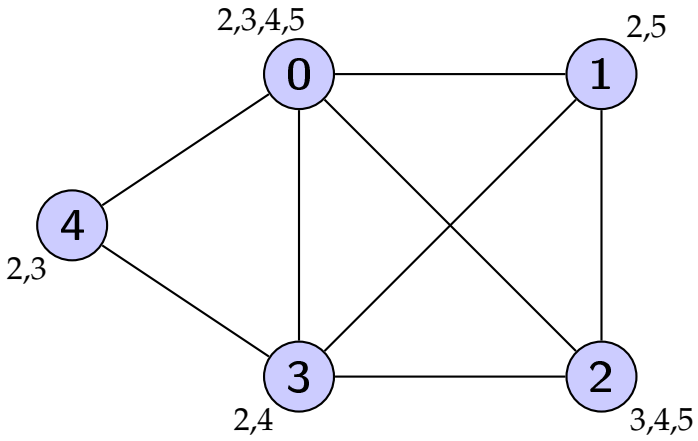

Abbildung 7: CMI-Instanz für Facettenbeispiel

Hierbei stellen die Ziffern an jedem Knoten $v \in \{0, \ldots, 4\}$ die in diesen Knoten verfügbaren Schnittstellen I_v dar.

Die facettendefinierenden Ungleichungen des zugehörigen CMI-Polyeders wurden mit Hilfe von PORTA bestimmt. Unter anderem ergab sich hierbei die Ungleichung

$$2x_{0,2} + x_{0,3} + x_{0,4} + x_{0,5} \geq 3. \tag{10}$$

(4.8) Satz
Die Ungleichung aus (10) lässt sich nicht als $\{0, \frac{1}{2}\}$-Schnitt der CMI-Modellungleichungen der CMI-Instanz aus Abbildung 4.4 darstellen.

Beweis:

Angenommen diese Ungleichung ließe sich als $\{0, \frac{1}{2}\}$-Schnitt erster Ordnung darstellen. Um die rechte Seite von (10) zu gewährleisten muss man bei der $\{0, \frac{1}{2}\}$-Schnitt-Bildung Ungleichungen mit positiver rechter Seite *mit einbeziehen*, d.h. die entsprechenden Gewichte im Vektor u auf $\frac{1}{2}$ setzen. Diese Ungleichungen sind bei dieser Instanz:

(a) $z_{\{0,1\},2} + z_{\{0,1\},5} \geq 1$,

(b) $z_{\{0,2\},3} + z_{\{0,2\},4} + z_{\{0,2\},5} \geq 1$,

(c) $z_{\{0,3\},2} + z_{\{0,3\},4} \geq 1$,

(d) $z_{\{0,4\},2} + z_{\{0,4\},3} \geq 1$,

(e) $z_{\{1,2\},5} \geq 1$,

(f) $z_{\{1,3\},2} \geq 1$,

(g) $z_{\{2,3\},4} \geq 1$,

(h) $z_{\{3,4\},2} \geq 1$.

Dabei ist jedoch zu beachten, dass durch jede gewählte Ungleichung im resultierenden $\{0, \frac{1}{2}\}$-Schnitt auf der linken Seite $z_{\{v,w\},i}$ -Variablen auftreten würden, welche jedoch in (10) nicht vorkommen. Eine Möglichkeit dies auszugleichen, wäre es, für jede $z_{\{v,w\},i}$ -Variable eine Ungleichung der Form $-z_{\{v,w\},i} \geq -1$ mit in den $\{0, \frac{1}{2}\}$-Schnitt einzubeziehen. Jedoch würde dies den Wert der rechten Seite um mindestens so viel senken, wie er durch die Hinzunahme einer der Ungleichungen (a)-(h) erhöht worden wäre. Von daher ist dieses Vorgehen nicht hilfreich.

Die einzige verbleibende Möglichkeit die $z_{\{v,w\},i}$ -Variablen zu entfernen ist es, für jede $z_{\{v,w\},i}$ -Variable eine Ungleichung der Form $x_{v,i} - z_{\{v,w\},i} \geq 0$ mit in den $\{0, \frac{1}{2}\}$-Schnitt einzubeziehen. Ist jedoch hierbei $x_{v,i} \notin \{x_{0,2}, x_{0,3}, x_{0,4}, x_{0,5}\}$, so würde dies wieder dazu führen, dass auf der linken Seite $x_{v,i}$ -Variablen auftauchen, welche in (10) nicht vorkommen. Dies wiederum ließe sich nur durch Hinzunahme einer Ungleichung der Form $-x_{v,i} \geq -1$ korrigieren, was jedoch den Wert der rechten Seite um mindestens so viel senken würde, wie er durch die Hinzunahme einer der Ungleichungen (a)-(h) erhöht worden wäre.

Die einzige sinnvolle Herangehensweise ist es also, von den Ungleichungen (a)-(h) nur jene in den $\{0, \frac{1}{2}\}$-Schnitt mit einzubeziehen, welche durch Ungleichungen der Form $x_{v,i} - z_{\{v,w\},i} \geq 0$ mit $x_{v,i} \in \{x_{0,2}, x_{0,3}, x_{0,4}, x_{0,5}\}$ ausgeglichen werden können. Diese Ungleichungen sind bei dieser Instanz:

(i) $x_{0,2} - z_{\{0,1\},2} \geq 0$,

(ii) $x_{0,2} - z_{\{0,3\},2} \geq 0$,

(iii) $x_{0,2} - z_{\{0,4\},2} \geq 0$,

(iv) $x_{0,3} - z_{\{0,2\},3} \geq 0$,

(v) $x_{0,3} - z_{\{0,4\},3} \geq 0$,

(vi) $x_{0,4} - z_{\{0,2\},4} \geq 0$,

(vii) $x_{0,4} - z_{\{0,3\},4} \geq 0$,

(viii) $x_{0,5} - z_{\{0,1\},5} \geq 0$,

(ix) $x_{0,5} - z_{\{0,2\},5} \geq 0$.

Wie man sieht, kommen somit von den Ungleichungen (a)-(h) nur die Ungleichungen (a)-(d) in Frage. Dies ist jedoch eine Ungleichung zu wenig, um die recht Seite von (10) zu gewährleisten. Somit ist (10) kein $\{0, \frac{1}{2}\}$-Schnitt erster Ordnung. □

Anmerkung: Jedoch ist (10) ein $\{0, \frac{1}{2}\}$-Schnitt zweiter Ordnung:

Bildet man einen $\{0, \frac{1}{2}\}$-Schnitt aus den Ungleichungen (ii)-(vii), (ix), (b)-(d), so erhält man die Ungleichung

$$x_{0,2} + x_{0,3} + x_{0,4} + x_{0,5} \geq 2. \tag{11}$$

Bildet man nun einen $\{0, \frac{1}{2}\}$-Schnitt aus den Ungleichungen (11), (i)-(iii), (v), (vii), (viii), (a), (c), (d), so erhält man

$$2x_{0,2} + x_{0,3} + x_{0,4} + x_{0,5} \geq 3,$$

also genau die Ungleichung (10).

§5 Approximationsalgorithmen für CMI

Da die in Kapitel 4 gefundenen Resultate nicht ausreichen, um eine zeitgerechte Lösbarkeit allgemeiner CMI-Instanzen zu gewährleisten, wenden wir uns im Folgenden Approximationsalgorithmen für CMI zu. Die Laufzeit und die praktische Approximationsgüte der hier vorgestellten Approximationsalgorithmen werden in der anschließenden Rechenstudie untersucht und präsentiert.

5.1 $(k-1)$-Approximationsalgorithmus

Der $(k-1)$-Approximationsalgorithmus wurde von KLASING ET AL. [24] beschrieben. Der Algorithmus sortiert zunächst die im Netzwerk verfügbaren Schnittstellen $i \in I$ nach ihren Kosten $c(i)$, so dass

$$I = \{i_1, \ldots, i_k\}, \quad c(i_1) \leq \cdots \leq c(i_k).$$

Anschließend wird für jede Kante $\{v, w\} \in E$ geprüft ob $i_1 \in I_v \cap I_w$. Ist dies der Fall, so wird i_1 in v und w aktiviert und $E := E \setminus \{\{v, w\}\}$ gesetzt. Diese Prozedur wird für $i_2, \ldots, i_{k-2}$ wiederholt. Abschließend wird die CMI-Teilinstanz

$$G = (V, E'), I' = \{i_{k-1}, i_k\}, W' : V \to 2^{I'}, v \mapsto I'_v := I_v \cap I'$$

mit

$$E' := \{\, \{v, w\} \in E \,|\, \{i_1, \ldots, i_{k-2}\} \cap (i_v \cap I_w) = \emptyset \,\}$$

in Polynomialzeit optimal gelöst und die Lösung $W'_A : V \to 2^{I'}$, $v \mapsto A'_v \subseteq I'_v$ wird mit der bereits vorhandenen Teillösung W_A der Hauptinstanz vereinigt.

Da der Lösungspolyeder einer solchen Instanz, wie in Kapitel 3.3.2 gezeigt, ganzzahlig ist, wurde in der hier verwendeten Implementation, statt dem von KLASING ET AL. [24] beschriebenen Polynomialzeitalgorithmus zum Lösen von CMI-Instanzen mit $|I| = 2$ (siehe Kapitel 3.2) das zugehörige ILP der Teilinstanz mit Hilfe von SCIP gelöst.

1: Sortiere I so, dass $I = \{i_1, \ldots, i_k\}$, $c(i_1) \leq \cdots \leq c(i_k)$.
2: **for all** $j = 1$ **to** $k - 2$ **do**
3: **for all** $\{v, w\} \in E$ **do**
4: **if** $i_j \in I_v \cap I_w$ **then**
5: $A_v = A_v \cup \{i_j\}$
6: $A_w = A_w \cup \{i_j\}$
7: $E = E \setminus \{\{v, w\}\}$
8: **end if**
9: **end for**
10: **end for**
11: Löse ILP für $G = (V, E'), I' = \{i_{k-1}, i_k\}, W' : V \to 2^{I'}, v \mapsto I'_v$
12: **for all** $v \in V$ **do**
13: $A_v := A_v \cup A'_v$
14: **end for**

Abbildung 8: $(k-1)$-Approximationsalgorithmus

Laufzeit: Wir beschreiben die Laufzeit der in Klasing et al. [24] beschriebenen Variante des Algorithmus, da wir über die genaue Laufzeit unserer Variante mit der ILP Lösung keine Aussage treffen können. Der hierdurch resultierende Unterschied in der Laufzeit dürfte jedoch für diese Rechenstudie nicht von Relevanz sein.

Das Sortieren von I in Zeile 1 ist zum Beispiel mit *Quicksort* von Hoare [19] in $O(|I|^2)$ durchführbar. Die Schleife in den Zeilen 2-10 hat wiederum eine Laufzeit in $O((k-2)|E|)$. Die Lösung der Teilinstanz in Zeile 11 ist wie schon in Kapitel 3.2 beschrieben mit Laufzeit in $O(|V|^3)$ zu finden und die Zusammenlegung der Lösungen in den Zielen 12-14 hat eine Laufzeit in $O(|V|)$. Zusammengenommen ergibt sich somit eine worst-case-Laufzeit in $O(\max\{|I|^2, (k-2)|E|, |V|^3\})$.

Approximationsverhältnis: Der $(k-1)$-Approximationsalgorithmus wählt sowohl in jedem Zuweisungsschritts der Schnittstellen $i_1, \ldots, i_{k-2}$, als auch bei der optimalen Lösung der Teilinstanz für die Schnittstellen i_{k-1}, i_k am Schluss die optimale Lösung für das Netzwerk das sich ergibt, wenn man das Ausgangsnetzwerk auf die jeweils betrachteten Kanten beschränkt. Die hierdurch verursachten Kosten sind jeweils durch den Wert der optimalen Lösung für die Ursprungsinstanz beschränkt. Da dies $k-1$ Schritte sind, ergibt sich ebenso $k-1$ als Approximationsverhältnis.

5.2 k-Approximationsalgorithmus

Der für diese Rechenstudie verwendete k-Approximationsalgorithmus ist eine eigens abgeschwächte Variante des in Kapitel 5.1 vorgestellten $(k-1)$-Approximationsalgorithmus. Anstatt zum Schluss für die beiden teuersten Schnittstellen i_{k-1}, i_k die optimale Teillösung zu ermitteln verwendet der k-Approximationsalgorithmus auch für diese Schnittstellen die am Anfang des $(k-1)$-Approximationsalgorithmus genutzte Vorgehensweise.

Wir betrachten diesen Algorithmus zur Überprüfung, wie signifikant die Auswirkung durch die im letzten Schritt des $(k-1)$-Approximationsalgorithmus ermittelten optimalen Lösung für die beiden teuersten Schnittstellen tatsächlich ist.

```
1:  Sortiere I so, dass I = {i_1, ..., i_k}, c(i_1) ≤ ··· ≤ c(i_k).
2:  for all j = 1 to k do
3:     for all {v, w} ∈ E do
4:        if i_j ∈ I_v ∩ I_w then
5:           A_v = A_v ∪ {i_j}
6:           A_w = A_w ∪ {i_j}
7:           E = E \ {{v, w}}
8:        end if
9:     end for
10: end for
```

Abbildung 9: k-Approximationsalgorithmus

Laufzeit: Das Sortieren von I in Zeile 1 ist wie bereits in Kapitel 5.1 beschrieben in $O(|I|^2)$ durchführbar. Der Test auf Verbindung durch die Schnittstellen $i_1, \ldots, i_k$ in den Zeilen 2-10 hat wiederum eine Laufzeit in $O(|I||E|)$. Zusammengenommen ergibt sich somit eine worst-case-Laufzeit in $O(|I|^2|E|)$.

Approximationsverhältnis: Das Approximationsverhältnis von k ergibt sich analog zur Ausführung für das Approximationsverhältnis des $(k-1)$-Approximationsalgorithmus in Kapitel 5.1.

5.3 $(1 + (k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus

Der $(1 + (k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus wurde von D'Angelo et al. [11] beschrieben. Der Algorithmus prüft zunächst, ob es Schnittstellen gibt, welche in jedem Knoten verfügbar sind, d.h. ob

$$\bigcap_{v \in V} I_v \neq \emptyset.$$

Ist dies der Fall, so wird von diesen die kostengünstigsten Schnittstelle in allen Knoten aktiviert. Im anderen Fall, werden in jedem Knoten des Graphen alle verfügbaren Schnittstellen aktiviert.

1: **if** $\bigcap_{v \in V} I_v \neq \emptyset$ **then**
2: $\quad A_v = \left\{ \text{argmin}\{ c(i) \mid i \in \bigcap_{w \in V} I_w \} \right\} \quad \forall\ v \in V$
3: **else**
4: $\quad A_v = I_v \quad \forall\ v \in V$
5: **end if**

Abbildung 10: $(1 + (k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus

Laufzeit: Die Prüfung auf $\bigcap_{v \in V} I_v \neq \emptyset$ in Zeile 1 lässt sich ohne Weiteres mit Laufzeit in $O(|I||V|)$ durchführen. Die anschließende Zuweisung der Lösung hat in beiden möglichen Fällen eine Laufzeit in $O(|V|)$. Zusammengenommen ergibt sich somit eine worst-case-Laufzeit in $O(|I||V|)$.

Approximationsverhältnis: Wir betrachten lediglich den Fall $\bigcap_{v \in V} I_v = \emptyset$, welcher offenbar konstant schlechtere Ergebnisse liefert als der Fall $\bigcap_{v \in V} I_v \neq \emptyset$. Sei W_A die optimale Lösung der betrachteten Instanz. Dann gilt für jeden Knoten $v \in V$:

$$\frac{\sum_{i \in I_v} c(i)}{\sum_{i \in A_v} c(i)} = 1 + \frac{\sum_{i \in I_v \setminus A_v} c(i)}{\sum_{i \in A_v} c(i)} \leq 1 + \frac{|I_v \setminus A_v| c_{max}}{|A_v| c_{min}} \leq 1 + \frac{(k-1) c_{max}}{c_{min}}$$

und damit ergibt sich $1 + \frac{(k-1)c_{max}}{c_{min}}$ als Approximationsverhältnis.

5.4 $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus

Der $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus wurde ebenso von D'Angelo et al. [11] beschrieben und verhält sich zunächst wie der $(1 + (k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus. Der Unterschied besteht in dem Verhalten, falls der Test auf $\bigcap_{v \in V} I_v \neq \emptyset$

negativ ausfällt. Der $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus wählt in diesem Fall für jede Kante $\{v, w\} \in E$ eine Schnittstelle $i \in I_v \cap I_w$ und aktiviert diese in v und w. Der Unterschied zu den in den Kapiteln 5.1 und 5.2 beschriebenen Algorithmen ist hierbei, dass nicht zwangsläufig die günstigste Schnittstelle gewählt wird. Eine Sortierung der Schnittstellen nach den Kosten $c(i)$ ist somit nicht notwendig.

1: **if** $\bigcap_{v \in V} I_v \neq \emptyset$ **then**
2: $\quad A_v = \left\{ \text{argmin}\{ c(i) \mid i \in \bigcap\limits_{w \in V} I_w \} \right\} \quad \forall\, v \in V$
3: **else**
4: $\quad$ **for all** $\{v, w\} \in E$ **do**
5: $\quad\quad$ Wähle $i \in I_v \cap I_w$
6: $\quad\quad A_v = A_v \cup \{i\}$
7: $\quad\quad A_w = A_w \cup \{i\}$
8: $\quad$ **end for**
9: **end if**

Abbildung 11: $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus

Laufzeit: Die Prüfung auf $\bigcap_{v \in V} I_v \neq \emptyset$ in Zeile 1 lässt sich ohne Weiteres mit Laufzeit in $O(|I||V|)$ durchführen. Die anschließende Zuweisung der Lösung hat in ersterem Fall eine Laufzeit in $O(|V|)$ und in letzterem eine Laufzeit in $O(|E||I|)$. Zusammengenommen ergibt sich somit eine worst-case-Laufzeit in $O(|I| \max\{|V|, |E|\})$.

Approximationsverhältnis: Sei W_A die optimale Lösung der betrachteten Instanz. Ist $\bigcap_{v \in V} I_v \neq \emptyset$, so aktiviert der Algorithmus eine Schnittstelle i^* in allen Knoten $v \in V$. Für jeden Knoten erhält man somit als Approximationsverhältnis

$$\frac{c(i^*)}{\sum_{j \in A_v} c(j)} \leq \frac{c_{max}}{c_{min}}.$$

Ist hingegen $\bigcap_{v \in V} I_v = \emptyset$, so aktiviert der Algorithmus im worst-case für jeden Nachbarn eines Knotens $v \in V$ eine Schnittstelle in v. Man erhält als Approximationsverhältnis

$$\frac{\sum_{j \in A'_v} c(j)}{\sum_{j \in A_v} c(j)} \leq \Delta(G)\frac{c_{max}}{c_{min}},$$

wobei A'_v die Menge der vom $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus aktivierten Schnittstellen in v ist. Da, bis auf Graphen mit $E = \emptyset$, die wir hier nicht betrachten wollen, stets gilt:

$$\Delta(G)\frac{c_{max}}{c_{min}} \geq \frac{c_{max}}{c_{min}},$$

ergibt sich $\Delta(G)\frac{c_{max}}{c_{min}}$ als Approximationsverhältnis.

5.5 Modifikation der $(1+(k-1)\frac{c_{max}}{c_{min}})$- und $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmen

Um die von den in den Kapiteln 5.3 und 5.4 beschreibenen Algorithmen gefundenen Lösungen zu verbessern, wurde zusätzlich ein eigener Ansatz zur Modifikation der Algorithmen verfolgt. Hierzu wurde die von den Algorithmen gefundene Lösung W_A betrachtet und für jeden Knoten $v^* \in V$ und jede Schnittstelle $i \in A_{v^*}$ geprüft, ob

$$W'_A : v \to 2^I, v \mapsto \begin{cases} A_v \setminus \{i\} & , v = v^*, \\ A_v & , \text{ sonst,} \end{cases}$$

ebenso eine zulässige Lösung ist und für diesen Fall mit dieser Lösung weiterverfahren.

```
1:  for all v ∈ V do
2:    for all i ∈ I do
3:      if i ∈ A_v then
4:        A_v = A_v \ {i}
5:        for all w ∈ N(v) do
6:          if A_v ∩ A_w = ∅ then
7:            A_v = A_v ∪ {i}
8:            BREAK
9:          end if
10:       end for
11:     end if
12:   end for
13: end for
```

Abbildung 12: Modifikation der Algorithmen

Laufzeit: Die Laufzeit der Modifikation ist anhand der verwendeten Schleifen einfach zu bestimmen und ist offenbar in $O(|V||I||E|)$. Damit ergibt sich sowohl für den modifizierten $(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus, als auch für den modifizierten $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus eine Laufzeit in $O(|V||I||E|)$.

5.6 Vorläufiges Fazit zu den Approximationsalgorithmen

Da sowohl die Laufzeit als auch das Approximationsverhältnis der vorgestellten Algorithmen von jeweils unterschiedlichen Größen abhängen, ergibt sich zunächst

kein klares Bild welcher der Algorithmen den anderen gegenüber allgemein im Vorteil sein könnte. Es ist lediglich klar, dass der $(k-1)$-Approximationsalgorithmus und die modifizierten $(1+(k-1)\frac{c_{max}}{c_{min}})$- und $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmen auf Grund der Konstruktion der Algorithmen keine schlechteren Approximationen liefern werden als ihre jeweils schwächeren Versionen.

Wir erhoffen uns ein deutlicheres Bild durch die Ergebnisse aus der folgenden Rechenstudie.

§ 6 Rechenstudie

Zum Abschluss der Masterarbeit fügen wir der bisherigen theoretischen Arbeit praktische Resultate in Form einer Rechenstudie bei. Wir stellen verschiedene Approximationsalgorithmen und Testgruppen mit insgesamt 18000 CMI-Testinstanzen vor. Anschließend stellen wir anhand der Testgruppen erzielte Laufzeitergebnisse dar und stellen die durch die Approximationsalgorithmen erzielten Ergebnisse den optimalen Werten für die Testinstanzen gegenüber.

Sämtliche vorgestellten Rechenergebnisse wurden auf einem Computer mit 2,93 GHz Intel Xeon Quad-Core-Prozessor, 12 Gb RAM und Betriebssystem Ubuntu in der Version 10.04 durchgeführt.

Sowohl CMI als auch alle in Kapitel 5 vorgestellten Approximationsalgorithmen wurden in der Programmiersprache C++ implementiert. Zur Modellierung und Lösung von ILPs in C++ wurde hierbei die frei erhältliche Software SCIP (siehe Achterberg [2]) in der Version 2.1.1 verwendet.

6.1 Die Testgruppen

Für die Durchführung der Rechenstudie wurden Testgruppen mit je 1000 Instanzen generiert. Die Testgruppen unterscheiden sich in den Merkmalen

- Knotenanzahl,
- Kantendichte,
- Anzahl im Netzwerk verfügbarer Schnittstellen.

Zur Erzeugung zufälliger Graphen wurde das von Gilbert [17] vorgestellte $G(n, p)$-Modell verwendet. Hierbei wird zunächst ein leerer Graph mit n Knoten gebildet, welchem jede mögliche Kante mit einer Wahrscheinlichkeit von p hinzugefügt wird. Ein $G(n, p)$ Graph enthält somit im Erwartungswert $p\frac{n(n-1)}{2}$ Kanten.

Für die Verteilung verfügbarer Schnittstellen W wurde für jede Kante $\{v, w\} \in E$ im Eingabegraphen eine zufällige Schnittstelle $i \in I$ gewählt und den Schnittstellenmengen der Endknoten I_v, I_w hinzugefügt.

```
for all {v, w} ∈ E do
  wähle zufälliges i ∈ I
  I_v = I_v ∪ {i}
  I_w = I_w ∪ {i}
end for
```

Abbildung 13: Schnittstellenverteilung der Testgraphen

So wurde sichergestellt, dass jede Testinstanz zulässig ist.

Als Knotenanzahlen wurden für die Testgruppen die Werte $n \in \{20, 50, 100\}$ gewählt. Die Intention hierbei war es, Netzwerke unterschiedlicher Größen zu simulieren, jedoch eine bestimmte Graphengröße nicht zu überschreiten, um eine angemessene Rechenzeit zu gewährleisten (mehr dazu in den Kapiteln 6.3.1 und 6.3.2). Bei den Anzahlen an im Netzwerk verfügbaren Schnittstellen wurde sich, mit dem Ziel einen eher niedrigen und einen eher hohen realitätsnahen Wert zu wählen, für $k \in \{4, 8\}$ entschieden. Die Kosten $c(i)$ einer Schnittstelle $i \in I$ wurden für jede Testinstanz zufällig aus $\{3, \ldots, 20\}$ gewählt.

Bei der Wahl der Wahrscheinlichkeitswerte p ist es notwendig, Werte in Abhängigkeit von der Knotenanzahl der Graphen zu wählen. Es muss darauf geachtet werden, p nicht zu klein zu wählen, um möglichst keine isolierten Knoten in den Graphen zu erhalten. Einen Richtwerte hierfür liefern Erdős und Rényi [13], welche zeigen, dass ein Graph $G(n, p)$ im wahrscheinlichkeitstheoretischen Sinne fast sicher zusammenhängend ist, falls gilt:

$$p > \frac{\ln n}{n}.$$

Für die hier verwendeten Knotenanzahlen lauten diese Werte

$$\frac{\ln 20}{20} \approx 0,15\,, \quad \frac{\ln 50}{50} \approx 0,08\,, \quad \frac{\ln 100}{100} \approx 0,05\,.$$

Ebenso darf p nicht zu groß gewählt werden, da sonst zwangsläufig die Schnittstellendichte pro Knoten zu groß wird und die Lösung der Instanz unnatürlich vereinfacht würde. Um hier eine Entscheidung zu fällen, wurden für verschiedene p die Wahrscheinlichkeiten für $I_v = I, v \in V$ betrachtet:

$p = 0,06$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	0,000	0,000	**0,381**
$k = 8$	0,000	0,000	**0,000**

$p = 0,09$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	0,000	0,094	**0,711**
$k = 8$	0,000	0,000	**0,011**

$p = 0,10$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	0,000	**0,234**	0,781
$k = 8$	0,000	**0,000**	0,028

$p = 0,12$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	0,000	0,381	**0,875**
$k = 8$	0,000	0,000	**0,093**

$p = 0,15$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	0,000	**0,513**	0,947
$k = 8$	0,000	**0,000**	0,248

$p = 0,20$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	**0,094**	**0,781**	0,987
$k = 8$	**0,000**	**0,028**	0,531

$p = 0,30$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	**0,381**	0,947	0,999
$k = 8$	**0,000**	0,248	0,859

$p = 0,40$	$n = 20$	$n = 50$	$n = 100$
$k = 4$	**0,623**	0,987	1,000
$k = 8$	**0,002**	0,531	0,962

Tabelle 2: Wahrscheinlichkeiten für $I_v = I$

Anhand dieser Ergebnisse wurde entschieden folgende Wahrscheinlichkeiten p für die Testgruppen zu wählen (die entsprechenden Werte in den obigen Tabellen sind fettgedruckt):

Knotenanzahl	Werte für p		
20	0,20	0,30	0,40
50	0,10	0,15	0,20
100	0,06	0,09	0,12

Tabelle 3: Gewählte Wahrscheinlichkeiten p im Überblick

Insgesamt ergeben sich damit folgende Testgruppen:

Testgruppe	n	p	k
1	20	0,20	4
2			8
3		0,30	4
4			8
5		0,40	4
6			8
7	50	0,10	4
8			8
9		0,15	4
10			8
11		0,20	4
12			8
13	100	0,06	4
14			8
15		0,09	4
16			8
17		0,12	4
18			8

Tabelle 4: Überblick über die Testgruppen

6.2 Rechenresultate für $\{0, \frac{1}{2}\}$-Schnitte erster Ordnung

Um die in Kapitel 4.4 geschilderte Beobachtung bezüglich $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung weiter zu verfolgen, wurde die CMI-Implementation in C++ und SCIP unter Verwendung des $\{0, \frac{1}{2}\}$-Schnitt Seperators von Koster et al. [27] so abgeändert, dass zum Finden der ganzzahligen Lösung lediglich $\{0, \frac{1}{2}\}$-Schnitte erster Ordnung gebildet und dem Lösungspolyeder hinzugefügt wurden. Anschließend wurde die lineare Relaxierung des Problems gelöst. Somit konnten Instanzen, für welche das Hinzufügen von $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung zum Erhalt einer optimalen ganzzahligen Lösung nicht ausreicht, dadurch identifiziert werden, dass die resultierende Lösung nicht ganzzahlig war.

Die nachstehende Tabelle veranschaulicht für wie viele der 1000 Testinstanzen in den jeweiligen Testgruppen durch Hinzunahme von $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung und Lösung der linearen Relaxierung keine optimale Lösung gefunden werden konnte.

Testgruppe	Anzahl	Anteil in der Testgruppe
1	10	1,0 %
2	5	0,5 %
3	38	3,8 %
4	94	9,4 %
5	70	7,0 %
6	233	23,3 %
7	62	6,2 %
8	113	11,3 %
9	108	10,8 %
10	386	38,6 %
11	75	7,5 %
12	475	47,5 %
13	131	13,1 %
14	297	29,7 %
15	119	11,9 %
16	621	62,1 %
17	51	5,1 %
18	673	67,3 %

Tabelle 5: $\{0, \frac{1}{2}\}$-Schnitt Ausnahmen in den Testgruppen

Die Rechenergebnisse zeigen, dass es bei relativ kleinen Graphen mit wenig verfügbaren Schnittstellen und wenigen Kanten meist ausreicht, dem Lösungspolyeder $\{0, \frac{1}{2}\}$-Schnitte erster Ordnung als Schnittebenen hinzuzufügen. Bei zunehmender Größe einer Testinstanz ist dies jedoch immer seltener gegeben.

6.3 Rechenresultate für CMI und Approximationsalgorithmen

In diesem Kapitel betrachten wir die durch CMI-Implementation und Approximationsalgorithmen erzielten Rechenresultate. In den Tabellen und Abbildungen dieses Kapitels werden für die in Kapitel 5 eingeführten Approximationsalgorithmen die folgenden Abkürzungen verwendet:

Abkürzung	Approximationsalgorithmus
Apx. 1	$(k-1)$-Approximationsalgorithmus
Apx. 2	k-Approximationsalgorithmus
Apx. 3	$(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus
Apx. 4	Modifizierter $(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus
Apx. 5	$\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus
Apx. 6	Modifizierter $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus

Tabelle 6: Abkürzungen für die Approximationsalgorithmen

6.3.1 Laufzeitschranke der Testinstanzen

Da die Laufzeit mancher Testinstanzen unter Verwendung der normalen CMI-Implementation mehrere Stunden oder gar Tage betrug, wurde schließlich entschieden für die Lösung in SCIP eine Laufzeitschranke von einer Stunde zu setzen, um einen Durchlauf der Testgruppen in angemessener Zeit zu gewährleisten. Dennoch betrug die gesamte Rechenzeit für alle Testgruppen knapp 7 Tage. Wir geben einen Überblick über die Anzahl hiervon betroffener Testinstanzen pro Testgruppe.

Testgruppe	Anzahl	Relativer Anteil
1-14, 17	0	0,0 %
15	1	0,1 %
16	26	2,6 %
18	82	8,2 %

Tabelle 7: Laufzeitschranke-überschreitende Testinstanzen

Selbstverständlich wurden die Ergebnisse der entsprechenden Instanzen bei den in Kapitel 6.3.2 angegebenen Mittelwerten nicht berücksichtigt. Wir geben einen Überblick über die durchschnittliche Differenz der durch SCIP zum Zeitpunkt des Abbruchs gefundenen oberen und unteren Schranken für den Wert der Lösung.

Testgruppe	Differenz	Differenz in % der unteren Schranke
15	56,27	2 %
16	93,90	4 %
18	80,73	4 %

Tabelle 8: Differenz oberer und unterer Schranke

Ebenso stellen wir die durchschnittliche Knotenanzahl dar, welche in SCIP im Rahmen des Branch&Bound-Verfahrens für den Branch&Bound-Baum bis zum Zeitpunkt des Abbruchs erzeugt wurden.

Testgruppe	Knotenanzahl
15	76079
16	30336
18	20205

Tabelle 9: Erzeugte Branch&Bound-Knoten bei Laufzeitschranke

6.3.2 Laufzeitvergleich

Dieses Kapitel stellt die Laufzeitergebnisse der CMI-Implementation und der Approximationsalgorithmen vor. Wir stellen die durchschnittliche Laufzeit Implementationen für die Testgruppen dar.

Testgruppe	Durchschnittliche Laufzeit in Sekunden						
	CMI	Apx. 1	Apx. 2	Apx. 3	Apx. 4	Apx. 5	Apx. 6
1	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
2	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
3	0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
4	0,02	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
5	0,03	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
6	0,17	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
7	0,04	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
8	0,05	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
9	0,28	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
10	1,90	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
11	0,43	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
12	8,73	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
13	0,55	0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
14	2,84	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
15	4,60	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
16	60,12	0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
17	3,01	0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01
18	105,47	0,01	< 0,01	< 0,01	< 0,01	< 0,01	< 0,01

Tabelle 10: Laufzeiten für die Testgruppen

Wie man deutlich sieht, arbeiten sämtliche Approximationsalgorithmen sehr schnell mit einer durchschnittlichen Laufzeit von unter 0,01 Sekunden. Auffällig ist ein wenig der $(k-1)$-Approximationsalgorithmus, dessen durchschnittliche Laufzeit für manche Testgruppen mit Graphen mit 100 Knoten bereits genau 0,01 Sekunden beträgt. Dieser Laufzeitunterschied dürfte jedoch in der Praxisanwendung nur bei sehr großen Instanzen von Relevanz sein.

Die Laufzeiten für CMI zeigen deutlich, dass alle drei Auswahlkriterien einen großen Effekt auf die Laufzeit der Instanzen haben. Die folgende Tabelle zeigt das durchschnittliche Wachstum der Laufzeit in den Testinstanzen bei der Steigerung genau eines Auswahlkriteriums zum nächst höheren Wert mit ansonsten gleichen Parametern. Zum Beispiel geht in die durchschnittliche Steigerung als Folge der Erhöhung der Kantendichte die Laufzeiterhöhung um 75,43% zwischen den Testgruppen 16 und 18 mit ein, deren Instanzen beide über acht Schnittstellen und 100 Knoten verfügen, wobei die für die Kantendichte ausschlaggebende Wahrscheinlichkeit p in Testgruppe 18 mit 0,12 den nächst höheren Wert besitzt als in Testgruppe 16 mit $p = 0,09$.

Knotenanzahl	Kantendichte	Schnittstellenanzahl
2789,09 %	738,92 %	907,41 %

Tabelle 11: Effekte der Auswahlkriterien auf die Laufzeit

Etwas aus dem Rahmen fällt bei den Laufzeitergebnissen die Testgruppe 17, welche durchschnittlich eine geringere Laufzeit aufweist als die Testgruppe 15 mit geringerer Kantendichte und sonst gleichen Parametern. Wir erklären dies mit unserer Wahl von p und der damit für diesen Fall einhergehenden Wahrscheinlichkeit für $I = I_v$ für einen Knoten $v \in V$ von 0,875 (siehe Tabelle 2), dem höchsten Wert innerhalb der gewählten Testgruppen.

Wir betrachten die beiden Testgruppen 16 und 18, welche deutlich die höchsten Laufzeiten aufweisen, genauer im Hinblick auf die Laufzeitenverteilung. Es ergibt sich:

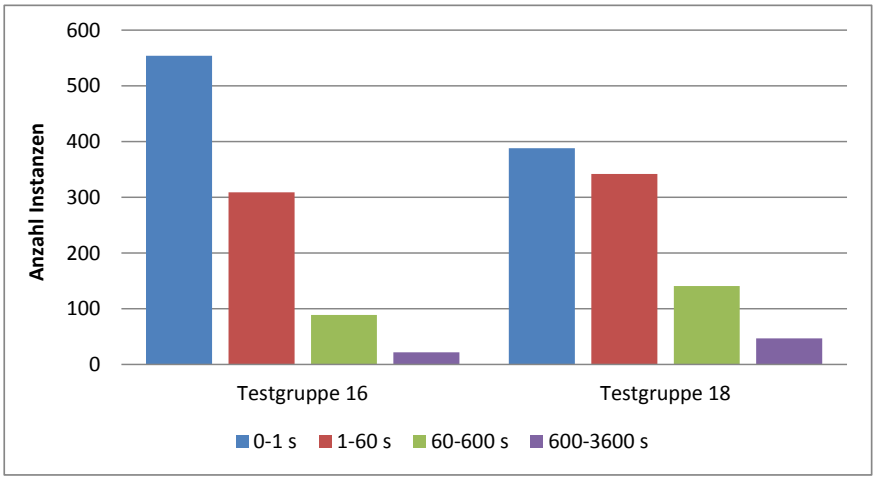

Abbildung 14: Laufzeitenverteilung in den Testgruppen 16 und 18

Man sieht eindeutig, dass die erhöhte durchschnittliche Laufzeit in beiden Testgruppen jeweils von relativ wenigen Testinstanzen mit hoher Laufzeit verursacht wird.

Wir geben außerdem einen Überblick über die durchschnittliche Knotenanzahl, welche in SCIP im Rahmen des Branch&Bound-Verfahrens für den Branch&Bound-Baum erzeugt wurden.

Testgruppe	Anzahl
1	1
2	1
3	1
4	1
5	1
6	3

Testgruppe	Anzahl
7	1
8	1
9	2
10	29
11	3
12	132

Testgruppe	Anzahl
13	7
14	88
15	61
16	667
17	13
18	630

Tabelle 12: Erzeugte Branch&Bound-Knoten

Wie zu erwarten, sind die Laufzeit in den Testklassen und die Anzahl der für den Branch&Bound-Baum erzeugten Knoten stark korreliert. Den im Vergleich zur Testgruppe 15 geringeren Wert in der Testklasse 17 erklären wir uns wie bereits oben. Auffällig ist jedoch außerdem, dass die durchschnittliche Knotenzahl in der Testgruppe 16 höher ist als diejenige in Testgruppe 18. Eine Erklärung für die dennoch höhere Laufzeit in dieser Gruppe bietet möglicherweise die höhere Kantendichte, wodurch die zu lösenden relaxierten LPs im Branch&Bound-Verfahren deutlich mehr Variablen und Nebenbedingungen besitzen als in der Testgruppe 16.

6.3.3 Güte der approximierten Lösungen

In diesem Unterkapitel stellen wir die durch die Approximationsalgorithmen erzielten Werte den durch die CMI-Implementation ermittelten optimalen Werten für die Testinstanzen gegenüber. Anhand der sehr guten Laufzeitergebnisse sämtlicher Approximationsalgorithmen aus Kapitel 6.3.2 ist es offenbar ohne Weiteres mit guter Laufzeit möglich, für eine Instanz alle sechs beschriebenen Approximationsalgorithmen durchzuführen und aus allen erhaltenen Ergebnissen das Beste auszuwählen. Wir verfolgen diese Idee in diesem Unterkapitel weiter und stellen die zugehörigen Ergebnisse unter der Bezeichnung *Minimum-Algorithmus* bzw. *Min.* in den kommenden Tabellen und Abbildungen dar.

Die folgende Tabelle enthält die durchschnittlichen Zielfunktionswerte der Lösungen, welche durch die CMI-Implementation und die Approximationsalgorithmen ermittelt wurden.

Testgr.	Durchschnittliche Zielfunktionswerte der Lösungen							
	CMI	Apx. 1	Apx. 2	Apx. 3	Apx. 4	Apx. 5	Apx. 6	Min.
1	295,18	313,95	318,52	567,66	342,04	374,63	353,07	303,59
2	395,71	434,92	438,00	712,50	446,44	496,14	455,27	416,44
3	273,71	294,95	297,58	672,87	348,70	377,26	359,86	284,10
4	396,87	445,28	446,92	965,76	500,90	551,15	509,08	431,04
5	240,53	263,38	264,12	615,39	316,43	329,41	320,72	247,72
6	352,14	399,42	400,09	1148,32	494,83	554,10	509,76	389,04
7	742,21	779,91	787,59	1623,19	925,96	958,95	920,73	769,96
8	1000,79	1094,94	1099,75	2140,01	1207,18	1334,56	1237,98	1081,65
9	643,63	674,08	676,51	1930,85	916,28	921,14	901,83	665,81
10	911,29	1001,47	1003,13	2823,42	1284,35	1383,11	1295,78	997,62
11	536,32	563,85	564,25	1943,25	840,43	843,77	836,47	552,96
12	740,05	814,50	814,75	3285,65	1189,74	1289,48	1223,07	811,37
13	1431,15	1489,73	1502,37	3373,56	1870,36	1847,18	1803,64	1481,65
14	2006,37	2186,59	2194,45	4691,82	2530,48	2727,36	2553,30	2177,03
15	1194,88	1226,17	1228,57	4111,61	1858,26	1783,14	1765,66	1221,62
16	1681,24	1823,89	1825,69	6170,64	2590,76	2692,46	2539,42	1822,68
17	1018,61	1043,96	1044,30	4308,26	1745,38	1580,19	1577,44	1037,24
18	1388,68	1498,11	1498,43	6843,76	2405,84	2554,45	2417,15	1497,06

Tabelle 13: Durchschnittliche Zielfunktionswerte

Setzt man die Zielfunktionswerte der Approximationsalgorithmen in Relation zu den Werten der CMI-Implementation, so ergibt sich folgendes Bild:

Testgruppe	Durchschnittliches Relationsverhältnis zur CMI-Lösung							
	CMI	Apx. 1	Apx. 2	Apx. 3	Apx. 4	Apx. 5	Apx. 6	Min.
1	1,00	1,06	1,08	1,92	1,16	1,27	1,20	1,03
2	1,00	1,10	1,11	1,80	1,13	1,25	1,15	1,05
3	1,00	1,08	1,09	2,46	1,27	1,38	1,31	1,04
4	1,00	1,12	1,13	2,43	1,26	1,39	1,28	1,09
5	1,00	1,10	1,10	2,56	1,32	1,37	1,33	1,03
6	1,00	1,13	1,14	3,26	1,41	1,57	1,45	1,10
7	1,00	1,05	1,06	2,19	1,25	1,29	1,24	1,04
8	1,00	1,09	1,10	2,14	1,21	1,33	1,24	1,08
9	1,00	1,05	1,05	3,00	1,42	1,43	1,40	1,03
10	1,00	1,10	1,10	3,10	1,41	1,52	1,42	1,09
11	1,00	1,05	1,05	3,62	1,57	1,57	1,56	1,03
12	1,00	1,10	1,10	4,44	1,61	1,74	1,65	1,10
13	1,00	1,04	1,05	2,36	1,31	1,29	1,26	1,04
14	1,00	1,09	1,09	2,34	1,26	1,36	1,27	1,09
15	1,00	1,03	1,03	3,44	1,56	1,49	1,48	1,02
16	1,00	1,08	1,09	3,67	1,54	1,60	1,51	1,08
17	1,00	1,02	1,03	4,23	1,71	1,55	1,55	1,02
18	1,00	1,08	1,08	4,93	1,73	1,84	1,74	1,08
Gesamt	1,00	1,08	1,08	2,99	1,40	1,46	1,39	1,06

Tabelle 14: Relationsverhältnis zum optimalen Zielfunktionswert

Als Erstes fällt auf, dass sowohl der $(k-1)$- als auch der k-Approximationsalgorithmus mit durchschnittlichem Approximationsverhältnis von 1,077 bzw. 1,081 in der Praxis sehr gut angenäherte Ergebnisse liefern. Interessanterweise ergibt sich ebenso, dass die theoretische Überlegenheit des $(k-1)$-Approximationsalgorithmus gegenüber dem k-Approximationsalgorithmus durch die optimale Lösung für die beiden teuersten Schnittstellen in der Praxis kaum von Belang zu sein scheint, da letzterer fast genauso gute Resultate liefert wie sein optimiertes Pendant.

Wie nach Konstruktion des Algorithmus nicht anders zu erwarten, liefert der $(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus das mit Abstand schlechteste Approximationsverhältnis, wohingegen der $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus noch recht passable Ergebnisse aufzuweisen hat. Es zeigt sich außerdem, dass die Modifikation beider Algorithmen tatsächlich eine Verbesserung der Ergebnisse bewirkt. Besonders hervor sticht die Verbesserung durch die Modifikation beim $(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus, wo das Approximationsverhältnis durch die Anwendung der Modifikation mehr als halbiert wird. Beim $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus jedoch scheint die Modifikation nur wenig Verbesserung zu bringen, so

dass die durchschnittlichen Approximationsverhältnisse beider modifizierten Varianten nahezu identisch sind.

Wie die Ergebnisse des Minimum-Algorithmus zeigen, gibt es offenbar Fälle in denen der $(k-1)$-Approximationsalgorithmus von seinen insgesamt schwächeren Gefährten ausgestochen wird. Die Kombination sämtlicher Algorithmen ist demnach die beste Lösung zur Ermittlung einer Näherungslösung einer CMI-Instanz.

Wir gehen einen Schritt weiter und stellen die Verteilung der Approximationsverhältnisse der Algorithmen über allen 18000 Testinstanzen dar.

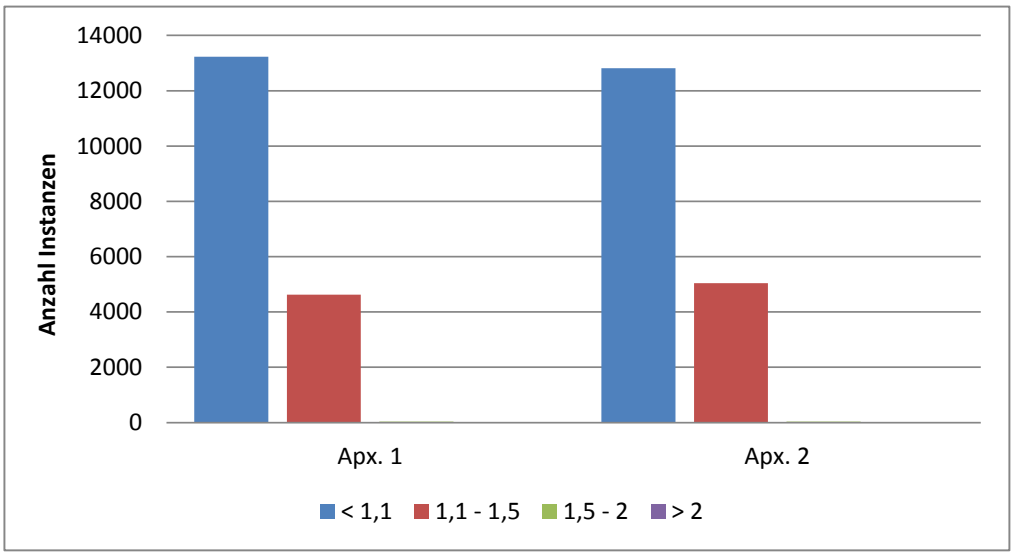

Abbildung 15: Häufigkeiten der Approximationsverhältnisse 1

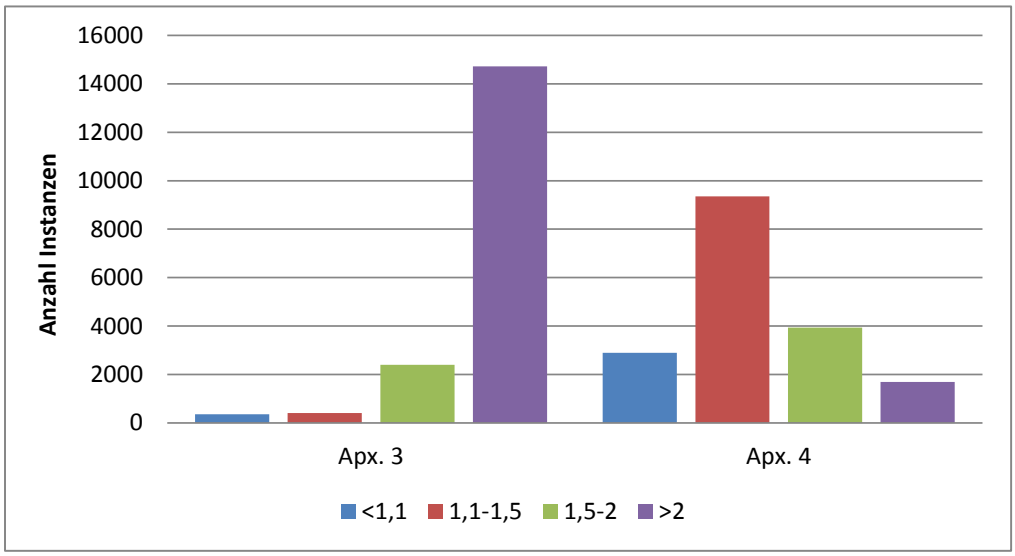

Abbildung 16: Häufigkeiten der Approximationsverhältnisse 2

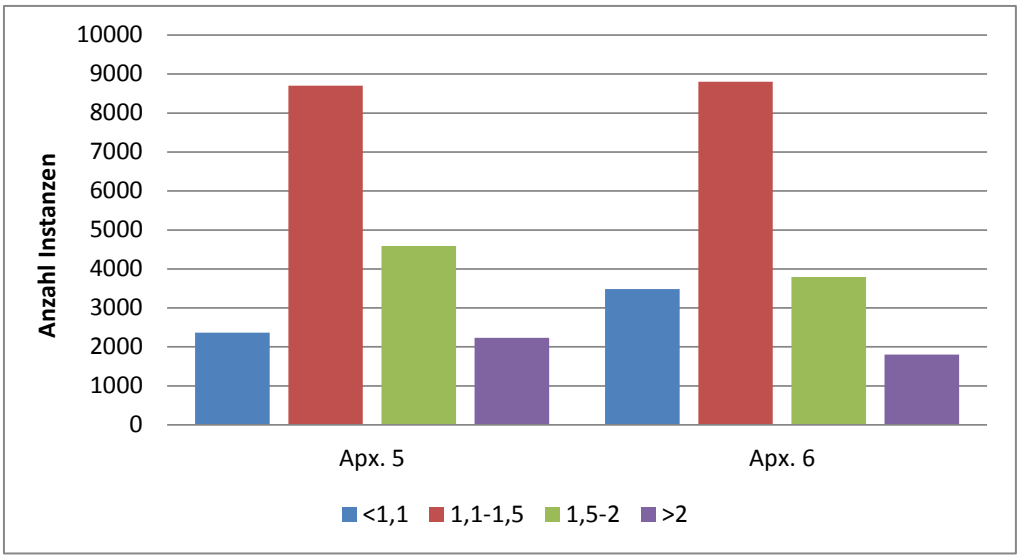

Abbildung 17: Häufigkeiten der Approximationsverhältnisse 3

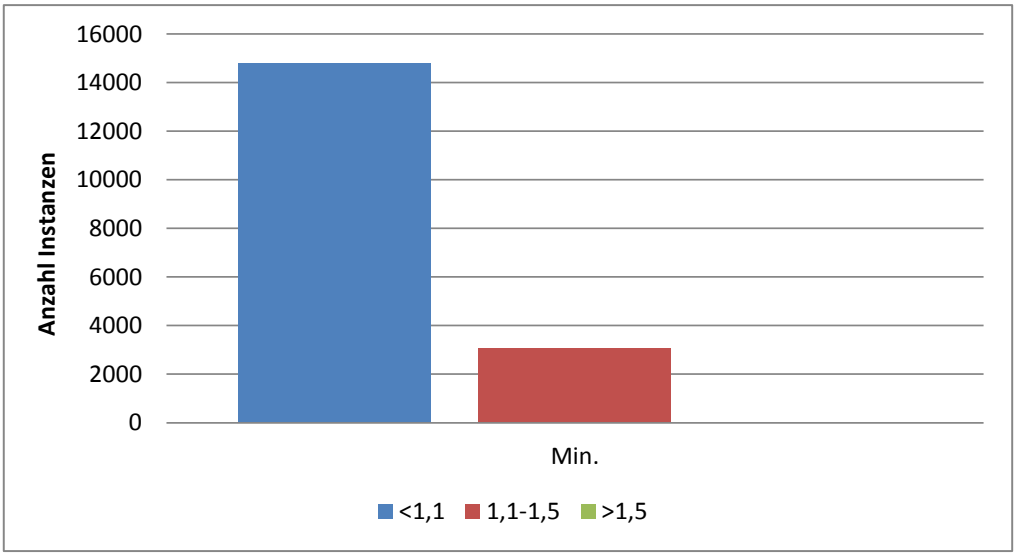

Abbildung 18: Häufigkeiten der Approximationsverhältnisse 4

Die Histogramme ergeben ein genauso deutliches Bild wie bereits Tabelle 6.3.3. Der $(k-1)$- und k-Approximationsalgorithmus liefern in unserer Rechenstudie zu 99,78% ein Approximationsverhältnis von unter 1,5 und in keinem Fall ein Approximationsverhältnis von über 2. Der $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus und die Modifikationen des $(1+(k-1)\frac{c_{max}}{c_{min}})$- und $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmus liefern ein eher durchwachsenes Bild ab, während der $(1+(k-1)\frac{c_{max}}{c_{min}})$-Approximationsalgorithmus für die Praxisanwendung als unbrauchbar erscheint.

Erstaunlicherweise stellt sich heraus, dass der Minimum-Algorithmus für sämtliche Testinstanzen ein Approximationsverhältnis von unter 1,5 liefert. Somit werden der $(k-1)$- und der k-Approximationsalgorithmus also genau bei den Instanzen übertroffen, in denen sie selber ihre am schlechtesten angenäherten Lösungen liefern. Der Minimum-Algorithmus, also die Kombination der in Kapitel 5 vorgestellten Algorithmen eignet sich somit hervorragend für die Bestimmung guter Näherungslösungen für CMI-Instanzen.

6.4 Fazit der Rechenstudie

In dieser Rechenstudie wurden 18 Testgruppen zu je 1000 Testinstanzen mit unterschiedlichen Attributen vorgestellt und verwendet. In Kapitel 6.2 wurde gezeigt, dass sich zwar viele der Instanzen bereits nach Hinzufügen von $\{0, \frac{1}{2}\}$-Schnitten erster Ordnung zum Lösungspolyeder mit Hilfe der linearen Relaxierung und den Methoden für allgemeine LPs lösen lassen, jedoch wurde auch deutlich, dass dies kein verlässliches Mittel ist, um zur optimalen Lösung einer CMI-Instanz zu gelangen.

In Kapitel 6.3.2 wurde gezeigt, dass CMI für kleine Testinstanzen durchaus schnell optimal gelöst werden kann. Jedoch wurde ebenso dargelegt, dass die Rechenzeit für eine optimale Lösung bei größeren Instanzen bereits viele Stunden betragen kann. Auf Grund der allgemein sehr guten Laufzeitergebnisse der Approximationsalgorithmen aus Kapitel 6.3.2 und den Ergebnissen für die Güte der Approximationsalgorithmen aus Kapitel 6.3.3 liegt es daher Nahe, in der Praxis zur Lösung von CMI-Instanzen sämtliche vorgestellten Approximationsalgorithmen zu verwenden und die beste Lösung auszuwählen, falls man nicht auf eine optimale Lösung für eine CMI-Instanz angewiesen ist. Natürlich reicht hierfür die Verwendung des $(k-1)$-Approximationsalgorithmus und der modifizierten $(1+(k-1)\frac{c_{max}}{c_{min}})$- und $\Delta(G)\frac{c_{max}}{c_{min}}$-Approximationsalgorithmen aus, welche auf Grund ihrer Konstruktion ihr jeweils schwächeres Pendant stets dominieren.

§7 Schlusswort

In dieser Arbeit wurde das Problem *Kostenminimierung in Multi-Interface Drahtlosnetzwerken* vorgestellt und mit den Methoden der ganzzahligen linearen Optimierung untersucht. In Kapitel 3 wurde eine mathematische Beschreibung für CMI geliefert sowie bisher bekannte Ergebnisse für dieses Problem vorgestellt. Anschließend wurde eine ILP-Beschreibung für CMI präsentiert.

Im Rahmen der Lösbarkeit von CMI mit dem Branch&Bound- bzw. Branch&Cut-Verfahren wurde in Kapitel 4 eine polyhedrale Studie für die natürliche Formulierung der ILP-Berschreibung durchgeführt. Es wurde die Dimension des Polyeders bestimmt und es wurden Fälle aufgezeigt, für welche Modellungleichungen facettendefinierend sind. Zwar wurde dargelegt, dass allgemeine facettendefinierende Ungleichungen oft als $\{0, \frac{1}{2}\}$-Schnitte der Modellungleichungen dargestellt werden können, jedoch gilt dies nicht für alle facettendefinierender Ungleichungen.

Vor dem Hintergrund der praxistauglichen Lösbarkeit von CMI wurden in Kapitel 5 diverse Approximationsalgorithmen für CMI vorgestellt und ihre theoretischen Laufzeiten und Approximationsverhältnisse präsentiert.

Diesen Ergebnissen wurde in Kapitel 6 eine umfassende Rechenstudie gegenübergestellt, um die Berechenbarkeit von CMI in der Praxis und die Güte von Näherungslösungen mit Hilfe der in Kapitel 5 vorgestellten Approximationsalgorithmen zu testen. Es wurde gezeigt, dass sich relativ große CMI-Instanzen zwar noch in akzeptabler Rechenzeit optimal lösen lassen, dass jedoch eine gute Rechenzeit nicht immer garantiert werden kann und dass die Berechnung einer optimalen Lösung in Extremfällen viele Stunden oder gar Tage dauern kann. Es wurde jedoch aufgezeigt, dass manche der vorgestellten Approximationsalgorithmen, in der Praxis ein gutes Approximationsverhältnis liefern und damit für die praktische Anwendung als durchaus geeignete Alternative erscheinen, wenn eine optimale Lösung einer Instanz nicht notwendig ist.

Offen ist nach dieser Arbeit weiterhin, ob es für die gewählte ILP-Beschreibung von CMI eingrenzbare Klassen gültiger (facettendefinierender) Ungleichungen für P_{CMI} gibt und ob diese die Laufzeit einer CMI-Implementation deutlich reduzieren können.

Denkbar wäre der Ansatz, eine alternative ILP-Beschreibung für CMI aufzustellen und zu untersuchen und durch die dadurch andersartige Struktur facettendefinierender Ungleichungen bei der Analyse von Testinstanzen ein Muster zu entdecken, welches zu verallgemeinerbaren Klassen facettendefinierender Ungleichungen führt.

Weiterhin ist es möglich, dass durch die Kombination der hier vorgestellten und weiterer, hier nicht betrachteter, Approximationsalgorithmen für CMI ein praktisches Approximationsverhältnis geschaffen werden kann, welches das des in Kapitel 6.3.3 betrachteten Minimum-Algorithmus übertrifft.

Kostenminimierung in Multi-Interface Drahtlosnetzwerken ist ein überaus interessantes Problem mit hohem Praxisbezug und großer aktueller Relevanz. Es ist damit sehr wahrscheinlich, dass sich in Zukunft noch weitere Arbeiten mit diesem Thema beschäftigen und hoffentlich weitere interessante Ergebnisse hervorbringen werden.

Literatur

[1] ABRAMSON, N. THE ALOHA SYSTEM: another alternative for computer communications. pages 281–285, 1970.

[2] ACHTERBERG, T. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1): 1–41, 2009. `http://scip.zib.de/`.

[3] BACHEM, A. and GRÖTSCHEL, M. Characterizations of adjacency of faces of polyhedra. *Mathematical Programming at Oberwolfach*, pages 1–22, 1981.

[4] BEUTELSPACHER, A. Lineare Algebra, 2003.

[5] CHRISTOF, T. and LÖBEL, A. PORTA - POlyhedron Representation Transformation Algorithm. `http://typo.zib.de/opt-long_projects/Software/Porta/`.

[6] CHVÁTAL, V. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4): 305–337, 1973.

[7] COOK, S. A. The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[8] COOK, S. A. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[9] COOK, S. A. and RECKHOW, R. A. Time bounded random access machines. *Journal of Computer and System Sciences*, 7(4):354–375, 1973.

[10] DAKIN, R. J. A tree-search algorithm for mixed integer programming problems. *Computer Journal*, 8: 250–255, 1965.

[11] D'ANGELO, G. and DI STEFANO, G. and NAVARRA, A. Minimize the maximum duty in multi-interface networks. *Algorithmica*, 63(1-2): 274–295, 2012.

[12] DANTZIG, G. B. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, pages 339–347, 1951.

[13] ERDŐS, P. and RÉNYI, A. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5: 17–61, 1960.

[14] GAREY, M. R. and JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman&Co, San Francisco, 1979.

[15] GENTILE, C. and VENTURA, P. and WEISMANTEL, R. Mod-2 cuts generation yields the convex hull of bounded integer feasible sets. *SIAM Journal on Discrete Mathematics*, 20(4): 913–919, 2006.

[16] GHOUILA-HOURI, A. Caractérisation des matrices totalement unimodulaires. *Comptes Rendus de l'Académie des Sciences Paris*, 254: 1192–1194, 1962.

[17] GILBERT, E. N. Random graphs. *Annals of Mathematical Statistics*, 30: 1141–1144, 1959.

[18] GOMORY, R.E. Outline of an algorithm for integer solutions to linear programs. *Bulletin American Mathematical Society*, 64: 275–278, 1958.

[19] HOARE, C. A. R. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[20] HOFFMAN, A. J. and KRUSKAL, J. G. Integral boundary points of convex polyhedra. *Annals of Mathematics Studies*, 38: 223–246, 1956.

[21] JANSEN, K. and MARGRAF, M. *Approximative Algorithmen und Nichtapproximierbarkeit*. de Gruyter, 2008.

[22] JONGEN, H. T. and MEER, K. and TRIESCH, E. *Optimization theory*. Kluwer Academic Publishers Boston, 2004.

[23] KHACHIYAN, L. G. A polynomial algorithm in linear programming. 20:191–194, 1979.

[24] KLASING, R. and KOSOWSKI, A. and NAVARRA, A. Cost minimisation in multi-interface networks. *Network Control and Optimization*, pages 276–285, 2007.

[25] KOSTER, A. M. C. A. and KUTSCHKA, M. and RAACK, C. *Next Generation Internet (NGI): Towards robust network design using integer linear programming techniques*. 2010.

[26] KOSTER, A. M. C. A. and MUÑOZ, X. *Graphs and algorithms in communication networks. Studies in broadband, optical, wireless and ad hoc networks.* Springer, 2010.

[27] KOSTER, A. M. C. A. and ZYMOLKA, A. and KUTSCHKA, M. Algorithms to separate $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts. *Algorithmica*, 55: 375–391, 2009.

[28] KUTSCHKA, M. Algorithmen zur Separierung von $\{0, \frac{1}{2}\}$-Schnitten, December 2007.

[29] LAND, A. H. and DOIG, A. G. An automatic method for solving discrete programming problems, 1960.

[30] NEMHAUSER, G.L. and WOLSEY, L.A. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[31] SCHRIJVER, A. On Cutting planes. *Annals of discrete mathematics*, 9: 291–296, 1980.

[32] SCHRIJVER, A. *Theory of integer and linear programming*. Wiley, Chichester, 1986.

[33] TOLLA, P. A survey of some linear programming methods. *Combinatorial optimization*, 1: 157–188, 2010.

[34] VOLKMANN, L. *Graphs on all vertices and edges (Graphen an allen Ecken und Kanten)*. RWTH Aachen, Lehrstuhl II für Mathematik, zweite Version, 2011.

[35] WEYL, H. Elementare Theorie der konvexen Polyeder. *Commentarii Mathematici Helvetici*, 7(1): 290–306, 1935.

Selbstständigkeitserklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate gekennzeichnet zu haben.

Aachen, 27. September 2012

(Sébastien Auroux)

Danksagung

Zunächst einmal möchte ich mich bei Herrn Professor Arie Koster bedanken, welcher mich bei meiner Masterarbeit unterstützt und mir regelmäßig mit wertvollen Hinweisen und Tipps zur Seite gestanden hat. Ich bedanke mich außerdem bei meinem Zweitkorrektor Herrn Professor Berthold Vöcking. Ein großer Dank geht ebenso an die Assistenten des Lehrstuhls II für Mathematik der RWTH Aachen, insbesondere an Martin Tieves und Manuel Kutschka, die mir durch ihr Fachwissen und ihre Anregungen sehr oft weitergeholfen haben.

Ebenfalls bedanke ich mich bei meinen guten Freunden und Bekannten, welche mich während der Erstellung dieser Arbeit motiviert und unterstützt haben. Ein besonderer Dank gilt denjenigen, die sich bereiterklärt haben meine Masterarbeit gegenzulesen, sich nicht gescheut haben, Kritik zu üben und Verbesserungsvorschläge vorzubringen.

Nicht zuletzt gilt mein besonderer und tiefer Dank meinen Eltern, die mir mein Studium ermöglicht und mich stets moralisch unterstützt haben und ohne die ich niemals da wäre, wo ich jetzt bin.

Abschließend bedanke ich mich bei all jenen, die nun nicht namentlich genannt wurden und dennoch einen Beitrag zu dieser Arbeit geleistet haben.