Masterthesis

For obtaining the degree of

Master of Science

Frequency Assignments in Slow Frequency-Hopping GSM Networks

-A MIP Approach-

Second chair of Mathematics, **RWTH Aachen**

Produced by:	Martin Tieves
Matrikel-No.:	273955
Email:	Martin.Tieves@rwth-aachen.de
Presented at:	11/09/2011
Advisor :	Prof. PhD. Koster
Second advisor :	Prof. PhD. Lübbecke

Statutory Declaration

I assure that this thesis is the product of my own work and that it contains no plagiarism. I duly referenced all sources and marked all parts of this work that have been literally or analogously obtained from these sources. Finally, I assure that no work in this style or similar has previously been presented to any other board of examination.

Aachen, den 11/09/2011

_____

Martin Tieves

Acknowledgments

First, I want to thank Prof. PhD. Koster for supporting me whenever I needed his help or advice. Second, I want to thank Prof. PhD. Lübbecke for his valuable input with respect to the topics of column generation, generously sharing his experience with me.
I further want to thank my partner and my family for encouraging and supporting me throughout my studies. Last but not least, I would like to thank everyone who supported me in writing this thesis.

Table of Contents

List of Figures

List of Tables

List of Algorithms

§1 Preface

1.1 — *Introduction* —

This Master's thesis contributes to analysis and improvements in the field of mobile communication by providing a mathematical approach to optimize the quality of frequency assignments in mobile communication networks.
The quality of said networks is determined by factors like signal strength, propagation and signal interference. Signal strength is evidently important when delivering information over a distance, propagation describes the signal availability. For example, a car-radio might fail to receive the signal from a station when the car enters a tunnel. Signal interference occurs when the wavelengths of two or more signals overlap. Staying with the car-radio example, such an overlap would mean that you could hear a noisy combination of several radio stations at once (see section 1.3 for a more detailed annotation). Since exhaustive research on all influencing topics is beyond the scope of a Master's thesis, this thesis concentrates on improvements of signal quality in mobile networks by minimizing interference in a physically existing network.

To be more precise, the focus lies on generating frequency assignments. The mode of a network (soft constraint) is treated and not a networks outer organization (e.g. transceiver positions - hard constraints). These frequency assignments shall be determined as good as possible (which is specialized later on). Especially a mathematically optimal planning method and not a generic/heuristical one shall be used. Due to the high importance of GSM technology (the most used mobile communication standard – compare section 1.2 for more information), these analysis will specialize on GSM networks. Nevertheless, some of these results may be applied to other technologies for radio transmissions as well, since this work rather incorporates common radio transmission features than special technical GSM features. During this thesis, generalizations are given, whenever possible.

Going into detail on frequency assignments, the classic model (where a single frequency is assigned to a single carrier) is not treated, because it has already been thoroughly researched by others (representative, one can mention [Eis01], where exhaustive work on the classic frequency assignment problem is done). The focal point of this approach will be the concept of slow frequency-hopping. This is an assignment model, where multiple frequencies are assigned to a single carrier, as an advancement of the classic model. This challenge is approached with the facilities

and resources of "mixed integer programming", as a tool of choice. Mixed integer programming is chosen, because it is one of the backbones of the author's mathematical education, as well as a key element of the chairs topics. Hereby, the most important tool is provided by the concept of "column generation" (a description is given in section 4.1), which was used (among others) for obtaining the later presented results. In the course of this work, many aspects of [MT95], which performs column generation to address a similar problem, are used.

Addressing readers with little or no experience on this subject, some of the technical terms given above are introduced and explained later on.

Summing the structure of this thesis up, at first a short introduction into the functionalities of mobile communication, with respect to GSM technologies and their characteristics is given. Second, certain problems with capacity and quality restrictions of the GSM technology are pointed out and a possible solution in the form of slow frequency-hopping is offered. This will serve as a motivation for the later work. In the third section, the problem of finding the best frequency assignment for a slow frequency hopping GSM network is aggregated into a mathematical model and transported into the formalism of mixed integer programming. Consequential, this model and it's solutions are assessed and analyzed. This leads to the refinement of splitting the model into two stages, thus enforcing the practical computability.
The first stage involves a column generation approach and is described in the sections four and five. In the sixth chapter, the second stage is presented. The following section (7) gives an overview over the obtained results. Hereby, it is to mention, that no "optimal" assignment could be obtained. Nevertheless, the results are analyzed and the reasons for that drawback as well as positive aspects of these approaches are pointed out. The section concludes with an evaluation of the suitability of the chosen means.
The thesis concluded with section eight, the appendix, where as well some background information and additional work is presented. Finally, an overview of the used literature and some additional relevant work is given, in order to help the reader integrate the information presented in this thesis.

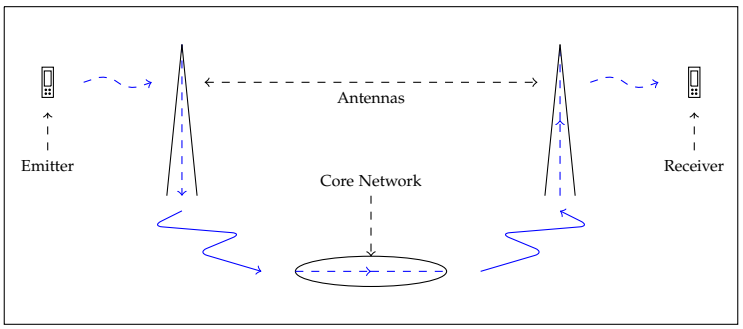

Figure 1: Schematical Routing

1.2 — *Mobile Communication in GSM networks* —

"Global System for Mobile Communications" (GSM) is a technical standard, which specifies interfaces and settings for mobile communication. GSM was originally introduced in the second generation of wireless communication technology. The GSM standard is incorporated and built upon in the third and fourth generation of mobile communication systems as well. It specifies, how two mobile partners communicate via mobile devices, such as mobile phones or laptops, by organizing and standardizing the process of communication within (typically) large scale infrastructures like (public) telephone networks. Without giving to much technical details, figure 1 depicts how both partners communicate: the first participant uses radio transmission to communicate with the nearest network antenna, which forwards the call via "physical" transmission (copper or fiber wire) to the antenna closest to the second participant. This second antenna uses radio transmission again, to communicate with the second participant. A mobile communication between two participants will henceforth be referred to as a "mobile call".

Until recently, the story of GSM has been a story of success and growth. The GSM Association (see [Ass]) estimated that the technologies defined in GSM are used in about 80% of the global mobile market. This market has more than 1.5 billion users across more than 212 countries and territories and has been growing ever since. In the second quarter of 2009, about 3 billion calls have been made with GSM techniques within the GSM Association alone. However, there is a drawback to the extensive use of GSM technology. GSM networks cannot grow (in the sence of capacity) indefinitely because they operate on a limited resource, namely the (finite) frequency spectrum. In principle, each mobil call needs some separate space in this spectrum, such that the amount of simultaneous calls at a certain location is limited.

Figure 2: Typical Antenna

Resulting a highly efficient usage of this resource is recommended for large scale networks.

To reiterate, every mobile call is passed to some antenna, which then forwards the call. Since a call can take place everywhere in a given region equipped with a GSM infrastructure, every spot within the region must have access to at least one antenna. This obviously is the idealized way. Due to territorial issues, it might not be possible to reach an antenna everywhere (e.g. below bridges no connection can be established). Naturally, mobile communication companies try to establish a sufficient degree of coverage and capacity, that is, they build more antennas or acquire additional frequencies (if possible) for their network. Resulting, rooftop-mounted antennas (figure 2) are commonplace now, providing efficient positions for spreading radio signals. In this context, the recent auction of additional frequencies for mobile communications in Germany (see [Bun10] for more information) is to mention. High scored prices, and the political charge to increase the coverage of high speed Internet with these new frequencies emphasize both, the importance of the resource frequency and the need for it's sensible usage.

In a mathematical point of view, a region needs to be covered by transceiver sites (with eventually more than one antenna - for increased capacity), so that an antenna is reachable form every possible spot in this region. For that reason, a GSM network can be described as (digital) cellular network, which will be an important characteristic later on. Some sketch of this structure is given in figure 3. However, this sketch is heavily simplified, in the following sense:
Clearly, the resolution of this model is far to low, there are to few cells, for covering

whole Germany. In reality, it should be clear, that the different sites may not be disjoint, may not cover the whole region or may not be homogeneous in signal strength and spreading. But in general, the figure provides an idea of this structure, especially since this coverage model was in use (in the sense that planning was done with it), in the beginning of mobile communication (see section 2.2.1 for more information of frequency pattern reuse).
At this point, some necessary formalism in the context of this structure, is given. The regional unit seen in figure 3 is called site. At one site, there may be multiple antennas, which serve all calls of the whole site. Each of this antennas defines one sector. For example, one antenna can form one sector (360° opening angel), three antennas may form 3 sectors with an opening angle of 120° each. In each sector, the antenna has certain number of transceivers, abbreviated TRX. Each of whom operates on it's own frequency and is responsible for the communication with the (mobile) participant. Herby, the management, which call is treated by which TRX is not important here. Every TRX communicates in up to 8 different time slots, so it can handle up to 8 different calls (on the same frequency) at the same time. Resulting, a communication is split into discrete time slots, that dignified, that the human ear is not able to recognize it. Concerning this settings, especially the time frames, more details are provided in chapter 2.
Again it holds, that matters are more complex then suggested here, but the general idea is sufficient in this work. Repeating: The concept is, to "cover" a certain area with TRX, but due to territorial influences, the area covered by TRX has no hexagonal shape in reality (the shapes are mostly chaotic, they even need not be connected). This is reasoned by territorial issues (e.g cement may prevent signal expansion), such that the TRX have no uniform area, on which they can serve. Nevertheless, from a theoretical point of view, the hexagonal forms are a useful model to do some research on.

While the above mentioned characteristics shall give a rough overview on a networks organisatorial structure, in the following it is commented on the "radio" aspect of this communication mean. Mobile communications can possible take place anywhere. So they cannot be passed by some hardware like telephone cable (e.g. copper / optical fiber). Therefore these calls are transmitted in the same way, like tv or radio signals, namely in the radio spectrum.
This spectrum spans from zero (physical: extremely long wavelength) to 300 GHz. It is divided into several parts due to the fact that many different users depend on it: radio, television, radar, the military and GSM all have exclusive access to a unique part of this spectrum. To date, it is impractical to use higher frequencies due to

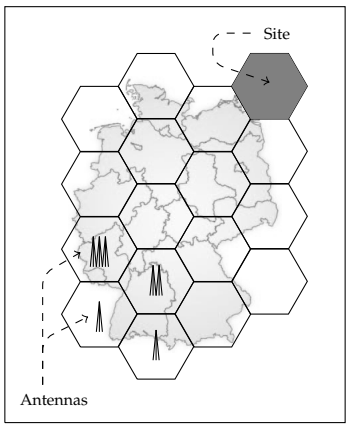

Figure 3: Cellular Structure

the fact that the earth's electromagnetic radiation disrupts signals sent at this wavelength, a technological restriction that may be overcome in the future.
Whereas the exact spectrum of a special purpose (e.g. emergency frequencies) might differ among national borders, in Europe, GSM may generally be used within 880 to 915, 925 - 960 MHz (GSM 900) as well as within 1710 - 1785 and 1805 - 1880 MHz (GSM 1800) (more detailed presented in [oSA09]). The GSM 900 spectrum comprises 124 channels (uplink, outgoing data) between 890 - 915 MHz and 124 channels (downlink, incoming data) between 935 - 960 MHz, each channel having a bandwidth of 200 kHz. Each of these channels operates on eight time slots, such that it can serve eight different calls at once. So, putting it in a nutshell, given the same position and the same time frame, a signal antenna in the GSM 900 spectrum (with enough TRX for every channel) can sustain 992 different calls at the very most. This number is very limited, even when adding the other GSM spectrum's capabilities as well. The conclusion is, that capacity enhancements can only be achieved by increasing the number of antennas (and therefore the TRX operating on the same frequency), thus leading to an "overcrowding" of the frequency spectrum. This leads to significant problems. For example, comparing to normal radio channels: Different mobile calls, placed next to each other in the frequency spectrum (often observed at borders of countries, for example with car radio) disturb each other / produce atmospheric noise. Accordingly, given the history of GSM, which can be described as an ever growing of demanded capacity, there arise some problems at this point. These problems are: increasing the networks capacity correlates with the networks transmission quality, which is carried out in the following section. At the end, thee problems lead to the subject of this work: Given a network (with a certain capacity),

how is it organized at best (by a frequency assignment), with respect to transmission quality?

1.3 — *Transmission quality in radio networks* —

As mentioned before, quality in radio network depends on many different factors. At this point, some examples are more convenient, than a concrete definition of transmission quality (which would lead to a rather technical point of view).
Most of this influencing factors do not only appear in GSM networks, but can be found similarly in normal radio transmissions (e.g. car-radio), which may serve as a parallel. As a first example, landscape (mountains, buildings etc.) can be mentioned. While the selection of special frequencies might reduce the landscape's effect (low frequencies are physically better adapted for steep slopes like in street canyons, high frequencies have a wider fade radius and are therefore better adapted for flat, thin populated areas), the effect of the antenna's position itself is far more important (an antenna in a street canyon will be significant inferior to one on a skyscraper, concerning it's transmission range, though some street canyons might not be reached). Frequency planning may reduce some of this effects but this correspondence is not treated in this work, since influences of specific frequencies are very hard to formalize. Another factor might be the strength of an emitted signal or just the number of simultaneous calls at a certain time and place. These factors are surely important, but are not treated in this paper directly.
The focal points of this work, are the influences addressed by the label: interference. Interference or mutual signal/transmission disruption arises if (two or more) transmissions are too close to each other in the frequency spectrum and (both) participants can receive both transmissions. In most cases, it is sufficient to consider interference caused by a channel difference of at most one (co- or adjacent channel interference). The bigger the differences within the spectrum, the smaller the possible amount of disruption. Concerning practical issues, no noise is emitted by a distance of more then two. If those conditions are met, both participants receive noise (since the other transmission is not understandable, because of encryption) from the other transmission. Typically, those interference structures are sparse, in the sense, that only locally close calls can interfere another. For example a TRX in Munich will probably not interfere a TRX in Berlin. A further example is given by radio transmissions, again. If outer conditions are met (strong wind, or being at the border of two or more sending stations), one might be able to receive "two channels at the same time", or in other words, has a lot of noise on one channel. One could say, that the signal suffers from signal overlay (see figure 4 for example). Due to technical

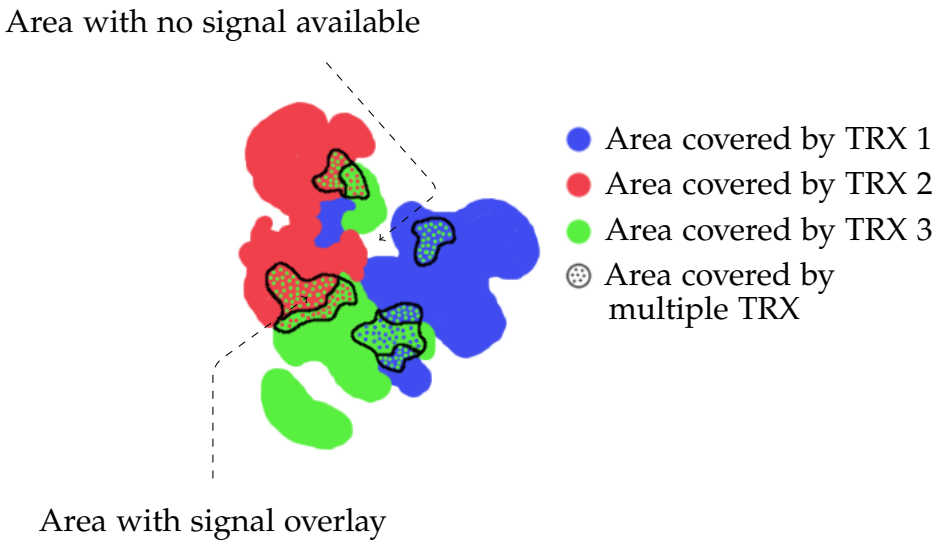

Figure 4: TRX Serving Area

issues, transmission signals are only interpretable, if the signal to noise relation is above a certain limit. Consequentially, the most important question, when installing a (radio) network is: How can calls be placed in the available frequency spectrum, so that their mutual interference is acceptable? In other words, since the TRX determines, where in the spectrum the call is placed: Which frequencies must be assigned to which TRX, so that the transmission quality is the best? Here, the special site/cell structure, mentioned above needs to be taken into account. Regarding the network's organization, this translates into: Which TRX potentially disturb each other to a significant degree and may the use of a certain frequency at a special TRX prohibit the use of that frequency on another TRX? Hereby, the potential disturbances are aggregated into fixed interference values, arising if two TRX use similar frequencies, such that dynamic structures like capacity utilization and fractional load disappear. In figure 5, the interference relations of a frequency assignment (in this case kindly provided by atesio - Berlin) is shown, as an example. The colors and thickness of the links between the TRX indicate the induced interference. In most cases, it is not possible to find an assignment with no interference induced (since the number of TRX / the demanded capacity is far to big). So the aim is to minimize the total interference in an assignment, therefore reducing the average interference, which is the standard approach (see [Eis01]) for frequency assignments. Another approach for a "good transmission quality" might be, to restrict the interference values of every single TRX (or even incorporate dynamic structures like capacity utilization). Ne-

vertheless, the first approach is chosen for the sake of comparability to other works.

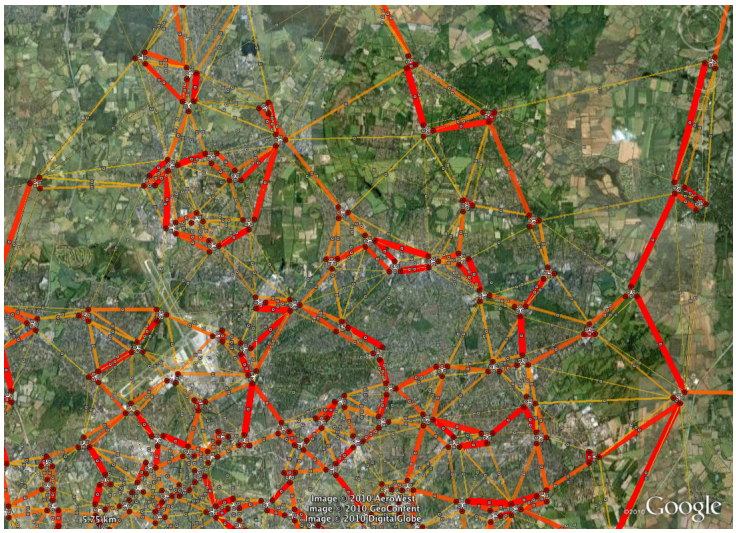

Figure 5: Interference Relations

Therefore, the question, which will be the basis for the future work is: Ignoring all other influences but interference, given all data about the available spectrum and all potential interferences between two sites a well as all demanded amounts of frequencies at the sites, how can the available frequencies be assigned optimally. Optimal in the sense, that the overall existing interference between each pair of transmitters is minimized. Herby interference is divided into co-channel and adjacent channel interference. Co-channel interference describes the amount of disturbance, induced by two transmitters, sending on the same frequency and adjacent-channel interference is the amount of disturbance, induced by two transmitters, sending on neighboring frequencies. Using this model, all other possible interferences of lower order (channel difference of two or more) are ignored for decreasing the problems complexity. This problem is well known as the (classical) frequency assignment problem (FAP). In the Appendix (chapter 8), a short description, which formalizes the FAP model and explains the relation between the site/cell structure and the interference occurrence is given.

For the later work, it is assumed, that the reader is aware of the GSM structure as well as of the characteristics of the FAP. Meaning he knows, which problems are addressed with the FAP, why they are important and how they are related to the GSM techniques and structures.
The information given up to now are the backbone for the future work. In the following, an enhancement (slow frequency hopping) for the GSM technology is des-

cribed, leading to the central aim of this thesis: Developing optimal frequency assignments for slow frequency hopping GSM networks.

§ 2 Slow Frequency Hopping

2.1 — *Intentions and Purposes* —

In the last section, some overview on (classic) radio communication, especially on GMS architecture, was given. But as mentioned in the introduction, much effort has been put into the classical frequency assignment problem, before (compare the bibliography for example). At this point, the concept of slow frequency hopping is introduced as an advancement (concerning capacity / speech quality) of the FAP model. In the classical view, each transceiver operates on exactly one frequency, the whole time. As a result, a network may have an acceptable total interference (which is minimized by the FAP), or in other words it has a minimized average interference. So there may be pairs of TRX with a very high interference and pairs with a very low interference. While the high interference levels totally prevent mobile communication, the very lowest interference values do not offer an experienced gain in speech quality towards an average case, such that the network's quality is used inefficiently. In slow frequency hopping networks, this should be changed: Every transceiver shall change it's frequency periodically. That is to say, a call is not handled on a single channel anymore but alternates it's frequency periodically. Resulting, speech quality is averaged. While there may be still frames with high/low interference, the average is a mixture of both and more important, neighboring time frames have probably not the same interference levels. Consequential, sequential information loss is prevented, such that average experienced speech quality is raised (single lost frames can be reconstructed by error recognition techniques) - the peaks in both directions are removed.
While a more detailed description is given in the following, note that many different technical realizations (for example randoms versus cyclic hopping, for more information see section 8.4) are possible, in general. Nevertheless, the following characteristics were chosen, because they seemed reasonable and support a mixed integer programming approach. Frequency hopping should be applied at random. That is, in every time frame, the TRX choses it's current frequency out of it's assigned frequencies equally distributed, at random. Resulting, and contrary to the classic FAP, there is no definite interference amount at every time frame which can be minimized, but an expected value of interference (see section 3.1 for more details).
For a better visualization on channel hopping, some short explanation on the sending structure of an antenna is given: As mentioned above, each frequency consists of a 200 kHz band of the radio spectrum. A TRX transmits it's data in an eight - frame time cycle. Every time slot is defined to consist of eight frames, each of length

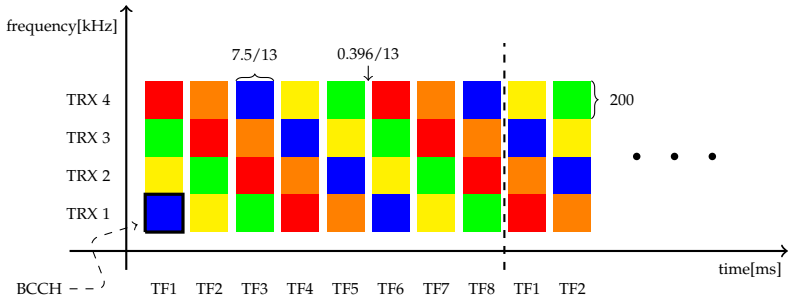

Figure 6: Sector/Frequency Diagram with SFH

7.5/13 ms, with breaks of 0.396/13 ms in between. In that way, a TRX may serve up to 8 communications at the same time. If a TRX changes its frequency at every frame, this is called "slow frequency hopping" (SFH). For completeness sake, fast frequency hopping has, opposed to slow frequency hopping, maximal 1 bit per hop. Depending on the special size of a sector, there might be some special TRX / frames. This is to say: The first TRX in a sector at the first time frame is called "broadcast control channel" (BCCH) and has to transmit cell organization information. So it is permitted to change it's frequency. Resulting, in a SFH network, no hopping is applied to the BCCH carrier (TRX 1, time slot 1). With growing sector size (TRX amount) there may be other channels, which behave the same, but they are not mentioned here in detail. For visualizations sake, a sector / frequency diagram might look like in figure 6.

In detail, the concept of slow frequency hopping should address the following points:

- Signals are subject to fading. Fading is dependent on the local position of transmitter and receiver as well as of the used frequency. For example, one frequency might spread better through some material then another. While changing the net's physical conditions is mostly not an option, the use of different frequencies is an important influential factor. Because of this, the aim is to spread the call (meaning to use several frequencies) in the frequency spectrum (frequence diversity) for preventing *sequential* information loss (substantial fading on some frequencies / time slots, but not on all). Error recognition and correction is relatively far advanced. As long as only single blocks (e.g. one time period) are lost, this can be corrected, if more blocks are lost, quality decreases (information is irrevisibly lost).
- In the classical FAP, signals are subject to continuous interference, thus, if in-

terference occurs, there is sequential information loss, since both calls stay on the same frequencies the whole time. So the risk of sequential information loss (and therefore bad speech quality) is relatively high. Similar to above, the aim is changing frequencies to get an averaged interference (interference diversity). In frequency hopping networks, the two calls will iteratively change their frequency. The aim is, to organize the hopping in the way, that calls may interfere in one time slot, but probably not in the following, as well. So in total, the quality is averaged. Non interfering and interfering frequencies are mixed, therefore sequential information loss is prevented. An important feature hereby: An average speech quality is sufficient for communication, while the best possible quality is not necessary (experienced quality does not increase in relation to average quality) and the worse qualities totally prevent communication. As a result, the averaging yields an improvement in experience quality, by cutting of the peaks in both extremes.

The overall aim of this concept is, to improve signal (and therefore speech-) quality by (compared to FAP) introducing another aspect to frequency assignments (no static frequency usage any longer). While the description above is sufficient for the following work, additional information can be found in [ONS95]).
Note, that quality is tightly connected to capacity. High quality networks can sustain additional traffic, while low quality networks may break down by additional users (induced interference). In this thesis, optimal frequency assignments for this concept shall be provided and at the same time, FAP as an origin of slow frequency hopping should be emphasized. Resulting the slow frequency hopping model is developed on the background of the FAP model, namely on the basis of co- and adjacent channel interference. Resulting, SFH should be examined with respect to FAP and may be treated as some sort of generalization of FAP. While FAP provides the possibilities to bear with interference, there is no direct way for including signal strength or signal fading. As a consequence in this work's approach signal fading (frequency diversity) is neglected and the mathematical model is build on the basis of the interference diversity gain.

While a frequency assignment for slow frequency hopping networks is defined by the information, which TRX should hop over which frequencies, a further important question corresponds to the hopping type. There are some different possible solutions, the most common are iterating (cyclic or at random) through the assigned frequencies, which can be specified further, according to the random distribution, or by some type of TRX offset, in cyclic hopping (see section 8.4 for more details on slow frequency hoping concepts). Nevertheless, this thesis focuses on equally distri-

buted random frequency hopping, mostly because it nicely fits into the context of linear programs. Cyclic hopping, on the opposite, bears the difficulty to create the hopping sequences at runtime (and evaluating their interference values). Further, choosing between different sequences has not only a very big solution space, but their influence on interference is very difficult to calculate in a linear programming. Though an approach on these settings is invoked in [BVY05], for example (there in the context of a simulated annealing algorithm).
Before the concrete (mathematical) model is introduced, some confinements on generality are given. Obviously, an optimal slow hopping frequency plan should contain the information, which frequencies are assigned to which TRX. Further, the problem could be described as: Given the amount of TRX, the available frequency spectrum and the possible interference (co- and adjacent channel) relations. How many and which frequencies need to be assigned to which TRX, such that the expected amount of total interference is minimal in an equally distributed randomized frequency hopping. Especially, this means that solving this problem with full generality implies, that the amount of frequencies assigned to each TRX is not known at the beginning. But since the concrete amount of channels per TRX influences the possible amount of received and transmitted interference, this to complex, to be treated in a linear model. Compare a linear program formulation in section 3.2 for more details: changing the amount of frequencies per TRX has a non linear influence on the objective function. Consequential, for the sake of computability some restrictions on generality are made for the future work. While in the general problem, groups of TRX, which have the same frequencies assigned, are implicitly formed, these are assumed given at the beginning, together with the amount of frequencies assigned to them.

Though these restrictions limit the generality, they are somewhat justified by practical aspects. Having the cellular (cell/site) structure of GSM in mind, many data is only available at site level and many settings may only be applied to the site as a whole (and not to a special TRX). As a result, the above mentioned groups, as formed by all TRX of a site, is an assumption, which is reflects commonplace settings for slow frequency hopping GSM networks. So, the mathematical restriction is induced by practice anyways.
Concerning the limitations on the amount of frequencies per STRX group, research on that matters states (for example in [ONS95]), that the quality increase, achievable through randomizing over more frequencies, decreases on the long run. That means, most quality gain of slow frequency hopping can be achieved by assigning as few as 3 or 4 more frequencies than inbound TRX to each group. Resulting, the gain of leaving that amount of frequencies for each group variable is relatively small compa-

red to the increased computational effort. Thus, assuming the number of demanded frequencies per group given as the number of inbound STRX plus some constant offset seems a reasonable trade of.
Note, that with both settings, some minimum co-channel interference is already determined, namely the induced interference from each STRX (which could potentially be relatively high, if the site structure from above was chosen). This amount should be chosen as low as possible beforehand, but is not treated here in detail, since these values will not influence the following optimization problems (concerning their objective value, this additional interference is just a constant offset).

For the following work, the central problem, which is the frequency assignment problem for slow frequency hopping networks (FAPH), can be defined as: Given the amount of TRX (Super TRX, STRX), which hop over the same frequencies, given their potential co- and adjacent channel interference relations, the available frequency spectrum and the frequency demand (number of frequencies to hop over) of each group (STRX). Which channels need to be assigned to which STRX, such that the total expected interference is minimal. While a mathematical expression of this problem can be found in section 3.4, this is formulated as mixed integer program in section 3.

2.2 — *Referential Work* —

Before presenting own approaches for obtaining frequency assignments in slow hopping networks, referential work on this topic is presented in this section. As hinted before, the telecommunication market is growing ever since. Resulting, this sector is interesting for science as well, leading to more and more scientific results in that matters, including research on slow frequency hopping. In the following section, sources with different approaches to this topic are presented as an example. Nevertheless, these presentations are only able to show comparable methods (algorithms) rather than numerical results. This is due to the fact, that slow frequency hopping can be implemented in many different ways (e.g. like mentioned in section 8.4, random hopping vs. cyclic hopping) and on different data (key questions are: how is the gain of frequency hopping measured etc.). This is enforced by the fact, that no central data sets are existing. All in all, different papers work on different scenarios, making it hard to compare numerical values. Even when working on the same data (like from fap.zib.de, the cost256 project), the effect of slow frequency hopping (e.g. the modification of interference values) on the there presented (classical) FAP instances is interpreted different in most cases. Resulting, the following remarks rather intro-

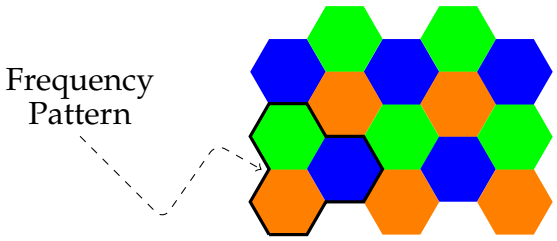

Figure 7: Frequency Reuse (c=2)

duce alternative methods for obtaining slow frequency hopping assignments than compare numerical results.

In general, similar to the classic frequency assignment problem, two different methods for generating frequency assignments are used. The first method is called "frequency reuse patterns" and describes a generic way of spreading frequencies over STRX. The second one concerns a dynamic, possibly optimal frequency assignment. Since frequency reuse has not been introduced in this work yet, this is done in the following. On the background of centering on optimal frequency assignments and not on generic ones, the introduction and the presented results are only outlined and not presented in detail.

2.2.1 Frequency Reuse

Having in mind, that GSM networks offer a cellular structure, frequency reuse describes the usage of a specific pattern of frequencies for covering all STRX, that means, the frequencies are regularly reused (for a more detailed introduction, one may consider the article [Lin]). Hereby, the cellular structure is mostly modeled as a hexagonal structure, but other structures are possible, too. The main parameter, influencing this concept is the (frequency) reuse distance c, which denotes the number of cells, which needs to lie between two cells (in the sense of the shortest way) sharing the same frequency. In general, $c > 1$, since adjacent cells are normally not allowed to share the same frequency. It applies, that the number of needed frequencies is dependent on that parameter. Figure 7 shows an example of $c = 2$, forcing a frequency pattern of three different frequencies. Note, that this can be applied on frequency hopping networks as well (assign multiple frequencies).

Obviously, such a model is dependent on various influences. Key aspects hereby are

(besides the above mentioned reuse distance) the way the cellular structure is derived (hexagonal, differently sized cells in practical aspects, etc.), which pattern is used exactly, or whether different patterns are used for certain part of the structure. Much effort has been put into this, on various specifications. For example, in the paper [KI01]: "Fundamentals of Dynamic Frequency Hopping in Cellular Systems" a dynamic adaption of reuse patterns is analyzed. As another example, the paper [Eea98]: " Multiple reuse patterns for frequency planning in GSM networks" focuses on research on practical aspects and yields a frequency reuse factor of 7.5.

Though this concept is deployed in praxis and provided useful results, it is getting outdated. In generic frequency assignments specific interference values cannot be considered, it is only possible to have a low level of interference in general. Therefore results are not comparable to optimal frequency assignments, with respect to interference values. Nevertheless, one of it's mayor advantages over optimal assignments lies in the relatively ease to generate a frequency assignment, compared to the computational effort in the other case.

2.2.2 Optimal Assignments

The counterpart to frequency reuse patterns is the idea of creating dynamic optimal (or optimized) frequency assignments. While frequency reuse offers a generic way of creating assignments, local properties of single TRX (STRX), like specific interference values, are ignored. On the other hand, using local data on STRX level creates very huge amount of data, making calculations very difficult in a practical aspect. Nevertheless, the quality gain when respecting this data is so promising, that this effort is taken into account. Hereby, an optimal assignment is preferable, of course, but appropriate approximations are often sufficient, if this optimum is not reachable. Similar to frequency reuse, much effort has been put into this topic, before. Hereby, a distinction can be made between approaches (heuristics), derived from the original FAP problem and the ones "newly" introduced for slow frequency hopping. For the first option, any source describing a mathematical approaches for obtaining frequency assignments may be consulted, in this work, the PhD thesis [Eis01] is recommended. This thesis offers a broad overview on various mathematical heuristics, creating not optimal, but good frequency assignments, most of them easily convertible to frequency hopping conditions. Quality wise, the therefore generated solutions have the same characteristics than in the classic FAP - fast, but not optimal (see section 4.5, where some of these heuristics are incorporated into this work).

Other work is directed mainly to slow frequency hopping. Here the most prominent example (using very similar problem settings to our approach) is the paper "Optimized planning of frequency hopping in cellular networks" ([BVY05]). In this paper, the problem is formulated as choosing a list of frequencies (to cycle through) for every STRX. Due to the size of the problem, finding an optimal list of channels to hop over for every STRX, this problem is not solved optimal. Instead, a solution is presented, which uses a simulated annealing heuristic, achieving success in an interference reduction at runtime.
A further interesting paper in this context bears the title "Assignment of Frequency Lists in Frequency Hopping Networks" ([MHS05]). There, the above mentioned paper is expanded in various directions like pregeneration of the frequency lists before assigning or modifying existing lists, showing improvement capabilities of this measures.

All in all, these papers show that a quality gain through optimized frequency assignments can be achieved towards the concepts of frequency reuse. Serving as some sort of justification for the next section, where this approach is sustained and even strengthened. There, it will be tried to obtain an optimal assignment, not by the use of meta heuristics but by the use of integer programming.

§ 3 A MIP approach

3.1 — *Problem Description and Input Data* —

In the current chapter, the problem of finding an optimal frequency assignment for a "slow frequency hopping" network is developed out of the classic FAP description. Therefore the same input data is kept, as far as possible (see section 6.3 for more details). However, the aim is the same: Creating a feasible frequency assignment with minimal inbound expected (total) interference. While a definition as optimization problem can be found in 3.4, a mixed integer formulation for that problem is given in this section. So at the beginning, constraints for forming a feasible assignment and evaluating it's quality are needed.
Repeating, for a slow frequency hopping frequency assignment, the following input is needed: Similar to the classic FAP, the set of available frequencies, as well as the number of TRX and their corresponding interference relations is needed, from which the data on STRX level will be aggregated. Further, secondary information like local blockings, and separation issues between these TRX can be given. Additional, and not contained in the FAP model before, the following data needs to be specified.
It is known beforehand, which TRX have the same frequencies to hop over. These groups are called STRX (super transceiver) in the future. In practical aspects, these STRX are often formed according to the cell/ site structure, though this is not necessary, in general. Further, it is assumed, that for every STRX I, the number of needed channels $k_I \in \mathbb{N}$ is known beforehand, too. So the actual goal is to choose the specific frequencies, but not to build the concrete grouping (STRX) structure or determine the corresponding number of needed frequencies. Hereby, for a further understanding, the chapter before is referred. Using the given data on "TRX" level, these are aggregated on STRX level in the following way:
Interference relations (e.g. co-channel) are defined for TRXs v and w within STRXs I and J (e.g. co(v,w)=...) and not directly for the STRX themselves. These are generalized by means of probability distributions of expected (interference) values: Given an equally distributed probability, that TRX v in STRX I gets frequency f at a certain time slot t, the expected interference with another TRX w in STRX J can be calculated. So the sum over all expected values between all TRX combinations between two STRX gives the total (expected) interference between these two STRX. This is explained, by means of an example:
Given two STRX I and J, and two lists of assigned frequencies FL_I and FL_J. Let $|I|$ denote the number of TRX in I and $|FL_I| = k_I$ the number of frequencies the TRX in I might hop on, analogue for J. Further, let $y_{I,J}$ denote the size of the intersection of

FL_I and FL_J. Given random and equally distributed frequency hopping in I and J, in every time slot, the (co-channel) interference, which is imposed from a TRX $v \in I$ on a TRX $w \in J$ is regarded. There is only one chance of co-channel interference, if v uses a frequency which is in FL_I and FL_J at the same time. This is the case with likelihood $y_{I,J}/k_I$. So w needs to use the same frequence, which occurs with probability $1/k_J$. Then, a co-channel interference amount of co(v,w) is created. Following this line of thought, the expected total co-channel interference, opposed from I on J is given by:

$$co(I,J) := \sum_{v \in I} \sum_{w \in J} \frac{y_{I,J}}{k_I \cdot k_J} \cdot co(v,w). \tag{1}$$

While this value is formally an expected value of a probability distribution, it will be denoted as the arising (continuous) interference value between the STRX I and J for shortness sake.
Note, that both interference measures explicitly allow $co(I,I)$ and $ad(I,I)$ values different to zero (because of the aggregation from TRX to STRX level). While the induced co-channel interference is already determined by the STRX structure (see section 2) and can be regarded as a constant offset therefore, the adjacent channel interference has still to be determined. Though this does not change the computational aspects of the (future) models, it is mentioned here for completeness sake.

A similar construction is used for the adjacent channel interference ($n_{I,J}$ denotes the number of neighboring channels from I in J):.

$$ad(I,J) := \sum_{v \in I} \sum_{w \in J} \frac{n_{I,J}}{k_I \cdot k_J} \cdot ad(v,w). \tag{2}$$

The secondary data can be aggregated in a similar manner: The FAP model provides the possibility of separation constraints on TRX level. It is assumed that these constraints are lifted on STRX level, so that every STRX I now has a parameter d_I, which represents the frequency separation within this STRX. Further, every pair of STRX I, J has a parameter $d_{I,J}$, which represents the separation requirements between these two STRX (matches the separation requirements between all TRX $v \in I$ and $w \in J$). Obviously inner STRX separation is easily achievable by imposing the single, most striking separation constraint between two TRX within one STRX on all TRX within that STRX. Similar holds for separations between the STRX. In comparison to the old FAP, this amplifies the old constraints and may therefore decrease the amount of feasible solutions. In scenarios, where this method is too restrictive, one may consider a different STRX structure or an omitting of any of those constraints. In this context, it

might be interesting or necessary, to treat some single / special TRX as STRX (STRX with exactly one inbound TRX). Especially, this applies to the BCCH carriers: Since they spread broadcasting control information, they are not allowed to change their frequency. So every BCCH carrier forms a single STRX, with the demand of exactly one frequency, each.
In this context, local blocking constraints are mentioned, for completeness sake. In the FAP model, it is possible to block frequencies on local TRX. Obviously a similar approach as above can be made to produce feasible assignments, with the same drawbacks. In the following, the set of blocked frequencies for a STRX I is denoted with B_I.
Summing up, with some additional information (STRX structure, $k[I]$ values) the constraints for forming feasible frequency assignments can be derived from the classic FAP model. The mean by which these assignments are measured are (similar to FAP) the expected total inbound (co- and adjacent channel) interference:

$$\sum_{I \in \mathcal{S}} \sum_{J \in \mathcal{S}} \left(co(I,J) + ad(I,J) \right),$$

whereas $\mathcal{S}$ denotes the set of STRX.

3.2 — *A mixed integer formulation* —

As mentioned above, the aim of this model, is to provide a feasible (optimal) frequency assignment (given the input from above). Herby, the mathematical tools are the ones, provided by mixed integer programming. So in this section, a mixed integer formulation, addressing the above mentioned problem is given.
At first, some notational issues (referring to the input data), for this section and whole following work, are presented. Hereby, most notations are similar or equal as in the standard FAP problem.

- $\mathcal{S}$, for the set of all STRX.
- F, the set of all available frequencies.
- k_I, the amount of (different) frequencies, demanded by STRX I.
- $co(v,w)$ ($ad(v,w)$) potential co- (adjacent-) channel interference value between TRX v and TRX w.
- d_I , inner STRX (I) separation and $d_{I,J}$ for separation between the SRX I and J.
- B_I for the set of frequencies locally not available at STRX I.

Concerning the slow frequency hopping, frequency assignment problem, a formulation as an integer program may be given, using the following variables:

$x_{I,f} \in \{0,1\}$ $\quad \forall\ I \in \mathcal{S}, f \in F.$

$x_{I,f} = 1 \quad \Leftrightarrow$ frequency f is used at STRX I.

$z_{I,J,f} \in \{0,1\}$ $\quad \forall\ I, J \in \mathcal{S}, f \in F.$

$z_{I,J,f} = 1 \quad \Leftrightarrow$ Frequency f is used at both STRX I and J

$\Leftrightarrow \quad x_{I,f} = x_{J,f} = 1.$

$b_{I,J,f} \in \{0,1,2\}$ $\quad \forall\ I, J \in \mathcal{S}, f \in F.$

$b_{I,J,f} = x \quad \Leftrightarrow$ x adjacent channels of frequency f from STRX I in STRX J

$$\Leftrightarrow \quad b_{I,J,f} = \begin{cases} 0 & \text{if } x_{I,f} = 0 \\ 1 & \text{if } x_{I,f} = 1, (x_{J,f-1} \text{ xor } x_{J,f+1}) = 1 \\ 2 & \text{if } x_{I,f} = x_{J,f-1} = x_{J,f+1} = 1 \end{cases}.$$

$y_{I,J,} \in \mathbb{Z}_+$ $\quad \forall\ I, J \in \mathcal{S}.$

$y_{I,J} = x \quad \Leftrightarrow$ x equal channels from STRX I in STRX J

$$\Leftrightarrow \quad y_{I,J} = \sum_{f \in F} z_{I,J,f}.$$

$n_{I,J,} \in \mathbb{Z}_+$ $\quad \forall\ I, J \in \mathcal{S}.$

$n_{I,J} = x \quad \Leftrightarrow$ x adjacent channels from STRX I in STRX J

$$\Leftrightarrow \quad n_{I,J} = \sum_{f \in F} b_{I,J,f}.$$

This leads to the following objective function (with the notations above):

$$Min: \quad \sum_{I \in \mathcal{S}} \sum_{J \in \mathcal{S}} \sum_{v \in I} \sum_{w \in J} \sum_{f \in F} \left[z_{I,J,f} \cdot \left(\frac{1}{k_I} \frac{1}{k_J}\right) \cdot co(v,w) + b_{I,J,f} \cdot \left(\frac{1}{k_I} \frac{1}{k_J}\right) \cdot ad(v,w) \right].$$

By the (optional) constraints

$$\sum_{f \in F} z_{I,J,f} = y_{I,J} \qquad \forall\ I, J \in \mathcal{S},$$

$$\sum_{f \in F} b_{I,J,f} = n_{I,J} \qquad \forall\ I, J \in \mathcal{S},$$

this is is equivalent to

$$\begin{aligned} \textit{Minimize}: \quad & \sum_{I\in\mathcal{S}}\sum_{J\in\mathcal{S}} y_{I,J}\cdot\overline{co(I,J)} + n_{I,J}\cdot\overline{ad(I,J)}, \\ \textit{Where is}: \quad & \overline{co(I,J)} := \frac{\sum_{v\in I}\sum_{w\in J} co(v,w)}{k_I k_J} \qquad (3) \\ \textit{and}: \quad & \overline{ad(I,J)} := \frac{\sum_{v\in I}\sum_{w\in J} ad(v,w)}{k_I k_J}. \qquad (4) \end{aligned}$$

The corresponding constraints are the following: at first, allot every STRX it's corresponding number of frequencies:

$$\sum_{f\in F} x_{I,f} = k_I \qquad \forall\ I\in\mathcal{S}.$$

Further, count the amount of equal frequencies in each two STRX:

$$\begin{aligned} z_{I,J,f}+1 &\geq x_{I,f}+x_{J,f} && \forall\ I,J\in\mathcal{S}, f\in F, \\ \sum_{f\in F} z_{I,J,f} &= y_{I,J} && \forall\ I,J\in\mathcal{S}. \end{aligned}$$

Here, the conditions

$$z_{I,J,f}\leq x_{I,f} \text{ and } z_{I,J,f}\leq x_{J,f} \quad \forall\ I,J\in\mathcal{S}, f\in F$$

are not necessary, because of z's (positive) objective function's coefficient and the "urge" to minimize. Similarly, the number of adjacent channels for each pair of two STRX is counted:

$$\begin{aligned} b_{I,J,f} &\geq 2(x_{I,f}-1)+x_{J,f-1}+x_{J,f+1} && \forall\ I,J\in\mathcal{S}, f\in F, \\ \sum_{f\in F} b_{I,J,f} &= n_{I,J} && \forall\ I,J\in\mathcal{S}. \end{aligned}$$

Similarly as above, the conditions

$$b_{I,J,f}\leq x_{I,f} \text{ and } b_{I,J,f}\leq x_{J,f-1}+x_{J,f+1} \quad \forall\ I,J\in\mathcal{S}, f\in F$$

on b are not necessary, again. At last, some separation constraints:

$$\begin{aligned} x_{I,f_1}+x_{I,f_2} &\leq 1 \qquad \forall\ I\in\mathcal{S}, f_1,f_2\in F, |f_1-f_2|\leq d_I, \qquad (5) \\ x_{I,f_1}+x_{J,f_2} &\leq 1 \qquad \forall\ I,J\in\mathcal{S}, f_1,f_2\in F, |f_1-f_2|\leq d_{I,J}, \qquad (6) \\ x_{I,f} &\leq 0 \qquad \forall\ I\in\mathcal{S}, f\in B_I. \end{aligned}$$

Where the first conditions represent separations between the TRXs of one STRX (inner STRX separation), the second separations between TRX of two different STRX (outer STRX separation) and the third some local (in relation of a STRX) blocked frequencies. In the later context, this model is referred to as [FAPSFH].

3.3 — *Computational aspects* —

In this section, the model given above is analyzed from a practical / computational point of view. As a first advantage the formulation itself is to mention. It is very straight forward, in the sense of an easy to gasp approach: Assign enough frequencies and calculate the interference by the means of penalty variables (if two STRX have a frequency in common, a certain penalty has to be paid). Nevertheless, the problem is purely integer, with no continuous variables at all. Although this is not bad in general, it mostly leads to excessive branching during the solving process, opposed to continuous variables, which can be addressed (locally) optimal rather easily, in all stages of the solution, by the simplex algorithm.
An additional disadvantage is the huge amount of variables used by this formulation. Variables in the order of $O(|\mathcal{S}| \cdot |\mathcal{S}| \cdot |F|)$ are used, as well as constraints in the same order. Given a realistic problem size, of 500 STRX and 75 frequencies, this number is about as high as 18 million. At this point, problems concerning memory usage arise as early as in the problem's initialization. Even though the problem has two major advantages /characteristics (symmetry and a certain "sparse" structure), which can be exploited, concerning runtime this size is still significantly big and problematic.
Nevertheless, these characteristics are shortly outlined in the following: At first, the formulation's symmetry is to mention. If there is potential interference (both STRX transmit on the same or on neighboring frequencies, though the interference value / penalty might be equal to zero) between STRX I and J, the (potential) interference between J and I arises as well. By defining

$$co(\{I, J\}) := co(I, J) + co(J, I),$$

it is sufficient to treat sets of two STRX $\{I, J\}$ and not pairs (I, J). Thus effectively reduces the problem's size by a factor of two (exchanging ordered tuples with two elementary sets).
The second characteristic is the sparse structure of the (interference) relations between the STRX. This property is typically for FAP problems and therefore inherited by slow frequency hopping assignment problems: With growing problems size (in the sense of rising amounts of STRX), normally the scale of the area covered by this STRX rises as well. Giving a sufficient large area, most STRX (the inbound TRX) can not disturb each other. Since the signals are fading by distance, interference relations can only take place on a local neighborhood. So most objective's function coefficient of the variables y or n will be equal to zero, so that these can be ignored (in a computational point of view: they need not be created).

A further, and by far more important disadvantage (than the pure size) lies in the (integer) formulation, with penalty variables, in the structure of $x = y = 1 \Leftrightarrow z = 1$ (or similar) itself. Concerning the solution process of such a LP, the linear relaxation of a problem is solved, it is then branched for integrality and the process is repeated again (as is shortly described in section 8.2). In this process, the linear relaxation provides (dual) bounds on the solution, in general. In MIPs containing the above structure and especially in the [FAPHSFH] model, the following problem arises: Since a minimization problem is regarded, the solution process will obviously try to avoid the penalty variables and still fulfill the requirements for a feasible solution. The penalty values are only taken into account, when certain variables (the $x_{I,f}$) are used with a degree, strictly higher then 0.5. But this is easily avoidable, since the requirements (assigning k_I frequencies to STRX I) can be fulfilled by small fractional values on all frequencies. For example, $x_{I,f} = 1/k_I$ is possible in most cases – ignoring separation and blocking constraints, for the sake of simplicity here. The same holds true, in general, too, since there are mostly only few of these constraints. Resulting no penalty has to be paid at all. So the relaxation will always give a dual bound of zero, which is of no worth at all (obviously the theoretically best solution can not induce less than zero interference). As a consequence, one could argue, that the solution process emphasizes on a total enumeration of the possible integer solutions, rather than on the usage of the simplex method (in the sense of solving the linear relaxation).
Another problem lies in the model's symmetry. This is not to be mixed up with the symmetry mentioned above. While the one above is an advantage in the problem's initialization (e.g. no need for creating the symmetric variables) and it's memory usage, the other influences the branch and bound process of the solution in the following way:
A solution (a feasible assignment) is symmetric in the way, that there are various other assignments possible, with the same interference values. As above, ignoring separation and blocking constraints as well as the adjacent channel interference, this is evident (gaining another solution by a one to one mapping on the frequencies). However, the same holds true, with these constraints as well, though to a lesser extend. This does not only apply to the final, integer solution, but to solutions of the linear relaxation as well. So a branching, for example on a $x_{I,f}$ variable, yields to an equivalent solution, with the old value of $x_{I,f}$ on another variable $x_{I,\overline{f}}$ ($f \neq \overline{f}$). Resulting, the branch and bound procedure will cycle on equivalent solutions for some time, not improving the solution quality at all. In the worst case (with the settings from above) there are exponentially many solutions (there are $2^{|\mathcal{S}|}$ mappings possible) to be cut away, so the time consumption might be significant. A further

drawback, besides this inefficient branching, is that the great number of equivalent solutions prevents an effective pruning in the branching process. Equivalent solutions cannot be cut away (since they are not worse then the comparing solutions), forcing a great amount of comparisons between (integer) solutions and therefore bringing the whole solving process closer towards a complete enumeration of the integer solution, slowing its runtime considerably. So all in all, this symmetry leads to a highly inefficient branching and pruning, therefore severely slowing the process down.
Summing it up, this model can be described as very intuitive, but with a bad solving behavior. Since the model, respectively the FAPH problem has it's origin in FAP and therefore in graph coloring, some similar approach, with similar analysis can be found in [MT95].
For evaluating this formulation in practical aspects and for emphasizing the problem's origin in FAP, the test scenarios have been taken from fap.zib.de. Referring to these scenarios (whose adaption to SFH is explained with more details in section 5.2, as well as the used soft- and hardware) the following results have been obtained: At first, the small test scenario "tiny" shows that this approach can provide feasible frequency plans, in this small case (7 STRX, 13 frequencies) within seconds. Other, more realistic sized problems like "k" (264, 50) and "siemens1" (506, 75) show that this approach is not helpful at all. While the initial (root) LP itself takes about $8 - 9$ Gigabytes (resulting, not much space for the branch and bound process is left – at least on usual in-trade PCs) the solution is of no worth at all. As shown above, the dual bound stays at zero and no primal bound (feasible solution) is found at all. Concerning even bigger scenarios like "Bradford-0-eplus" (1886, 75) this gets even worse in the point, that the problem could not be initialized (on a machine with 14 gigabyte working space) at all.

Though this formulation seems to be of little practical use, there are several workarounds (for increasing it's computability) possible. For example, there are some general or generic ways, to shrink a model's size. At first, the problem size itself may be taken care of, with some sort of preprocessing. This implies ignoring interference values below a certain bound or eliminating STRX without connection/relation to any other STRX (interference value of zero, no separation requirements – these can be assigned separately, since they do not influence the assignment of other STRX.), for example. This can be interpreted as focusing on the essential data. Nevertheless, there is no formal proof of optimality possible any longer and enforcing computability by ignoring enough interferences may omit so much data, that the result is not useful for practical applications any longer.

Another improvement might be made by adding cuts / feasible inequalities. While in theory, the existence of helpful cuts is evident (for example with the Gomorry procedure), no specific cuts can be presented in this practical case. Though the separation constraints (5 and 6) may produce some nice "stable set"- shaped cuts, these are surely not enough (since there are to few separation constraints in realistic instances) to obtain a significant speed-up for the complete model.
At last, the solution process (especially the problems with the LP relaxation, mentioned above) might be improved by concepts like separation of feasible inequalities and column generation (see section 8.2 for more information). Nevertheless, a further discussion on that matters is delayed to the end of this section. Before that, a short complexity analysis of the FAPH problem ([FAPHSFH] formulation) is given, for completeness sake.

3.4 — *Problem Hardness* —

Concluding this chapter, a short description of the "hardness" of the FAPH problem, in terms of complexity analysis is given. Hereby, a basic knowledge of the theory of complexity classes or of the complexity of algorithms is assumed, so that this section will rather give some results than a whole overview on complexity analysis, definitions and other results. A short introduction on this matter is given in section 8.1, nevertheless.
Obviously, the linear relaxation of [FAPHSFH] is in P, since the variables and constraints are "polynomial sized" (in $O(|\mathcal{S}| \cdot |\mathcal{S}| \cdot |F|)$). On the other hand, the combinatorial (not relaxated) variant is NP-complete. This is induced by the evolution of FAPH from FAP and therefore graph coloring, which are both NP complete problems, as well. More details, concerning the NP-completeness of FAP, can be derived from [Eis01], where this matters are shown by a reduction from the minimum edge deletion k-partion problem. In the following, a formal proof of the NP-completeness of FAPH is given. Hereby, the optimization problems are treated, rather than the corresponding decision variants (which would ask, whether a solution below a certain value exists).

As mentioned above, this prove is given by a reduction of FAP to FAPH. Since both problems have only been introduced as mixed integer programs (which is formally not sufficient; for example, a minimum spanning tree can be formulated as MIP as well, though it is not NP-complete), in the following, both problems are redefined. Hereby, the definition of FAP is taken from [Eis01]:

The input of the FAP problem consists of a seven-tuple (carrier network)

$$N := (V, E, F, \{B_v\}_{v \in V}, d, co, ad).$$

Hereby, $G = (V, E)$ is a graph, with a vertex for every TRX. The edges are defined by the three functions d, co and ad, giving to each edge the corresponding separation, and the co-channel and adjacent channel interference (greater or equal than zero). F denotes the set of available frequencies (positive integers) and B_v denotes the set of blocked frequencies for every TRX.
A feasible solution (frequency assignment) for such a carrier network is a function

$$y : V \to F,$$

such that

$$\begin{aligned} &y(v) \in F \setminus B_v && \forall\ v \in V, \\ &|y(v) - y(w)| \geq d(vw) && \forall\ (v, w) \in E. \end{aligned}$$

Then the FAP can be formulated as: given a carrier Network, the problem

$$\min_{y} : \quad \sum_{\substack{(v,w) \in E \\ y(v) = y(w)}} co(v, w) + \sum_{\substack{(v,w) \in E \\ |y(v) - y(w)| = 1}} ad(v, w)$$

for a feasible solution y, is called the frequency assignment problem (FAP).

The input for the FAPH problem is, very similar to the input of the FAP problem, a nine-tuple (STRX graph):

$$M := (\mathcal{S}, E, F, \{B_I\}_{I \in \mathcal{S}}, k_I, d_1, d_2, co, ad).$$

Hereby, the graph $(\mathcal{S}, E)$ is formed by the STRX (nodes) and the edges give relations between these STRX (within the same notation, co and ad for interferences and d_1 for inner and d_2 for outer STRX separations). Blockings are now defined for every STRX and the new parameter k_I denotes how many frequencies have to be assigned to every node/STRX I. F denotes the available frequency spectrum. As above, a feasible solution (a feasible assignment) is a function

$$y_2 : \mathcal{S} \to 2^{|F|},$$

mapping each vertex to a set of frequencies, such that

$$\begin{aligned} &|y_2(I)| = k_I && \forall\ I \in \mathcal{S}, \\ &y_2(I) \subseteq F \setminus B_I && \forall\ I \in \mathcal{S}, \\ &|f_1 - f_2| \geq d_1(I) && \forall\ I \in \mathcal{S},\ f_1, f_2 \in y_2(I) \text{ with } f_1 \neq f_2, \\ &|f_1 - f_2| \geq d_2(I, J) && \forall\ (I, J) \in E,\ f_1 \in y(I),\ f_2 \in y_2(J). \end{aligned}$$

Then, a definition of FAPH is analogous: Given a STRX graph, the problem

$$\min_{y_2}: \quad \sum_{(I,J)\in E} \sum_{\substack{y_2(I) \\ y_2(I)\in y_2(J)}} 1 \cdot co(I,J) + \sum_{IJ\in E} \sum_{\substack{y_2(I): \\ y_2(J)+1\in y_2(J) \\ y_2(I)-1\in y_2(J)}} 1 \cdot ad(I,J)$$

for a feasible solution y_2, is called the frequency assignment problem for slow frequency hopping networks (FAPH).

While FAP is stated to be in NP, with [Eis01], it is formally left to show, that FAPH is in NP as well. Referring to section 8.1, it is sufficient, that a solution of FAPH can be validated within polynomial time. Since a solution can be calculated from the underlying graph, which is polynomial in the amount of STRX, this is an obvious consequence. For proving the NP-completeness from FAPH, the reduction from FAP to FAPH is still to show. With these two definitions above, a reduction r from FAP to FAPH can be given in the following way:

$$\begin{aligned} r: \; & N \to M \\ & N \mapsto \left(V, E, F, \{B_v\}_{v\in V}, 1, 0, d, co, ad\right). \end{aligned}$$

Obviously, r is in P, as a composition of constant and identity functions. With this reduction, the FAP problem can be treated as an instance of the FAPH problem. Resulting, since FAP is NP-complete, FAPH must be NP-complete as well (FAPH is at least as hard as FAP).

In addition, a (not necessarily polynomial) reduction the other way around, is possible, too. Given a FAPH instance, the transformation works the following way: Every node I (STRX) in FAPH, is substituted by k_I different nodes $v_1, ..., v_{k_v}$. For every edge between these nodes, (v_j, v_j) ($i \neq j, 0 < i, j \leq k_I$) it holds, that $co(v_i, v_j) = ad(v_i, v_j) = 0$ and $d(v_i, v_j) = d_1(I)$. The edges (v_i, w_j), with $0 < i \leq k_I$ and $0 < j \leq k_J$ have the weights $co(v_i, w_j) = co(I, J)$, $ad(v_i, w_j) = ad(I, J)$, and $d(v_i, w_j) = d_2(I, J)$. For every other possible edge, all three weights are equal to zero. The resulting graph can serve as an input for FAP, solving the corresponding FAPH instance.
Nevertheless, this reduction is formally not in P. Denoting $\bar{k} := max_I \{k_I\}$, the new graph has nodes in the order of $\bar{k}$ times the previous nodes ($|\mathcal{S}|$) and edges in the order of $|\mathcal{S}|^2 \cdot \bar{k}^2$, which is not polynomial in the FAPH description size. While $\bar{k}$ is mathematically unbounded, in practical applications, this value can be bounded by the number of frequencies in the frequency spectrum (which is constant over all instances). So for the subclass of realistic sized problems, there is a polynomial transformation in the other direction as well.

All in all, it holds, that

- FAP is reducible to FAPH, FAPH is at least as hard as FAP and therefore NP complete.
- Restricted to practically sized instances, FAPH is reducible to FAP. Both, FAPH and FAP, are equivalently difficult (hard).
- Since FAPH is NP-hard, this may be seen as a reasoning or justification for the bad solution behavior of [FAPHSFH]. The other case, a very good solution behavior of straight forward formulation would have been astonishing.

Nevertheless, keeping in mind, that though both problems have very similar graph representations, FAPH instances and FAP instances differ in their meanings. While the FAP produces a frequency assignment and the calculated interference can theoretically be measured at every moment in the physical network, the FAPH solution produces a frequency assignment, which interference is an expected value. While the FAP problem is fitted directly into a graph (nodes to TRX), the FAPH problem is generally more abstract.

In the next section, ending this chapter, some conclusion of the MIP approach is drawn, regarding both, the computational results and the theoretical hardness, proven in this section.

3.5 — *Reducing complexity: Decomposition in a two stage problem* —

As can be derived from section 3.3, the [FAPHSFH] model fails in practical aspects, in a computational point of view. Since the only information provided by this approach was a dual bound of zero for realistic sized instances (which is a trivial information), no further progress in the solution (like feasible bounds) could be made, at all. Resulting, the question for the following work is, whether this approach can be improved, with sensible effort, to produce some non trivial results (like feasible upper or lower bounds). Hereby, the aim mentioned in the preface persists: The FAPH problem should be solved by linear programming and not by the means of (polynomial time) approximations (heuristics).
Though some improving concepts have been mentioned at the end of section 3.3, a reasonable potent improvement seems to be hard to achieve. The formulation, consisting of penalty values (e.g $x = y = 1 \Rightarrow z = 1$), shows a (too) bad solution behavior in general and offers (too) little possibilities to work with, concerning improvements

by valid inequalities. Hereby, having in mind that this problem is NP-hard, it might be necessary to refine this problem for an easier approach, in the sense of, being more prone to a computational evaluation (e.g. having a structure, that yields faster results in the linear relaxation).

The here proposed refinement is the decomposition of the problem into two main stages / sub problems, therefore reducing the problem's complexity. Since co-channel interference can be interpreted as of higher order (concerning the interference values) than adjacent channel interference, it seems reasonable to split this computation into two steps.
Resulting, in the first stage, co-channel interference is addressed (and in the latter adjacent channel interference). Hereby, it is exploited that co-channel interference can be treated without the knowledge of the specific frequency shared by various STRX (it is sufficient to know, that STRX share a frequency, the interference amount is not dependent on the specific frequency). So, an assignment can be "prepared", such that the co-channel interference is already completely determined (for the whole assignment) after the first step. In other words, it is determined which groups of STRX share a frequency among each other (e.g. a result of step one might be: the STRX $\{1, 4, 5, 8\}$ have one frequency in common).
In the second step, this assignment is "finished", with respect to adjacent channel interference. That is, the groups of STRX sharing the same frequency determined in step one, get the specific frequency assigned (e.g. all the above mentioned STRX get frequency 3 assigned). So the whole assignment has as total interference the sum of the co-channel interference determined in step one plus the adjacent channel interference of step two.
Without going into detail yet, both models are easier to solve (in a computational point of view), than the [FAPHSFH] model itself, though both problems will be NP-hard as well.
Nevertheless, this decomposition forces some concessions to optimality. Without presenting the particular models here, the combined solution of the two models is not globally optimal (for the FAPH problem) any longer. Since an assignment is chosen optimal to co-channel interference (primary) and only secondary optimized to adjacent channel interference, it needs not be optimal for the combined interference any longer (the combined optimal assignment needs not be optimal restricted to co-channel interference). Further, the structure of both problems forces to drop additional constraints like separation and local blockings (or reducing them to the most important constraints, at last). Both restrictions are reasoned by the particular LP formulations, given and commented in the later work. Nevertheless, these concessions

seem reasonable for gaining more insightful results.
A further gain of the reformulation into two stages lies in the implementability of modern solution concepts like column generation or cutting planes algorithms. Both are used for accessing larger problems (both stages will be very large problems, still), from a computational point of view and both could not be embedded in the [FAPHSFH] model before.

In the following chapter (4), the first (stage) problem is given, describing the co-channel interference. Hereby, a column generation approach is made, since the size of the first stage problem will still be very large (exponentially many variables). This is complemented by section 3.3, where computational results and statistics are presented. In chapter 6, the final results are processed towards adjacent channel interference and in the preceding chapter, a final evaluation is given.

§ 4 The first Stage

4.1 — *Column Generation in General* —

Before going into detail with the first stage problem, the mean of choice – column generation (or often referred to as „pricing“, especially in the context of “branch and price“) is outlined here. Nevertheless, elementary knowledge on the solution algorithms of linear programs is assumed in the following. Therefore, basic knowledge on linear programming is provided in the appendix (section 8.2) or can be obtained in nearly every operations research lecture, e.g. in [HL09] or in the standard work [Chv83]. Readers who are sufficiently experienced in this context may skip this subsection and continue with the application of this concept. For an easily accessible example, the paper [MT95], can be mentioned, which shows a column generation approach to graph coloring. More theoretical descriptions on that matters are given by [LD05], in the book [DDS05] respectively.
Given a linear program (LP)

$$\begin{aligned} &min && c^T \cdot x \\ &s.t. && A \cdot x \leq b, \\ & && x \geq 0. \end{aligned}$$

with $A \in \mathbb{R}^{m \times n}$, $x, c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, the standardized way of solving LPs is to apply one of the various simplex algorithms (in most cases, ellipsoid method's run times are not comparable). Hereby, the solution process can be described as an iterated calculation on the constraint matrix A of the LP. In this matrix, every variable x_i $(i = 1, ..., n)$ in the LP corresponds to a specific column. Resulting, without going into detail with the solution algorithms, increasing amounts of variables lead to an increasing runtime, in general (simplex algorithm have an exponential time complexity in theory). Further, storing a complete matrix for maybe hundreds of thousands variables takes a significant amount of memory.
Bringing the overall aim to mind, the simplex's purpose is, to produce an optimal basis solution. Especially, this means, that given a starting basis (= a feasible solution), the algorithm iterates through various other bases (in the following denoted as “sequence“) until it terminates at an optimal one. For many problems it holds, that these bases contain a very small amount of columns (variables with value not equal to zero), each. This is, compared to the amount of available variables in general. Note, that a basis can be described as a regular sub matrix of the columns of A, so that an upper bound of the basis variables is the row amount of A (including slack variables, which are not contained in the x vector). Here, the knapsack problem could

be mentioned as an example. Given a relatively small capacity and relatively many items with high weight, every possible solution will only consist of very few items. Resulting, in each computational step, just a small subset of the constraint matrix (columns) needs to be known (and stored in memory). So if one is able (which is the crucial point of column generation) to recognize and create the necessary columns / variables for the next basis solution at runtime, without knowing the complete constraint matrix, considerable computational effort can be saved. At least compared to the standard simplex algorithm, which uses computations on the whole constraint matrix to determine the next basis solution. Hereby, an important aspect is, that in many linear programs, the columns of the constraint matrix correspond to combinatorial objects (for example stable sets). Resulting, these columns (in the example, the characteristic vector of such a stable set) can be created at runtime. The matrix A needs not be stored beforehand (for a lookup of these columns), but can be computed if needed. So the question is, which of all variables improve the basis. This is commented on, in the course of this section.
All in all, just a small subset of the variables is treated explicitly in each step, or in other words: for the solution process, the knowledge of the above mentioned sequence of basis columns is sufficient. So there is no need to do calculations on the complete constraint matrix (or even store it completely), if knowing this sequence. Just the actual basis needs to be stored in memory. This, however, is a very theoretical point of view. As presented here, the knowledge of this sequence is needed beforehand, which is clearly not trivial for most problems. So without this knowledge, calculations on the whole constraint matrix are needed to recognize the next basis. Nevertheless, a customized run of the simplex algorithm (using only that sequence of variables) may save considerable amounts of time and memory. In a practical point of view, this approach is helpful for problems, which are too huge to compute in "one run", with the positive aspect of generating primal solutions (bounds) in each step.

The resulting approach of just identifying the needed / helpful variables and performing the solution process on just these columns is called column generation. This is shaped in the following way: giving a starting basis (a feasible solution for the LP), it is checked, whether an improving solution (e.g. a solution with a better objective value) exists. If there exists one, by exchanging one (or more) basic columns, switch to that solution and start again. If there exists none, the solution is already optimal. Solving the LP by this procedure can be regarded as repeatedly finding improving variables (leading to an improved solution) or proving that none exists. This can be visualized as in figure 8. Since there are only finitely many different bases existing,

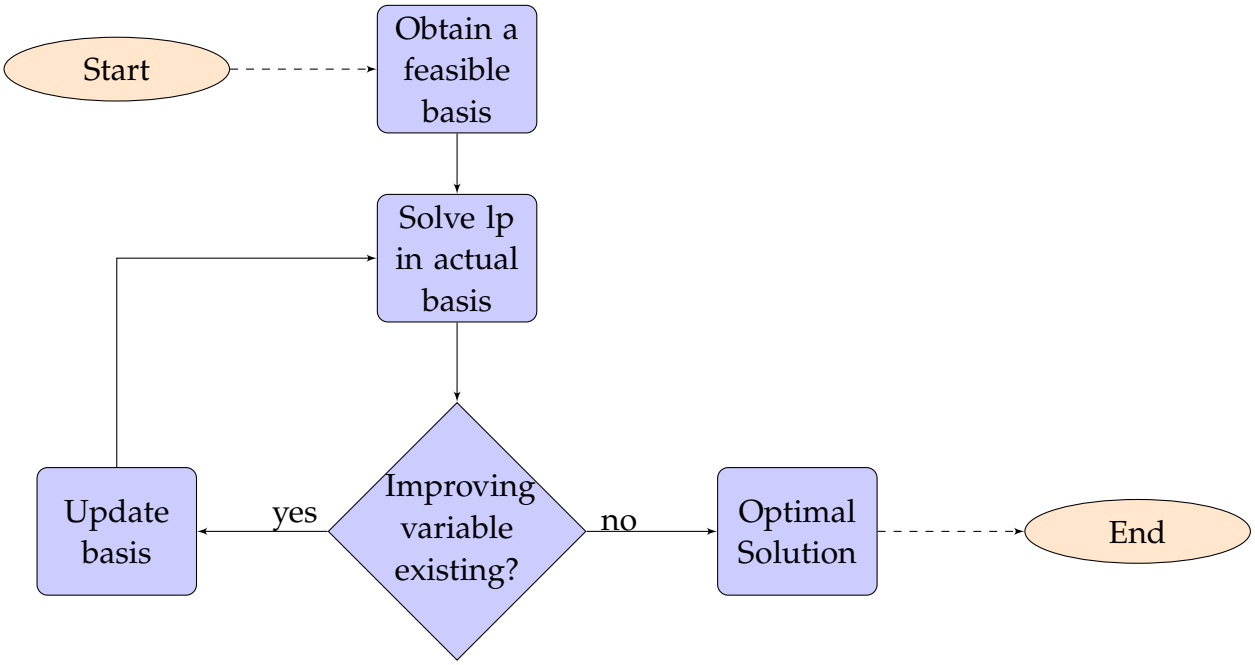

Figure 8: Schematic Column Generation Diagram

this process terminates in finite time. The key element of this procedure is the recognition of improving variables. This can be done, by the means of reduced costs. A variable improves the current solution if it has negative reduced costs (for a minimization problem). Note, that the following remarks apply similar to maximization problems, as well. Recall, that a variable's j reduced cost r_j is computed via

$$r_j = c_j - (A^t y)_j,$$

whereas y denotes the dual variables' solution vector (which is computed at very low costs during each Simplex step) and c_j is the variable's objective functions coefficient. Hereby, the number r_j denotes, how much the objective value of the current base solution improves per unit added of the variable j. Accordingly, the solution space is iteratively scanned for variables with negative r_j for recognizing the improving variables. Since an exhaustive search for every variable is too costly regarding a huge number of variables, a more purposeful way of finding one variable with negative reduced cost, can be formulated as a minimization problem:

$$\min_{j \in J} : \quad c_j - (A^t y)_j. \tag{7}$$

Hereby J denotes the set of all variables / columns. Note, that in the context of iteratively improving the current solution, this problem either gives an improving

variable (the argmin) or proves, that no improving variable exists (minimum is greater or equal than zero). Nevertheless, for an improvement a variable with negative reduced costs is sufficient, the explicit solution of (7) is not necessary. Interestingly, this problem can be formulated as an (integer) program itself, very often. In the literature (and in the following text), this is often referred to as pricing problem. Consequential in a column generation approach, the solution process is formally divided into the master (original LP) and the sub (pricing) problem.
The concept of (negative) reduced costs has a very special characteristic. A variable with negative reduced costs will improve the current solution (if it is used in the next base solution). Nevertheless, there is no mean known to forecast ,which (if more than one variable is found) variable will lead to the fastest improvement in the long run. A faster improvement at the first pricing steps may enforce more pricing steps in total, compared to choosing variables with barely negative reduced costs in the beginning. Further, a variable with negative reduced costs may not have reduced costs in the next basis solution. So there is always a trade off between finding more than one variable (for a potentially faster improvement in the long run or a higher improvement in the current step) and a bigger time consumption.
Nevertheless, since only one variable with negative reduced costs is needed (not necessarily the optimum of (7)), this problem can be treated with fast heuristics, very often. On the other hand, evaluating the optimum of (7) as greater or equal than zero proves that no improving variable exists (the solution is optimal). So the recognition of improving variables can entirely be handled by examining this problem. In practical aspects, it is mostly tried to find a variable with heuristics and if none was found, a formal evaluation of (7) follows.
The main difference between a "normal" simplex run and a column generation step is, that the simplex calculates the reduced costs for all variables (calculation on the whole constraint matrix), while the column generation focuses on finding one variable with negative reduced costs (calculation on a few columns only). Nevertheless, this gain is paid for, by the need to solve another, possibly hard problem.
Summing up, column generation allows to treat a linear program with a huge constraint matrix on very small space (only the current basis needs to be stored, the simplex operates only on the old and new basis variables), thus enhancing computability. This is paid for, by some drawbacks. At first, one must be able to compute $(A^t y)_j$ detached of the knowledge of the whole matrix A for every variable. At second, while this process converges to the optimum in theory, the correspondence

$$OPT \;=\; (\; SEP \;=\;)\; PRICE$$

(optimization = separation = pricing) states (for example in [GLS95]), that solving the original problem is as hard as finding an improving variable (or a valid cutting

plane). Hence, this concept cannot reduce the problems theoretical difficulty/complexity (it just enhances computability, in the sense of bypassing memory restrictions).
As mentioned above, every variable with negative reduced costs improves the current solution, but it is not known, how to calculate the shortest/fastest sequence leading to the optimum. A variable with negative reduced costs very close to zero might be better than the argmin of (7) or vice versa. In practical issues, this is addressed by generating sufficiently many variables with negative reduced costs and hoping for the best. Nevertheless many positive practical experiences have been made, e.g. in [MT95].

This concept is made for linear programming. Using it in integer programs (on the linear relaxation) needs to be done carefully, since branching in the original problem may lead to further inequalities, which may disturb the (generic) computation of (7), since it introduces new dual variables (so every node in the branching tree would have a different pricing problem). At a side note: a nice way of dealing with the above mentioned problem is shown in the graph coloring example [MT95]. Branching is realized in a modification of the (input) data (in this case a modification of the underlying graph) and not with additional explicit constraints. Nevertheless, this way of treating mixed integer problems is called "branch and price" and is increasingly used in research, mainly for huge problems, instead of a "branch and bound" approach. Similar problems occur when using feasible inequalities: generically adding them might cause the same problems with the pricing problem.
In general, it is to mention, that column generation might be seen as the dual / complementary expression of generating valid inequalities (column generation versus row generation). Additional information and the theoretical background of column generation can be found in [LD05], for example.

In the following section, a formulation for the first stage of the FAPH problem is introduced, which will allow a column generation approach more easily than the old [FAPHSFH] model. Hereby, it is focused on a column generation approach on the linear relaxation. A possibly following branching for integer solutions is left out (compare section 6.3).

4.2 — *Reformulated model* —

In this section, a formulation of the above presented first stage of the FAPH problem, on the background of developing a column generation accessible model, is

given. Hereby, due to the similarity of FAPH towards the classical frequency assignment problems (FAP) and therefore graph coloring, this approach is inspired by [MT95], since a similar approach is invoked there.
Due to the above presented splitting into two problems, the aspect of adjacent channel interference is totally left out, as well as the aspect of separation constraints and local blockings. Resulting, adjacent channel interference is addressed with a separate, second stage optimization problem, which is presented later on in chapter (6). As a first step, a frequency assignment problem, minimizing total co-channel interference is outlined, and in the second step, this frequency plan is finished with respect to adjacent channel interference.

This first stage problem is a restricted FAPH problem and in the following denoted with FAPHRES. While a formal (mathematical) definition is given in section 4.3, this problem can be described as: Given the set of STRX and their demanded frequencies, the set of available frequencies and the potential co-channel interferences. Which feasible frequency assignment minimizes the total co-channel interference.
Resulting, in the following MIP formulation (a formal prove that this is a formulation for FAPHRES can be found in section 4.3), there is only one type of variables, any longer. For every subset $T \subseteq \mathcal{S}$, the (integer) variable

$$x_T \in \mathbb{Z}_+$$

denotes, whether all STRX in T share x_T frequencies (in the final assignment). Obviously these are exponentially many variables ($2^{|\mathcal{S}|}$, in the order of all possible subsets of STRX), which is the reason for a column generation approach. Then, the whole problem can be formulated as:

$$min : \sum_{T \subseteq \mathcal{S}} co(T) \cdot x_T \tag{8}$$

with constraints:

$$\sum_{\substack{T \subseteq \mathcal{S} \\ I \in T}} x_T = k[I] \qquad \forall\ I \in \mathcal{S} \tag{9}$$

$$\sum_{T \subseteq \mathcal{S}} x_T = |F| \tag{10}$$

Hereby, the notation of the preceding chapters is maintained. That is:

- For all $I \in \mathcal{S}$, $k[I]$ denotes the amount of frequencies, which STRX I needs to have.

- $|F|$ is the amount of available frequencies.
- $co(T)$ gives the amount of co-channel interference, induced by the STRX in $T \subseteq \mathcal{S}$.

Exactly spoken, the co-channel interference of a set $T \subseteq \mathcal{S}$ is (derived from the definition of interference on STRX level, see equation 3) defined by

$$\begin{aligned} co(T) &:= \sum_{I \in T} \sum_{J \in T} \overline{co(I,J)} \\ &= \sum_{I \in T} \sum_{J \in T} \sum_{v \in I} \sum_{w \in J} \left(\frac{1}{k_I} \cdot \frac{1}{k_J}\right) \cdot co(v,w). \end{aligned}$$

Note, that though the notations of induced co-channel interference are very similar for sets, STRX and TRX, it should be clear by the actual context, which measure is meant. In the following, this model is referred to with [FAPHCG], in contrast to the first model, [FAPHSFH]. Note, that in the constraints (9) and (10) the demanded equality may be changed into greater or equal ("$\geq$") and lesser or equal ("$\leq$"), respectively. Since this is a minimization problem and more assigned frequencies (at last in not degenerated scenarios) induce more interference, the constraints will be fulfilled with equality anyways. Similar for constraint (10), fewer available frequencies will induce more interference. Resulting the maximum number of frequencies will be used, such that (10) will be fulfilled with equality, too.
For completeness sake, some short explanations are given, showing that the interference measure of (co-channel) interference in the old model and in the new model are equivalent (concerning their integer solutions). Starting with a fixed frequency assignment and the total interference in the old model, it follows that

$$\begin{aligned} \sum_{I \in S} \sum_{J \in S} \overline{co(I,J)} y_{I,J} &= \sum_{I \in \mathcal{S}} \sum_{J \in \mathcal{S}} \left(\sum_{v \in I} \sum_{w \in J} \sum_{f \in F} \left[z_{I,J,f} \cdot \left(\frac{1}{k_I} \frac{1}{k_J}\right) \cdot co(v,w) \right] \right) \\ &= \sum_{f \in F} \left(\sum_{I \in \mathcal{S}} \sum_{J \in \mathcal{S}} \left[z_{I,J,f} \cdot \overline{co(I,J)} \right] \right) \\ &= \sum_{f \in F} co(T_f). \end{aligned}$$

Here T_f denotes the set of all STRX which have frequency f in common (given an assignment/result of the model [FAPHSFH]):

$$T_f := \left\{ I \in \mathcal{S} \mid x_{I,f} = 1 \right\}.$$

It follows, that

$$\sum_{f \in F} co(T_f) = \sum_{T \subseteq \mathcal{S}} co(T) \cdot x_T.$$

The last equality holds, by equation 10. So both models measure the occurring (co-channel) interference in an equivalent way. Resulting, the input can be treated with both models with comparable results.

Before going into detail concerning the solution process, a short example for visualizing this model functionality, is presented. Given three STRX, with a demand of 2, 3 and 5 frequencies respectively, and 7 available frequencies in total, this leads to the following equations:

$$\begin{aligned}
x_{\{1\}} + x_{\{1,2\}} + x_{\{1,3\}} + x_{\{1,2,3\}} &= 2 \\
x_{\{2\}} + x_{\{1,2\}} + x_{\{2,3\}} + x_{\{1,2,3\}} &= 3 \\
x_{\{3\}} + x_{\{1,3\}} + x_{\{2,3\}} + x_{\{1,2,3\}} &= 5 \\
x_{\{1\}} + +x_{\{2\}} + x_{\{3\}} + x_{\{1,2\}} + x_{\{1,3\}} + x_{\{2,3\}} + x_{\{1,2,3\}} &= 7
\end{aligned}$$

With a feasible solution of

$$\begin{aligned}
&x_{\{1\}} = 2 \quad x_{\{2,3\}} = 3 \quad x_{\{3\}} = 2 \\
&x_T = 0 \qquad \forall\, T \subseteq \mathcal{S}, T \notin \{\{1\}, \{3\}, \{2,3\}\}.
\end{aligned}$$

In this context, the conditions (9) provide enough frequencies for every STRX and the condition (10) limits the amount of used frequencies to the available ones. Hereby, repeating that the whole problem is detached of the explicit frequencies. It calculates the arising interference via groups of STRX sharing a (not determined) frequency, only. This results in the following facts: Out of every frequency assignment, the sets containing the STRX transmitting on the same frequency are constructed easily, but the other way is not unique (the assignment, resulting of the solution of the above problem, is not unique). In the above example, STRX 1 could have the frequencies 1 and 2 assigned, as well as the frequencies 5 and 7. This is the same phenomena, which leads to the bad branching behavior in the old FAPH model [FPHSFH] (see section 3.3), the symmetry of frequency assignments (though this symmetry is lessened, when respecting adjacent channel interference). On the one hand, this is a positive aspect, since it gives the opportunity to assign the specific frequencies in a second stage optimization (choose the best of the symmetric assignments, with respect to minimal adjacent channel interference). On the other hand, this is the reason, why adjacent channel interference can not be incooperated in this model in the first

stage: Adjacent channel interference is dependent on the sequence of the assigned frequencies, which is not resolved in this model. As a consequence, this model can create an optimal frequency assignment (which is not uniquely determined) with respect to co-channel interference, but no globally optimal assignment (respecting co- and adjacent channel interference). Bearing in mind, that the first model failed in a computational aspect, this may be a sensible trade off between solvability and optimality.

In the version above, this model has no separation and blocking constraints incorporated. This is reasoned in the formulation itself. As mentioned above, it is only calculated, which STRX share a frequency, but the exact frequency is not determined. But most of these (additional) conditions (like blockings and separations bigger than one) can only be treated, when assigning specific frequencies, as e.g. in the second stage problem (6.2). Nevertheless, having in mind that a final frequency assignment will assign frequencies to already fixed variables / sets x_T, a later processment towards this constraints may need some special treatment in the first stage (here), too. Explicitly, imposing $d_{I,J}$ values bigger than zero prohibits the usage of variables x_T with I and J in T:

$$d_{I,J} > 0 \Rightarrow x_T = 0 \quad \forall\, T \subseteq \mathcal{S}, I, J \in T.$$

Resulting, some of these constraints (like outer STRX separation of the value one) could be incorporated in this model by restricting the corresponding variable usage. A similar discussion about further secondary constraints (though they are not treated in general) can be found in section 6.2.

As a conclusion and identifying similar LP structures, this problem can be described as a minimal weighted (objective function) set partitioning (10) problem with minimum (node) levels (9). In the next section, before the column generation approach is introduced in detail and the specific pricing problem for the [FAPHCG] model is announced (section 4.4), the complexity of FAPHRES is analyzed and compared to the old formulation.

4.3 — *Complexity Analysis and a comparison to [FAPHSFH]* —

Since the FAPHRES problem ([FAPHCG] formulation) is closely connected to the FAPH problem ([FAPHSFH] formulation), the FAPHRES problem is NP-complete, as well. Nevertheless, FAPHRES as standalone problem, is a severely simplified version of the FAPH problem, namely adjacent channel interference is omitted, as well

as local blockings and separation issues. Accordingly, a formal prove is given in the following (for a short introduction on complexity analysis, section 8.1 is referred to, again).

For completeness sake, the FAPHRES problem is formally defined here, again (compare the section before for a spoken definition). Analogous to the FAPH problem, the (restricted) input can be described as a six-tuple (or a "restricted STRX graph")

$$Q := (\mathcal{S}, E, F, k_I, co),$$

with the same notations as in section 3.4. Similar, a feasible solution (a feasible assignment) for such a tuple is a function

$$y_3 : \mathcal{S} \to 2^{|F|},$$

mapping each vertex to a set of frequencies, such that

$$|y_3(I)| = k_I \qquad \forall I \in \mathcal{S}.$$

Then, FAPHRES is defined by: Given a restricted STRX graph, the problem

$$\min_{y_3} : \quad \sum_{(I,J)\in E} \sum_{\substack{y(I) \\ y(I)\in y(J)}} 1 \cdot \overline{co(I,J)}$$

for a feasible solution y_3, is called the restricted frequency assignment problem for slow frequency hopping networks (FAPHRES).

By the above definition, the [FAPHCG] model is a formulation of the FAPHRES problem, a very restricted subclass of the FAPH problem. Note, that hereby a straight forward representation of a node's assignment with variables of the form $x_{I,f}$ is not chosen, but a formulation denoting which nodes sharing an assignment (similar to the different possible graph coloring models).

This can be derived from the following line of thoughts: Given a solution of the FAPHRES, the sets of STRX sharing a frequency can be obtained easily. These sets can be interpreted as a solution for [FAPHCG] (8) (these sets form the variables x_T, with the amount of shared frequencies as solution value). Since only |F| frequencies have been chosen, constraint 10 holds, and $|y_3(I)| = k_I$, this solution is feasible. If this solution was not optimal for [FPHCG], this would lead to a contradiction. Namely, assumed that a solution with strictly lower objective value is existing, from that solution, a solution for FAPHRES could be obtained by assigning to every set

x_T frequencies, not assigned to other sets before (that is possible, since [10] holds). As both models measure the interference equivalently (see the section above), this would be the mentioned contradiction. Resulting, the [FAPHCG] problem is a formulation of FAPHRES.
Though FAPHRES is a simplification of FAPH, it is still NP-complete. In the following, this is shown, by the same reduction as from the minimum edge deletion k-partion problem to FAP, cited in section 3.4 from [Eis01] (where the NP-completeness of this problem is shown as well). For completeness sake, the definition of the NP-complete "minimum edge k-partitition problem" is repeated here:
An instance of the minimum edge deletion k-partitition problem consists (MEDKP) of a Graph $G = (V, E)$, a weighting $w : E \to \mathbb{Z}_+$ of the edges, and a positive integer $k \leq |V|$. The objective is to find a partitition of V into at most k disjoint sets $V_1, ..., V_p$ such that

$$z := \sum_{l=1}^{p} \sum_{\substack{(i,j)\in E \\ i,j\in V_l}} w_{ij}$$

is minimized.

As mentioned above, the reduction showing NP-completeness will be MEDKP $\to$ FAPHRES, similar to the MEDKP $\to$ FAP reduction in [Eis01]. This reduction can be described via

$$(V, E, w, k) \mapsto (V, E, \{1, .., k\}, 1, w).$$

Hereby, the graph partitions and the frequency assignments correspond to each other, such that the FAPHRES problem is NP-complete, as well.

Since both, FAPHSFH and FAPHRES are equally hard, the question arises, why the [FAPHCG] model is preferable to the [FAPHSFH] model. The reason is, that the [FAPHSFH] model is limited to the features, treated in [FAPHCG] as well, in the following. This means, adjacent channel interference and separations/blockings are left out. Even with these restrictions, the problems with the computability (with the linear relaxation) mentioned in section 3.3, persist (without showing this in details here, though). On the other hand, the [FAPHCG] model will show a significant improvement, there. Hereby it is to mention, that both formulations (their linear relaxations, respectively) are compared by their solution values and not in the sense of polyeder theory (the one solution polyeder is contained in the other and therefore the first formulation is better or vise versa). In the following, an example is given,

which underlines the superiority of the [FAPHCG] formulation. Though this is an example, only, the characteristics can be generalized for all (not degenerated) input instances. In the [FAPHSFH] model, a fractional solution will always induce an objective value of zero. Contrary, in the [FAPH] problem, every chosen set T will most likely produce some interference, it is not possible to fulfill constraints 9 without inducing interference.
To compare both models' linear relaxation and emphazising the above statement, the following example can be regarded:

$$\begin{aligned} &\mathcal{S} = \{1,2,3\}, && k_I = 1 && \forall\, I \in \mathcal{S}, \\ &F = \{1,2\}, && co(I,J) = 1 && \forall\, I,J \in \mathcal{S} \\ & && \Rightarrow co(\{1,2,3\}) = 6. \end{aligned}$$

Obviously, every integer solution will induce an interference value strictly above zero (at least two STRX will interfere each other), the minimal induced interference of an frequency assignment is exactly one), since the number of available frequencies is too low to give every STRX it's own frequency. Nevertheless, both models relaxations will produce more or less helpful/sensible results in approximating these "integer interference" level.
The [FAPHSFH] model will choose every frequency at every node with value 0.5, namely

$$x_{I,f} = \frac{1}{2} \qquad \forall\, I \in \mathcal{S}, f \in F.$$

Resulting, the penalty values need not be above zero for all pairs (I,J) of STRX, namely

$$z_{I,J,f} = 0 \quad \forall\, f \in F \quad \Leftrightarrow \quad y_{I,J} = 0.$$

Leading to a[FAPHSFH] solution (for the linear relaxation) of value zero.

On the other hand, the [FAPHCG] model will produce a solution for the relaxation, which is a better approximation (in this example, it is even the solution for the optimal integer solution). The only possibility of having an objective value of zero would be, to exclusively choose one-elementary sets for every STRX (assign every STRX a different frequency), which is not possible (three needed frequencies). Without using too much formalism, the best solution will use as many one-elementary sets as possible, completing the assignment with sets inducing interference. For example, an (optimal, by a tedious calculation with the simplex algorithm) solution is

given by

$$x_{\{2,1\}} = x_{\{3\}} = 1$$
$$x_{\{1\}} = x_{\{2\}} = x_{\{2,3\}} = x_{\{1,3\}} = x_{\{1,2,3\}} = 0.$$

This solution (for the linear relaxation) induces an interference value of 1, and is already the optimal (integer) solution as the one of [FAPHSFH]. Though an optimal solution for the integer problem cannot be expected of the linear relaxation of [FAPHCG] in general, the solution of the linear relaxation from [FAPHCG] is always better than the one of [FPAHSFH]. As soon as variables / sets are introduced into [FAPHCG] (with a value of above zero, even if this is very close to zero), they will have induced interference of above zero (with a very high probability) and therefore forcing the solution value away from zero. Contrary, in the [FAPHSFH] model, variables may be used (e.g. the $x_{I,f}$ variables) without changing the objective function), confining the solution of [FAPHCG] and [FAPHSFH]. Even within the settings of a column generation approach, this result remains true (in the context of the linear relaxation, column generation can be regarded as solving on a subset of all variables). Therefore, the change from [FAPHSFH] towards [FAPHCG] is justified, (at least) concerning the sensibility (quality of the approximation) of solutions of the linear relaxation.

4.4 — *Pricing in Detail* —

As mentioned earlier, the [FAPHCG] model is built by relatively few inequalities (in the order of the amount of STRX) but exponentially many variables (one for every possible subset of $\mathcal{S}$). On the other hand, in every feasible (integer) basis solution, just a very small part is not equal to zero (see constraint (10), at most $|F|$). As mentioned in the abstract about column generation in general, this can be seen as a classic setting for a (branch and) price approach. In the following, the constraint matrix of [FAPHCG] is denoted with A, thus it holds, that

$$A \in \mathbb{R}^{|\mathcal{S}|+1 \times 2^{|\mathcal{S}|}}.$$

At the beginning of the pricing algorithm, a feasible basis solution is easily obtained, with the variable / set containing all STRX (hereby, no degeneration in the input data, like less available frequencies than demanded frequencies is assumed). Clearly, this solution is "bad" in the sense, that it will produce the maximum amount of interference possible. Other methods for producing better or more appropriate solutions are presented in section 4.5.

Then, according to section 4.1, the pricing problem of [FAPHCG] can be formulated as

$$\min_{T \subseteq \mathcal{S}} \quad co(T) - (\chi_T)^t \cdot y - y_0.$$

Giving a short explanation: a variable (a subset T of $\mathcal{S}$) is searched, with minimal costs. The costs are build up from the set's original objective function's coefficient, which is it's induced co-channel interference and the product of the corresponding column of the matrix A^t and the vector of the constraint's dual values. During this work, this is often referred to as "a STRX corresponding dual value". Since every STRX corresponds to exactly one constraint, this constraint's dual value in the current LP relaxation is meant and therefore well defined. Having a closer look at A, it appears, that for every set T, the corresponding column in A^t is it's characteristic vector χ_T, attached with a "one" at the end (from the last constraint (10)). The vector of the constraint's duals can be divided into two parts, $y \in \mathbb{R}^{|\mathcal{S}|}$ (from (9)) and $y_0 \in \mathbb{R}$ (from (10)). Consequential the last entry from the duals can be detached for every set T, in the calculation of the objectives value, since it is a constant offset in every pricing step.
Since y_0 is constant, a mixed integer formulation of the pricing problem [FAPHCG-PRICE], giving an optimal subset T, is given by

$$\begin{aligned}
min: \quad & \sum_{I \in \mathcal{S}} \sum_{J \in \mathcal{S}} co(I,J) \cdot z_{I,J} - \sum_{I \in \mathcal{S}} x_I \cdot y_I \\
subto: \quad & z_{I,J} \geq x_I + x_J - 1 && \forall\ I, J \in \mathcal{S}, \\
& z_{I,J} \in \{0,1\} && \forall\ I, J \in \mathcal{S}, \\
& x_I \in \{0,1\} && \forall\ I \in \mathcal{S}.
\end{aligned}$$

Here, x_I denotes whether STRX I is in the solution set T and the $z_{I,J}$ are needed to calculate the induced interference of the current set. The value of y_0 is left out, since it is a constant offset. Resulting, the outcome for the pricing problem is strictly lower then y_0, if a improving variable exists and greater or equal to y_0 if the solution is already optimal.
It is remarkable, that the structure of this problem (the constraints) is closely related to the original [FAPHSFH] problem (similar usage of penalty variables). Nevertheless, it is a lot smaller, since the z variables are not dependent on the specific frequency any longer (and the adjacent channel interference settings are ruled out). Another fact, which makes up for a big difference in runtime is, that the mixed integer program does not need to be solved to optimality in most cases. As presented in the

section about column generation, one variable with reduced cost is already enough. This opens up the door for (a) fast heuristics further (b) limits the branch and bound process of the pricing problem to finding a feasible solution with corresponding objective value (no need for a total processment of the complete branch and bound tree in most cases). Nevertheless, in cases, where the explicit pricing problem has to be solved, because heuristics have not been successful, this is possible (contrary to the [FAPHSFH] model), but very slow. This means, the pricing problem as standalone is solvable, but problems with memory usage could still arise, when solving in a column generation process (since column generation may use significant amounts of memory as well, such that the combined memory claims may be to big).
Furthermore it is repeated, that the pricing problem is in NP and especially not in P, since it holds that it is equally hard as it's corresponding master problem ("optimization=pricing", [GLS95]). So improving variables might be found easily (heuristics), but an explicit solution of the pricing problem remains theoretically hard (we will come back to this in the section about dual bounds of the current solution of the linear relaxation, 5.3).

In the next three subsections some improving work on the solving / pricing process is presented. At first, explanations on generating good starting solutions are given. In the second section, some heuristics are presented, exploiting the fact that the pricing problem is seldomly needed to be solved optimally and so, speeding up the solution process significantly. In the third section, comments are made, on various problems, which arising, when the new (outpriced) variables are embedded.

4.5 — *Starting heuristics* —

As mentioned above, obtaining a feasible starting solution is quiet easy (with one variable/set containing all STRX), but with the drawback of a very bad objective value. Since the runtime of column generation cannot be forecasted, there is no definitive answer, which starting solution is better, concerning the runtime of the whole solving process. Nevertheless, starting with a solution closer to the optimal seems quiet logical. At least, with a better starting solution, all following solutions have a certain minimum "quality" (maximum level of interference), which may be desired in practical aspects.
Obviously it holds, that the more time is invested, the better (starting) solutions can be obtained. In this attempt, a starting solution is obtained with a polynomial time starting heuristic (which is called DSATUR). This refers to a method, originally inspi-

red by a graph-coloring heuristic and later adapted to FAP (for example in [Eis01]), which can be applied here, too. The basic concept is, to assign frequencies dynamically, in the order of the STRX "difficulty". This is meant in the following way: It is checked, which STRX is the currently hardest to assign (hardest, by the means of highest inducing interference, which is explained in detail, later) and which would be the best channels to assign to it. These channels are allocated to the STRX and the process is repeated until every STRX is assigned (hereby assigning frequencies may change the difficulty of assigning other STRX, as well as it influences the frequency choices of other STRX). Resulting, a complete frequency assignment is created by a dynamic sequence. So, this heuristic belongs to the class of starting / construction heuristics.res to a method, originally inspired by a graph-coloring heuristic and later adapted to FAP (for example in [Eis01]), which can be applied here, too. The basic concept is, to assign frequencies dynamically, in the order of the STRX "difficulty". This is meant in the following way: It is checked, which STRX is the currently hardest to assign (hardest, by the means of highest inducing interference, which is explained in detail, later) and which would be the best channels to assign to it. These channels are allocated to the STRX and the process is repeated until every STRX is assigned (hereby assigning frequencies may change the difficulty of assigning other STRX, as well as it influences the frequency choices of other STRX). Resulting, a complete frequency assignment is created by a dynamic sequence. So, this heuristic belongs to the class of starting / construction heuristics.

Going into detail (having a computational implementation in mind), the whole algorithm operates on a cost-matrix

$$A := a(I, f) \in \mathbb{R}^{|\mathcal{S}| \times |F|}$$

(not to be confounded with the constraint matrix of the [FAPHCG] problem). This matrix gives the (co-channel) interference it would cost, to assign frequency f to STRX I in the current (partial) assignment. Hereby, if an assignment is created sequentially, a partial assignment is an assignment, in which not all STRX are completely assigned. Interference values in this partial assignment are the induced interference values of this assignment. Mathematically:

$$a(I, f) := \sum_{\substack{J \in PA, \\ \text{J has f assigned}}} co(I, J) \qquad \forall\ I \in \mathcal{S}, f \in F,$$

whereas PA denotes the maximal (partial) assignment, meaning the maximal subset of all STRX, whose STRX have been assigned at least partially (at least one frequency assigned to every STRX). Further, a cost vector is needed, which stores the sum of the

interferences $c(I)$, potentially induced by STRX I in the current assignment (which is the measure of "hardness" for assigning a specific STRX). This is defined as the sum over all frequencies of the current costs of that frequency - STRX combination:

$$c(I) := \sum_{f \in F} a(f, I).$$

All in all, the heuristic consists of repeatedly assigning STRX and updating the cost-matrix and vector, until the assignment is complete. For clearness sake, the algorithm is outlined in pseudo code in the following section.

The DSATUR heuristic allows to generate solutions, which are not trivially improvable. Note, that DSATUR, as presented here, generates a frequency assignment in the notation of the [FAPHSFH] model, from which the description in the way of [FAPHCG] can be obtained easily (compare section 4.3). Nevertheless, this heuristic has some drawbacks. Because of it's heuristical character, DSATUR will probably not produce the optimal solution and at the same time, no quality guarantees on solution quality can be made. A further drawback is, that DSATUR produces integer solutions. This is useful concerning the overall solution as a frequency assignment, but is of limited use concerning the solution of the linear relaxation. Since no investigations in the direction of heuristics for the linear relaxation were made in this work, there might still be room for improvements.
Though the above mentioned DSATUR heuristic produces acceptable starting solutions, a further (improvement) heuristic, a typical 1-opt step, was implemented. Given a complete assignment, e.g from DSATUR this heuristic orders the STRX I regarding their induced interference $\mathrm{Ind}(I)$, defined by

$$\mathrm{Ind}(I) := \mathrm{Ind}(Ass) - \mathrm{Ind}(Ass \setminus I),$$

whereas $\mathrm{Ind}(Ass)$ describes the total interference of the current assignment and $\mathrm{Ind}(Ass \setminus I)$ describes the interference induced by the resulting partial assignment, when STRX I is unassigned/ignored. In the next step, it is subsequently tried, whether a better assignment can be made for every STRX, namely in unassigning it and reassigning it with the frequencies, which would induce the least interference in the total assignment. If an improvement could be made, the process is started all over again (therefore the heuristic classically belongs to the class of improvement heuristics). While this heuristic worked as desired and improved given (integer) solutions, using this heuristic is very costly since comparing different "new" assignments with respect to their induced interference needs many computation on the whole interference map. For every STRX I there are $O(|F|^{k[I]})$ possible assignments, with

costs for recognizing it's interference in the order of the adjacent STRX of I (at most $O(|\mathcal{S}| \cdot |\mathcal{S}|)$), with all calculations repeated for every improvement made.
While this time consumption is okay as a one time operation at the beginning, it restricts the heuristic's later usage, as is described in the next but one section. Some pseudo code of the 1-opt step can be found, in the next section.

Note that both heuristics (DSATUR and 1-opt) deal with integer solutions (or with the improvement of integer solutions). While this concept is helpful concerning the overall aim of feasible frequency assignments, both are limited in their quality in relation to the linear relaxation, since they cannot exploit fractional structures/values. Nevertheless, they serve as a sufficient starting point for column generation, whose practical results on some instances are presented in section 5.1.

4.5.1 Pseudo code Starting heuristics

Referring to the description of DSATUR above, the algorithm is outlined in the following pseudo code:

```
Initialize:   // Cost matrix / -vector

double a(I,J) = 0;   // For all I in STRX, J in F
double c(I) = 0;     // For all I in STRX

Algorithm:

while ( Assignment not complete )
{
  Choose I[max];
  // STRX I with highest c(I) value, I not assigned

  For (int j = 1; j < k[I[max]]; j++)
  {
    Choose f[min];
    // Frequency f with lowest a(I[max],f) value,
    // f not already assigned to I

    Assign f -> I;

    // Update cost-matrix and total costs
```

```
    a(J,f[min]) += co(J,I[max]);
    c(J)        += co(J,I);
  }
}
```

Algorithm 1: DSATUR Heuristic

Note, that an implementation, where the chosen STRX is not "fully assigned" is possible too. That means, after every single frequency assignment, the costs are updated and a (potential) new maximum STRX is chosen.

At the end of this section, the pseudo code of the 1-opt step is presented. Starting with a complete assignment, in the notation of the section before, the 1-opt algorithm can be described as:

```
Initialize:

bool Improvement = true;
double old_costs = Ind(Ass);
Algorithm:

While (Improvement)
{
  Calculate: Ind(I);

  Choose I[max];
  // Maximum Ind(I) value

  Unassign I[max];

  Choose f[min];
  // k[I[max]] Frequencies with would induce
  // the least interference

  Assign f[min] -> I[max];
  double new_costs = Ind(Ass);

  if (old_costs <= new_costs)
    Improvement = false;
```

```
  else
    old_costs = new_costs;
}
```

Algorithm 2: 1-opt Step

Here, different approaches as reassigning all $k[I]$ frequencies or just a single frequency (e.g. the one inducing most interference) can be made, similar as in DSATUR.

4.6 — *Pricing heuristics* —

Since the aim of the pricing is, to find appropriate variables (objective function of the pricing problem strictly lower than zero) iteratively for every basic solution, the pricing problem needs to be solved very often and therefore, as fast as possible. Nevertheless, the pricing problem needs not to be solved to optimality, such that it can be treated by heuristics in many cases. In extreme cases, when the heuristics are always able to produce feasible solutions, the pricing problem just needs to be solved once, at the end to prove optimality, and not to generate new columns. Nevertheless, the here presented heuristics and their variants may not be able to produce feasible results, so that in this case, the pricing problem needs to be solved, until it finds a feasible solution (not necessarily optimal) with negative reduced costs. Since the pricing problem is not very restrictive in it's constraints (all in all they just tell to choose an appropriate (low cost) subset of STRX), there is only the objective function, to work with. Since a variable can be expressed as a subset of STRX, both expressions are used synonymous in the following. For a heuristic's success, it all depends on a selection of some STRX, whose inbound interference is low, compared to their corresponding (constraint) dual values. In the following, the purpose and the general idea behind the here presented heuristics are explained, using the notation of the pricing problem [FAPHPRICE] in section 4.4. Detailed pseudo-code can be found at the end of this section, again.
Here, the focus lies on two (main) variations of greedy heuristics. Both aim is, to create adequate subsets of STRX, and both kind of operate inversely to each other. The key element of both heuristics is the ordering of STRX, on which is commented later on, in more detail. Nevertheless, here are different variations possible again, each with different consequences. So in the following, the two types of greedy heuristics (increasing / decreasing) and their variants (different ordering priorities) are introduced.

The first heuristic starts with an empty set of STRX and tries to expand it as much

a possible (as long as the set's reduced costs stay negative). This is done the following way: The STRX are ordered decreasingly, according to their corresponding dual values. Then the STRX are added into the set, according to that order, as long as the objective value stays below y_0.
The second heuristic operates the other way around. It starts with the set of all STRX with positive dual value (and therefore the objective value /reduced cost is most likely above y_0) and subsequently eliminates STRX (again, the STRX with the highest dual value first) until the objective value is below y_0. Note, that in both cases, if no STRX with a positive dual value exits, the current solution is already optimal (since interference is always positive, the objective function can not become negative in this case).

Both heuristic's runtime are polynomially bounded, since the most costly operations are the ordering of the STRX and the (more expensive) calculations of the inbound interference, which are possible in polynomial time, both. So they are very fast and can potentially be applied in every pricing step. Both generate new variables during the whole pricing process and are very helpful in that perspective (detailed results are presented in section 5.1). Nevertheless, in the proposed way, both use the dual values of the STRX as decision criterion, only. This addresses only "half" of the objective function. Without going into detail too much, both heuristics can be extended to consider the interference, too (as another variation of the heuristic types). For instance, the STRX are added to the set, if the decrease through dual values is higher than the increase through inbound interference. Nevertheless, since the increased amount of interference costs calculations are rather expensive, this method has it's drawbacks, too. At the end of this section, the pseudo code of this variations, is presented, too.

Interesting to mention are the following characteristics: The first heuristic apparently creates relatively small and the second heuristic relatively large sets. This has to be noted carefully. Assuming some kind of even distributed interference and a "dense enough" interference graph (though this assumption may contradict to reality, since the interference relations in typical instances are mostly "sparse"), it is most likely that in an optimal frequency assignment, all frequencies are used and that all sets sharing the same frequency will be of about the same size. The reason is, that every STRX in a set T with $|T| = n$ increases the number of potential interferences (the number of STRX interfering each other, not the interference value) by a factor of n (the potential interference increases with $O(n^2)$. For example in a set with one STRX, there is no potential interference, in a set with two STRX, there is one poten-

tial interference, in a set of n elements, there are $n \cdot (n-1)$ potential interferences. So the number of potential interferences rises quadratically and therefore they will probably induce too much interference if the number of STRX grows too big. From a stochastical point of view, the sets' sizes will center around $\lceil n/f \rceil$. On the background of that thought, most variables created by heuristic one (two) are to small (big). So it might be necessary, to rephrase them, e.g. for producing sets close to the desired size.
In this work, the following trade-off was used. Since the heuristics are relatively fast, they can be run many times, without major impact on the performance. Resulting, these heuristics were applied in some different ways. At first, in the way mentioned above, and in the following, as a further criterion, the set's size was limited (if possible) to the range of

$$|T| \in \left\lfloor \frac{f}{n} \right\rfloor \pm 2.$$

Though the here presented heuristics and their variants offer a lot of possibilities, there is surely room for other and potentially better heuristics, in future research. Nevertheless, within the purpose of this thesis, sufficient results could be obtained, which are presented in section 5.1. In the following, the pseudo code of the above heuristic is presented.

4.6.1 Pseudo code

The first heuristic, generating variables / sets increasingly:

```
Initialize:

  set T = {};                  // The new variable
  double current_costs = 0;    // cost of current T
  int demanded_size = x;       // If desired
  bool feasible;

  sort(STRX);                  // According to dual values

Algorithm:

  for ( I in STRX )            // In the order of above, highest first
  {
    // costs(T) = interference(T) - dual_values(for all I in T)
```

```
    if ( costs(T union I) < current_costs)
    {
      T := T union I;
      current_costs = costs(T);
      ((re)sort(STRX); )     // If ordering includes interference
    }
    STRX = STRX \ I;

    // If a special size is desired
    (if (size(T) = demanded_size) break; )
  }
  if (size(T) > 0)
    feasible = true;
  else
    feasible = false;
```

Algorithm 3: Greedy Heuristic (Increasing)

Note, that the ordering may also imply interference (as another variant). So the "costs" of a STRX might be it's dual value or it's dual value minus it's additional induced interference in the current assignment. Apparently, there are many different variations possible.
A similar variant be can expressed as a "stable-set" approach. As above the STRX are sorted according to their dual values, but they may only be inserted into the new variable as long as the inbound interference of the new variable / set is zero. Concerning the interference relations, a weighted (dual values) stable set is created in a greedy way. Naturally, this approach is kind of limited, since no interference is accepted at all, but it still produces (different) feasible solutions and is therefore useful in the above mentioned context.

The opposite heuristic, generating sets decreasingly:

```
Initialize:

  set T = { STRX | duals >= 0 };
  double current_costs = costs(s);
  int demanded_size = x;
  bool feasible;
  sort(T);              // E.g. according to dual values
```

Algorithm:

```
While(current_costs > y_0 && 0 < size(T) < demanded_size )
{
  T := T − T[max];
  current_costs = costs(T);
  ((Re)sort(T);)
}
if (size(T) > 0)
  feasible = true;
else
  feasible = false;
```

Algorithm 4: Greedy Heuristic (Decreasing)

Clearly, if the maximum element (with respect to the dual value) is selected, one can consider a variant, where the minimal dual value is selected instead of the maximum, as well. Contrary, in the example above, a "minimum" increase is not useful, since bigger sets imply a higher interference (loss) compared to the dual value (gain). As suggested above by the "stable set" approach, many different heuristics may be applied at this stage, mostly having some greedy characteristic in common but differing within the details. Since reasonable solutions could be obtained from these heuristics, there was no need to develop further ones, though there might be a chance to improve time consumption, or heuristics success (finding a negative reduced cost variable) by other algorithms.
Note, that similar to the heuristic before, many variations in the STRX costs (for the ordering) are possible as well. Possible costs are: a STRX's dual value, it's reduced costs allotment (induced interference of the STRX in the current assignment minus it's dual value), among other.

4.7 — *Problems with new Variables* —

This section is dedicated to a phenomenon concerning the inclusion of new variables. In the practical context of section 5.1, the procedure described above works as desired, up to the point, that new (theoretically) improving variables are generated and inserted into the LP. The problem that arises, is that these variables are not used (they stay equal to zero) in the next basis solution, in most cases. Recalling the pricing procedures background, the aim was, to produce variables, which improve the current LP solution. Therefore adding an improving variable should theoretically

imply, that this variable is used in the next basis solution (at least if only one variable is added). Nevertheless in the instances presented here, this is not the case. The reason for this can be found in the constraints of the main problem: The constraints are sharp in the sense, that it is impossible to exchange exactly one (or very few) basis variables. Each change in one variable needs to be compensated by many other changes on other variables. Resulting, having a rather small pool of variables available, it is difficult to assign a value above zero to new variables.
Since the amount of basis variables or their total sum is limited to $|F|$ (because of constraint (10), adding a variable (with value higher than zero) results in disappearance (or a decrease in the value) of another. In most cases, this change, in the sense of maintaining feasibility, cannot be compensated by the available set of variables.This can be emphasized by the following example (for the sake of simplicity, an integer example is regarded, but the same holds similar for fractional values). Note, that the example above with 3 STRX combined with it's frequency demand is to small here, so a new example is used. Assume, the following instance given:

$$\mathcal{S} = \{1,2,3,4\}, \qquad k_I = 1 \qquad \forall\, I \in \mathcal{S},$$
$$F = \{1,2\}.$$

A feasible basis solution could consist of two variables:

$$x_{\{1,2\}} = x_{\{3,4\}} = 1,$$
$$(x_T = 0 \qquad \forall\, T \subseteq \mathcal{S}, \{1,2\} \neq T \neq \{3,4\}),$$

Assuming $\{\{1,2\},\{3,4\}\}$ the only variables, which are available at this pricing round and that the new variable would be $x_{\{1,3\}}$. By incorporation of $x_{\{1,3\}}$ with a value of one, at least one of $x_{\{1,2\}}$ and $x_{\{2,3\}}$ needs to be set to zero to fulfill constraints (10) and resulting (9). Consequentially, without loss of generality: $x_{\{1,2\}}$ is chosen. So STRX 2 has not enough frequencies any longer, the change cannot be compensated by the other variable (setting both existing variables to zero and adding the variable $x_{\{2,4\}}$ could be a solution). So the new variable would not be used for the next basis solution. The reason is clear, forcing one variable to appear in the next basis, it might be necessary to add other variables for maintaining feasibility as well. These variables (in the following denoted as surrounding or auxiliary variables) need not necessarily have negative reduced costs as well and can therefore not be found by the procedure shown above, consisting of adding only improving variables in each pricing step. Resulting, adding an improving variable might force to add a (lesser) corrupting variable, as well.

In general, it is very hard to address this problem, little research is available on

that matters. In this thesis, the following workarounds have been tested:
As a very first approach, ignoring this behavior and just adding lots of "out-priced" variables may produce enough variables to be able to compensate most changes by further added variables, through pure mass. But there is no guarantee of success, as mentioned above, there might be the need of adding corrupting variables, as well).
As a second approach, during the later presented examples / computations, the DSATUR heuristic was used to generate a "surrounding" solution for an "out - priced" variable. In detail, the DSATUR heuristic was used under the assumption that a special variable has a value of one (and therefore reducing the demanded frequencies $k[I]$ of the inbound STRX) and then, the outpriced variable were added, as well as the ones produced by DSATUR. Comparing to the first approach, it has the advantage of creating corrupting variables as well. Obviously this approach has no claim for generating the best compensating / surrounding variables either, but among various implementations, a significant speed up in the solution process could be regarded. But, for every outpriced variable, $|F| - 1$ other variables have been added as well. Nevertheless, the quality of the standard DSATUR solution was increased, by assigning the new variable prior to running the DSATUR algorithm and respecting this during the DSATUR run (concerning the induced interference of this assignment). Since the algorithm is fast enough, all possible assignments to the new variable can be tested and the best solution over all possible prior assignments is chosen. Note, that another possible solution would be, to choose the frequency for the new set after the DSATUR algorithm is run, not beforehand. But since both solutions have approximative character, there is no forecast possible, which one will be better. The latter variant is faster, while the first usually yields better results in the test scenarios.
Referring to the one-opt step from above, this is only applied on top, when the DSATUR solution is not to far away from the current objective function (for all scenarios, a difference of plus/minus two at most seemed appropriate), since it is relatively slow. Resulting: If the chance of obtaining a new basis solution via DSATUR and one-opt is relatively high, it is used, but not in all cases. But since the solutions of DSATUR are always integer, the linear relaxations solution value generally differs by more than two, in the long run. Resulting, the (slow) one-opt heuristic is only used in the beginning. Note, that even though the DSATUR solution offered no new basis solution in nearly all cases, the added (potentially corrupting) other variables still helped to obtain new basis solutions. Hereby, this improvement has been paid off, by adding (many) surrounding variables on the one hand and on the other hand, these variables are not the best possible, which means that an improving variable may not be used to it's full potential (the improvement is less than indicated by the reduced cost). Both aspects slow the improvement process down severely: more variables

result in slower computations in each simplex step, each for less improvement as potentially possible.

Summing the last section up, the column generation algorithm combined with these heuristics leads to an improvement concerning the solution's behavior compared to the [FAPHSFH] model. In general, this means no optimal (because of the above mentioned problems), but a "relatively good" solution of the linear relaxation (a primal bound) could be produced. In more detail, some results of the first stage and the here presented column generation procedure are presented in the next section (5.1).

§5 Computational Results

In this section, a detailed view on the computational results of the above mentioned procedure for the first stage problem, on various instances, is given. Hereby statistics are produced and interpreted, in the following way: a collection of the scenarios of *fap.zib.de* is used with some modifications (because of the transition to slow frequency hopping, in detail explained in section 3.1 and 5.2). At first, the results of the "Siemens1" instance are commented on in detail, representatively. In the following section, results for more instances are given from a less detailed perspective. Hereby, the test environment was a processor with 4 cores ($4x\ Intel(R)\ Xeon(R)\ CPU\ W3540@2.93GHZ$) and 12 Gigabyte Ram, though the quell code was not especially parallelized.
As a first, general result, it has to be clarified, that the [FAPHCG] model could not be solved optimally in most scenarios. As presented in section 4.7, improving variables can hardy be used to their full computational potential. The amount of additional/-surrounding variables (slowing the simplex algorithm down, overcrowding memory) and the inefficient usage of the improvement potential of an outpriced variable hampered the solution process. On top of that, in cases, when the heuristics were not successful, improving variables had to be computed via an explicit solution of the pricing problem, whose bad computational structure (see section 4.4) even increased the time consumption. As a consequence, the tree factors

- Too many surrounding variables $\rightarrow$ Memory and simplex speed issues.
- Inefficient improvement per variable $\rightarrow$ Decreased improvement speed overall.
- Slow explicit pricing problem $\rightarrow$ Decreased improvement speed overall.

prevented an optimal solution. Nevertheless, the column generation approach was successful in obtaining an approximation on a dual bound (an approximation – primal bounds – of the linear relaxation). Progress could be made in or towards the linear relaxation (contrary to the first presented model [FAPHSFH]), but it could not be solved to optimality either (not to mention a branch and price / branch and bound framework).

Nevertheless, some statistics on the linear relaxation are presented in the following (pricing was applied, until no significant progress could be made with reasonable effort any longer). To complete the two staged approach, in the next chapter, the best integer solutions found are recorded, for obtaining a feasible frequency assignment (especially with respect to adjacency interference).

5.1 — *Statistics of a key example* —

Regarding the "siemens1" modification, the following section gives detailed statistics concerning the first stage problem, accessed with column generation (at least on the linear relaxation). Hereby, the solution process was sustained, until it ran out of memory, at about $50,000$ added variables after about $80,000$ seconds. At this point, the solution improvement was so slow (figure (16)), that solution probably was close enough to the optimal value (the solution value improved by about 0.0002 per pricing step). Further processment would have taken great effort for a rather low advance. Before commenting on the objective value in detail, the column generation process with respect to th created variables is analyzed.
The graphics ((9) and (10)) show the amount of created variables (with negative reduced costs) in relation to the time consumption and the pricing steps. Important is, that the number of totally added variables is (because of the creation of surrounding variables) significantly higher.
These graphics show a well known characteristic of column generation. The first variables can be created relatively easily, the heuristics find many solutions but as the time progresses, improving variables are harder and harder to acquire. The heuristics produce less useful solutions (especially in the long run, only one variable with negative reduced costs is found per pricing step). This is exactly the behavior, which is expected for a column generation / improvement process. At early stages, improvements can be made easily, but in later stages this becomes more difficult. Nevertheless, it is remarkable, that the process of "increasing hardness" is not unlimited. Even in the long run, the heuristics produce at least one result (variable) in most cases (pricing steps). Resulting, the explicit pricing problem solvings are equally distributed over the whole time frame (figure 13). In contrast to first expectations, there is no need to solve the pricing problem relatively more often, in the long run. Though this cannot be claimed with generality, since the instance was not solved to optimality. Theoretically, this (expected) behavior could still take place, in a wider time frame than examined here.
At the same time, regarding the "hardness" of the pricing LP over time, figure (14) suggests that the problem gets more difficult over time (here, with great variance, it's solution time has roughly doubled). So the explicit pricing problem's hardness rises over time, while the frequency of it's solvings stays constant.
This behavior of the added variables differs in amounts, but not in structure, if the surrounding variables are counted as well (see figure (11) and (12)). Note, that the surrounding solutions will add at least $|F| = 75$ variables per solution (variable with negative reduced costs) per pricing step. As a result, about $50,000$ variables have been created at the end, thus effectively overcrowding the memory and putting the

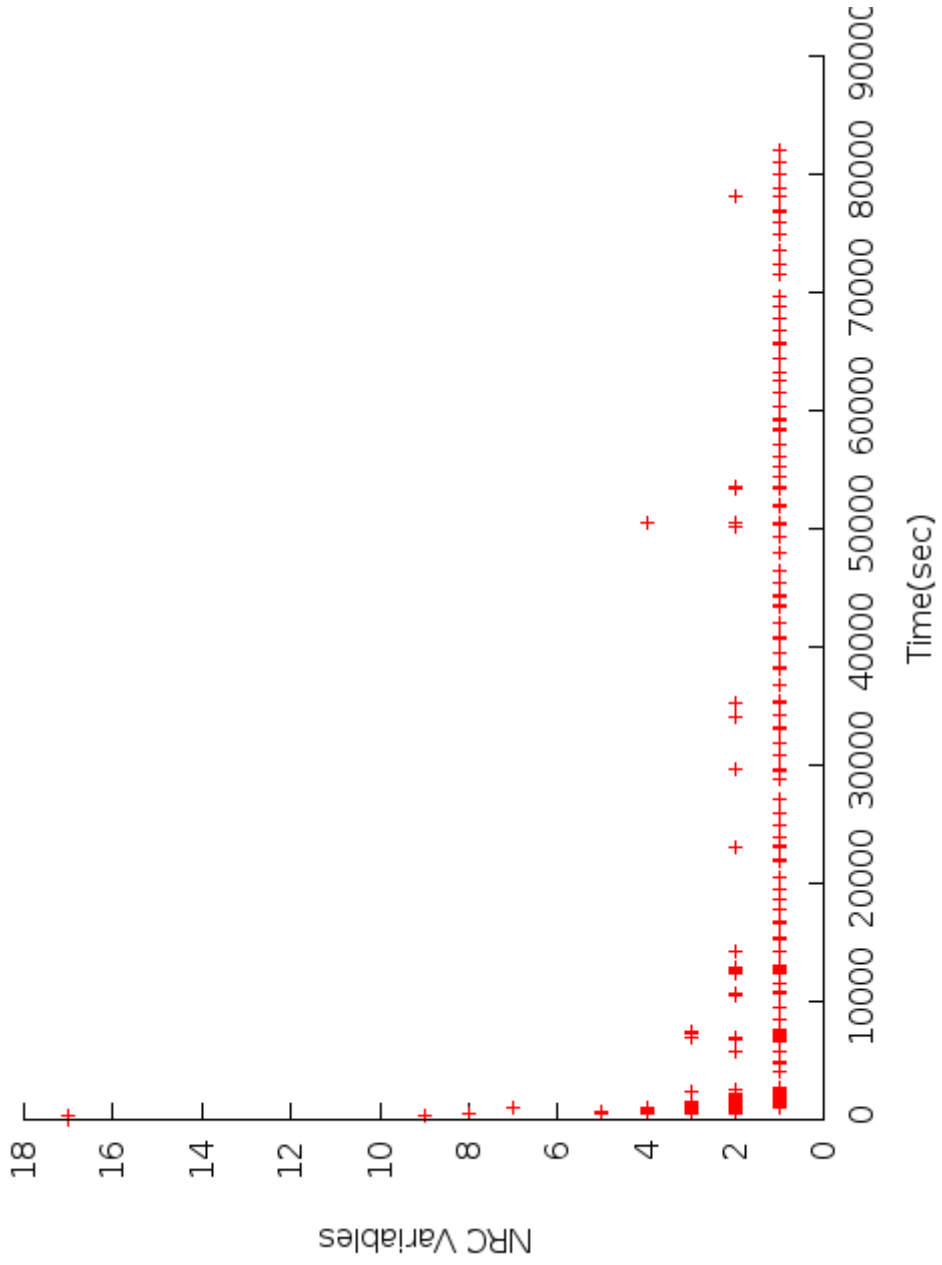

Figure 9: Negative reduced costs variables found over time

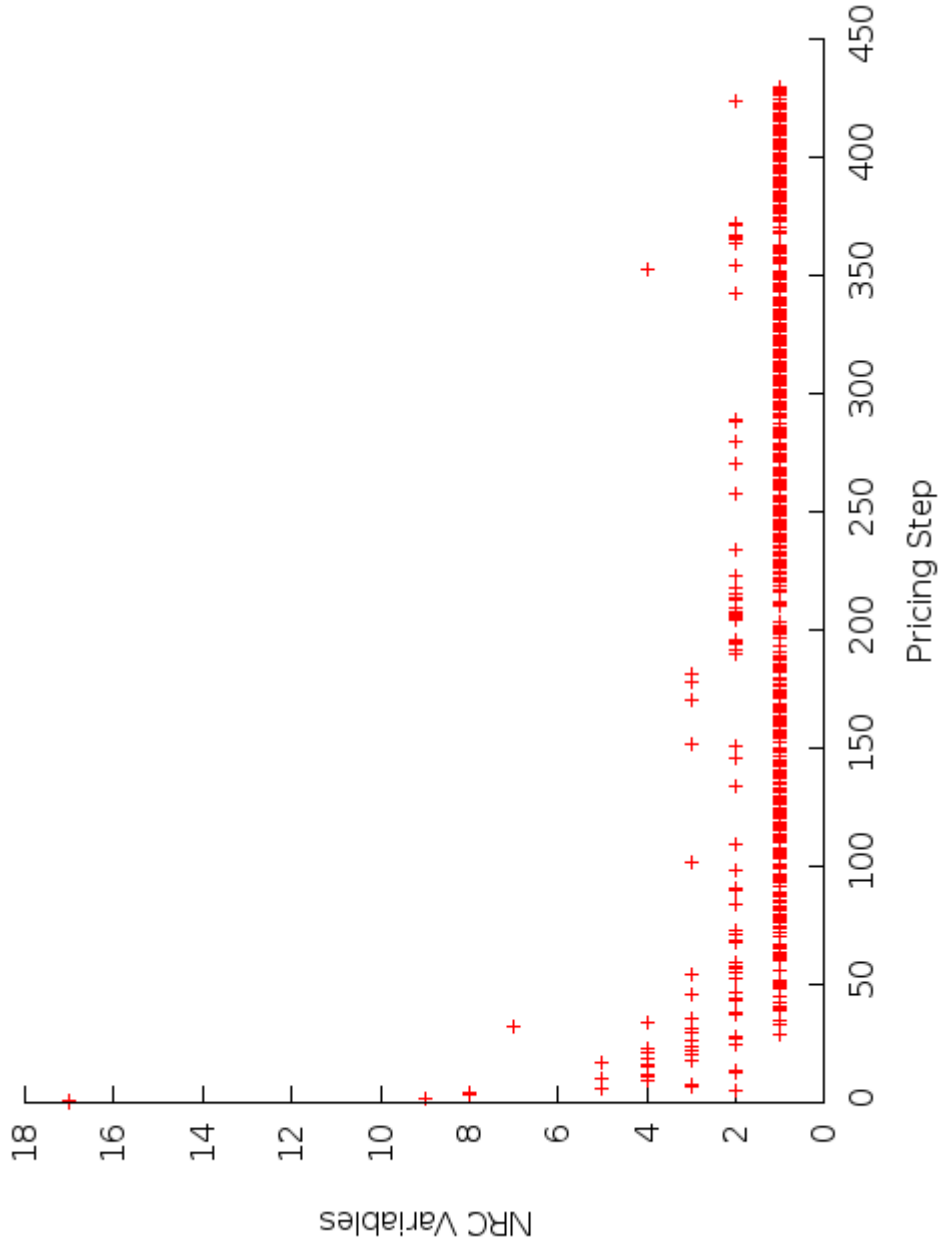

Figure 10: Negative reduced costs variables found per pricing step

solution process to an end (actually, scip 2.01 cannot delete variables, it can only fix them to a certain value, such that these memory restrictions take place). In addition, having the speed of the solution / success of the heuristics in mind, picture 15 shows, which heuristics found solutions when the pricing LP was not run. Hereby, if a heuristic produced an improving variable, the heuristic's number is plotted for this pricing step, or if it did not find an improving variable, zero was plotted for that pricing step (this holds for all heuristics, such that zero may be plotted multiple times during each pricing step). Note, that the scale of this graphic is somewhat compressed, not all heuristics are generating solutions at the same time (compare figure 10, in the long run, all heuristics produce only one solution in sum, per pricing step). Nevertheless, some observations can be made:

- The "original" heuristics and the ones demanding special sizes in the created sets (denoted with "... pieces") create variables in the same frequency, both variants are equally successful. The exception is the stable set heuristic. While the "pieces" variant was not successful at all, the original variant was successful during the whole time. This apparently is reasoned by the fact, that it is rather easy to create small stable sets (e.g. of size 2), but it gets increasingly harder (by greedy heuristics) to generate bigger ones.
- The "maximal increase" heuristics (especially the ones demanding a special size) fail to produce solutions on the long run.
- The "maximum reduce" heuristics are somewhat more successful, than the others (the density of their findings is higher than the ones from the others).

As a comparison, the behavior of the solution values over time or pricing step (figures 16 and 17) are outlined. At first glance, these values show exactly the typical behavior of column generation algorithms. While a fast decrease (in minimization problems) can be achieved at the beginning, the quality gain is diminishing in the log run.

Remarkably, the solution values behavior shows a high coincidence to the variable creations. The progress in quality decays by time. Apparently, the more improving variables are found, the greater progress in solution quality might be achieved, so the progress is faster in the beginning. A further interesting point can be regarded in the "long term" behavior of the solution and in the variable-addition behavior. After some time, the amount of variables added per pricing step is close to one. Nevertheless, the solution improvement decays even further. So on the one hand, the amount of variables added has an impact on the solutions improvement (as well as in quality as in speed) but on the other hand, the "ability to gain quality" is diminishing even

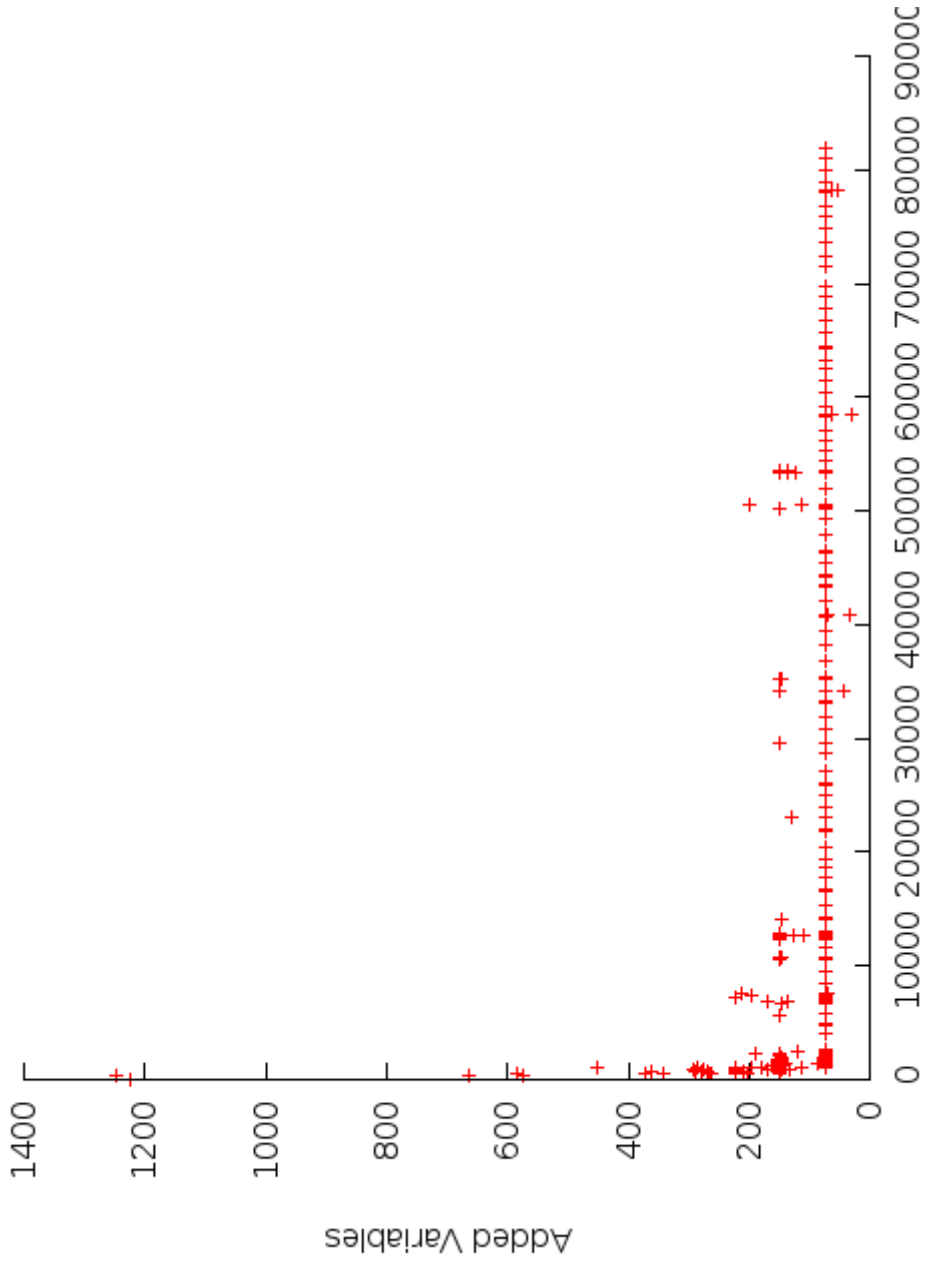

Figure 11: Totally added variables over time

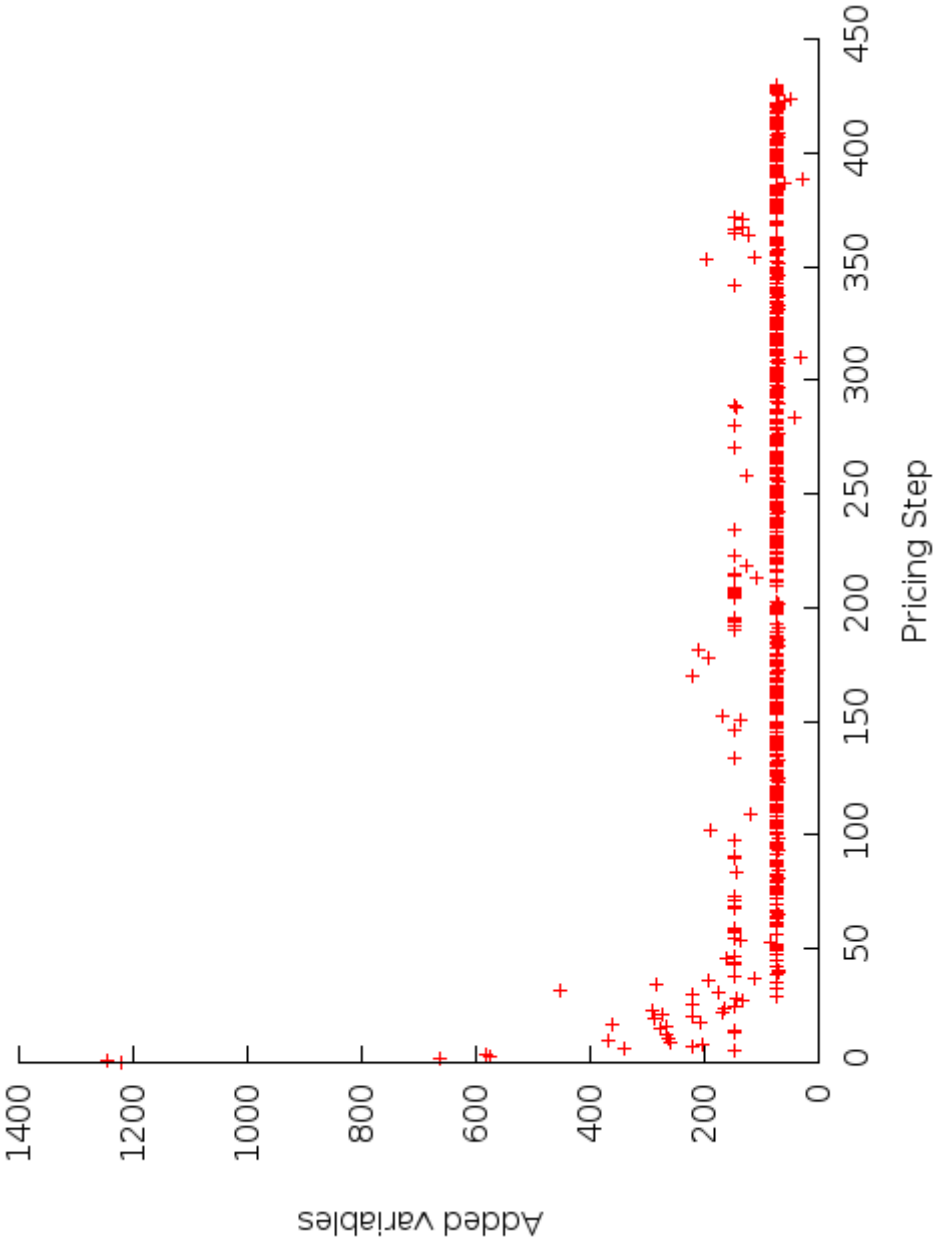

Figure 12: Totally added variables per pricing step

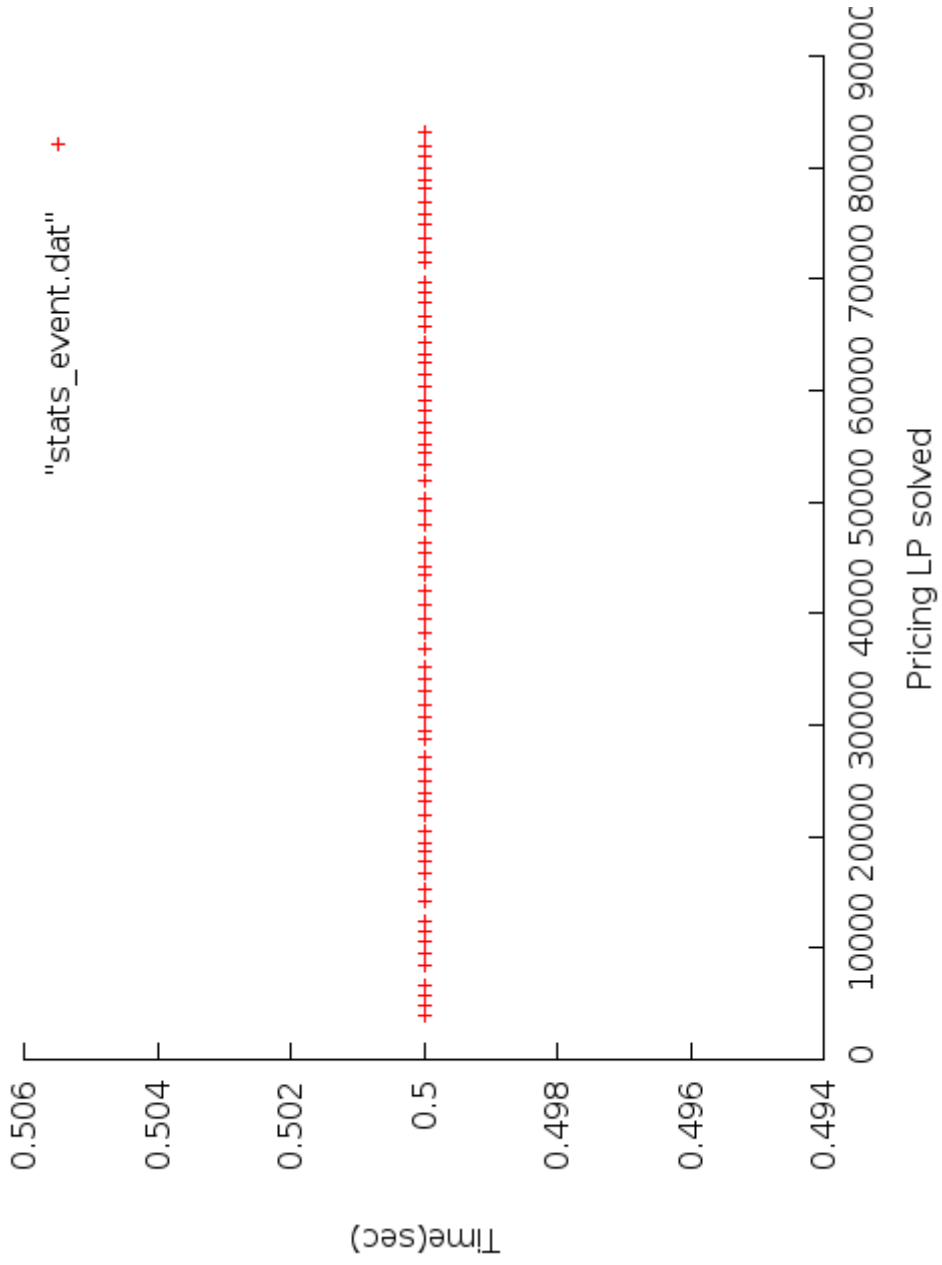

Figure 13: Pricing LP solves per time

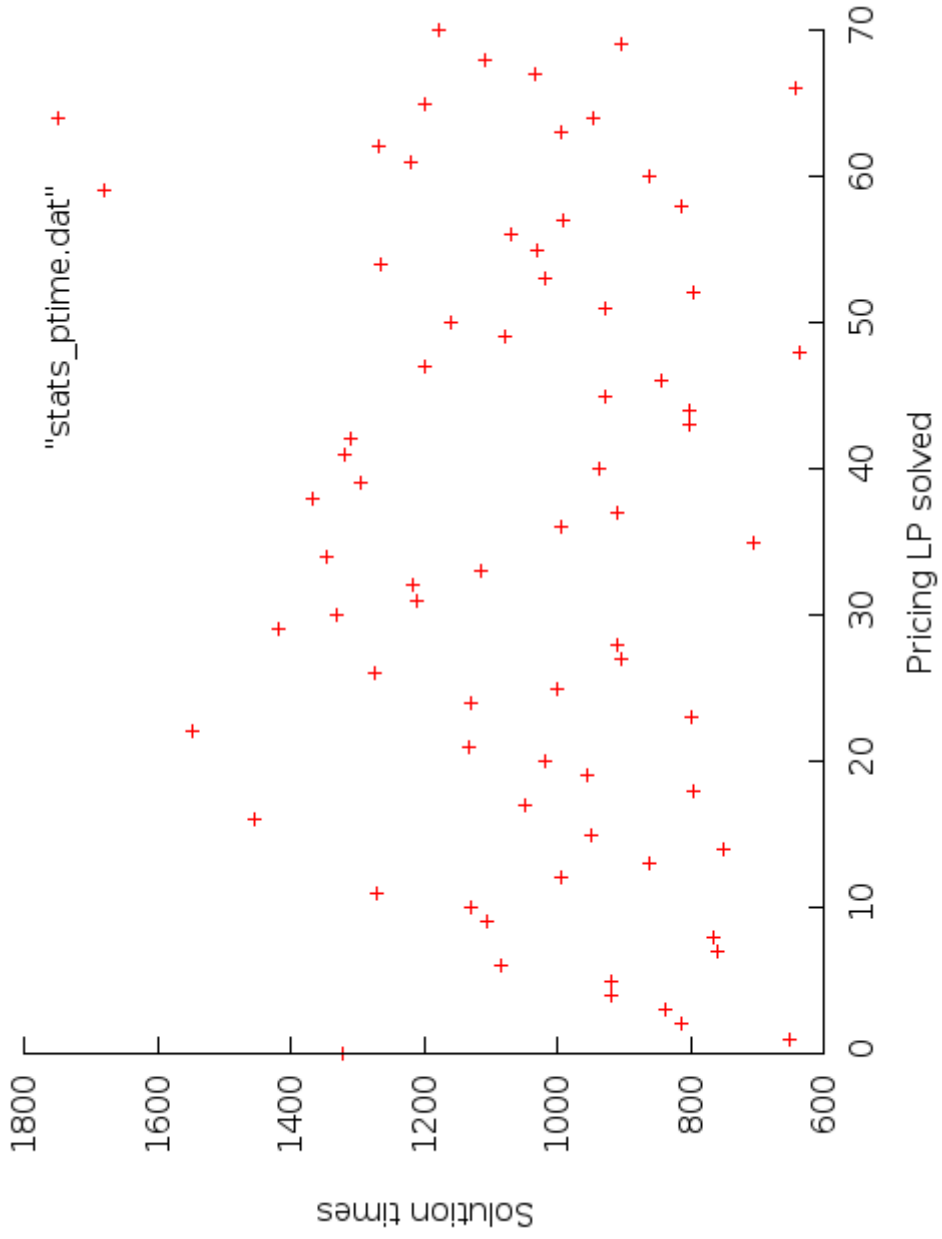

Figure 14: Time used per pricing LP solve

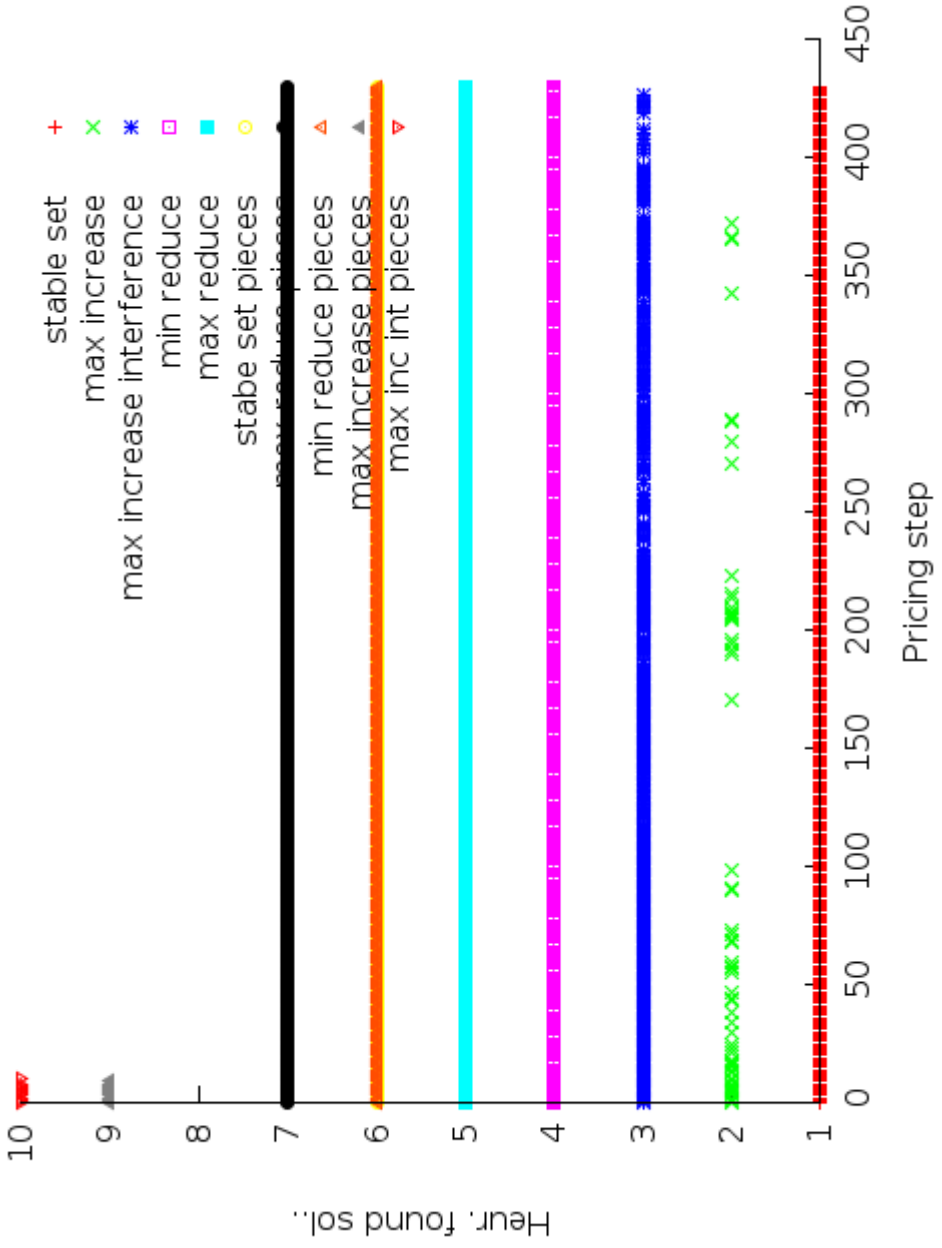

Figure 15: Heuristics effectivity

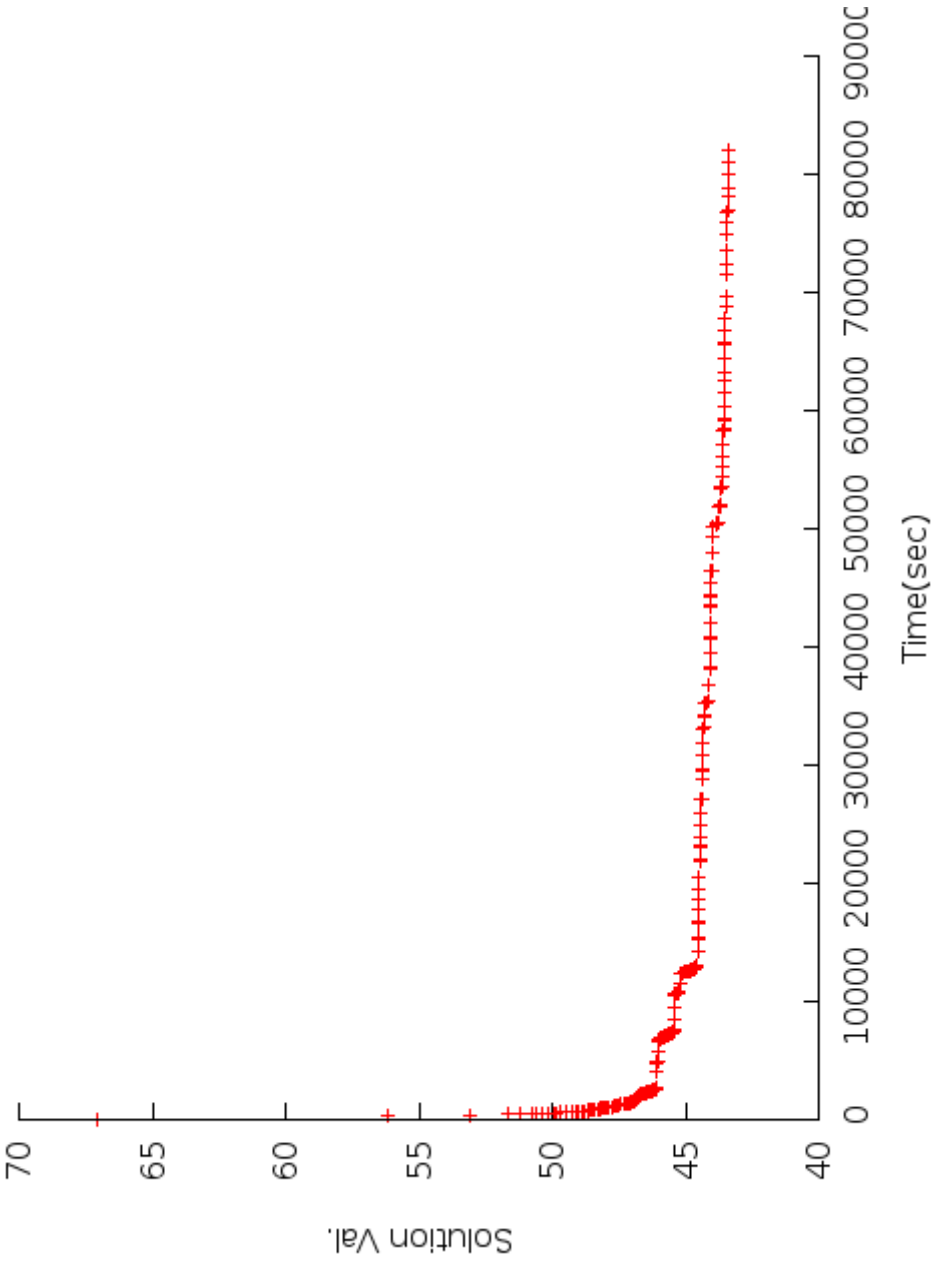

Figure 16: Solution value over time

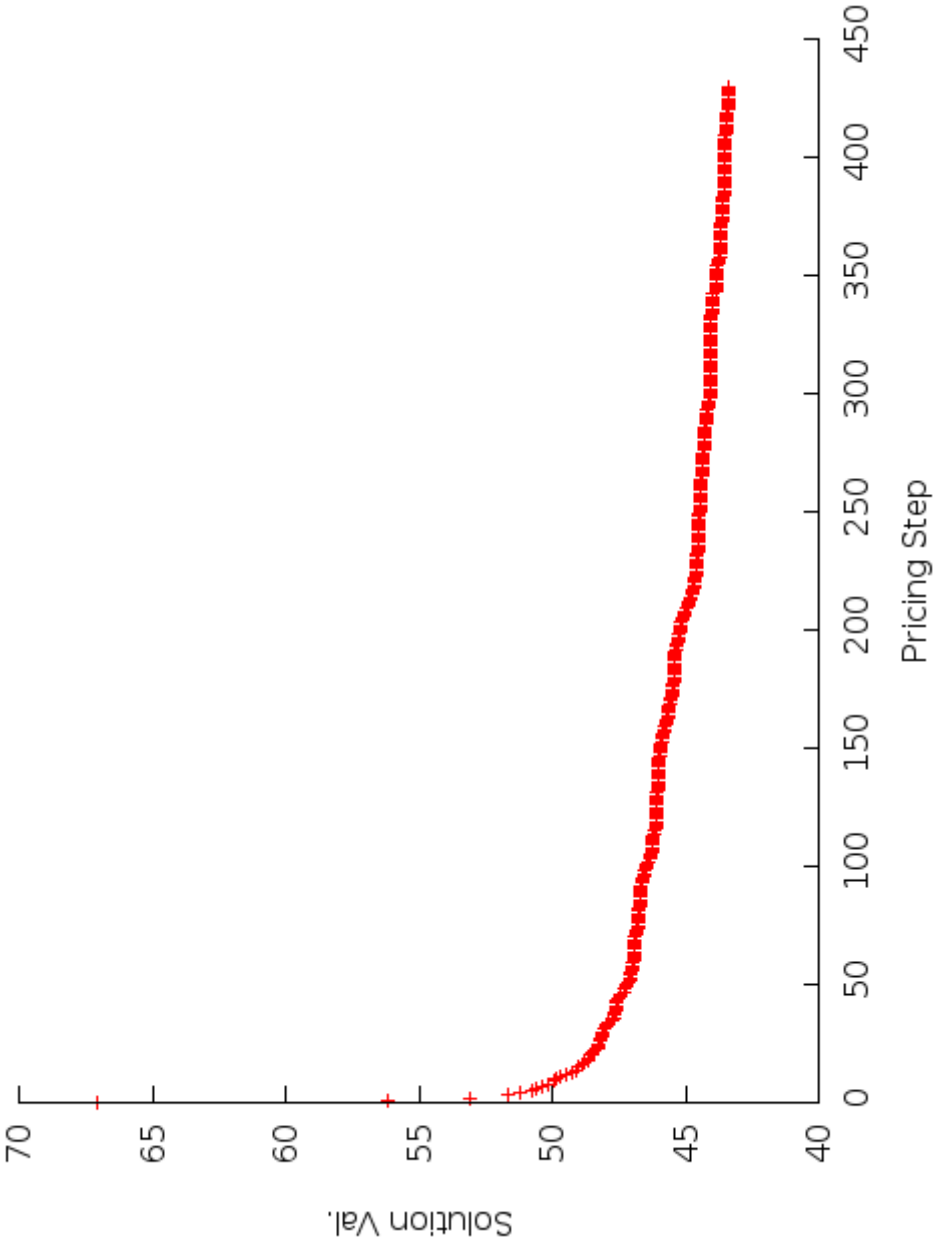

Figure 17: Solution value per pricing step

more than the "ability" to find improving variables.

Evaluating the column generation process, it is to say, that the time and memory consumption is still significant, which causes, that instead of an optimal solution, only a primal bound (on the linear relaxation or an approximation on a dual bound of the integer problem) could be acquired. This is mainly reasoned in the problems in embedding new variables. As shown above, it is possible to find improving variables, but these may only be used by adding many surrounding variables as well (which overcrowds the memory in the long run) and significantly increasing the amount of simplex calculations between the pricing steps. With this approach, progress can be made at the beginning, but the optimal solution is not reachable. Nevertheless, this behavior could be improved by better heuristics or a more targeting incorporation of new variables, in future research. A further improvement could be made, in a technical point of view. Since the used software (scip 2.0.1) does not allow to delete variables (it only allows to fix variables to zero), memory restrictions are rather striking. Advancement in the software (or in creating less and better improving variables and embedding them) could lead to an improvement here, so that the column generation approach does not fail in general, like the [FAPHSFH] model.

In the next section, more results, on other instances are given, though not as highly detailed as in the present section.

5.2 — *More general results* —

In this section, more examples like the above, but from a less detailed perspective, are shown. Hereby, the three instances "Koeln", "Bradford-0-eplus" and "Swisscom" are treated (and compared with the "Siemens1" scenario). All are taken from the "cost256" project from fap.zib.de, like the "siemens1" instance and as well adapted for slow frequency hopping. Furthermore, all the scenarios are solved for the given number of frequencies, as well as for 50% and 150% of the available number of frequencies.

For completeness sake, the above mentioned modification is explained in detail, here: every cell is interpreted as STRX, with the TRX demand (from that cell) interpreted as the number of inbound TRX. The demanded number of frequencies per STRX ($k[I]$) is set as the amount of inbound TRX plus four (more information on that parameter can be found in section 3.5). Notable is, that the most common setting for the inbound TRX is equal to one. But since the important influential factor for the

LP is is given by the demanded frequencies, which are not equal to one, generality is not decreased in this aspect. As mentioned in section 3.5, minor constraints like local blockings and separations are left out. Interference relations are modified as explained in section 3, equation 1. Hereby, the TRX interference values are the same than the ones of their corresponding site, the site's relations are multiplied for the inbound TRX. The available frequency spectrum stays the same, though it is transformed to begin with frequency number one.
It is to mention, that with these modifications, some characteristics of the test scenarios are diminishing. While the relatively high density of interference relations of the *Koeln* scenario is contained, the key characteristic of *Swisscom* (many local blockings) is omitted (in this case, combined with few interference relations, leading to a trivial scenario).

For completeness sake, an overview on the characteristics of these instances is given in table (1). Most of the time, no optimal solution could be acquired, so an uniform

Name	**STRX**	**Frequencies**	**50%**	**150%**
Siemens1	506	75	37	112
Bradford-0-Eplus	1886	75	37	112
Koeln	264	50	25	75
Swisscom	148	68	34	102

Tabelle 1: Overview Scenarios

computation time of 100,000 seconds was used for every scenario. Nevertheless, if a scenario's linear relaxation was solved optimally, this is denoted with (o) and if the computation was aborted before the 100,000 seconds were reached, this is denoted with (a). Note, that such a termination could be caused by generating to much variables or the pricing problem using too much memory on top of that. Resulting, the results presented in table 2 have been obtained (denoting k for 1000). Interesting to mention is, that the $Bradford$ instance with 75 frequencies could be solved optimally with $50k$ variables and a solution value of 3241.45 (in $250k$ seconds), while other instances ($Siemens1$ with less STRX but more frequencies) were aborted due to memory issues.

From this data, some general trends can be derived. At first, it is confirmed, that interference assignments become (drastically) better with more available frequen-

Name	Solution value / Generated Variables	50%	150%
Siemens1	44.0(a) 50k	767.20 30k	0.85 (a) 80k
Bradford-0-Eplus	3264.2 (o) 23k	11661.4 10k	1321.84 32k
Koeln	882.75 10k	2707.7 6k	384.92 12k
Swisscom	0 (o) 90	11.48 (o) 4k	0 (o) 90

Tabelle 2: Overview on First Stage Results

cies. Further, it seems, that with increasing available frequencies, the problem gets easier accessible. In the same period of time, more improving variables could be created (this means more successful heuristics or less solvings of the explicit pricing problem demanded), throughout all instances (with the exception of *Swisscom*). On the other hand, since no relation to an optimal solution can be shown, it is not proven that more variables correspond to a better approximation of the optimum. Hereby, having in mind that the number of feasible results is directly dependent on the number of available frequencies (e.g. more STRX combinations are allowed, for example the ones forming smaller subsets), such that the solution space increases with increasing frequencies. Informally spoken, the problem size increases, while the problems difficulty decreases.
Interference values obviously depend on the special interference data, but in generally (as the contrast between *Siemens*1 and *Bradford* shows), more STRX (and the same amount of available frequencies, same "overall level" of interference) lead to an increasing interference value. On the other hand, comparing *Siemens*1 to *Koeln*, the above trend is not true, if the interference levels differ (overall the *Koeln* scenario has much more interference relations). In general, this was to be expected. More STRX within the same area (higher density, higher average interference per STRX pair) will induce more interference than the same STRX on a wider area (lower overall interference level). As a result, the interference amount is not directly dependent on the number of STRX, but rather on the inbound interference relations.
A further remarkable difference between these scenarios can be observed at the num-

ber of added variables. Though a uniform computation time was chosen, these differ a lot. This is reasoned in the pricing problem. While the heuristics are equally successful over all scenarios (the relation of the amount of pricing steps, in which it was necessary to solve the explicit pricing problem to the amount of steps where the heuristics were successful, is the same), the difference lies in the pricing problem solutions them self. Because of the pricing problem's structure (penalty variables), the explicit solution (until a solution with negative objective value is found) may take considerable amounts of time. These time amounts are not directly dependent on STRX or frequency amounts and are therefore not comparable between different scenarios. As a result, these differences in created variables are rather differences in the made pricing steps, which are mostly determined by the time consumption of the explicit pricing problem solves.
Since the above instances could not be solved optimally (integer), the best recognized integer solutions have been noted in table 3 (these are further processed in the second stage, section 6.2). Note, that all of these solutions have been found relatively early in the solution process. This is remarkable. Contrary to the solution of the linear relaxation, the integer solutions(heuristically produced) are not improved on the long run during the column generation process. All of these solutions have been

Name	**Solution value** / **Time**	**50%**	**150%**
Siemens1	58.81 384	830.93 400	4.27 188
Bradford-0-Eplus	3333.79 14*k*	11706 44*k*	1390.66 22*k*
Koeln	945.20 135	2754.7 134	432.50 102
Swisscom	0 0	23.75 42	0 0

Tabelle 3: Best Integer Solutions

obtained by the DSATUR heuristic, combined with the one opt step, so they have no claim for optimality (for the integer problem), in general. Recall, that integer solutions are produced in each pricing step, when surrounding variables are created (compare section 4.7).

Since the linear relaxation of the first stage was not solved, the values of table 2 cannot serve as dual bounds for the integer problem, either (though they are a feasible primal bound for the linear relaxation and are most likely to apply as dual bound as well). Nevertheless, these solutions cannot be improved trivially, such that they are sufficient for an exemplary calculation in the second stage optimization (adjacent channel interference). Concerning the solution value of the integer solutions, it is to say, that they clearly follow the trend of the solution of table 2. In a nutshell, more frequencies lead to better solutions and the STRX amount is not as important as the underlying (corresponding) interference relations.

All in all, the here presented results offer some useful insights into frequency planning, though they have some major drawbacks - the most important, that no optimal solution (at least of the linear relaxation) could be obtained. Other drawbacks are, that the approximation of the linear relaxation does not produce formal dual bounds (for the integer problem), though they are most likely to apply (given a long enough runtime). On the other hand, heuristical integer solutions are produced, which can be further processed towards adjacent channel optimization.
Furthermore, there is still improvement potential in this approach. Better heuristics (e.g y exploiting parallel computing) or another concept of incorporating improving variables, combined with an increased runtime may lead to an optimal solution, at least of the linear relaxation.
Concluding this chapter, some thoughts on a possible evaluation of the above results by feasible dual bounds (and how to obtain them) are given in the sequent section.

5.3 — *Quality Estimations: Lower Bounds* —

Having the above presented results in mind, the question of their quality appears, especially on the background of their approximative character. Since feasible approximations to the linear relaxations have been found, their objective values may serve as primal, or upper bounds to this relaxation. Nevertheless, while an optimal solution is not available, the only evaluation possible, is by the means of lower bounds on the optimal solution. In the following, two potential ways of obtaining feasible, non trivial lower bounds (for the linear relaxation) are presented. Note, that these lower bounds apply for the non relaxated problem as well, though they are not as tight. For completeness sake, it is repeated, that a trivial lower bound is zero (since an assignment can induce not less than zero interference at all).

As a first possibility of obtaining a dual (lower) bound, the mean of a lagrangian

relaxation is given. Though this is condensely presented in the following, it is referred to [JF10] for more information. Given a linear program (row wise defined, I denotes the set of all rows),

$$\begin{aligned} z := min \quad & c^t \cdot x \\ s.t. \quad & A_i \cdot x \geq b_i \quad \forall\, i \in I, \\ & x \in X. \end{aligned}$$

a lagrangian relaxation (lagrangian dual) is obtained by "dualizing" a subset $J \subseteq I$ of the constraint into the objective function by

$$\begin{aligned} z(\lambda) := min: \quad & c^t \cdot x + \sum_{j \in J} \lambda_j \cdot \left(b_j - A_j \cdot x\right) \\ s.t. \quad & A_i \cdot x \geq b_i \quad \forall\, i \in I \setminus J, \\ & x \in X. \end{aligned}$$

whereas $\lambda \in \mathbb{R}^{|J|},\ \lambda \geq 0$ holds component wise. In the following, this problem is referred to as $LR(\lambda)$ (with Lagrange multipliers λ). While from a computational point of view, this concept allows to get rid of "difficult" constraints (by moving them into the objective function and paying a penalty if not fulfilling these constraint), it holds that $z \geq LR(\lambda)$, thus $LR(\lambda)$ is a formal relaxation of the above LP. Resulting a solution of $LR(\lambda)$ would yield a feasible lower bound.
Now, the Lagrangian relaxation is applied to the model in section 4.2. Hereby, in every pricing step, choose $\lambda = \pi^\star$, as the vector of the dual solution values of the current basis solution, and dualize all constraints. As a result, in every pricing step a dual bound can be computed via the following problem. Hereby, denote I for the set of constraints and $\mathcal{S}$ as set of STRX, $T \subseteq \mathcal{S}$ for a set of STRX (corresponds to a variable) and $a(i, T)$ as the induced constraint matrix. Note, that by using the (equivalent) "$\geq$" constraints in (9) and (10), the dual variables are always equal or bigger than zero. Then, a dual bound on the optimum of the linear relaxation z is computed via

$$\begin{aligned} z^{\geq} z(\pi^\star) &= min \left(\sum_{T \subseteq \mathcal{S}} co(T) \cdot x_T + \sum_{i \in I} \pi_i^\star \left(k[i] - \sum_{T \subseteq \mathcal{S}} a_{i,T} \cdot x_T \right) \right) \\ &= min \left(\sum_{i \in I} \pi_i^\star \cdot k[i] + \sum_{T \subseteq \mathcal{S}} co(T) \cdot x_T - \sum_{i \in I} \sum_{T \subseteq \mathcal{S}} \pi_i^\star \cdot a_{i,T} \cdot x_T \right) \\ &= min \left(\sum_{i \in I} \pi_i^\star \cdot k[i] + \sum_{T \subseteq \mathcal{S}} co(T) \cdot x_T - \sum_{T \subseteq \mathcal{S}} \sum_{i \in I} \pi_i^\star \cdot a_{i,T} \cdot x_T \right) \end{aligned}$$

$$
\begin{aligned}
&= min \left(\sum_{i \in I} \pi_i^\star \cdot k[i] + \sum_{T \subseteq \mathcal{S}} \left(co(T) - \sum_{i \in I} \pi_i^\star \cdot a_{i,T} \right) \cdot x_T \right) \\
&= \sum_{i \in I} \pi_i^\star \cdot k[i] + \; min \left(\sum_{T \subseteq \mathcal{S}} (co(T) - \pi_i^\star \cdot \chi_T) \cdot x_T \right) \\
&=: \underbrace{\sum_{i \in I} \pi_i^\star \cdot k[i]}_{obj(\pi^\star)} + min \quad \sum_{T \subseteq \mathcal{S}} red(T) \cdot x_T.
\end{aligned}
$$

Hereby, the only constraints for the minimization problem are the (from the linear relaxation inherited) $x_T \in [0,1]$ conditions for every set T. In this context, $red(T)$ (for every variable T) denotes the reduced costs, appearing in the column generation process as well and $obj(\pi^\star)$ denotes the current LP solution. Note, that this dual bound is computed via an optimization problem again. Though the solution is easy to see (current objective value minus all negative reduced costs) it is hard to compute (because of the exponentially many variables).

Another mean of calculating a dual bound is presented in [LD05]. Since the LP bears the information, how many variables are used at most, in every solution (constraint (10): at most $|F|$ frequencies / variables), a dual bound (in every pricing step) is given by

$$
\begin{aligned}
\underline{z} &:= \sum_{i \in I} \pi_i^\star \cdot k[i] + |F| \cdot min \left\{ \{ red(T) | T \subseteq \mathcal{S} \}, 0 \right\} \\
&= obj(\pi^\star) + |F| \cdot min \left\{ \{ red(T) | T \subseteq \mathcal{S} \}, 0 \right\}.
\end{aligned}
$$

This holds, since the reduced costs show, how much the objective function changes per unit of this variable. In contrast to the calculations above, the evaluation of all reduced costs is exchanged with the problem of finding the lowest reduced costs, which is exactly the pricing problem. Since a column generation approach is invoked here, the latter formulation seems preferable. Nevertheless, during column generation, it is not necessary to solve the pricing problem optimally in most cases, such that the above data may not be available.
Both concepts have the same basis in common. A dual bound is computed by the current LP solution, decreased by some reduced costs. As a consequence, these concepts do not produce sensible results / bounds from the start onwards. Reasons are given in the following: Since a dual bound strictly above zero is desired, $||F| \cdot red(T)|$ needs to be smaller than the current solution value. As a result (and having some numerical results in mind, e.g. $obj(\pi^\star) = 45$, $|F| = 75$ in the "siemens1" example) the (minimal) reduced costs need to be close to zero. This will probably not hold

true at the beginning, but at the end of the pricing process, since this process converges to the optimum, the reduced costs must rise to zero at some point. Resulting, these dual bounds should only be computed towards the end (delayed) of a column generation approach.

Summing up, both possibilities show some theoretical ways of calculating feasible dual bounds. Unfortunately, both turned out to be inappropriate for the here presented solution approach. Since this is obvious for the lagrangian dual (exponentially many calculations necessary), the reasons for the second option are presented below. As the above calculations relied on heuristics rather than on an explicit solution of the pricing problem, the calculation of the optimal solution (respective the calculation of all negative reduced costs) was too expensive (in it's time consumption) to be done at runtime in each pricing step. Consequential, these bounds were only computed, when the column generation approach was aborted (with the final solution) and not in each pricing step. A straight forward implementation showed the same drawbacks (because of penalty variables) as presented in section 3.3, such that no non trivial dual bounds could be presented here. This data is supported by the "Siemens1" example. After the above presented $100,000$ seconds runtime of the first stage problem, the explicit pricing problem was run. While it could not be solved optimally by straight forward implementation because of memory restrictions, a dual bound proved, that the minimum laid below -7. This is nowhere as near to zero (see above) as it should be for a sensible dual bound. Having this in mind, this concept for obtaining dual bounds may only be helpful at later stages of the approximation, which can't be given here, because of the reasons presented in section 6.3.
Nevertheless, non trivial dual bounds may be obtained by this procedure, if a better approximation of the linear relaxation (more advance in the column generation procedure) is available, e.g. in a wider time frame, or in future, when the problem with the embedding of new variables is resolved.

In the next section, the further processment of an (integer) first stage solution towards adjacent channel interference (the second stage problem) is shown.

§ 6 The second Stage

Referring to section 3.5, the FAPH problem was split into a two staged approach. While in the sections before a column generation approach was analyzed for the first stage, this section is dedicated to the second stage problem. Therefore, an integer solution of the [FAPHCG] model is needed and should be processed, towards adjacent channel interference. Since the [FAPHCG] model could not be solved optimally, obtaining (good) integer solutions is not trivial. The following section shows, that it is impossible, to enforce integrality straight forward, on a solution of the linear relaxation of [FAPHCG].

6.1 — *Enforcing Integrality* —

While the column generation approach was helpful in steadily improving the objective value towards the optimum of the linear relaxation, but could not solve the problem optimally, the solution itself could be processed further for creating feasible (in the sense of integer) frequency plans (as input for the second stage). It seems reasonable to generate a frequency plan out of that approximation, even when the (co-channel) interference guarantees or bounds of the first optimization step can not be kept by that processing.
A naive way of enforcing integrality would be, to solve the problem again, on the subset of variables, that is different to zero in the above mentioned solution and additionally enforce integrality (a branching routine restricted to that subset of variables). Hereby having in mind, that the number of variables is low enough, such that a branch and bound procedure may yield a solution (repeating, that a complete branch and bound process is not possible here). Unfortunately, this does not work, since solvability by fractional variables does not imply solvability by integer variables. So the problem of solving (branch and bound) "on the fractional solution" for an integer solution is not feasible. This can be made clear by the following example. Given

$$\mathcal{S} = \{1, \cdots 8\}, \qquad k_I = 1 \quad \forall\, I \in \mathcal{S},$$
$$F = \{1, 2, 3\},$$

a solution might look like:

$$x_{\{1,3,4\}} = x_{\{2,3,4\}} = x_{\{1,2\}} = \frac{1}{2}$$
$$x_{\{5,7,8\}} = x_{\{6,7,8\}} = x_{\{5,6\}} = \frac{1}{2}.$$

Hereby, all other variables are not available in that pricing step. On these six variables, an integer solution is clearly impossible. For example choosing one of the first three variables (which is necessary in every solution) yields to the usage of a second one of that group, since at least one STRX of $\{3,4\}$ is not covered. The same argumentation applies to the second group, from which two variables need to appear in every integer solution, either. Combining both arguments, an integer solution would need at least 4 frequencies, when operating on that variables, which is infeasible. So a feasible solution with fractional values cannot be transfered to a feasible solution with integer values in general. Especially, this means, that the induced solution polyeder (derived from the variables with positive solution value, which is iteratively adapted during column generation) needs not to contain any integer points inbound.

Currently, there is no way known, to transfer the fractional solution of the first stage problem into the second stage problem, without loosing the quality guarantees. So, in the following stage, the best known integer solution (heuristically produced during the column generation process) is used, gaining at least some quality level (determined interference amount) out of the first stage.

6.2 — *Adjacency Optimization* —

Given a (feasible) integer solution of the FAPHRES problem ([FAPHCG] formulation), the amount of co-channel interference is fixed. On the other hand, this solution does not uniquely determine, which specific channels are assigned to which STRX, since this is not clear without ambiguity. As a result, a solution contains no explicit frequency assignment, but can be processed towards adjacent channel interference, obtaining a well defined frequency assignment. Formally spoken, the second stage problem (FAPHRES_A) could be defined as: Given a set of sets of STRX, each with a demanded number of frequencies (x_T values), the adjacent channel interference relations and the available frequency spectrum. How can the frequencies be uniquely (one frequency to exactly one set) assigned to these sets, such that the induced adjacent channel interference is minimal. Note, that a mathematical formulation is given in section 6.2.1.
This processment can be expressed as a linear program, as well. The solution of [FAPHCG] consists of |F| sets of STRX T_i, sharing the same frequency. Hereby,

$$T_i \in \mathcal{S} \quad \forall\, i = 1, \cdots, |F|,$$
$$\overline{T} := \bigcup_{i=1}^{F} T_i, \quad \overline{T} \subseteq \mathcal{P}(\mathcal{S}).$$

These sets / variables have a solution value, denoted with $\overline{x_{T_i}} \in \mathbb{Z}$, $x_{T_i} > 0$. In the following, these sets get frequencies assigned, inducing the lowest total adjacent channel interference value. This can be solved by the following LP, [FAPHAC], using the notation of the sections above:

There are three types of variables, namely

$$\begin{aligned} &x_{T,f} \in \{0,1\} && \forall\ T \in \overline{T}, f \in F, \\ &n_{T_1,T_2,f} \in \{0,1,2\} && \forall\ T_1, T_2 \in \overline{T}, f \in F, \\ &y_{T_1,T_2} \in \mathbb{Z} && \forall\ T_1, T_2 \in \overline{T}. \end{aligned}$$

The first variable $x_{T,f}$ describes, which frequency is assigned to a certain set and the second ($n_{I,J,f}$) indicates, how many adjacent channels f in I has in J. The third variable, y_{T_1,T_2}, denotes how many adjacent channels from I are in J (zero, one or two). This leads to the objective function

$$min: \quad \sum_{T_1 \in \overline{T}} \sum_{T_2 \in \overline{T}} ad(T_1, T_2) \cdot y_{T_1,T_2}.$$

Hereby, for a pair of sets T_1 and T_2, the number $ad(T_1, T_2)$ denotes the estimation of induced adjacent channel interference between the STRX in T_1 and the ones in T_2, whereas

$$ad(T_1, T_2) := \sum_{I \in T_1} \sum_{J \in T_2} \overline{ad(I, J)}.$$

Note, that this estimated value is additive, with respect to the number of adjacent channels, such that the total interference can be obtained by the number of adjacent channels multiplied with a interference amount per neighboring channel. So via equation (4), with the numbers $n_{I,J}$ denoting the amount of neighboring channels from I in J it holds, that the interference measure is equivalent to the one in section 3.2. Note, that $n_{I,J} = n_{K,L} \forall\ I, K \in T_1,\ J, L \in T_2$ and that this number is the same for every pair of STRX in the current sets, namely $n_{I,J} = y_{T_1,T_2}$. It follows, that the interference of an assignment is measure equivalently in both models:

$$\begin{aligned} \sum_{T_1 \in \overline{T}} \sum_{T_2 \in \overline{T}} ad(T_1, T_2) \cdot y_{T_1,T_2} &= \sum_{T_1 \in \overline{T}} \sum_{T_2 \in \overline{T}} \sum_{I \in T_1} \sum_{J \in T_2} \overline{ad(I, J)} \cdot y_{T_1,T_2} \\ &= \sum_{T_1 \in \overline{T}} \sum_{T_2 \in \overline{T}} \sum_{I \in T_1} \sum_{J \in T_2} \sum_{v \in I} \sum_{w \in J} \frac{n_{I,J}}{k_I \cdot k_J} \cdot ad(v, w). \end{aligned}$$

For completeness sake, the constraints can be written as

$$\sum_{f \in F} x_{T,f} = \overline{x_T} \qquad \forall\, T \in \overline{T},$$

$$n_{T_1,T_2,f} \geq 2\left(x_{T_1,f} - 1\right) + x_{T_2,f-1} + x_{T_2,f+1} \qquad \forall\, T_1, T_2 \in \overline{T}, f \in F,$$

$$\sum_{f \in F} n_{T_1,T_2,f} = y_{T_1,T_2} \qquad \forall\, T_1, T_2 \in \overline{T},$$

$$\sum_{T \in \overline{T}} x_{T,f} = 1 \qquad \forall\, f \in F.$$

That notation is similar, to that ones in section 3.2. Hereby, the first constraint determines the demanded frequencies for every, the second recognizes if two sets use neighboring frequencies, the third counts the number of neighboring frequencies and the last constraint preserves the settings from the FAPHRES problem, or from the [FAPHCG] model, equation (10), namely using different frequencies for every set (not reusing single frequencies). Note, that the value of $\overline{x_T}$ in the first constraint may be integer, though in most cases, it will be equal to one.
Remarkable, the structure of this problem is very similar to the FAPH problem (the [FAPHSFH] formulation). This is related to the formulation with penalty variables (y) and results in the same advantages and disadvantages (symmetry on the one side, bad linear relaxation behavior on the other). Nevertheless, since this problem is significantly smaller (variables and constraints in the order of $|F|^3$), it is much more likely to be solvable (analysis concerning the "hardness" of the problem can be found below, in section 6.2.1).

At the end of this section, some details on additional constraints (separations and blockings) ignored in the FAPHRES problem, are given. As mentioned in section 3.5, these constraints are omitted in general, but in the following, it is explained how they could be added, at least to a small extend. Since these constraints need the concrete frequency assignment, which was not explicitly given in [FAPHCG], they need to be treated in the second stage. The first, inner $STRX$ separation (for STRX I) can be addressed easily with

$$x_{T_1,f_1} + x_{T_2,f_2} \leq 1 \qquad \forall\, T_1, T_2 \in \overline{T}, I \in T_1, I \in T_3, \quad f_1, f_2 \in F \text{ and } |f_1 - f_2| \leq d_I.$$

Outer STRX separation has to be taken care of at the first stage problem, as well. Since

$$d_{I,J} > 0 \quad \Rightarrow \quad \forall\, T \in \overline{T}: \quad I \in T \; xor \; J \in T,$$

this can be expressed as: For all STRX I and STRX J, it holds

$$x_{T_1,f_1} + x_{T_2,f_2} \leq 1 \qquad \forall\, T_1, T_2 \in \overline{T}, I \in T_1, J \in T_2, \quad f_1, f_2 \in F \text{ and } |f_1 - f_2| \leq d_{I,J}.$$

The last constraint (blocking conditions) can be expressed similar. For all STRX I:

$$x_{T,f} = 0 \qquad \forall\, T \in \overline{T}, I \in T, \quad f \in F \text{ and } f \in B_I.$$

This is the only possibility to respect these (additional) conditions in the two staged model. But these constraints have a significant drawback, namely they are to striking. Generally spoken, a condition on $x_{T,f}$ does not only appeal to a STRX $I \in T$, but to all STRX in T. This has not only the effect of imposing conditions on STRX, which did not have them originally (as was done by generalizing FAP to FAPH, too), but may significantly harm feasibility, too. Further, these conditions are heavily influenced by the solution obtained from [FAPHCG]. Different sets induce different conditions in the following problem, therefore guarantees about feasibility can not be made and are dependent on the special solution of [FAPHCG]. One result may stay feasible, while the other does not. At this point, the focus swifts from the two staged approach, primary minimizing co-channel interference, to an approach, where the focus lies on the secondary constraints. As a result, only the most important constraints should be represented in the LP formulations or they should entirely left out, if possible (and done in the following work).

6.2.1 Problem Hardness

Before some results of the FAPHRES_A problem are presented, some difficulty analysis of the second stage problem is given. Having in mind, that this problem has additional character, being on top of the column generation approach for the first stage problem, it needs to be solvable in practical instances with sensible effort. Nevertheless, it is shown in the following, that this problem has a very high theoretical difficulty (it is NP complete, see section 8.1 for an introduction on that matters), as well as the last (FAPHRES problem with [FAPHCG] formulation). Hereby, the additional constraint mentioned above are left out, due to the above argumentation.
While a spoken definition of FAPHRES_A was given in the last section, this problem can formally be defined as follows: Let the input be given as a five-tuple

$$R := \left(\overline{T}, E, F, x_T, ad\right).$$

Hereby, $\left(\overline{T}, E\right)$ is the graph, induced by the results (sets) of the first stage as nodes and edges E between the nodes, with the induced adjacent-channel interference as weights between these sets (ad). Further, $|F|$ denotes the available frequency spectrum, still, and x_T denotes, how many frequencies every node gets assigned.
A feasible solution / assignment to that input is a function

$$y_4 : \overline{T} \to 2^{|F|},$$

mapping each set to a subset of frequencies (uniquely to one or more frequencies), such that

$$\begin{aligned} |y_4(T)| &= x_T \qquad \forall\ T \in \overline{T}, \\ y_4(T_1) \cap y_4(T_2) &= \emptyset. \end{aligned}$$

Then, the FAPHRES_A problem is defined by: Given the above input, the problem

$$\min_{y_3} : \quad \sum_{\substack{(T_1,T_2)\in E \\ f\in y_4(T_1) \\ f\pm 1\in y_4(T_2)}} ad(T_1,T_2)$$

for a feasible solution y_4, is called the second stage restricted adjacent channel assignment problem for slow frequency hopping networks (FAPHRES_A).

At first, some important observations of the FAPHRES_A problem, restricted to

$$x_T = 1 \quad \forall\ T \in \overline{S}.$$

is given. Every set T should have some frequencies between one and $|F|$ assigned, in a unique way (bijection between sets and frequencies). Hereby a penalty value arises (adjacent channel interference), when two sets have neighboring frequencies. Further, every set can only induce two penalty values and it is not important if the interfering channels are e.g. 3 and 5 or 7 and 9. In other words, a special sequence of these sets is searched. A graph description/interpretation can be given in the following way. Every set T forms a node and two nodes T_1 and T_2 are linked (not directed) if both sets induce potential adjacent channel interference, with the interference value $(ad(T_1,T_2) + ad(T_1,T_2))$ as weight. Accordingly the subgraph is complete, though a link may have weight zero. At the end, a (start and end) node in this graph, connected with every other node (weight zero) is added. Now, the aim is to create a tour, containing all nodes exactly once and with minimal length (from which a frequency assignment is easily reconstructed). Resulting (with a polynomial reduction: creating the graph), the (restricted) FAPHRES_A problem can be formulated as traveling salesman problem [TSP]. It is assumed, that LP formulations of the TSP are known to the reader, so more details are omitted here (a [TSP] formulation can be found in the following section (6.3)). Therefore the restricted FAPHRES_A is obviously NP-complete and "theoretically hard".
For completeness sake, since $FAPHRES\_A$ contains the above special case, it must be NP-complete, as well.

Though the settings of separation and blocking constraints are left out, they are mentioned here for completeness sake. These conditions could be partially incorporated in the TSP transformation as well. Inner or outer separation constraints between the variables / sets ($x_{T_1} + x_{T_2} \leq 1$, for $T_1, T_2 \in \overline{T}$, in the notation of the section above), can be reflected in the subgraph by missing links between the nodes for T_1 and T_2, at least if the separation demand is not bigger than one. Bigger separation or blockings may need a special TSP formulation and can probably not be applied in all of them. Complexity wise, this problem does not get easier, when applying these conditions, so that the problem is at least NP-hard, still (without going deeply into formalism, here).

For practical applications, with $x_T = 1 \; \forall \; T \in \overline{S}$, the above reduction to [TSP] yields to an useful computational approach. Even though the problem is NP-complete as well as the TSP, much effort has been put into [TSP] problems and since the instances are relatively small (|F| nodes), a good solvability can be claimed. Since this restriction holds for all test cases from section 6.3, this approach is chosen in the following section.
In the case, that not all x_T are equal to one, a reduction to TSP is still possible. Hereby every set T with $x_T > 1$ is replaced by x_T different nodes. Each two of these new nodes are connected by an edge with weight $ad(T, T)$ (which needs not be zero) and each other node (e.g. node S, the rest of the graph is formed like above) is connected with these new nodes by edges with weights $ad(T, S) + ad(S, T)$. As a result, in both cases the resulting graph has |F| nodes and is therefore easily accessible/computable by a TSP formulation. Further note, that since this graph can be constructed in polynomial time in both cases –same amount of nodes– a direct reduction from FAPHRES_A to TSP would have been possible as well.
In the following section, some solutions of the FAPHRES_A problem, obtained with a [TSP] formulation are given and explained.

6.3 — *Computational Results* —

In this section, the results of the second stage problem are presented. Since this problem was reduced to the well known traveling salesman problem, the results are not as presented detailed as the ones from the first stage problem. For the computation, the same hardware and software as presented in section 3.3 was used. For completeness sake, the used MIP formulation is the well known “two -connectivity without sub tours“. Denoting $\overline{T}$ as the set of vertices, ad_{T_1,T_2} describes the potential adjacent channel interference between the nodes. The binary variable x_{T_1,T_2} denotes,

whether this edge is used in the optimal tour (the graph is obtained with the above mentioned processing). Than, the [TSP] can be formulated as

$$
\begin{aligned}
&min && \sum_{\substack{T_1\in\overline{T}\\ T_2\in\overline{T}}} ad_{T_1,T_2}\cdot x_{T_1,T_2} && \\
&subto && \sum_{T_1\in\overline{T}} x_{T_1,T_2} = 2 && \forall\ T_2\in\overline{S},\\
& && \sum_{\substack{T_1\in S\\ T_2\notin S}} x_{T_1,T_2} \geq 2 && \forall\ S\subset\overline{S}, 1\leq|S|\leq|\overline{S}|-1, \qquad (11)\\
& && x_{T_1,T_2}\in\{0,1\} && \forall\ T_1,T_2\in\overline{S}.
\end{aligned}
$$

Since a node corresponds to a set of STRX and we are interested in a visiting sequence for that nodes, the underlying graph is complete (all potential visiting sequences are allowed). Further, since the edges' weights denote the potential adjacent channel interference between the nodes, this graph is not metric (the weights do not obey the triangle inequality). Additional, it is very likely, that nearly all edges will have strictly positive edge weight, such that the graph is complete.
This model can be easily accessed by a cutting plane approach (see section 8.2.2 for a short introduction into cutting plane algorithms), similar to the one presented in the scip TSP - example [AB], separating the conditions (11). As mentioned in the beginning, the TSP problem is well researched, so that very fast heuristics are available. At this point it is to say, that referential implementations, like the one in scip, mentioned above or the famous Concorde solver [Coo] are very fast, but could not be used here, since they demand metric settings. Nevertheless, expanding research on separating subtour constraints in TSP problems was not the focal point of this work, so that a very basic implementation of the problem was sufficient.
Especially, this means, that for every integer solution for the restricted problem (without the conditions (11)), it is tested whether these contain a subtour (which is possible in polynomial time). While it is formally necessary to solve a separation problem (in this case a MINIMUM-CUT problem) to find the corresponding constraint, cutting away the infeasible solution, this it not needed here. Since the problem instances are relatively small, it is sufficient to recognize the infeasible nodes in the branch and bound tree. If an infeasible node can be branched further (e.g. not all integer/binary variables fixed), this is done, if not this node is marked infeasible and closed. This has the drawback of bringing the solution process closer to a complete enumeration of all feasible integer solutions, thus increasing runtime, but on the other hand, it prevents, that the MINIMUM CUT problem needs to be explicitly threaten (similar to the pricing problem). As mentioned above, the instances are

small enough, such that additional branching instead of solving a MINIMUM CUT problem is sufficient. The given instances could be solved (optimally) in reasonable time (mere minutes). Nevertheless note, that potential improvements can be made, by explicitly creating those separated constraints. But since this is done already in the above mentioned solvers, restricted to the metric case, few further knowledge could be gained, such that the above implementations are sufficient for our matter. Resulting, a runtime analysis is not invoked here.

In the following (table 4), the results of the second stage problem from the problems instances of section 5.2 are shown. Herby, it is to mention, that the key difference between these scenarios is the amount of available frequencies, effectively determining the size of the TSP instance (note, that all "TSP graphs" are complete, with positive edge weight nearly everywhere). Opposing to the first stage problems, these TSP instances are relatively small ($O(|F|)$ nodes, $O(|F|^2)$ variables / edges), so that they can be solved to optimality (integer) .
By a rough overview on this data, one can say, that the results behave similar to the first stage results. The *Bradford* scenario (more inbound STRX) produced higher values than the *Siemens*1 solution, and more than the *Koeln* and the *Swisscom* scenarios. Further, the interference values decrease with more frequencies available.
But having a closer look on these results, it appears, that the second stage results

Name	**Solution value**	**50%**	**150%**
Siemens1	125.89	363.76	43.07
Bradford-0-Eplus	207.11	516.96	108.02
Koeln	28.55	83.02	11.03
Swisscom	0	0	0

Tabelle 4: Overview Second Stage Results

are sometimes higher than the corresponding first stage results. This is remarkable, since adjacent-channel interference was supposed to be of lower order than co-channel interference. For example having two TRX v and w, their potentially induced adjacent-channel interference is always lower than their potentially induced co-channel interference. Nevertheless, this is a straight forward consequence of a two staged result and exactly the point, where this concept looses in comparison to a combined approach: An optimal "co-channel assignment" (or a sufficient close approximations) may contradict an optimal "adjacent channel assignment" (and both need not be optimal in a combined approach).

Summing the results of the first stage up, it is to say that the second stage problem can be computed very well and that the solutions behave (as expected) similar to the first stage results. In the following section, a final evaluation of the two staged approach is given, judging the here presented models and results for practical suitability.

§7 Evaluation

In the following, an overview on the results of this thesis is given. Hereby, it is to recall, that the aim was, to provide methods for producing optimal frequency assignments (out of the context of integer programming) for practical purposes.
As a first result, it was shown that a straight forward mixed integer program, likewise presented in section 3, for the here presented specifications of slow frequency hopping, is not helpful at all. Even, when exploiting characteristical features, like symmetry or a certain sparse structure, no success (not even in the linear relaxation) could be achieved, and this probably applies to the near future (concerning computer memory advances) as well.

As an alternative, the decomposition into two problems, for co- and adjacent channel interference was proposed. Hereby, the hope for an optimal solution was abandoned, since an assignment, primary optimal for co-channel interference and secondary optimal for adjacent channel interference needs not be optimal in the sum of both cases. Further, some minor constraints like local blockings or separations have been omitted, as a trade of for computability. This decomposition provided some success in the following way:
The first stage problem could not be solved optimal, neither the mixed integer program, nor the linear relaxation. Nevertheless, some approximation (or in other words, asymptotical dual bounds for the integer problem) could be obtained by a column generation approach. In this approach, the pricing problem could be treated with greedy heuristics in most cases, it was not necessary to solve it explicitly. The main problem, preventing an optimal solution, appeared in embedding the outpriced variables. The first stage result is rather strict in it's constraints, in the following way. Having a feasible basis solution, in most cases it is impossible, to change exactly one basis variable, to obtain another feasible (better) solution. These other variables are called surrounding variables. During a column generation process, these surrounding variables may not be known, which caused these problems. The here presented workaround, creating some surrounding variables heuristically, worked in the way, that an improvement of the basis solution was obtained. Though, these variables will not ensure the maximal quality gain, out of the new variable, thus the solution process was slowed down significantly. Further this had the drawback of adding very many variables. This foreclose an optimal solution in the following way: On the one hand, these (dimensions of) variables overcrowded the memory and on the other hand, every simplex calculation takes more time than the one before. Therefore, the solution process slows down even more. On top of that, it is to men-

tion, that if the pricing problem was solved explicitly, because no heuristic had been successful, this took a significant amount of time as well. As a consequence of the tree factors (improvement not used to it's full extend, many surrounding variables, slow explicit solution of the pricing problem), this stage could not be solved optimal. At the same time, an evaluation of the approximation by lower bounds (as proposed in section 5.3) was not possible. This is reasoned by the following facts: On the one hand, an explicit solution of the pricing problem is not trivial and on the other hand, the reduced costs needed to reach a certain maximum level, which was not given in the here obtained approximations.
Nevertheless, with further, potentially more powerful heuristics and a better concept of finding these surrounding variables, this problem may be solved optimally in the future (and may therefore be used for creating frequency assignments in practical circumstances).
The second stage problem, optimizing a first stage solution towards adjacent channel interference was formulated as a variation of the famous TSP problem. Therefore, it could easily be solved for the instances derived from feasible solution of the first stage problem (since these are typically relatively small). Hereby, the means of choice was a cutting plane algorithm, though not to much effort has been put into this, since sufficient work onto the TSP problem is available already.

All in all, the here presented work shows some promising approaches for obtaining optimal frequency assignments. Though it was not possible to obtain optimal solutions (in the first stage), the reasons could be identified and may potentially be fixed in future research, so that an optimal planing may become a useful alternative to the usage of heuristics/simulated annealing or similar.

§ 8 Appendix

8.1 — *Complexity Analysis* —

Developing (optimal) solution algorithms is always related to a proper understanding of the treated problem. Hereby, the measure of "hardness" or "complexity" of a problem is used to indicate the difficulty of a problem and therefore giving the background for analyzing an algorithm's quality or suitability. The aim is, to give some system of difficulty classes, in which every problem can be sorted into, for the mean of comparing different problems. The here presented overview, shall outline the concepts for such a complexity analysis, as far as used in this work. This means, a problem recognized to be a member of a certain complexity class for indicating it's potential computational behavior. For a more detailed and a more formal information on that matters, the script [Voe09] or the standard work [GJ79] might be taken into account.

For such an analysis, the following Landau - notation is used for describing asymptotical behavior of functions. A function f is called to be in $O(g)$ (or of order $O(g)$), with g a function as well, if and only if

$$0 \leq \underset{x\to\infty}{limsup}, \frac{f(x)}{g(x)} < \infty.$$

So $O(g)$ is the set of all functions, being of lesser order then g. In the following, the concept of measuring complexity by a "worst case" runtime is proposed. Hereby, there are some formal issues when speaking of "basic computation steps", as a mean of measure. The concepts of RAM (Random Access Machine) and TM (Touring Machine) are formally important, but the imagination of the costs of a calculation as being proportional to the addressed bits is sufficient here.

At first, when talking about time consumption of an algorithm, the definition of (worst case) runtime is needed. An algorithm is said to have a runtime of $t_A(n)$ for $n \in \mathbb{N}$, if any input of length n can be treated in at most $t_A(n)$ basic calculations. Resulting, an algorithm is said to be "polynomial bounded" (or short polynomial) if

$$\exists\ \alpha \in \mathbb{N} \text{ with:} \qquad t_A(n) \in O(n^\alpha).$$

With that expression, the first complexity class / category of difficulty can be given. Derived by the definition above, the first complexity class introduced is called P (for

polynomial): P is the class of all problems, for which a deterministic polynomial (solution) algorithm exists. So the most common way, to prove that a problem in P, is to give the corresponding algorithm.
Before introducing more complexity classes, some short explanations of the importance of this class are given. In general, this class is treated as the class of problems, which are efficiently computable (time consumption is polynomial related to the input). Without formally going into detail on "efficiency", it is meant, that these problems are "practically solvable". Important examples in this context are shortest path problems, sortings or linear programming.

The next complexity class pursues the above concept. While P treats the problems solvable by deterministic polynomial algorithms, the class NP contains the problems being polynomially solvable by a not deterministic algorithm. Since this a very theoretical construct, a problem is proven to be in NP, if every given solution is polynomially verifiable (which symbolizes the randomness, the "guessing" of a non deterministic algorithm). Note, that NP does not mean "not in P", which is a very common mistake. Characteristic for this class is, that all known (deterministic) algorithms, solving problems of this class, have exponential runtime, rendering this a class of "not efficiently" computable problems. As above, some referential problems are the traveling salesman problem, knapsack problems or coloring problems. Clearly (since the deterministic algorithms can be interpreted as a subclass of non-deterministic or randomized algorithms), both classes are related via

$$P \subseteq NP.$$

There resulting question is, whether $P = NP$ holds is very famous, since it belongs to the Millennium Problems (some of the most famous unresolved problems in mathematics, see the article [Ins] for more details). However, this (or the contrary) is not proven yet, though it is assumed that P is strictly contained in NP.
Contrary to the class P, NP symbolizes the problems, which are not efficiently computable, or informally spoken, "which take ridiculous time to compute". So characterizing a problem as a member of one class (in the sense of "lowest" possible) generalizes it's difficulty or complexity. Therefore encouraging the computation of optimal solutions (if in P) or the use of approximations (if in NP).
Obviously, further complexity classes can be derived in a similar manner, too. The most common ones are EXP (exponential time consumption) or LOG (logarithmic). Nevertheless, P and NP are the most commonly used ones, for the sake of ordering problems into "good and bad" computability.

So how is a problem sorted into one of that classes? The easiest way is to find

an algorithm, e.g. deterministic polynomial, for that problem and the result is, that it lies in P. Since this is relatively hard with the first definition of NP (finding a randomized polynomial algorithm), the mean of verifying the solution in P is often used.
Another important mean is the (polynomial) reduction. This means a problem p_1 is polynomially reducible (time consumption) to another problem p_2, if and only if the input of p_1 can be transformed to an input for p_2 in P. This is denoted with

$$p_1 \leq_p p_2.$$

With this conditions, it holds, that if p_2 is in P, the same holds for p_1 (and analogous for NP; time consumption of the reduction is neglectable, there). Note, that this is often used the other way around, in the NP case: If p_1 is in NP, p_2 must be in NP as well (assumed $P \neq NP$), since a polynomial time algorithm would have been found for p_1, otherwise. Furthermore, with the concept of polynomial reduction, the notation of NP completeness can be introduced. A problem p is NP complete, if and only if

1. $p \in NP$.
2. $\forall\, \overline{p} \in NP: \quad \overline{p} \leq_p p$.

An example is the SAT problem, among others. As a consequence, and respecting a problems reduction: If $P \neq NP$, no polynomial time algorithm for NP complete problems can exist, given a warranty of of how "good" (or fast) an algorithm for that problem can be.
Resulting, the NP complete problems are the "hardest" among NP and one can imagine the relation between the mentioned classes in the way, shown by figure 18 (with the assumption, that $P \neq NP$, which would collapse the figure, else). Mathematically spoken, modulo polynomial reduction (or modulo "efficient computing") the NP complete problems are all from the same difficulty (since they are reducible to each other). So the concept of polynomial reduction offers a straight comparability between the problems in one class, or offers the possibility to compare a certain problem to other, already analyzed ones.

Summing this section up, a problem is classified in a complexity class, often the decision between P and NP, estimating a potential good (bad) solving behavior and offering an analogy to other problems in this class. As mentioned in the beginning, this section may be seen as an introduction into complexity analysis, rating a problems difficulty. For further information, specific literature like the book [GJ79] or the lecture script [Voe09] is advised

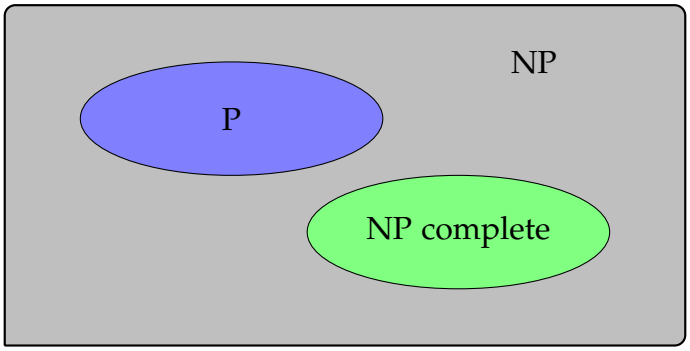

Figure 18: Complexity Classes

At the end of this section, it is to mention, that all the above concepts center around runtime. Nevertheless this can easily be converted on other resources, used for computations, as well. Hereby, the memory usage is the most significant other aspect, for which the classes P and NP and others may be distinguished as well.

8.2 — *Linear Programming* —

8.2.1 The standard form

Linear programming is one of the key aspects in operations research. Many "real life problems" can be expressed by a linear program (LP), reasoning this concept's importance. Famous examples, closely related to linear programming (restricted to integer solutions) are knapsack or traveling salesman problems. In the following section, backbone ideas and concepts of linear programming are summarized, as they are the most important mean in this work. Hereby a rough overview on techniques and principles is rather necessary then a formal prove. Resulting, some introduction on linear programming can be found here, but for a more formal and more detailed view some other source, like the books [Chv83] or [HL09] is recommended.
Mathematically spoken, a (standardized) LP consists of two vectors, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ with $n, m \in \mathbb{Z}_+$ and $n \times m$ matrix $A \in \mathbb{R}^{n \times m}$. The aim is to find a vector $x \in \mathbb{R}^n_+$, which minimizes

$$c^T \cdot x,$$

with respect to the conditions, given buy

$$A \cdot x \leq b,$$

whereas $\leq$ is declared component wise. The name "linear program" comes from the fact, that the variables x may only occur in linear form an not quadratic, exponentially or likewise. An (2 dimensional) geometrical representation about the solution space, described by the constraints $A \cdot x \leq b$, is given by figure 19. Some variations like "maximization" or "$\geq$" constraints are possible but can equivalently be expressed in the above form. Mostly, some direct conditions on x like $x \geq 0$ are given separately and are not incooperated within A.
A solution x is called feasible, if it satisfies the conditions $A \cdot x \leq b$. This solution is called optimal, if it is feasible and minimizes $c^T \cdot x$. Nevertheless, an optimal solution may not be unique, which is explained later on. Every feasible solution gives a primal bound, an upper bound (in the case of minimization) to the objective function. If $x = 0$ is a feasible solution, the value zero is a trivial primal bound.
Additional to feasible and optimal solutions, the whole LP can have no (not degenerated) solution at all. This is the case, if it is infeasible (if no feasible solution exists) or unbounded (no maximal objective value exists).

In the context of linear programs, the concept of duality needs to be mentioned

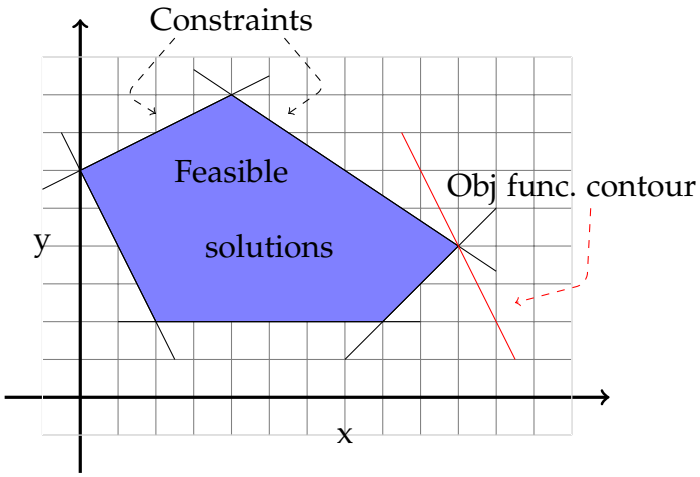

Figure 19: Typical 2 dimensional LP

as well. As hinted above, every minimization problem corresponds to a maximization problem (and vice versa). The original problem is called the primal problem and the derived one the dual problem. This relation can be expressed via

$$\begin{array}{llcll} min: & c^T \cdot x & \Leftrightarrow & max: & b^T \cdot y \\ subto: & A \cdot x \leq b, & & subto: & B^T \cdot y \geq c. \end{array}$$

where x and y is greater or equal then zero, component wise. Some of the key facts about the relationship of both problems are the following:

- The dual problem has a (bounded) optimal solution if an only if the original LP has one as well (and both problems have the same objective value in the optimum).
- If the primal problem has no feasible solution, the dual problem is unbounded or has no feasible solution either.
- If the primal problem is unbounded, the dual problem is not feasible.

While this relation is important in general and more complicated then suggested here, it is not used to it's full extend in this work, so [Chv83] is referred for further details. In the following, solution algorithms for LPs are lined out.

8.2.2 Solution of LPs

Generally speaking of solutions of LPs, their special geometrical view plays an important role. The space of all feasible solution, is described by $A \cdot x \leq b$, which forms a (hyper dimensional) polyeder. One can show, that an optimal solution (if existing) must be placed on a vertex (if unique) or on an edge (not unique). Such a solution and a vertex is called basis solution. This is due to the fact, that the solution corresponds to an invertible sub matrix from A, which is exploited by the simplex algorithm, explained below. For completeness sake it is to mention, that an infeasible LP corresponds to an empty ployeder (and an unbounded LP to and unbounded polyeder).
The standard solving process of such LPs is called "simplex" (algorithm). Hereby, the simplex operates the following way. Beginning with a start solution (usually $x = 0$ is feasible) it is subsequently tried, whether some neighboring vertex offers an improving objective function value (by the means of reduced costs, like shown in section 4.1). If this is the case, this solution is the new starting solution and the algorithm starts again, if this is not the case, the solution must be optimal. Without going into detail again, it can be proven that this algorithm is correct (and terminates at finite times). The backside of this algorithm is it's theoretical runtime, which is exponential, though it can be carried out with basic calculations on the columns and rows of A.
On the other hand, there are other algorithms (ellipsoid methods), which guarantee solvability in P, for every LP. Nevertheless, these algorithms have very high exponents/coefficients, such that the simplex method is used in most/practical cases (though the simplex runtime is exponential theoretically, it is far better in many practical applications). In this context, it is to add, that some different variations of the simplex algorithm are possible. While the normal simplex (as presented above) is depended on the choice of the pivot element (when more then one improving vertex exists, which to choose), there are variations like the dual simplex (= simplex on the dual LP) available. For computational issues it is mostly not clear beforehand, which is the best / fastest to use. Note, for that matters the concrete constraint structure is important. As a side note, a mean for modifying the constraint structure for enhancing the computability is the lagrangian relaxation. Here, constraints are "dualized into the objective function", such that they need not be regarded as explicit constraints any longer. Fore more information, it is referred to section 5.3, where this context is shorty explained (a more detailed introduction is given in [JF10]).

While LPs have a generally good computational behavior, some problems occur when dealing with extremely big instances. Very large input, e.g. exponentially si-

zed columns (variables) or constraints (rows) leads to problems when calculating or even storing the matrix A. In many practical problems, the entries of A can be calculated easily, so that there is no need to store the whole matrix in a practical sense. This can be used for two mayor approaches: column generation and feasible inequalities (cutting planes). While column generation is explained in details in section 4 (for an even wider perspective on that topic, it is referred to the book [DDS05]), cutting plane approaches (more details for example in [GLS95]) focus on reducing the amount of used constraints to a minimum. The key idea is, that (starting with a fixed subset of constraints and a corresponding optimal solution to that subset), all other constraints are (implicitly) tested, whether they are satisfied as well. If this is the case, the solution is optimal, if not, the injured constraint is added to he subset and a new solution of that subset is created via the simplex.
Both approaches have the same concept in common: Reducing the calculation on the needed minimum but preserve the usage of the simplex algorithm on a sub matrix of A. Both approaches may may improve the runtime (compared to the "normal" simplex), but at the same time, it holds that

$$Optimization = Separation = Pricing,$$

which means that the theoretical complexity / hardness of the problem remains unchanged (see [GLS95] for more details). Finding improving variables or injured inequalities is as hard as solving the original problem. This especially, holds true for integer problems, as presented in the following section.
Further options, when dealing with huge models can be found in the range of decomposition schemes (here, equivalent decompositions are meant, where the decomposition preserves optimality). Without going into details here (this approach is not used in this work), this deals with dividing a LP into different equivalent subproblems. As an example, if A a consists of two blocks, this may look like

$$\begin{aligned} &min: && (c_1, c_2)^T \cdot (x, y) \\ &subto: && \begin{pmatrix} A & - \\ - & B \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \\ & && x, y \geq 0. \end{aligned}$$

This can be reformulated with two problems, namely

$$\begin{aligned} &min: && c_1^T \cdot x && \text{and} && min: && c_2^T \cdot y \\ &subto: && A \cdot x \leq b_1, && && subto: && B \cdot y \leq b_2, \\ & && x \geq 0, && && && y \geq 0. \end{aligned}$$

Nevertheless, in general, identifying structure for a decomposition is not as easy as in this example. All in all, these are the most common possibilities to deal with inputs, bringing the computers memory to it's limits, without making restrictions / shrinking the model down (which is often called a "adequate" preprocessing).

8.2.3 Mixed integer programming

While linear problems as presented above are solvable in most practical aspects, the biggest problem occurs when focusing on integer variables. Formalized, this is a shift away, form linear programming toward mixed integer programing (MIP). This means, instead of $x \in \mathbb{R}^n$, $x \in \mathbb{Z}^n$ is demanded (since many practical/combinatorial problems demand an integer or even binary decision, e.g. with $yes = 1$ or $no = 0$, and no fractional values allowed) for a subset of the variables x. With this restrictions, the problem is not only not linear any longer, but NP hard.
As consequence, the solution process complicates even further. Hereby, the standardized approach is an extended simplex algorithm. Before going into detail, the expression "linear relaxation" is needed. The linear relaxation is an omitting of the integer constraint in the MIP, forming a LP again (making the problem linear, again). This problem can be solved in the above manner, but it will not produce integer solutions, in general. While a feasible solution for the MIP yields a primal bound, the solution of the relaxation is not feasible (for the MIP) in general and therefore offers a dual bound.
The whole process of solving MIPs is called "branch and bound", which describes how the above concept is used. At first, the (starting) MIP is relaxated and solved, which produces a fractional solution, in general. Resulting, a further condition is added to the MIP, which prohibits the fractional solution (it is sufficient to treat one fractional entry of the vector x), e.g.

$$x = \frac{3}{2} \quad \Rightarrow \quad \begin{cases} x \geq 2 \\ x \leq 1 \end{cases} .$$

Hereby both conditions contradict each other, but it is not known, which of both is obeyed by an optimal solution (fractional values are evaded by a case by case analysis). Resulting, the MIP is split up into two different problems of the same size / complexity, each containing one of the new constraints. Then, the whole process starts again. Each of the MIPs is relaxated, solved and potentially split again. Resulting, the problem is solved by exponentially often solving linear programs (graphically spoken, a branch and bound "tree" is created, with a LP at each node). In every node, a LP is solved, giving a dual bound (lower in the case of minimization) to the objective value, a better value then a feasible solution might achieve. At the same time, every primal solution offers a primal bound. Though much effort can be saved by sensibly branching and bounding (e.g. ignoring LPs , whose result is worse then earlier obtained results, sensible handling of dual and primal bounds), this process can degenerate to a complete enumeration of the possible integer solutions in the

worst case. As a consequence, the solving behavior of MIPs is much worser than the one of LPs, in general.
Concerning the approaches for very large instances (column generation and cutting planes), both can can be combined with the branching process of MIPs, which is then called branch and price and branch and cut respectively. A branch and cut approach is relatively easy to set up. Similar to above, a LP may be solved by a cutting plane approach, resulting in a potentially fractional solution. Then the problem is split up again and two new problem are created, which are solved again with cutting planes. Hereby possibly adding inequalities totally independent of each other. A column generation approach has some difficulties, on the other hand. Referring to 4.1 for more details, in a branch and price process, the pricing problem may change with every added inequality. This results in difficulties when generically creating and solving this problem at runtime.

Concluding this section, linear and mixed integer programming are an important aspect of mathematical optimization. The above section shall give a first introduction on that matters, but for more detailed information, the corresponding literature, like the above mentioned [Chv83] or [HL09] should be accessed.

8.3 — *The classical Frequency assignment Problem (FAP)* —

The classical frequency assignment problem can be described the following way: Given the set of all TRX (e.g. $TRX := \{a, b, c\}$) and the available frequency spectrum (e.g. $F := \{1, ..., 5\}$). Given the possible interference data between all pairs of TRX (e.g. $co(a, b) = 0.5$ and $ad(a, b) = 0.3$) and separation constraints (e.g. $d_{a,} = 2$, frequencies of A and B must have a distance of two channels in the radio spectrum). Further, given a list of locally blocked channels (e.g. $B_a = \{2, 4\}$) for each TRX, the frequency assignment problem is the following: Assign a channel to each TRX, such that separation constraints are met and local blockings are obeyed, with minimal total interference (between all pairs of TRX). Note, that a formal (mathematical) definition was already given in section 3.4.

In a mixed integer, program, this can be expressed the following way. A binary variable $x_{v,f}$ denotes, whether frequency f is assigned to TRX v. Two auxilliar (binary) variables $y_{v,w}$ and $n_{v,w}$ notify if there occurs co-channel (and respectively adjacent channel) interference between TRX v and TRX w. Using this and the notation from above, this leads to:

$$\text{min:} \sum_{v \in TRX} \sum_{\substack{w \neq v \\ w \in TRX}} co(v, w) \cdot y_{v,w} + ad(I, J) \cdot n_{v,w} \tag{12}$$

subject to:

$$\sum_{f \in F \setminus B_v} x_{v,f} = 1 \qquad \forall\, v \in TRX, \tag{13}$$

$$x_{v,f_1} + x_{w,f_2} = 1 \qquad \forall\, v, w \in TRX, |f_1 - f_2| \leq d_{v,w}, \tag{14}$$

$$x_{v,f} + x_{w,f} \leq 1 + y_{v,w} \qquad \forall\, v, w \in TRX, f \in F \setminus B_v \cup B_w, \tag{15}$$

$$x_{v,f} + x_{w,f \pm 1} \leq 1 + n_{v,w} \qquad \forall\, v, w \in TRX, f \in F \setminus B_v, f \pm 1 \in F \setminus B_w, \tag{16}$$

whereas

$$x_{v,f} \in \{0, 1\} \qquad \forall\, v \in TRX, f \in F,$$
$$y_{v,w} \in \{0, 1\} \qquad \forall\, v, w \in TRX,$$
$$n_{v,w} \in \{0, 1\} \qquad \forall\, v, w \in TRX.$$

Hereby the constraints 13 (each TRX gets one locally available frequency) and 14 (obey separations between every pair of TRX) provide feasibility, while constraints 15 and 16 register interference potential. At the end, the objective function 12 gives

the best solution, with respect to the minimal total interference. Various research has been done on this model before, some analysis and a compilation of results can be found in [Eis01], for example.
At the end, some interesting facts on this model are mentioned. At first, from a mathematical point of view, FAP bears a high analogy to graph coloring. Interpreting the TRX as nodes and possible interference relations (co- or adjacent- channel interference values strictly higher then zero) as edges between them, one gets the input for a graph coloring instance. By definition, every coloring obeys a separation of at last one per pair of TRX, so every interference (co- or adjacent channel) is permitted by this setting. Ignoring separation constraints above one, coloring brevodes feasible solution, without any interference and a minimal amount of frequencies (which still might be higher then the amount of available frequencies). In a way, FAP might be seen as a generalization of coloring. On the one hand, some operation constraints and local blockings are introduced and the amount of frequencies is limited (which "hardens" the problem). On the other hand, one is interested in a "weighted" coloring, so adjacent TRX might be colored the same way, if their penalty (interference) is low enough, which provides feasibility, concerning the limited amount of frequencies. As a result of this similarity, many results or algorithms of one model can be used adeptedly for the other. Some more detailed work on this topic can be found in [Eis01], too. Both problems are NP complete, as is shown in [Eis01], as well. This reasons, that in practical applications, frequency assignments are generated heuristically, rather then optimally.
Another interesting aspect on this model is the independence on special GSM properties. This model just uses separation and availability constraints and generates (and judges) solutions on the basis of interference. As a result, it may also be used in other radio systems, for example UMTS or such. So, this model is not restricted to GSM frequency assignment but may be used for more general purposes in frequency planing as well.

8.4 — *Slow Frequency Hopping: Technical Terms* —

Research in literature concerning slow frequency hopping is often connected with some technical terms, concerning the technical realization of hopping etc. In the following, some explanation to the most frequent appearing terms and their relevance in this work is given.
Going into detail with the concept of frequency hopping, there are some fundamental options or hopping modes, which are given in the GSM standards and need to be specified for a clear understanding.

- **Random / cyclic hopping**. This mode describes the special hopping type. A TRX may change its frequency at random or in a cyclic way (e.g. iterating through some frequency list). Since true interference diversity is only achieved by random hopping, (as stated in [Ce96]), the first type is treated in this work.
- **Baseband / synthesized hopping**. Both hopping modes have a rather technical influence. Referring to figure 6, [Eis01], p. 17 is cited: "With synthesized frequency hopping, each TRX of a sector transmits successive bursts on different channels. ... If more than one TRX is used for a sector, baseband frequency hopping can be applied alternatively. Each TRX uses a fixed channel, and the code words constituting a flow of communication are dispatched to changing TRX". Sloppy spoken, in synthesized hopping, a call stays on one antenna, which changes its frequency, while in baseband hopping, a call is passed over to different antennas, each sticking to their own frequency. Mathematically (concerning the LP model), this can be caught with a specified generation of the set of the STRX (baseband hopping means: all TRX of one sector form one STRX, with as much frequencies assigned as the amount of TRX inside). From this point of view, the different modes have rather no influence on the (LP) model itself. This thesis focuses on synthesized hopping, since it provides a greater margin in the model's creation, while baseband hopping can be regarded as a special case of synthesized hopping, in a mathematically point of view.
- **Schematic frequency reuse** (frequency patterns) or **optimal frequency assignments**. Making this point here is mostly for completeness' sake, much literature centers around schematic frequency allocation since this was the first possible way of assigning frequencies (historically spoken). On the other hand, optimal (or optimized) frequency assignments provide better results in general and so, the resulting problems are addressed by this master thesis. Nevertheless, one should be aware, that much literature and many results base upon schematic frequency reuse.

Some terms often appearing together with cyclic frequency hopping are **MAL**, **MAIO** and **HSN**. MAL (Mobile allocation list) describes the list of frequencies, over which to hop. HSN gives the hopping sequence number, which gives the special hopping sequence of the frequencies out of the MAL. At last, MAIO (Mobile Allocation index offset) describes the starting point in a MAL (to prevent interference if two STRX have the same MAL/HSN, so they do not hop synchron on the same frequencies). Since cyclic frequency hopping is not treated here, those terms are not used in in this thesis, directly.
Obviously, this list of terms does not claim to be complete, but should cover the most important terms.

§ 9 Bibliography

[AB] T. Achterberg and T. Berthold. http://scip.zib.de/examples.shtml.

[Ass] GSM Association. http://www.gsmworld.co.

[Bun10] Bundesnetzagentur. http://www.bundesnetzagentur.del, 2_2010.

[BVY05] P. Bjoerklund, P. Vaerbrand, and D. Yuan. Optimized planning of frequency hopping in cellular networks. *Computer and Operations Research 32*, 2005.

[Ce96] M. Chiani and e.al. Frequency and interference diversity in slow frequency hopping multiple access systems. *Seventh IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1996.

[Chv83] V. Chvatal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[Coo] Contact: W. Cook. http://www.tsp.gatech.edu/index.html.

[DDS05] G. Desaulniers, J. Desrosiers, and M. Solomon. *Column Generation*. Springer US, 2005.

[Eea98] S. Engstroem and et al. Multiple reuse patterns for frequency planning in gsm networks. *IEEE Vehicular Technology Conference*, 1998.

[Eis01] A. Eisenblaetter. *Frequency Assignment in GSM Networks: Models, Heuristics and Lower Bounds*. PhD thesis, Technische Universitaet Berlin, 2001.

[GJ79] M. Garey and . Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GLS95] M. Groetschel, L. Lovasz, and A. Schrijver. *Practical problem solving with cutting plane algorithms in combinatorial optimization*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science Vol. 20, 1995.

[HL09] F.S. Hillier and G.J Lieberman. *Introduction to Operations Research*. Mcgraw-Hill Higher Education, 2009.

[Ins] Clay Mathematics Institute. http://www.claymath.org/millennium/P_vs_NP.

[JF10] J.Foniok and K. Fukuda. *Integer Programming*. Insitute for Operations Research, ETH Zentrum Zurich, 2010.

[KI01] Z. Kostic and and X. Wang I, Maric. Fundamentals of dynamic frequency hopping in cellular systems. *IEEE Journal on selected areas in communications*, 2001.

[LD05] M. E. Luebbecke and J. Desrosiers. *Column Generation*, chapter A Primer in Column Generation. Springer, 2005.

[Lin] J. Linnartz. http://wireless.per.nl/reference/chaptr04/cellplan/reuse.htm.

[MHS05] J. Moon, L. Hughes, and D. Smith. Assignment of frequency lists in frequency hopping networks. *Assignment of Frequency Lists in Frequency Hopping Networks, Vol. 54, No. 3*, 2005.

[MT95] A. Mehrotra and M.A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 1995.

[ONS95] H. Olofsson, J. Naeslund, and J. Skold. Interference diversity gain in frequency hopping gsm. *Vehicular Technology Conference*, 1995.

[oSA09] Support officer: S. Antipolis. 3gpp ts 45.005 v9.0.0. Technical report, 3GPP, 2009.

[Voe09] B. Voecking. Berechenbarkeit und komplexitaetstheorie. Skribt zur Vorlesung, RWTH AAchen, 2009.