Olaf Maurer

Design and Scheduling Problems in Planning Optical Networks

Cuvillier Verlag Göttingen
Internationaler wissenschaftlicher Fachverlag

Design and Scheduling Problems in Planning Optical Networks

Design and Scheduling Problems in Planning Optical Networks

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
–Dr. rer. nat.–

im Fachbereich Mathematik
der Universität Kassel

vorgelegt von

Dipl. Math. Olaf Maurer

Promotionsausschuss

Erstgutachter : Prof. Dr. Andreas Bley
Zweitgutachter : Prof. Dr. Ir. Arie M.C.A. Koster

Tag der wissenschaftlichen Aussprache: 11.10.2016

Kassel 2016

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
1. Aufl. - Göttingen: Cuvillier, 2016
Zugl.: Kassel, Univ., Diss., 2016

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2016
Gedruckt auf umweltfreundlichem, säurefreiem Papier aus nachhaltiger Forstwirtschaft.

ISBN 978-3-7369-9387-7
eISBN 978-3-7369-8387-8

Contents

Introduction **7**

1 Technical and Mathematical Background **13**
1.1 Planning Optical Networks . . . 13
1.1.1 Network configurations . . . 14
1.1.2 Hierarchical network structure . . . 15
1.1.3 Optical access networks . . . 16
1.1.4 Collision avoidance . . . 17
1.2 Technical Background . . . 18
1.2.1 Hardware components . . . 21
1.2.2 Physical limits of optical networks . . . 26
1.3 Mathematical background . . . 27
1.3.1 Linear algebra . . . 28
1.3.2 Graph theory . . . 29
1.3.3 Computational complexity . . . 29
1.3.4 Linear and Integer Programming . . . 36
1.3.5 Algorithms for LPs and ILPs . . . 38
1.3.6 Modelling network problems . . . 41

2 Passive Optical Network Design **49**
2.1 The Two-Layer FTTX Network Design problem . . . 49
2.1.1 Introduction . . . 49
2.1.2 Problem setting . . . 49
2.1.3 Related work . . . 51
2.2 MIP model . . . 52
2.2.1 Problem input and constraints . . . 52
2.2.2 Valid inequalities for the aggregated MIP model . . . 56
2.3 Lagrangian Decompositions . . . 57
2.3.1 Feeder-Distribution Decomposition . . . 58
2.3.2 Fixedcharge-Flow Decomposition . . . 60
2.3.3 Generic Lagrangian Framework . . . 64
2.4 Computational results . . . 66
2.4.1 Branch&Cut algorithms . . . 66
2.4.2 Benchmark instances . . . 70
2.4.3 Computations . . . 72
2.5 Conclusions . . . 74

3 Node-Weighted Steiner Problems **81**
3.1 The Node-Weighted Dominating Steiner problem . . . 81

3.1.1 Introduction . . . 81
3.1.2 Problem setting . . . 81
3.1.3 Related work . . . 82
3.2 Integer programming formulation . . . 85
3.3 Polyhedral investigations . . . 86
3.3.1 Basic properties . . . 86
3.3.2 Model inequalities . . . 88
3.4 Partition inequalities . . . 92
3.4.1 Lifted Partition inequalities . . . 95
3.4.2 A complete description of P on a cycle . . . 96
3.4.3 Indegree inequalities . . . 110
3.5 Computational experiments . . . 111
3.5.1 Instances . . . 111
3.5.2 Implementation . . . 111
3.5.3 Results . . . 112
3.6 Conclusions . . . 113

4 Incremental Facility Location 115
4.1 The incremental UFL problem . . . 115
4.1.1 Introduction . . . 115
4.1.2 Problem setting . . . 117
4.1.3 Two-phase algorithm . . . 118
4.2 Analysis . . . 119
4.3 Lower bound . . . 121
4.4 Computational experiments . . . 122
4.5 Conclusions . . . 122

5 Incremental Connected Facility Location 125
5.1 The Incremental Connected UFL problem . . . 125
5.1.1 Introduction . . . 125
5.1.2 Problem setting . . . 125
5.1.3 Related work . . . 126
5.2 MIP models . . . 127
5.2.1 Valid inequalities . . . 129
5.3 Separation Algorithms . . . 131
5.4 Experiments . . . 132
5.5 Conclusions . . . 136

6 Frequency Assignment in Optical Networks 137
6.1 Spectrum Assignment problem . . . 137
6.1.1 Introduction . . . 137
6.1.2 Problem setting . . . 138
6.1.3 Computational complexity . . . 138
6.1.4 Related work . . . 139

6.2 Phase I: Articulation Point Heuristic . . . 140
6.2.1 Lower bounds . . . 141
6.2.2 Heuristic algorithm . . . 141
6.2.3 Hopcroft-Tarjan algorithm . . . 143
6.3 Phase II: Multicoloring formulation . . . 144
6.4 Phase III: Exact models . . . 146
6.4.1 Constraint Programming model . . . 147
6.4.2 Binary model . . . 147
6.4.3 Conflict-Graph Orienting model . . . 148
6.4.4 Branch&Price model . . . 149
6.5 Computational experiments . . . 153
6.5.1 Test instances . . . 153
6.5.2 Computational experiments . . . 153
6.6 Conclusions . . . 157

7 Online Scheduling 161
7.1 Online Scheduling problems . . . 161
7.1.1 Introduction . . . 161
7.1.2 Problem Setting . . . 162
7.1.3 Related work . . . 163
7.2 General simplifications and techniques . . . 165
7.2.1 Simplification within intervals . . . 166
7.2.2 Irrelevant history . . . 170
7.3 Abstraction of online algorithms . . . 177
7.4 Extensions to other settings . . . 182
7.4.1 Non-preemptive scheduling . . . 182
7.4.2 Scheduling on related machines . . . 188
7.5 Conclusions . . . 191

Bibliography 193

Symbol Index

$\subseteq$	subset or equal
$\subset$	subset and not equal
$\subsetneq$	subset and not equal, highlighting non-equality
$A \setminus B$	relative complement of set B in set A
$A \dot\cup B$	disjoint union of the sets A and B
$\mathbb{N}_0$	natural numbers, starting from 0
$\mathbb{N}$	natural numbers, starting from 1
$\mathbb{Z}$	integer numbers
$\mathbb{Q}$	rational numbers
$\mathbb{Q}_+$	nonnegative rational numbers
🦛	hippo
$\mathbb{R}$	real numbers
$\mathbb{F}^E$	set of all maps $f : E \to \mathbb{F}$
$[n]$	the set $\{1, \dots, n\}$
χ^S	characteristic $\{0,1\}$-vector of subset $S \subseteq E$
$\operatorname{rank}(X)$	linear rank of a set X of vectors
$\operatorname{arank}(X)$	affine rank of a set X of vectors
$\delta^{\text{out}}(v)$	arcs in a graph having v as source node
$\delta^{\text{in}}(v)$	arcs in a graph having v as target node
Γ^*_v	neighborhood of the node v without v
Γ_v	neighborhood of the node v including v
$\{0,1\}^*$	set of binary strings
$\lvert x\rvert$	length of a binary string x
$\lvert I\rvert$	encoding length of an instance I
$\delta(S)$	the cut induced by the node subset S of a graph
$\mathbb{R}_+$	positive real numbers
$\mathbb{R}^0_+$	nonnegative real numbers
$\mathbb{R}_{\geqslant 0}$	set of all $f : E \to \mathbb{R}^+_0$

Introduction

The topics of this thesis are mathematical optimization methods for the design, rollout and operating phases of optical networks. The work presented here was begun while the author was a member of the DFG[1]-Forschungszentrum "MATHEON: *Mathematik für Schlüsseltechnologien*" at the *Technische Universität Berlin* as part of the project *B21: Optical Access Networks*. It was continued and completed as a research assistant at the *Universität Kassel*, partially as a member of the *"Safe and Secure European Routing" (SASER)* project, subproject *SASER-Siegfried* which was led by Nokia and NSN Management International GmbH.

Focus of the thesis

This thesis focuses on different aspects of the design of optical networks:

- efficient design, i.e. cost minimization with a provable quality guarantee
- rollout, i.e. how to build the network when the design is complete
- how to operate the network, e.g. assigning the available spectrum to lightpaths when the physical network is established and traffic needs to be routed.

As this thesis is motivated by fundamental questions, we consider mainly the "big" aspects of planning and take an abstract view. In this spirit we try to tackle the main questions, while often ignoring some technical details in the process, as the latter can usually still be addressed once the general network design is already fixed.

Motivation

In the last decades, more and more communication services that were traditionally carried out using specialized platforms (e.g. mail, telephony, fax) have been converging towards the Internet. Until the early 1990s, voice traffic was the predominant cause of traffic in worldwide telecommunication networks, with a very predictable growth mainly driven by demographics. Then, non-voice data traffic began to surge and supersede the traditional voice traffic, mainly due to the emergence of the Internet.

Nowadays, the amount of data generated by voice traffic has long been overtaken by data traffic, which is now the predominant kind and shapes the worldwide traffic demands. There are a large range of applications ranging from high-definition video streaming to video telephony

[1]The Deutsche Forschungsgemeinschaft (DFG; German Research Foundation) is an important German research funding organization which supports research in science, engineering and the humanities. It is based in Bonn and is itself financed by the German states as well as the federal government.

that will keep requiring more and more bandwidth as more and more users in the world get connected (Cisco [2015]).

To cater for this ever-increasing demand, network technologies which provide very high bandwidth capacities are urgently needed.

One of the main technologies used for this purpose are optical networks. Their main advantages are the following: they can be used to send information across very long distances, they are insusceptible to electromagnetic radiation and allow transmission of large amounts of data due to their high bandwidth capacities. While due to physical reasons it is very difficult to exceed speeds of 10 gigabits per second using electrical transmission, transmission speeds exceeding one petabit per second across a distance of more than 50km have already been reached using optical technology (Takara et al. [2012]) and the current limits are still only based on the currently available technology, not on physical barriers.

Nowadays, optical technology is broadly employed in modern telecommunication networks. After optical technology was invented and made available commercially, it rapidly penetrated communication networks which had to handle large traffic loads. As research went on and the high physical capacities of optical fibers could be better exploited, usage of these networks increased even more.

While yielding lucrative investment opportunities for network providers, some hard to solve planning problems arise in the design and configuration of these networks. The problems arising in this context usually exhibit a high combinatorial complexity. There is a long-standing history of mathematical methods being applied to all kinds of different network design problems. The common difficulty among those optimization problems is the combination of several problem aspects, where the desired solution of one aspect depends on the outcome of one or even of all of the others. An example for this phenomenon is the dimensioning problem of links for a given network, while certain traffic requirements between endpoints are known and the routes between the endpoints have not yet been fixed. This results in a problem combining the dimensioning aspects of the network as well as the problem of fixing the traffic routing, where good solutions for the subproblems each depend on the other one. Often, these problems are even more complex. Building on the previous example, there is also the aspect of assigning intervals in the optical spectrum to chosen routes in such a way that routes using the same optical fiber do not occupy the same part of the spectrum. To not complicate these problems and models even further, in this thesis, we only consider the more or less high-level abstractions as described here and in the individual chapters and not the lots of intricate additional details coming into play when actually implementing these networks with real-world technology.

Another aspect that comes into play in the planning of these networks are the hard-to-predict traffic demands of modern usage scenarios. Unlike traditional voice traffic which causes a continous low-bandwidth stream, modern applications, for example large file transfers, typically cause bursts of high demand, followed by a period of almost no traffic. Networks employing traditional optical time-division multiplexing technology usually dedicate bandwidth to connections even when there is no traffic, which can lead to inefficient usage of the available bandwidth and result in unnecessary delays. To deal with these problems, infrastructure providers are also interested in ways to reconfigure the network on-the-fly to meet these rapidly changing demands.

This variability is connected to the need of solving planning problems of embedding virtual telecommunication networks into physical networks quickly. These networks need to be embedded in such a way that the demands (such as inter-node connection speeds, computing power or locality requirements) specified by the entity requesting the network can be satisfied.

In general, network planning problems can be further subdivided into the planning of final networks and the creation of rollout plans that specify in what order the network should be constructed. If the network is built in stages and is already usable at an intermediate stage, it can already start serving customers which reduces the total cost to the network provider. Efficient algorithms are needed to solve these sometimes very large problems. We call these problems *incremental* network design problems. This thesis also devotes two chapters to them.

Contributions and Outline of the Thesis

This thesis consists of seven chapters that cover different topics from problems in access networks to problems in core networks, ranging from rather theoretical to more applied topics as well as from problems related to the design phase to problems related to the rollout phase. We also investigate some theoretical problems that are motivated by the aforementioned network embedding problems. Except for Chapter 1, the different mathematical optimization problems are grouped into one problem per chapter. We now provide a short outline of the contents of the thesis.

Chapter 1: Technical and Mathematical Background In Chapter 1, we introduce the basic concepts of this thesis. We introduce the basic notations, explain the technical aspects as well as the mathematical aspects and backgrounds of the considered problems.

Chapter 2: Passive Optical Network Design In Chapter 2, we begin with a problem belonging to the design phase of a passive optical access network. The problem consists of simultaneously optimizing the placement and dimensioning of an access network such that a two-level tree-like network architecture is obtained. To obtain meaningful solutions, we have to deal with both topological questions and routing questions at the same time. We call this problem the *Two-Layer FTTX Network Design problem*. The problem is quite challenging from a computational point of view, so to reduce the complexity involved in the solution process, we introduce two novel decomposition approaches based on Lagrangian Decomposition. These lead to very good solutions very quickly compared to non-decomposed approaches.

Chapter 3: Node-weighed Steiner problems In Chapter 3, we consider the previous problem from a more abstract point of view. The chapter is concerned with a problem where a network has to be built to connect the core network to some aggregation points of lower networks. The problem has connections to the famous *Steiner Tree problem*. We introduce a model based on Integer Linear Programming that only uses node-based variables. We analyze the structure of the related polytope and identify several large classes of facet-defining inequalities.

Chapter 4: Incremental Facility Location In Chapter 4, we deal with an incremental problem stemming from the rollout planning phase of networks. We consider a combinatorial approach that solves a variant of the so-called *Facility Location problem*. In this incremental version of the Facility Location problem, we not only want to design a network, but also consider all the stages in which the network is rolled out. We provide a polynomial algorithm that computes an incremental solution which has a guaranteed quality in the competitive sense compared to a nonincremental optimal solution.

Chapter 5: Incremental Connected Facility Location In Chapter 5, we consider a variant of the problem from Chapter 4 with an additional connectivity requirement, called the *Incremental Connected Facility Location problem*. Here, the fundament again are Integer Linear Programming techniques. We introduce several MIP models for the problem as well as some valid inequalities and present a Branch&Cut algorithm to solve the problem and compare these approaches in a computational study.

Chapter 6: Frequency assignment in optical networks In Chapter 6, we present solution methods for the frequency assignment problem of a certain type of network called a *FlexGrid network*. The problem we solve is called the *Spectrum Assignment problem*. In this problem, we are given some demands that have already been routed on fixed paths between endpoints in a network and we would like to assign frequency intervals to these paths such that crossing paths do not have overlapping frequency ranges. Due to its mathematical connection to scheduling problems, it is also called the *Chromatic Scheduling problem*. We present a three-stage approach, where the first stage consists of combinatorial algorithms only, which are already able to solve many of our instances. The second stage uses LP lower bounding techniques, while the third stage builds an exact mathematical programming model, if the previous stages were unsuccessful. One of these third stage models has exponential size and is solved by a tailored Branch&Price algorithm.

Chapter 7: Online Scheduling In Chapter 7, we present a general framework for competitively optimal algorithms for online scheduling problems. In previous approaches for competitive online scheduling problems, worst-case instances and competitive algorithms were usually provided for each problem by hand. We show that it is at least theoretically possible to algorithmically find a competitive algorithm for the considered scheduling problem whose competitive performance guarantee is not worse than the best possible up to a factor that can be chosen to be arbitrarily close to 1. We present our algorithmic framework with a focus on online scheduling for minimizing the average weighed completion time. We do not apply the techniques in this chapter directly to optical networks, but we point out the connections to online embedding problems of virtual networks into physical architecture.

Acknowledgements

This thesis would not have been possible without many people which supported me throughout writing and who I wish to mention here so that their influences will not be forgotten.

First of all, I would like to thank my advisor, Andreas Bley, for all the guidance and encouragement through the years. Your advice really was invaluable and our discussions on all kinds of subjects were always enjoyable. I am greatly indebted for your patience and for all the understanding you gave me.

Next, I would like to mention my coauthors Ashwin Arulselvan, Andreas Bley, Stefan Gollowitzer, Ivana Ljubić, Elisabeth Lübbecke, Nicole Megow, Martin Skutella and Andreas Wiese with whom I worked on different parts of this thesis. My work has greatly benefited from their influences.

The COGA group at TU Berlin has also made the whole time in Berlin a wonderful one. I would like to thank especially my office mate Ashwin Arulselvan and former COGA member Torsten Gellert for our almost daily discussions on all kinds of things. The friendships I found at the group are still strong and I would like to thank the whole group for this experience.

While the move to Kassel University was not one I very much looked forward to in the beginning, my time in Kassel was shaped by the nice people I met there, some of whom became good friends as well. I enjoyed the time there more than I thought, especially in the summer, and learned about lots of different ways to spend time apart from work (thank you Jörn!). I especially would like to thank my office mate Frank Fischer for all the things he put up with and the many, many convenient rides in his car.

For proof-reading parts of the thesis and thereby helping me greatly improve the presentation, I would like to thank Anja Fischer, Frank Fischer, Martin Groß and Ute Skambraks. Their comments were invaluable for the clarity and readability of this thesis.

I would like to thank my partner Ute for all the love and encouragement through this sometimes also difficult time—coincidentally, our relationship started almost at the same time as the work on this thesis—and of course all our little friends, without whom this journey would have been quite different and much less enjoyable.

Lastly, I would like to thank my parents for their continued support, love and especially patience. Without them, this thesis would have never seen the light of day.

1 Technical and Mathematical Background

1.1 Planning Optical Networks

In a general networking planning problem, we have some locations, possibly of different types, and would like to connect these in some specified way using a network. Usually, some locations have some form of demand. In optical networks, these locations are usually called the *customers*. There are many different kinds of networks, for example telecommunication networks, transport networks and road networks. In the telecommunications setting, reasons for the building of a network can be manifold and include access to the Internet, telephony, fax or specialized applications like dedicated, highly tolerant or high-speed networks.

Because the building of telecommunication networks usually involves high investments incurred by the deployment of network infrastructure, a careful planning of these networks is very important to minimize the potential waste of resources.

In network planning problems, we distinguish between *greenfield* and *brownfield* planning. The former means the planning for areas where no infrastructure exists so far. In brownfield planning, we have to consider the infrastructure already in place and account for any problems resulting from the technological migration.

In this thesis, we only explicitly consider greenfield planning problems, but the presented techniques can also be adapted for brownfield problems. Also, we usually assume for the purpose of simplification that we know exactly which customers want to connect to our network and how much bandwidth they need.

In real planning scenarios, the latter assumption is of course somewhat unrealistic. Here, stochastic and robust methods come into play, where one can consider wide ranges of possible customer scenarios and try to build a network that performes reasonably well in all the likely settings. While this also leads to interesting questions, we concentrate on exploring other aspects of the planning problems in greater detail. Therefore, we assume that no uncertainty in data is involved.

The goal of network planning is to find an appropriate network configuration, where appropriate refers to the fact that the resulting network should be feasible for the given traffic requirements. Configuration is a term encompassing the arrangement of hardware, the routing inside the network and also several other aspects which become especially important for optical networks. These include the problem of assignining some part of the optical spectrum to connections in the network, so called *lightpaths*, and possibly solving network embedding problems, depending on the structure of the bandwidth requests.

1.1.1 Network configurations

From the planning perspective, finding a network configuration for an optical network consists of two tasks depending on one another. One task is to place hardware devices with sufficient capacities at allowed locations. The other task is the establishment of lightpaths inside the optical network. This has to be done in such a way that the requested traffic can be transferred through the network while observing the capacity limits implied by the hardware limitations and actual routing. We now explain these tasks in some more detail.

Hardware configuration

From a somewhat abstract point of view, a physical network consists of a set of *nodes* and a set of *links* connecting pairs of nodes. The nodes can be used as pure transit nodes or represent access points of the network, where access points in this context refer to nodes where there is a connection to some other network or network layer and the node serves as a connection between them.

For the technical realization of a network, hardware devices need to be installed mainly on the nodes of the optical network.[1] Links in the network consist of optical fibers between nodes. Some hardware components are also necessary to enable transmission through the optical links. The hardware configuration of the network encompasses all decisions about which hardware devices should be installed at which locations. These devices include transmitters, receivers, fibers, switches, regenerators, splitters, couplers and filters. The number of transmitters and receivers is usually fixed in our problems as it is determined by the bandwidth requests. Installed devices offer *transmission capacity* on the links and the ability to switch, regenerate, convert or split at the nodes.

All these hardware components determine the available capacity; this part of the planning is called the *dimensioning* part. Sometimes, we express bandwidth demands as the number of optical channels assigned, usually when the channel size is fixed. In more dynamic networks like Flexgrid networks, we also describe them as widths of frequency intervals.

Lightpath configuration

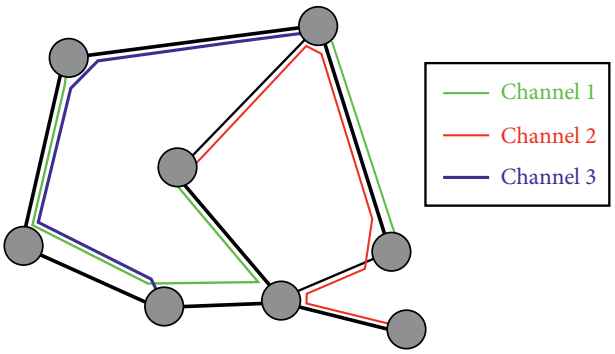


Figure 1.1.1: Lightpaths in a network

[1] Regenerator devices may also be placed on the links.

Connections in optical networks are realized by lightpaths, compare also Figure 1.1.1 on the previous page. The routing of the lightpaths determines the paths in the physical network. If we use optical technology that employs different wavelengths, a wavelength assignment has to be carried out as well, that is, a transmission wavelength needs to be assigned to each lightpath.

If the hardware configuration is fixed, the establishment of lightpaths is a pure software task since the hardware components can be reconfigured remotely. The problem of routing lightpaths through the network is called the *traffic engineering* part of the network planning problem.

1.1.2 Hierarchical network structure

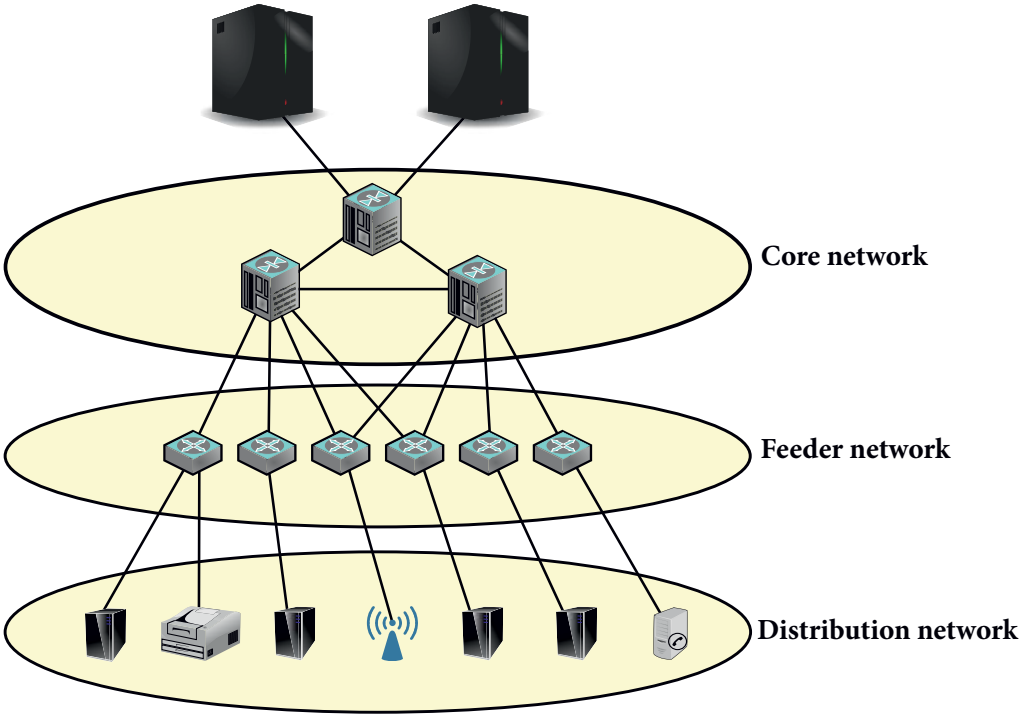


Figure 1.1.2: Hierarchical structure of a network

Network layers

Optical networks usually have a hierarchical structure, see also the example in Figure 1.1.2. Subnetworks at certain geographical regions constitute partially autonomous networks and are interconnected to other subnetworks at certain access points. From the lower to higher levels, more and more traffic is aggregated. The network with the highest level of aggregation is called the *core* or *backbone* network. Because highly aggregated traffic in telecommunication networks typically needs very high bandwidth capacities, the core networks especially tend to be constructed using optical technology. The network layer with the lowest aggregation levels is called the *access* network. Because of the technical breakthroughs in the last decades, optical technology is now available and cheap enough to be also used in access networks. The business acronyms for these technologies include Fiber-to-the-Home (FTTH), Fiber-to-the-Building

(FTTB) and Fiber-to-the-Curb / Cabinet (FTTC), depending on where the optical connection is terminated. From this termination point, some other technology is employed to connect to the customer on this so-called "last mile".

Virtual network layers

In telecommunication networks, a link generally corresponds to some kind of connection between nodes, for instance optical fibers or copper cables. This generally means some cables buried in the ground, so connections cannot be changed quickly or cheaply. To be able to provide computing networks with arbitrary topology quickly and flexibly, *network virtualization* introduces a virtual layer into the network.

This virtual layer corresponds to an overlay network on top of existing physical hardware. This reduces cost, because advanced network functionality can be moved to the overlay network, while using off-the-shelf switches and routers in the underlying physical network. These software-based solutions are typically much more portable and cost-effective than hardware-based implementations.

Using this construction, links in the overlay layer could correspond to paths in the physical network, for example. If a physical node providing the resources for a virtual node fails, the network might be reconfigured to simply use another physical node without propagating the failure to the overlay network, which leads to more fault-tolerant networks.

1.1.3 Optical access networks

In a typical access network architecture, the connections originate at one end from the Central Offices (COs) where optical transmitters are placed. The connections pass along optical fibers and through intermediate nodes we call *Distribution Points*, where optical splitters and also other devices may be installed. These networks typically use a Point-to-Multipoint (P2MP) architecture. The termination points at the other end of the network are called the *customers*.

We distinguish between two types of optical access networks, the *Active* and the *Passive* Optical Access Networks.

Active Optical Networks (AONs) Active Optical Networks require optical-electro-optical conversion and Medium Access Control (MAC) switching in the distribution points. Their real-world use is quite limited, as they have higher operating expenses due their active use of electricity than their passive cousins. From a lightpath point-of-view, they are based on single-wavelength point-to-point links.

AONs can also be used without distribution points by deploying fiber links from the CO to each customer. As no part of the physical infrastructure is then reused for other connections, this is a very costly approach. While AONs are technologically feasible, the usage of Passive Optical Networks (PONs) is usually more cost-effective.

Passive Optical Networks (PONs) In contrast to the previously described AONs, no active components are used in the distribution points of PONs, so there is no need for electricity at intermediate points of the network. Therefore, they create less operational expenditure.

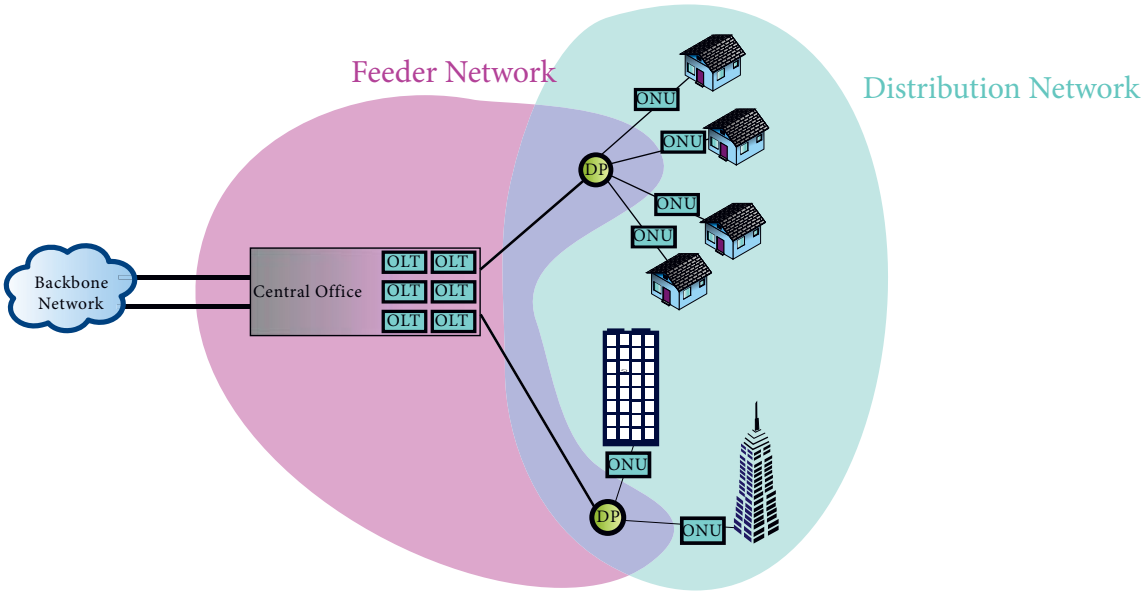


Figure 1.1.3: A passive optical network (PON)

In a typical PON architecture, an Optical Line terminal (OLT) at the Central Office (CO) is connected to several Optical Networks Units (ONUs) located at the customer end of the network. This is accomplished by splitting the lightpath at some Distribution Points (DPs). The connection between Optical Line Terminals and Distribution Points is called the Optical Feeder Network (OFN), while the connection between the Distribution Points and the Optical Network Units is called the Optical Distribution Network (ODN) or simply the Distribution Network. In the Central Office, each OLT services the OFN. To avoid collisions by frames sent by different ONUs in the Distribution Network at the same time, a protocol needs to be established to only allow one ONU to transmit, while the others have to wait for its transmission to complete.

Since the great advantage of PONs is the reuse of architecture, signals for and from each user need to be combined through multiplexing techniques in downstream and multiple access techniques in upstream. Time-Division Multiplexing (TDM) and Time-Division Multiple Access (TDMA) are the most commonly adopted solutions for these tasks. Wavelength-Division Multiplexing (WDM) accomplishes conflict avoidance by using different wavelengths, but also creates the need for more expensive hardware, as we will see later.

1.1.4 Collision avoidance

We now go into more detail concerning the collision avoidance techniques. While presented here for access networks, these techniqes are also employed in the other network layers. We concentrate on the most common approaches and thus distinguish between *Time-Division Multiplexing* and *Wavelength-Division Multiplexing*. While the former is based on sharing the available transmission time, the latter is based on sharing the available wavelength spectrum.

Time-Division multiplexing (TDM) This method of multiplexing was originally used for telegraphy and then employed for digital telephony, but is now also a prevalent way of sharing

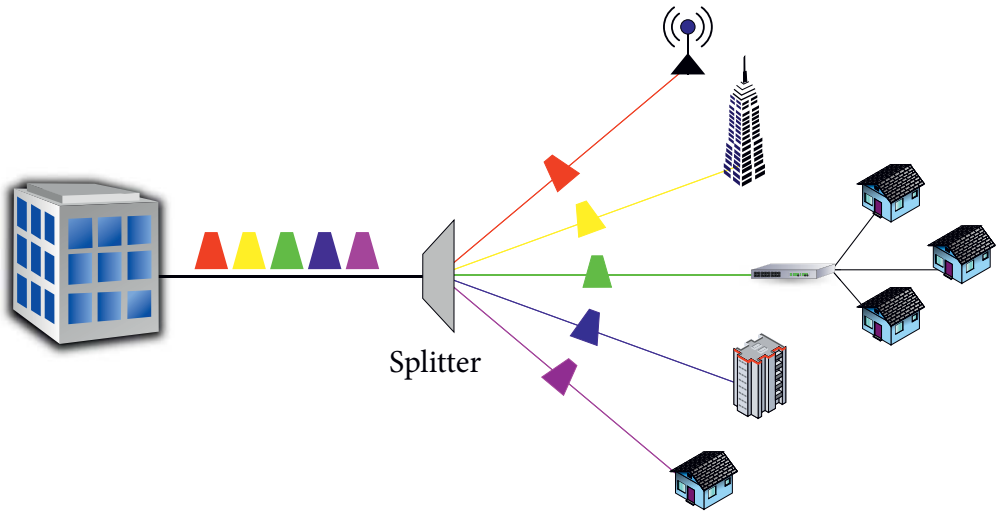


Figure 1.1.4: Standard WDM-PON architecture

an optical fiber between several parties. In TDM networks, time is divided into periodic time slots of fixed length, one for each transmission channel. For each channel, there is some time slot at which information for this channel can be transmitted.

On the upstream connections, Time-Division Multiple-Access (TDMA) is employed, which is a similar technology, but the signals come from different senders, multiple customers in our case. With *synchronous* TDMA, each sender gets a fixed time slot – more flexible approaches are called *asynchronous*. One asynchronous method, *Dynamic TDMA*, uses a scheduling algorithm that reserves a variable number of time slots in each transmission frame depending on the current traffic demand.

Wavelength-division multiplexing (WDM) In WDM networks, several optical signals may be transmitted on the same fiber. Conflict is avoided by using different wavelengths of light.

This method is very popular because it allows the network provider to increase the capacity without making changes to the fibers. This can be carried out simply by using better multiplexing hardware. On the other hand, additional costs are incurred by expensive equipment like tunable lasers which allow for transmission at varying frequencies of light.

As mentioned, an important advantage of PONs is that by splitting, less parallel fibers are needed to connect different customers; fibers can be reused, as long as the signals are not in conflict with each other. In WDM-PONs, these signals might use different wavelengths and could then be transmitted simultaneously on the same fiber. This also decreases the amount of optical fibers needed to achieve the same bandwidth capabilities compared to TDM-PONs.

1.2 Technical Background

In the first part of this section, we explain the physical basics of optical networks. We discuss some key hardware components used in optical telecommunication systems and identify the principle physical limits of optical communication. For further details and more in-depth

explanations of the used technology and physical phenomena, we recommend the books by Mukherjee [2006] and Kazovsky [2011].

A prototypical optical network In a fiber-optical network, light waves are used to convey information between endpoints. The simplest prototype of a fiber-optical network, compare also Figure 1.2.1, consists of three components: an optical transmitter to convert some information-carrying electrical signal into an optical one, an optical fiber through which the signal is then sent and an optical receiver at the other end of the fiber to reconvert the signal to an electrical one. In this way, the information can be recovered after the transmission.

In this prototypical network, the optical transmitter and receiver are responsible for the conversion between the electrical and the optical signal, while the fiber-optic cable acts as an optical waveguide, in which the light is trapped and forced to move along the cable. We now explain the physical background of optical waveguides.

Optical reflection The *refractive index* of a material is defined as

$$n = \frac{c}{v},$$

where c is the speed of light in a vacuum and v is the *phase velocity* of light in the material, which is a material constant. The *phase velocity* is given in terms of the wavelength λ and period T as $v_P = \frac{\lambda}{T}$. When a light wave reaches the boundary between two materials with different refractive indices, a part of the wave will be refracted, while another part will be reflected. The angle of refraction is determined by *Snell's law*, which states that if θ_1, θ_2 denote the angles measured from the normal directions of the surface boundary and n_1, n_2 denote the respective refractive indices of the materials, we have that

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}.$$

If $n_1 \gg n_2$ and $\sin\theta_1$ is close to one, we have

$$\frac{n_1}{n_2}\sin\theta_1 = \sin\theta_2 > 1$$

which cannot be fulfilled. In that case, a phenomenon called *Total Internal Reflection* occurs, in which the signal is not refracted, but totally reflected back into the original medium. The angle

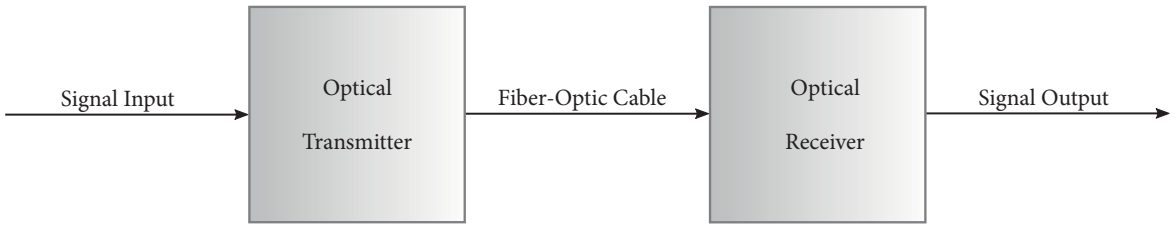


Figure 1.2.1: A prototypical optical network

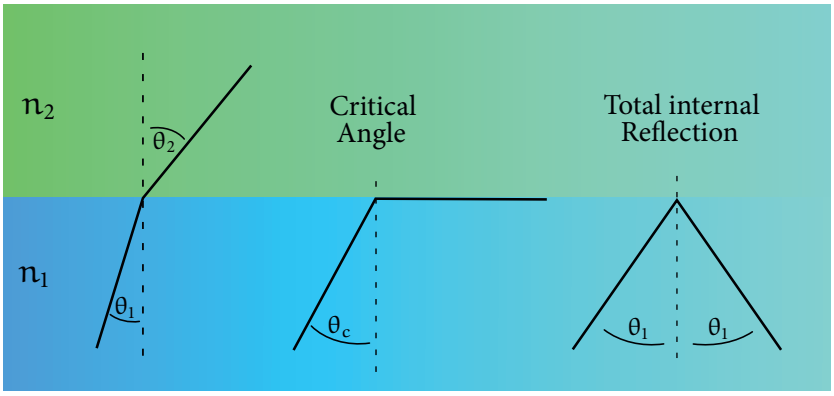


Figure 1.2.2: Optical reflection phenomena

at which this begins to occur is called the *critical angle* and can be found as $\theta_c = \arcsin \frac{n_2}{n_1}$. An illustration of these phenomena can be found in Figure 1.2.2.

Optical fibers work by utilizing this phenomenon to guide the light inside themselves.

Encoding information on optical signals

While a regular sine wave does not carry information, it can be modified or more precisely *modulated* to encode digital signals. There are several basic methods to encode information on electromagnetic waves.

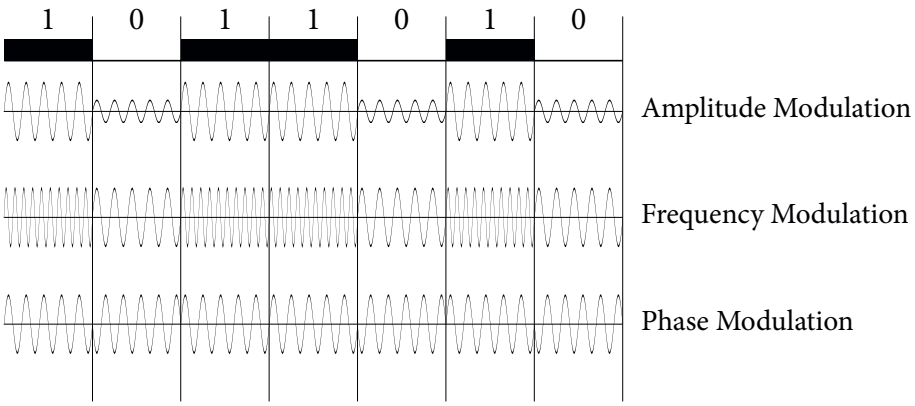


Figure 1.2.3: Signal modulation methods

In principle, modulation can be imposed on the phase, frequency, amplitude or polarization of the light beam, but most commonly, phase modulation is used. Amplitude modulation can be created via phase modulation with a device called a Mach-Zehnder interferometer, where the beam is split into two beams, one of them is phase-modulated and the beams are then recombined. By controlling the phase of the phase-modulated beam, interference will happen and can be constructive or destructive, thereby controlling the amplitude.

We do not go into any detail concerning these methods, but the ideas of these methods can be seen in a basic form in Figure 1.2.3 on the previous page.

1.2.1 Hardware components

We now introduce the main hardware components used to construct optical networks.

Optical fibers

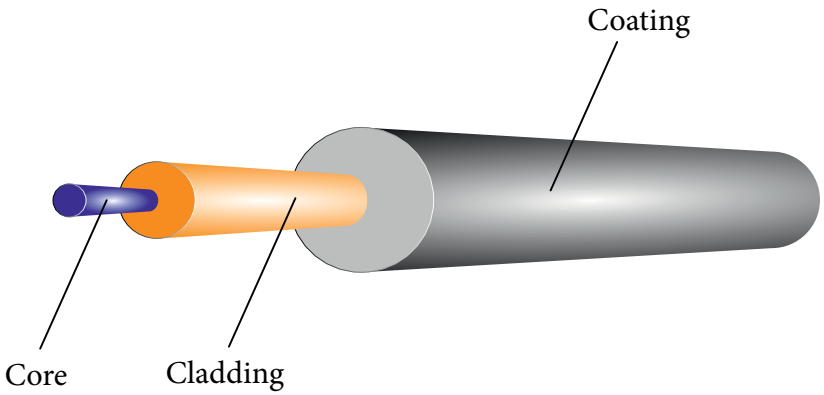


Figure 1.2.4: Cross section of an optical fiber

An optical fiber consists of a transparent core surrounded by a dielectric cladding material that is itself surrounded by a polymer buffer coating for protective purposes, compare Figure 1.2.4. The core and the cladding are most commonly made of silica glass, but are also available as plastics or a combination of both materials. Plastic fibers usually lead to higher attenuation compared to glass fibers and therefore limit the possible transmission distance. In contrast, silica fibers exhibit low attenuation over a wide range of wavelengths. Advantages of silica also include that they are relatively easy to splice and that they offer more resistance against pulling and bending than plastic fibers, making them more resilient and thus easier to place.

To trap the light waves inside the core and make the fiber act as a waveguide, a material with a lower refractive index than the one of the core is used for the cladding, utilizing the *Total Internal Reflection* phenomenon described earlier.

We distinguish between two main types of optical fibers, *Single-Mode Fibers* and *Multi-Mode Fibers*. We now explain their main characteristica.

Single-Mode Fibers have a diameter comparable to the wave length of an optical signal. The most common type has a core diameter between $8 - 10$ micrometers and is used in the near-infrared part of the spectrum. It supports only one optical signal at a time, but keeps the signal intact over a longer distance than a multi-mode fiber can. While the equipment needed to use single-mode fibers is more expensive than the equipment used for multi-mode fibers, the fibers themselves are cheaper to manufacture.

Due to less severe distortion phenomena, they support higher bandwidths and because of that, they are usually used for wide-area and metropolitan-area networks. Passive optical networks also employ them, achieving high data rates and long-distance transmission capabilities.

Multi-Mode Fibers are usually found in local area networks and have the advantage that they can carry many modes of light simultaneously. In this context, a *mode* of a a signal refers to the way it takes through the fiber. While a single-mode fiber is relatively narrow and can be approximated as a one-dimensional guide, a multi-mode fiber is much thicker and therefore light rays can take a rather direct route through the fiber or zigzag off the cladding that has a different refractive index. The transition between the core and the cladding is either realized as a step-index profile or a graded-index profile. *Step-index profiles* mean uniform refractive index in the core and a sharp decrease in the cladding. They are more common in single-mode fibers. In contrast, *Graded-index profiles* are usually used for multi-mode fibers. This implies that the parts of the core that are closer to the fiber axis have a higher refractive index than the parts near the cladding. Most commonly, a nearly parabolic index profile is created that decreases modal dispersion and continuously refocuses rays.

Optical transmitters

Optical transmitters convert electrical signals carrying information into optical signals. A key component of these devices is a light source, most commonly a semiconductor laser. There are two ways of encoding information in the light signal. One variant is to use the laser as a continuous laser, on which an optical modular encodes a signal. A cheaper alternative is to direcly modulate the light source.

Optical modulators exploit electro-optic effects, by which the optical properties of a material can be changed by the application of an electric field. The changes in the optical properties are caused by forces resulting from the electric field that change the position, shape or orientation of molecules inside the modulator material. A very important effect in this context is the so-called *Pockels effect* or *Linear Electro-Optic Effect*. In crystals exhibiting this effect, the refractive index can be modified in a way that is proportional to the strength of the electric field. These materials then exhibit *Birefringence*, where the refractive index depends on the polarization and direction of light.

A widely used optical modulator type uses a lithium niobate crystal. As described, the refractive index can be changed by changing the strength of the electrical field – using a stronger field will make light travel slower through the crystal. As the phase of the light leaving the crystal is determined by the length of time it takes the light to pass the crystal, this can be used to create phase modulation.

Optical receivers

After the optical transmitter has performed an electro-optic conversion and the signal has travelled through the network, the task of the *Optical Receiver* is to convert the optical signal back into an electrical one. In that way, the transmitted information can be recovered. The main components of an optical receiver are a photodetector that generates an electrical current that is proportional to the optical power, several amplifiers and an electrical circuit that recovers the information.

Photodetectors absorb photons and generate electrical current proportional to the power of the optical signal. In optical systems, this job is usually executed by *photodiodes*. They can be classified into PN and PIN diodes.

Photodiodes are like regular semiconductor diodes in the sense that they conduct current in only one direction and offer a high resistance against conducting into the other direction. The difference between photodiodes and regular semiconductor diodes is that the sensitive area of the diode is exposed in such a way that light can reach it.

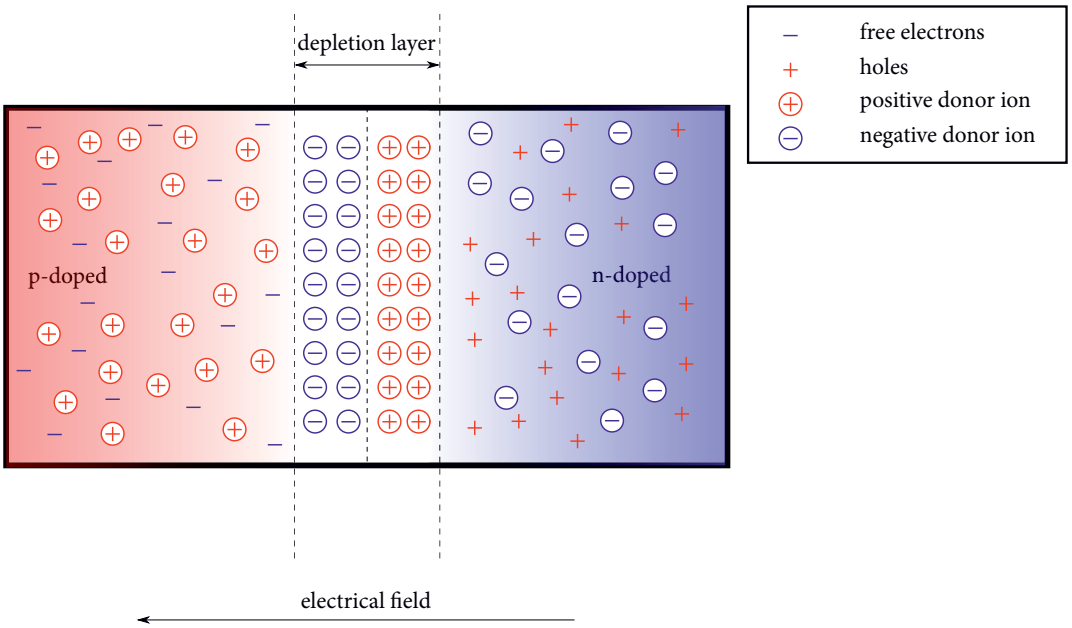


Figure 1.2.5: Schematic view of a p-n junction in an equilibrium state

Between resistant materials and insulators, there is a class of conducting materials called *semiconductors*, which can be crystalline or amorphous solids. They have the useful property that their conducting properties can be modified by introducing impurities into their highly pure base material, a process called *doping*.

The properties of semiconductors are explained using the movement of *charge carriers* inside a crystal lattice. Charge carriers can be electrons or the absence of an electron, where one could exist inside an atom or atomic lattice. The absense of an electron is called a *hole*, so there are two types of charge carriers in a semiconductor, electrons and holes.

The doping process increases the number of charge carriers. A semiconductor is said to be of *p-type* if its hole concentration is larger than its electron concentration, otherwise it is said to be of *n-type*.

The boundary area between a p-type and a n-type semiconductor is called a *p-n junction*. PN diodes are photodiodes based on p-n junctions, compare also Figure 1.2.5.

If no external voltage is applied, a p-n junction goes into an equilibrium state that exhibits a potential difference across the junction. This happens by the diffusion of electrons from the n-region into the p-region forming negatively charged ions in the p-region, while leaving behind positively charged ions in the n-region. This leaves no free charge carriers in the area of the junction, making it non-conductive. The area of non-conductivity is called the “depletion layer”.

By the resulting potential difference, an electric field is created, counteracting the diffusion movement, until an equilibrium state is reached.

By applying a current that is positive at the p-region and negative at the n-region, the non-conductive layer will be minimized. If we apply a current in the other direction, it will be enhanced, prohibiting the flow of current. This way, we have a diode phenomenon, where electrical current can only flow in one direction.

When a photon with sufficient energy arrives in the depletion layer or near it, it creates an electron-hole pair by the photoelectric effect. The hole moves to the anode and the electron to the cathode, producing a photo current.

While PN diodes are based on p-n junctions, *PIN diodes* use a similar construction known as a *PIN junction*. These junctions consist of an undoped semiconductor layer sandwiched in the center of a p-n junction. The advantage of this construction is a higher sensitivity to incoming photons, as the created depletion area is larger. This construction also strengthens the resulting electrical field, making PIN diodes operate much faster than PN diodes.

For long-haul communication systems, another type of diode is used, a so-called *avalanche photodiode*. They have the advantage of providing a larger photocurrent by themselves, because a certain amount of amplification is already taking place inside the diode.

Optical amplifiers and regenerators

Due to fiber loss, see also Section 1.2.2, the optical signal loses in strength during propagation. To successfully recover the signal after its journey, the optical power needs to exceed the receiver sensitivity.

A simple way of strengthening the signal at intermediate points is to convert it back to electrical form, amplify it electronically and then convert it back to an optical signal. With digital communication, this can remove all noise and distortion, but is expensive and introduces at least some amount of delay. The device used for this purpose is called an *Optical communications repeater* or simply an *OEO* (optical-electrical-optical). These devices can only be used for one wavelength, so in WDM systems, one device per wavelength has to be employed.

A more cost-efficient means of boosting the signal power without conversion is an *Optical Amplifier*. It can be located at an arbitrary point in the transmission path: either somewhere along the path or directly after the transmitter to boost its power or even as a preamplifier in front of the receiver. It is completely transparent to the data format and can be used to amplify several signals on different wavelengths simultaneously.

The most common kind of optical amplifiers are so-called *Doped Fiber Amplifiers (DFAs)*, one prominent example of them being *Erbium-Doped Fiber Amplifiers (EDFAs)*. In these devices, a pump laser and the signal are multiplexed into a doped fiber. The pump laser excites the doping ions into a higher energy, from which they decay via stimulated emission of photons at the signal wavelength. Characteristics of optical amplifiers are determined by the dopants. In an EDFA, the core of the silica fibre is doped with trivalent erbium ions.

As the signals weakens, it becomes more susceptible to noise, which is also amplified every time the signal is amplified. This sets a practical limit on the distance that can be reached using optical amplifiers only without OEO conversion.

Fiber-optic couplers / Optical splitters

A *Fiber-Optic Splitter* is used to spread an input signal into several output signals. In most common applications, apart from monitoring purposes, the signal power will be spread evenly onto the output fibers.

Splitting and Coupling devices function in both directions, meaning that in the reversed direction, an optical signal entering the common port will also appear on each of the branches with its signal strength divided by the number of branches. One then usually talks about *Fiber-Optic Splitters*. These are mainly used in Passive Optical Networks, connecting the distribution network cable and the drop wire to the customer premises.

We can differentiate between two different architectures for these devices, the *Fused Biconical Taper (FBT)* and the Planar Lightwave Circuit (PLC).

Fused Biconical Taper This is the most common architecture for splitters, but is also the older architecture. To construct these devices, two or more fibers are placed next to each other and then thermally fused. While this is already a highly mature technology and is quite cost-effective, PLC splitters are more accurate in the splitting ratio and exhibit even smaller losses. Also, FBT splitters are sensitive to wavelengths, which presents one of their most significant disadvantages. To achieve different splitting ratios, these splitters can be cascaded.

Planar Lightwave Circuit While more expensive to produce, the PLC splitters offer different splitting ratios more easily. These devices are constructed by using lithographic techniques on a silica glass substrate, which allows accurately selecting percentages of light that will appear on the waveguides. In that way, especially higher splitting ratios can be constructed more easily. They are also less sensitive to temperature fluctuations.

Power Losses In the downstream direction, the cost of doubling the split ratio is a 3 decibel loss in power, equalling roughly a halving of the input signal strength. Additional losses can occur due to the imperfect construction of splitter devices.

Bandpass filters

An optical filter is basically a bandpass filter through which only certain wavelengths can pass, while other wavelengths are rejected or at least heavily reduced in signal strength. Filters can be used in WDM systems to separate signals with different wavelengths as there might be unwanted signals on the fiber resulting from the splitting process. They are also used to suppress noise.

A common variant of an optical filter is called an *Etalon* or *Fabry-Péron interferometer*. It consists of two parallel highly reflective mirrors. An optical signal entering the area between the mirrors will be reflected back and forth. Everytime this happens, some part of the signal makes it through the output mirror. Between the mirrors, interference occurs. If the beams are out of phase, destructive interference occurs, destroying unwanted parts of the signal.

1.2.2 Physical limits of optical networks

Due to imperfections in manufacturing and other kinds of defects and necessities, there are some limitations as to the distance and bandwidth available using optical fibers. We discuss some of the limiting factors in this section.

Fiber loss

An optical signal in a network is attenuated in several ways throughout its journey. Although it takes a very long distance for the signal to disappear completely, optical receivers do need a minimum amount of optical power to recover the transmitted information. This fundamentally limits the possible transmission distance in such a network. Even though there are certain methods to mitigate that problem, for example OEOs, these also amplify spontaneous noise. Because the signal strength of an optical signal decays exponentially by attenuation, optical communications were not possible in a modern sense before the introduction of low-attenuation fibers.

Attenuation is caused by three major components: Material absorption, Rayleigh scattering and waveguide imperfections.

Material absorption Material absorption is caused mainly by imperfections in the optical fiber. This includes both fused silica and other impurities such as water and metals. Fused silica leads to strong absorption peaks in the ultraviolet region, while metal impurities lead to absorption in the wavelengths from 0.8 to 1.6μm.

Due to physical reasons, material absorption could not be avoided even in the complete absence of impurities.

Rayleigh Scattering As the propagation of light through the core of an optical fiber is based on total internal reflection, irregular surfaces can cause light rays to be reflected in random directions. This is called scattering. Variations in the refractive index of the fiber material caused by micro-structural defects cause Rayleigh-type scattering. It behaves inversely proportional to the fourth power of the optical wavelength and is therefore a much more significant factor for shorter wavelengths. This is one reason why infrared light is often used in optical networks.

Waveguide imperfections Radiative losses occur due to waveguide imperfections like bending of the fiber and variations in the core radius. While an optical fiber is ideally a perfect cylinder with constant core radius, this is not the case in practice. Small bends lead to a part of the optical signal being scattered into the cladding layer.

Fiber dispersion

Apart from the loss of signal strength, there are other phenomena limiting the possible transmission bandwidth at long distances. While a signal might still be strong enough to be picked up by the receiver, it might be unrecoverable due to a phenomenon called *Fiber dispersion*. This describes the effect of parts of the optical signal traveling at different speeds. In a multi-mode

fiber, this can mean the different modes, but it also happens in single-mode fibers, where different frequency components travel at different speeds. Also, different polarizations of light can exhibit different speeds due to non-uniformity of the fiber.

Modal dispersion Modal dispersion is significant mainly in multi-mode fibers. The fiber can carry many light waves simultaneously, but they enter the fiber with different angles respective to the fiber axis. If the angle is shallower, the path to travel is shorter. In this way, different components of the signal arrive at different times. This effect is structurally similar to multipath propagation in a radio signal.

Intramodal dispersion In multi-mode fibers, the phenomenon of intramodal dispersion does not play a significant role, as the effects created by modal dispersion far outweigh it. But in single-mode fibers, modal dispersion is effectively eliminated and intramodal dispersion becomes significant. This is the phenomenon of different frequency components of a single optical signal travelling at different speeds. This has two causes and can therefore be classified into two different subphenomena: *Material dispersion* is caused by the frequency-dependence of the speed of light in a medium. *Waveguide dispersion* is caused by differences in the refractive index of the core and cladding of the fiber, which in turn leads to different speeds of the signal in the core and in the cladding of the fiber.

Polarization Mode Dispersion In an ideal fiber, signals with orthogonal polarization follow the same propagation path. However, due to imperfections in the fiber, the mode indices and propagation constants can exhibit a slight difference. This phenomenon is known as polarization mode dispersion. It is relatively weak compared to intramodal dispersion, but can become significant in high-speed systems over a long distance, especially at wavelengths where the other dispersion phenomena are minimized.

Nonlinear effects

Even though silica is not a highly nonlinear medium, nonlinear effects can be observed even at low power levels in single-mode fibers. We distinguish between nonlinear scattering effects and nonlinear refraction effects.

Because these effects are usually outweighted by other physical effects, we will skip a detailed physical description of these effects and they will also be ignored in our models.

1.3 Mathematical background

In this section, we review the basic definitions and concepts in linear algebra, graph theory, computational complexity and mathematical programming used throughout the thesis. This section is not meant as an introduction to these concepts—we assume familiarity with the presented topics. It is rather meant as a reference for definitions and notations in the later chapters.

For an introduction to linear algebra and integer linear programming, we recommend the books of Bertsimas and Tsitsiklis [1997] and Grötschel et al. [1993]. The concepts of graph theory are can be found for example in Diestel [2010]. For the discussed topics of computational complexity theory, we recommend the books by Papadimitriou [1993] and Arora and Barak [2009]. Finally, on the topic of approximation algorithms for computationally hard problems, we recommend the book by Vazirani [2003].

1.3.1 Linear algebra

Numbers

We write $\mathbb{N}$ and $\mathbb{N}_0$ to denote the set of natural numbers excluding or including the number zero. The sets of integer, rational and real numbers will be denoted by $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$, respectively. The largest natural number n with $n \leqslant x$ will be written as $\lfloor x \rfloor$, while $\lceil x \rceil$ denotes the smallest natural number n with $n \geqslant x$. For $x \in \mathbb{N}_0$, we therefore have $\lceil x \rceil = \lfloor x \rfloor$.

Vectors

Let $\mathbb{F}$ be a base set and E a finite index set. By $\mathbb{F}^E$ we denote the set of vectors consisting of $|E|$ components with values in $\mathbb{F}$. For $E = [n] := \{1, \dots, n\}$, we simply write $\mathbb{F}^n$. Given a set $S \subseteq E$, we define the vector $\chi^S \in \{0,1\}^E$ with

$$\chi^S_f = \begin{cases} 1 & f \in S \\ 0 & f \in E \setminus S \end{cases}$$

and call χ^S the *characteristic vector* of the subset S.

The notation $\{0,1\}^*$ is used for the set of finite tuples whose elements are all 0 or 1. This set is also called the set of *binary strings*.

Operations on Vectors

By default, vectors will be considered as column vectors. We use the superscript T to denote the transposed vector. For any finite index set E, the sets $\mathbb{R}^E$ and $\mathbb{Q}^E$ naturally possess the structure of a vector space over their respective fields. We say that a vector $v \in \mathbb{R}^E$ is a *linear combination* of the vectors $x_1, \dots, x_k$ if there are coefficients $\lambda = (\lambda_1, \dots, \lambda_k)^T, \lambda_i \in \mathbb{R}$ such that $v = \lambda^T x$. If all λ_i are nonnegative, we have a *conic combination*. If the sum of the coefficients is equal to 1, we have an *affine combination*. A combination that is both conic and affine is called a *convex combination*.

A set $X \subseteq \mathbb{R}^E$ is called *linearly (affinely) independent* if none of its elements can be written as a linear (affine) combination of the other elements.

The *linear (resp. affine) rank* of a set $X \subseteq \mathbb{R}^E$, denoted by $\operatorname{rank}(X)$ resp. $\operatorname{arank}(X)$, is the maximum number of linearly (resp. affinely) independent vectors in X.

We define the *dimension* $\dim(X)$ of a set $X \subseteq \mathbb{R}^E$ as

$$\dim(X) := \operatorname{arank}(X) - 1.$$

If $\dim(X) = |E|$, X is called *full-dimensional*.

1.3.2 Graph theory

A pair $G = (V, E)$ of finite sets V, E with $E \subseteq V^2$ and $E \cap \{ (v_i, v_i) \mid v_i \in V \} = \emptyset$ is called a *simple directed graph* on V or a *simple digraph*. By default, we assume digraphs are simple and just talk about *digraphs*. If the set E is symmetric in the sense that for all $(u, v) \in E$, we also have $(v, u) \in E$, G is called *undirected* or simply a *graph*. In that case, elements of E will also be denoted as $\{u, v\}$.

The elements of V are called nodes, the elements of E edges. The two nodes an edge consists of are called its *endpoints*. In digraphs, we use the terminology of *arcs* instead of edges and if $a = (u, v)$ is an arc, we call the node u its *source* and the node v its *target*. For a fixed node v, we use the notation $\delta^{\text{out}}(v)$ to denote the arcs that have v as their source and $\delta^{\text{in}}(v)$ for the arcs that have v as their target.

Concerning the number of nodes and edges, we usually use the notation $n := |V|$ and $m := |E|$. Vertices and edges are by default denoted by lowercase letters, while sets are denoted by capital letters.

$G' = (V', E')$ is called a *subgraph* of $G = (V, E)$, if G' is a graph and $V' \subseteq V$, $E' \subseteq E$. We will write $G' \subseteq G$ to say that G' is a subgraph of G. G' is called an *induced subgraph*, if $E' = E \cap (V')^2$, that is, G' contains exactly those edges whose endpoints lie in V'.

A node v is called *incident* to an edge $e = \{w_1, w_2\}$ if $v = w_1$ or $v = w_2$. Two nodes $u, v \in V$ are called *adjacent* if there is an edge that has both u and v as endpoints. The set of nodes adjacent to a node $v \in V$ is called the *neighborhood* of v and is denoted by Γ_v^*. We set $\Gamma_v := \Gamma_v^* \cup \{v\}$.

A *path* in G is a tuple $(v_0, v_1, \ldots, v_k)$ with $(v_j, v_{j+1}) \in E$ for all $j \in \{0, \ldots, k-1\}$. The path *begins* at v_0 and *ends* at v_k. It is called *simple* if no nodes appear twice in the path. The number k is called the *length* of the path.

A *cycle* C in G is a path whose end node is the same as its beginning node. It is called *simple* if no nodes are repeated, except for the beginning and end node. The *length* of a simple cycle is the number of its edges, or equivalently, the number of its nodes.

A graph is called *connected* if for each pair of nodes $u, v \in V, u \neq v$, there is a path beginning at u and ending at v. Furthermore, a graph is called d-*connected*, if the removal of any $d-1$ nodes and their adjacent edges from the graph always leaves a connected graph.

1.3.3 Computational complexity

Turing machines To speak of the runtime of algorithms and the complexity of problems, we need to formally define these concepts. We begin with the definition of a *Turing machine*, which is a basic computing device that we use to execute prespecified instructions.

The type of Turing machine we consider here is a *three-tape Turing machine*. While there are many variants of Turing machines we could consider, we now concentrate on this one. Later, we will see that the precise variant of Turing machine we consider does not matter, as they still yield the same complexity classes.

We begin with an informal description. A three-tape Turing machine consists of three tapes, a state register, tape heads and a transition function. The three tapes are the *input tape*, *working tape* and *output tape*, see also Figure 1.3.1 on the following page. A tape consists infinitely many cells lying next to each other.

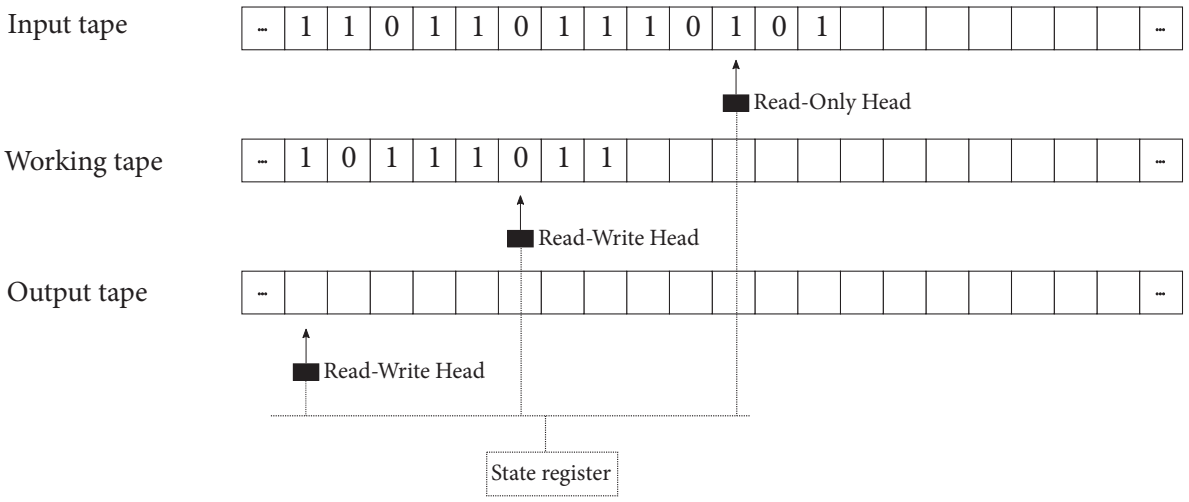


Figure 1.3.1: A three-tape Turing machine

Each cell on the tapes contains a symbol from some finite set which we call the *alphabet* of the Turing machine and denote by Γ. The alphabet contains at least one special symbol which we call the *blank symbol* and which we denote by b. The working tape and output tape will be filled with the blank symbol at all positions at the start of a computation, while the input tape has a finite segment where non-blank symbols are allowed and which we call the *input* to the computation.

Additionally, the Turing machine has *heads* which it uses for both writing to and reading from the tapes. While the input tape is assumed to be read-only, both the working and output tape can be written to. It also has a *state register* which stores the internal state of the Turing machine. There are only finitely many states the machine can be in, among them two special states. These are the state q_{start} which is the state the machine is in at the beginning of its computation and the state q_{halt} which means that the machine has terminated its computation process. We will denote the set of possible states by Q.

The computation is guided by the so-called *transition function* δ which takes as input the symbols currently on the tapes at the head positions as well as the internal state of the machine specified in its state register. It specifies the symbol the heads will write on the tape at the current position, how the heads should move on the tape ("left", "stay" or "right") and what the next internal state of the machine will be. While other machine models like Random Access Machines may allow accessing data in an arbitrary order, the Turing machine only allows each head to move by at most one cell at every executed instruction.

We now give a formal definition of the previous concepts. A Turing machine is given by a tuple $M = (\Gamma, Q, \delta)$ consisting of the alphabet set Γ, the state set Q and the transition function δ. We assume that Γ is a set of at least one element, that Q contains at least the states q_{start} and q_{halt} and the transition function δ is a function

$$\delta\colon Q \times \Gamma^3 \to Q \times \Gamma^2 \times \{L, S, R\}^3.$$

Any computational device acting to these rules is called a (three-tape) Turing machine.

Computing functions We now define what it means for a Turing machines to *compute* a function.

Let $f\colon \{0,1\}^* \to \{0,1\}^*$ and $T\colon \mathbb{N} \to \mathbb{N}$ be functions and M be a Turing machine. If M is initialized in state q_{start} and the tape head of the input tape is at the beginning of some string $x \in \{0,1\}^*$, assume that the Turing machine halts with $f(x)$ written on its output tape. If that property is true for every $x \in \{0,1\}^*$, we say that M *computes* f.

We also say that M *computes* f *in* $T(n)$*-time* if its computation on every input x requires at most $T(|x|)$ steps, where $|x|$ denotes the length of the input string x.

Universal Turing machine As we mentioned earlier, we could have used a different variant of a Turing machine. We could for example argue that three tapes are not really necessary, because intermediate results of a computation could also be saved on the input tape if it used a Read-Write Head, eliminating the need for the working tape.

Well-known results in computational complexity theory show that the standard complexity classes, which we begin to define in the next paragraph, are very robust against changes in the exact computing machines used to define them. One very important idea in these theorems is called the *simulation argument*, which is the following. For any Turing machine, we can write down a description of its inner workings on paper. This description can then be encoded using a binary string. We then can construct a so-called *Universal* Turing machine in the sense that given this binary description of another Turing machine, it can *simulate* the execution of any other Turing machine, taking not much longer for the computation than the original machine; see e.g. Arora and Barak [2009] for more details.

Decision problems Different functions can be grouped according to the time it takes to compute them using Turing machines. We call these groups *complexity classes*. For the purpose of their definition, we pay special attention to *boolean* functions, by which we mean functions having only one bit of output.

Many interesting computational questions can be reduced to whether for some input, which can always be assumed to be a binary string, a certain boolean function f has the value 0 or 1. In this way, functions $f\colon \{0,1\}^* \to \{0,1\}$ define *decision problems* or *languages* $L \subseteq \{0,1\}^*$ by $x \in L :\Leftrightarrow f(x) = 1$. We say that a Turing machine *decides* a decision problem if it computes the corresponding function f.

We define a language or decision problem L to be in $\mathrm{DTIME}(T(n))$ if there is a Turing machine M that decides L and runs in time $cT(n)$ for some constant $c > 0$.

For any fixed n, the *worst-case running time* of a Turing machine M is the longest time it runs on an input of size n. We use the worst-case running time of Turing machines and later of algorithms to compare the efficiencies of different approaches. We define a *polynomial-time Turing machine* as a Turing machine with a polynomial worst-case running time.

The class P We define $\mathbf{P}$, the class of all problems decidable in polynomial time, as

$$\mathbf{P} := \bigcup_{c \geqslant 1} \mathrm{DTIME}(n^c).$$

While it is not immediately clear that our definition of **P** does not depend on the choice of Turing machine, it can be shown that for all known standard models of computation, including other types of Turing machines, Random-Access Machines or the λ-calculus, polynomial is well-defined as it means the same for each of them.

In fact, the famous *Church-Turing thesis* states that every practically realizable computation device can be simulated by a Turing machine. This implies that the set of computable languages is the same. It does not imply that the set of problems that can be decided in polynomial time is the same for any of these machines. The strong form of the Church-Turing thesis even states that the other computation devices can be simulated with only polynomial overhead, but this is rather controversial, because *quantum computers* seem to violate this claim.

The complexity class NP Another important complexity class apart from **P** is the class **NP**. While **P** in some sense contains the problems that can be solved efficiently, **NP** contains the problems which may or may not be solvable in polynomial time, but given a solution to them, its correctness can be verified in polynomial time. The precise definition is as follows:

A language $L \subseteq \{0,1\}^*$ is in **NP** if there exists a polynomial $p\colon \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M, called the *verifier* for L, such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1,$$

where $M(x,u)$ denotes the output of the Turing machine M on its output tape, given x and u as input.

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, we call the binary string u a *certificate* for x with respect to L and M.

We directly observe that $\mathbf{P} \subseteq \mathbf{NP}$ as u can be an empty string and x can be trivially verified, as the problem can be solved in polynomial time.

Nondeterministic computation We can also define the class **NP** using so-called *nondeterministic* Turing machines. The difference is that a nondeterministic Turing machine has two transition functions δ_1 and δ_2 and a special state q_{accept}. When it computes a function, it makes an arbitrary decision at each step about which of the transition functions it uses. We define $M(x)$ to be 1 for an input x and a nondeterministic Turing-machine M if and only if there is some sequence of these arbitrary decisions that makes the machine reach q_{accept}. If every sequence of choices makes M halt without reaching the accept state, we define $M(x)$ to be 0. The class **NP** can then alternatively be defined as the class of languages such that there is a polynomial-time nondeterministic Turing machine deciding the language, which also explains the name of the class **NP**.

NP-hardness and Karp-reductions It turns out that there are some problems in **NP** that are at least as hard as any other problem in **NP**. This can be seen by showing that if these problems can be solved in polynomial time, then so can every other problem in **NP**, because every other problem can be *reduced* to them. These problems are called ***NP-complete***. The formal definition uses *Karp-reductions*, which we will now define.

A language $L \subseteq \{0,1\}^*$ is *polynomial-time Karp-reducible* to a language $L' \subseteq \{0,1\}^*$ if there is a polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that

$$x \in L \Leftrightarrow f(x) \in L' \, \forall x \in \{0,1\}^*.$$

We then write $L \leqslant_p L'$, because this implies that the complexity of L is not higher than the complexity of L'.

A problem L' is said to be ***NP**-hard* if $L \leqslant_p L'$ for every $L \in \mathbf{NP}$. We say that L' is ***NP**-complete* if L' is **NP**-hard and $L' \in \mathbf{NP}$.

As there might be no decision problem in **NP** to which every other problem in **NP** can be reduced, it is not clear by the previous arguments that a NP-complete problem exists. But we will now state such a problem.

First, we need to define what we mean by a *boolean CNF-formula*. We assume that we have some variables u_i that take values in the set $\{0,1\}$. A *literal* is a variable x_i or its negation $\bar{x}_i$. A disjunction of literals is called a *clause*. A *boolean CNF-formula* is then a conjunction of clauses. A boolean CNF-formula is *satisfiable* if there is an assignment of the truth values true and false to its variables such that the whole formula is true.

Fixing some efficient binary coding of boolean CNF-formulae, we denote the language of all satisfiable CNF formulae by *SATISFIABILITY*. The decision problem for this language is then to decide whether some given formulation is satisfiable or not.

One of the most famous results in the area of computational complexity is the following:

Theorem 1.3.1 (Cook-Levin '71)**.** *SATISFIABILITY is **NP**-complete.*

This means that every problem in **NP** can be reduced to the SATISFIABILITY problem. We will only sketch the proof here. We know that any decision problem in **NP** can be solved in polynomial time by a *nondeterministic* Turing machine. Let M be such a machine. For any input, we can construct a boolean CNF-formula whose truth value corresponds to the truth value of the statement "Given this input, the machine M runs correctly, halts and answers yes". Then, the boolean CNF-formula is satisfiable if and only if the machine M runs correctly and gives a positive answer with respect to this specific input.

Concerning the relationship between the classes **P** and **NP**, a major open question in computational complexity is whether these are really different classes or whether **P** equals **NP**. Due to the aforementioned reductions, it would suffice to give a polynomial algorithm for any of the many well-known **NP**-complete problems (see Garey and Johnson [1979]) to show the equality of these classes.

Optimization problems A related class of problems that we consider most of the time in this thesis are *optimization problems*. An ***NP**-optimization problem* Π can be defined, see e.g. Vazirani [2003], as a tuple consisting of

- a set of valid instances D_Π that is recognizable in polynomial time. We assume that all input numbers are rationals and denote for any instance $I \in D_\Pi$ the number of bits needed to write down I by $|I|$, assuming that binary encoding is used for the numbers,

- a set of feasible solutions $S_\Pi(I)$ for each instance $I \in D_\Pi$ with $S_\Pi(I) \neq \emptyset$. We assume that the encoding length of every solution $s \in S_\Pi(I)$ is upper-bounded by a polynomial in $|I|$ and furthermore that there is a polynomial time algorithm that, given a pair (I, s), decides whether $s \in S_\Pi(I)$,
- a polynomial time computable objective function obj_Π that assigns a nonnegative rational number to each pair (I, s) for instances I and $s \in S_\Pi(I)$ that is also called the *cost* of a solution,
- the specification whether the optimization problem is a minimization or maximization problem.

An *optimal solution* for an **NP**-optimization problem is a feasible solution that achieves the best possible objective value according to whether the problem is a minimization or maximization problem. Such a solution will be denoted by $\text{OPTSOL}_\Pi(I)$ or just by OPTSOL, if the context is clear. We denote the best possible solution value by $\text{OPT}_\Pi(I)$ or $\text{OPT}(I)$, respectively.

To each **NP**-optimization problem we can naturally associate a decision version for the purpose of classifying its complexity. The associated decision problem for a minimization problem is the question "Given an instance I and a rational number B, is there a solution $s \in S_\Pi(I)$ with $\text{obj}_\Pi \leqslant B$?". For a maximization problem, the question is constructed in the same way with the relation sign reversed.

A polynomial algorithm for the optimization problem Π can help solve the decision version by computing the cost of an optimal solution and comparing it with B. We call a **NP**-optimization problem ***NP-hard*** if the associated decision problem is **NP**-hard.

As the complexity of computing optimal solutions to **NP**-hard optimization problem is often too high to be feasible for practical purposes, it is useful to consider solutions that are non-optimal, but have some other provable quality guarantee, for example being only some constant factor more expensive than an optimal solution. While no polynomial algorithms are known for **NP**-hard problems, many of them allow polynomial algorithms with a provable approximation guarantee.

Approximation algorithms Let Π be a minimization (resp. maximization) problem and let δ be a map $\delta\colon \mathbb{N} \to \mathbb{Q}_+$ with $\delta \geqslant 1$ (resp. $\delta \leqslant 1$ for maximization). An algorithm $\mathcal{A}$ is said to be a δ*—factor approximation algorithm for* Π if for each instance $I \in D_\Pi$, the algorithm $\mathcal{A}$ produces a feasible solution s for I such that

$$\text{obj}_\Pi(I, s) \leqslant \delta(|I|) \cdot \text{OPT}(I)$$

in the minimization case resp.

$$\text{obj}_\Pi(I, s) \geqslant \delta(|I|) \cdot \text{OPT}(I)$$

in the maximization case and has a worst-case running time bounded by a polynomial in $|I|$ for all instances.

Instead of single approximation algorithms, we can also consider families of algorithms with increasing running time and constantly improving approximation guarantees. Let Π be a

NP-hard optimization problem with objective f_Π. An algorithm $\mathcal{A}$ is an *approximation scheme* for Π if on input (I, ε), where I is an instance of Π and ε is a desired degree of accuracy, it outputs a solution with

$$\mathrm{obj}_\Pi(I, s) \leqslant (1 + \varepsilon) \cdot \mathrm{OPT}$$

in the minimization resp.

$$\mathrm{obj}_\Pi(I, s) \geqslant (1 - \varepsilon) \cdot \mathrm{OPT}$$

in the maximization case. $\mathcal{A}$ is called a *polynomial time approximation scheme* (PTAS), if for each fixed $\varepsilon > 0$, its running is bounded by a polynomial in the size of the instance I.

Note that in the definition of PTAS, the running time of the algorithm may depend arbitrarily on ε. $\mathcal{A}$ is called a *fully polynomial approximation scheme* (FPTAS) if the running time is bounded polynomially in the size of the instance I and in $1/\varepsilon$.

Nonapproximability Not all problems that are **NP**-hard can be approximated equally well. While some problems for example allow a PTAS, other problems cannot be approximated better than a constant depending only on the problem. The latter problems are called *APX-hard*, where APX is the class of problems that allow an approximation algorithm with a constant approximation ratio.

We now introduce the techniques that are used to show that some problem cannot be approximated arbitrarily well unless $\mathbf{P} = \mathbf{NP}$. The basic idea is to construct a reduction from the SATISFIABILITY problem and show that such an approximation algorithm can be used to distinguish between satisfiable and nonsatisfiable instances.

In this sense, a *gap-introducing reduction* from SATISFIABILITY to a minimization problem Π has two parameter functions f and α. Given an instance I of SATISFIABILITY, it works in polynomial time to give an instance x of Π such that

- if ϕ is satisfiable, $\mathrm{OPT}(x) \leqslant f(x)$ and
- if ϕ is not satisfiable, $\mathrm{OPT}(x) > \alpha(|x|)f(x)$

Assume we have used a gap-introducing reduction to show the hardness of approximating some minimization problem Π_1. We then can transfer the nonapproximability result using a *gap-preserving reduction* to a maximization problem Π_2. This kind of reduction can be defined for all optimization senses for Π_2 and Π_2, but we only mention this one, the other cases are similar. This kind of reduction comes with four parameter functions, f_1, α, f_2, β. Given an instance x of Π_1, it runs in polynomial time and outputs an instance y of Π_2 such that

- $\mathrm{OPT}(x) \leqslant f_1(x) \Longrightarrow \mathrm{OPT}(y) \geqslant f_2(y)$
- $\mathrm{OPT}(x) > \alpha(|x|)f_1(x) \Longrightarrow \mathrm{OPT}(y) < \beta(|y|)f_2(y)$

with $\alpha(|x|) \geqslant 1$ and $\beta(|y|) \leqslant 1$.

We remark that this reduction only preserves the *existence* of an approximability gap, not its size.

Landau Symbols As we often want to speak about polynomial running times of algorithms and would like to ignore the polynomial terms of lower order, it is useful to introduce the so-called Landau Notation. Let $f, g\colon \mathbb{R} \to \mathbb{R}$ be two functions. We define the property of f not growing faster than g by an order of magnitude, written $f \in \mathcal{O}(g)$, as

$$f \in \mathcal{O}(g) \text{ as } x \to \infty :\Longleftrightarrow \limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$$

An equivalent definition is that $f \in \mathcal{O}(g)$ if and only if there exists a real number $M > 0$ and a real number x_0 such that

$$|f(x)| \leqslant M|g(x)| \text{ for all } x \geqslant x_0.$$

An algorithm then has a polynomial worst-case running time $f(x)$ if and only if there is a $k \in \mathbb{N}_0$ such that $f \in \mathcal{O}(n^k)$.

1.3.4 Linear and Integer Programming

Half-spaces and hyperplanes Given any non-zero vector $a \in \mathbb{R}^n$ and a scalar $\alpha \in \mathbb{R}$, we consider the sets defined by the linear inequality $a^T x \leqslant \alpha$ or the linear equality $a^T x = \alpha$. The set of points satisfying such a linear inequality is called a *half-space*, for an equality it is called a *hyperplane*. The vector a is then a normal vector of the defined hyperplanes for all values of α.

Solutions of linear inequality systems Combining inequalities into matrices, we can write systems of linear inequalities as $Ax \leqslant b, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ with $m, n \in \mathbb{N}$, where m is the number of rows and n the number of columns of the matrix A. The solution set $P_{A,b}$ of such an inequality system is called a *polyhedron*. If it is bounded, it is called a *polytope*.

An inequality $a^T x \leqslant \alpha$ is *valid* for a polyhedron P if P is a subset of the half-space defined by the inequality. We define the *face* of P *induced by* the inequality $a^T x \leqslant \alpha$ by

$$F(P, a, \alpha) := \left\{ x \in P \;\middle|\; a^T x = \alpha \right\}$$

If $F(P, a, \alpha) \neq \emptyset$, the inequality $a^T x \leqslant \alpha$ is called *tight* for P. If $\dim(F(P, a, \alpha)) = \dim(P) - 1$, we call the inequality $a^T x \leqslant \alpha$ *facet-defining* and the induced face a *facet* of P. The induced face is called a *vertex* of P if $\dim(F(P, a, \alpha)) = 0$.

The goal in Linear Programming is to minimize a linear objective function inside a polyhedron. This way, some combinatorial optimization problems can be solved efficiently with a unified method. But for many problems, no compact, linear model is known and is unlikely to exist.

A more general approach is called *Integer Linear Programming*, where feasible solutions not only have to satisfy the constraints given by the model, but also have to take integer values. The feasible solution set thus consists of the solution set of a linear program intersected with the high-dimensional grid $\mathbb{Z}^n$. For this problem, no general polynomial method is known and does not exist unless $\mathbf{P} = \mathbf{NP}$.

The feasible solutions in Integer Linear Programming consist of the integer points lying inside the convex polyhedron defined by the model constraints. The solution set is therefore only given

implicitly. This description is also not unique since there are infinitely many different linear descriptions that have the same set of feasible integer points.

The *Integer Linear Programming (ILP)* can be defined formally as follows. Given a matrix $A \in \mathbb{Q}^{m \times n}$, a vector $b \in \mathbb{Q}^m$ and a scalar $c \in \mathbb{Q}$, find a vector $x \in \mathbb{Q}^n$ such that $x = \arg\min\{c^\top x \mid Ax \leqslant b, x \in \mathbb{Z}^n\}$.

Given some candidate solution S, we can verify in polynomial time[2] that S satisfies the constraints of the model and also compute the solution value in polynomial time. Therefore, one of the first observations we make is that the general problem ILP lies in the set **NP**.

Restricting the integer variable range to the interval $[0, 1]$, we can interpret 0 and 1 as the boolean values *true* and *false* and reduce the **NP**-complete problem SATISIFIABILITY to ILP. The logical constraints can easily be translated to linear inequalities. This way, we can prove that the general ILP problem is **NP**-hard.

Duality in Linear Programming A linear program in so-called standard form with A, b and c as above is written as

$$(P) \quad \max\left\{ c^\top x \;\middle|\; x \in \mathbb{R}^n, Ax \leqslant b, x \geqslant 0 \right\}$$

With every such linear program, we can associate its dual program

$$(D) \quad \min\left\{ b^\top y \;\middle|\; y \in \mathbb{R}^m, y^\top A \geqslant c^\top, y \geqslant 0 \right\}$$

These programs are called the *primal* and the *dual* programs. A simple observation, called *weak duality*, is that the optimal value of the primal program is upper-bounded by the optimal value of the dual program, if both programs have feasible solutions. This can be seen quite easily, as for a feasible solution $x \in \mathbb{R}^n$ for the primal and a feasible solution $y \in \mathbb{R}^m$ of the dual problem, we get that

$$c^\top x \leqslant y^\top A x \leqslant b^\top y,$$

where the first inequality comes from the dual, the second from the primal feasibility combined with the factor that both x and y are nonnegative vectors. A stronger statement is the *Strong Duality Theorem* that makes the relation between the primal and dual program more precise, compare any textbook on linear optimization, e.g. Papadimitriou [1982]:

Theorem 1.3.2 (LP strong duality theorem)**.** *Let $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and consider the primal and dual programs (P) and (D) as defined above.*

1. *If both (P) and (D) have feasible solutions, there is a common value $k \in \mathbb{R}$ that is the solution value of optimal solutions for both programs.*
2. *If one the programs has no feasible solution, then the other program is either infeasible or unbounded.*

[2] One first needs to show upper bounds on the encoding length of feasible solutions to preserve the efficient verifiability, see for example Kannan and Monma [1978] or Papadimitriou [1981].

3. *If one of the programs is unbounded, then the other does not have a feasible solution.*

As feasible solutions to the dual problem yield valid bounds on the best possible values of the primal problem, the dual problem plays an important problem in practically solving linear and integer linear programs.

1.3.5 Algorithms for LPs and ILPs

Linear Programming

There are two major methods known for the solution of linear programs. These are the *Basis exchange* method and the *Interior Point method*. The *Basis exchange* method works by finding an initial vertex solution and then moving along the edges of the feasible polyhedron to an optimal vertex. The most famous Basis exchange method is known as the *Simplex Method*.

The *Interior point method* is based on moving through the interior of the polytope, as the name implies. There are many algorithms following this basic method. These include the *Ellipsoid Method* which was the first polynomial method for the Linear Programming problem, which suffers from bad numerical problems. Another famous method is the *Barrier function method*, which follows a path through the interior of the polytope and which uses a function that has small values inside the polyhedron and large values outside, called a *Barrier function*.

The Simplex method We first describe a geometric interpretation of the simplex method. As the simplex method operates on linear programs in standard form, the feasible region is a *pointed* polyhedron, meaning it has at least one vertex. For any linear objective function, there is an optimal solution to the linear program that is a vertex of the feasible region, which has only finitely many vertices. This way, we can restrict ourselves to a finite subset of solutions, making the problem a discrete one.

In the first phase, a feasible vertex solution is constructed. Given a polyhedron, its 1-skeleton is constructed by taking the vertices and edges of the polyhedron and forming an undirected graph from them. By the well-known theorem of Balinski [1961], the 1-skeleton of a convex d-dimensional polyhedron is d-vertex-connected. Therefore, for any two vertices of the feasible region, there is a path consisting of polyhedral edges connecting these two vertices. This idea is exploited in the simplex algorithm: In the second phase, we move along edges of the feasible region in directions where the objective function improves.

These vectors x^* of such vertices correspond to so-called *basic feasible solutions* which are characterized by the property that there are n constraints satisfied with equality in the inequality system $Ax^* \leqslant b$, where n denotes the number of variables. The switch from one vertex to an adjacent vertex works in the matrix form in such a way that we move from one *basis* to another, where a basis is subset of the columns that has full rank. Variables corresponding to a basis are called *basic variables* and the other *non-basic* variables. The basis gives rise to a *basic solution*, which is created by setting all non-basic variables to 0. As the basis columns have full rank, the basic variables suffice to solve the matrix inequalities. A basic solution is called a *basic feasible solution* if it also satisfies the nonnegativity constraints. We move from one basis to another by the following process. First, we select one non-basic column. This non-basic column then

enters the new basis and we remove another column in a process called *pivoting*. The way we select this non-basis column is most important, because different bases can yield the same basic solution. This leads to the problem of *cycling*, where we could move from one basis to the next and then end up with the same basis after a while, not making any progress in the solutions.

The rule by which we select the next non-basis column is called the *pivoting rule*. To avoid the problem of *cycling*, we can use rules that guarantue that we never have the same basis twice, for example the rule by Bland [1977]. The simplex method then can find an optimal vertex solution to the linear program in a finite amount of time.

While it usually runs very efficiently in practice, there is no known pivoting rule that makes it run in polynomial worst-case time.

Interior Point Methods Interior point methods move through the interior of the polytope instead of following the edges. This is accomplished by introducing a *barrier function* that has small values inside the feasible region and grows very quickly outside it. By creating a weighted linear combination of the barrier and the objective function, we can make sure to stay inside the polyhedron.

Interior-point methods follow the so-called *central path* which is a curve through the feasible polyhedron arising from linear constraints and quadratic constraints implied by the complementary slackness conditions. By using some form of Newton's method, the algorithms follows a piecewise-linear approximation to this central path until it reaches an optimal point of the desired accurateness.

By doing a so-called *crossing over* step, we can then find a vertex solution to the original problem.

Integer Linear Programming

While Linear Programs can be solved in polynomial time, there is no known method for the general ILP problem, which is known to be **NP**-hard.

Solving Integer Linear Programs optimally is usually accomplished by applying some variant of the Branch&Bound-Algorithm, which is basically a full enumeration of all possible solutions. The search space can be seen as a *search tree*. The "bound" part refers to the fact that we try to cut off branches off this search tree as early as possible by bounding the values of solutions that still exist in the remaining search tree. We use speeding up techniques like cutting plane methods, decomposition techniques, lower bounding, heuristics and integrate them into one algorithm.

While the Branch&Bound method works also in its most general form, one would like to make it fast in practice; this is accomplished by tailoring the generic method to the specific needs. For some problems, some techniques work a lot better than other techniques, but this is sometimes hard to tell in advance. In chapters 2, 3, 5 and 6, we use this procedure to find a good solution method. Usually, the end result uses one of these techniques as a major factor, but also employs other strategies at the same time.

If Branch&Bound is used together with the generation of cuts, it is usually called *Branch&Cut*, while in combination with column generation, it is called *Branch&Price*. One can also combine all of these techniques and get a *Branch&Cut&Price* approach.

Strong formulations For any polyhedron in $\mathbb{R}^n$, the minimal number of inequalities needed to fully describe it is equal to its number of facets. Up to multiplication with scalar factors, this gives rise to a canonical representation as a linear inequality system, where no inequality can be removed without changing the polytope.

If we consider the solution sets of Integer Linear Programs, the description is given more implicitly. We describe some polyhedron and then intersect it with the $\mathbb{Z}^n$. Let $P*$ be the smallest convex set containing this intersection, $P* = \text{conv}\{Ax \leqslant b, x \in \mathbb{Z}^n\}$, where the convex hull conv of a set A is defined as the intersection of all convex sets containing A. We call this construction the *integer hull* of a polyhedron P.

Given a full description of the integer hull, we can solve the ILP problem as a Linear Program – it is therefore not possible in most cases to get a complete description. The models we consider are more or less exact descriptions of the integer hull combined with the integrality constraint. Different models can then yield the same integer points.

By relaxing the integrality constraint, we get the so-called *LP relaxation* of a model. To use it to get a bound on optimal solution values of the ILP, it is useful to compare different LP relaxations. We say that a model M is *stronger* than another model N if the polyhedron defined by the LP relaxation of M is contained in the polyhedron defined by the LP relaxation of N.

Big-M constraints A common type of constraint causing weak LP relaxations, usually used to link integer to binary variables, is called a *big-M* type constraint. For a binary variable x_b and an integer variable x_i, this constraint often has a form like $M \cdot x_b \geqslant x_i$. It models an implication in the integer model: if we want the integer variable to have a strictly positive value, we need to set the binary variable to the value 1, allowing the integer variable to range in the interval $[0, M]$. The problem with this type of constraint is that in the LP relaxation, solutions are feasible even if x_b only takes the value x_i/M. This commonly leads to solutions with values that are factor of M away from any feasible integer solution. There are techniques to avoid using big-M constraints, for example the technique of *disassocation*. In the current example, we could introduce binary variables x_{ik} for $k = 1, \ldots, M$ and use the constraints

$$x_i = \sum_{k=1}^{M} x_{ik}$$

and

$$x_b \geqslant x_{ik} \quad \forall k = 1, \ldots, M.$$

This leads to a stronger variable linking in the LP, but also increases the size of the model.

Valid inequalities Given an ILP model, there are some techniques to improve the model. Improvement here refers to changes that make the LP relaxation stronger so that it yields better bounds. The basic idea is to add additional inequalities that do not remove any integer solution, but remove fractional solutions. Such an inequality is called a *valid inequality*.

This can also be done while running a Branch&Bound search for the optimal solution. At a node of the search tree, we consider the so-called *node LP* that results from the LP relaxation by also incorporating the branching decisions that led to this node. Solving it optimally will usually

still yield a fractional solution, which we then can exclude by finding a valid inequality that this solution violates. The process of finding such an inequality is called *separation* as the resulting inequality *separates* the feasible polyhedron from this point by a *cutting hyperplane* such that the fractional point lies on one side of the hyperplane, while all integer feasible solutions lie on the other side.

1.3.6 Modelling network problems

In this section, we introduce some standard models for the most famous network optimization problems. These models and variants of them will be used in later chapters to model various extensions of the optimization problems presented here.

Uncapacitated Facility Location problem (UFLP)

In the Uncapacitated Facility Location problem (UFLP), we are given two sets of nodes, the *facilities* F and the *customers* C. The objective function is a sum of two kinds of costs, the *facility opening costs* $f\colon F \to \mathbb{R}_+$ and the *service costs* $c\colon F \times C \to \mathbb{R}_+$. The goal is to find a set of facilities which we open and to assign each of the customers to some open facility, minimizing the total cost incurred. We denote $c(i,j)$ as c_{ij} and $f(i)$ as f_i.

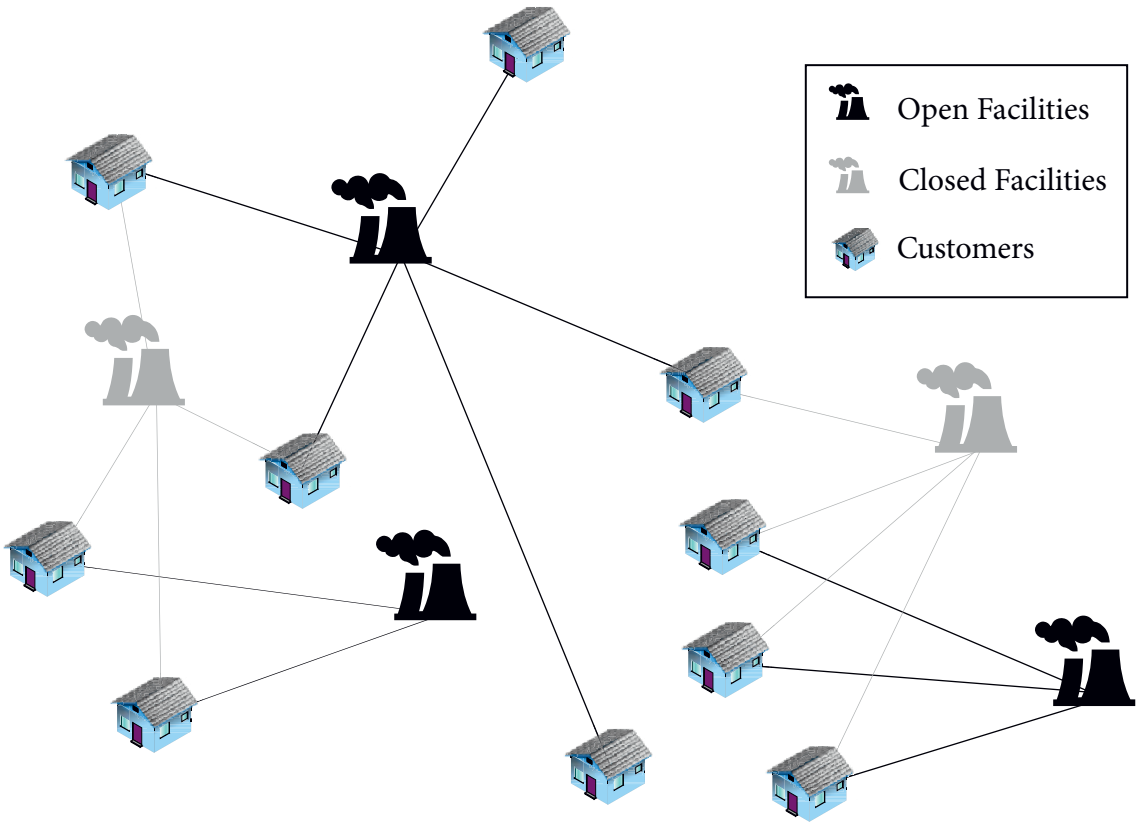


Figure 1.3.2: An exemplary solution of a facility location instance

An exemplary solution of a facility location instance can be seen in Figure 1.3.2 on the previous page. A standard ILP formulation for this problem is the following:

$$\begin{aligned}
&\text{(UFLP):} && \min \sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in C} c_{ij} x_{ij} && \\
&\text{subject to} && x_{ij} \leqslant y_i && \forall i \in F, j \in C && (1.3.1)\\
& && \sum_{i\in F} x_{ij} \geqslant 1 && \forall j \in C && (1.3.2)\\
& && x_{ij}, y_i \in \{0,1\} && \forall i \in F, j \in C &&
\end{aligned}$$

Relaxing to a linear program, we can write down the dual problem:

$$\begin{aligned}
&\text{(UFLP-D):} && \max \sum_{j\in C} v_j && \\
&\text{subject to} && \sum_{j\in C} w_{ij} \leqslant f_i && \forall i \in F && (1.3.3)\\
& && v_j - w_{ij} \leqslant c_{ij} && \forall i \in F, j \in C && (1.3.4)\\
& && w_{ij}, v_j \geqslant 0 && \forall i \in F, j \in C &&
\end{aligned}$$

The dual has an economic interpretation as prices the customers have to pay. The number w_{ij} then represents the amount customer j contributes to the opening of facility i. The constraints imply that no more money can be put into opening a facility than the facility costs (1.3.3) and no customer pays more than its contribution to the opening of a facility plus the service cost he has to pay to reach the facility (1.3.4).

In the *Set Cover* problem, we are given a ground set and a family of subsets with weights and ask for a weight-minimal selection of these subsets such that every element of the ground set is contained in at least one of the chosen subset. The Set Cover problem can easily be formulated as a special case of the UFLP. In this instance of the UFLP, the set C is identified with the ground set of the Set Cover problem, each facility corresponds to a subset of the ground set with the same cost as in the Set Cover problem and the assignment cost is 0 or ∞ depending on whether the subset contains the corresponding element.

Because the Set Cover problem cannot be approximated better than $\mathcal{O}(\log n)$ (Lund and Yannakakis [1994]; Feige [1998]) unless $\mathbf{P} = \mathbf{NP}$, the same is true for the Uncapacitated Facility Location problem.

Metric Facility Location problem In the metric facility location problem, we assume that the distances are part of a metric on $F \cup C$, which has to satisfy the triangle inequalities. For this variant of the problem, an 1.488-approximation algorithm is known due to Li [2011].

Steiner Tree problem

The original form of the Steiner Tree problem lives in the realm of Euclidean geometry. We call that problem the *Euclidean Steiner Tree problem*. In it, we are given n points in the Euclidean

plane and wish to connect them via straight line segments. Points can be connected directly or via other line segments and points. We can also use arbitrary additional points as endpoints of the line segments. An example instance of this problem can be seen in Figure 1.3.3.

To distinguish the original from the additional nodes, the original nodes are called *terminals* and the additional points *Steiner points*.

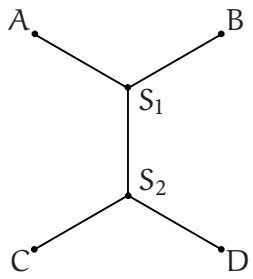


Figure 1.3.3: An optimal solution to the Euclidean Steiner Tree problem with 4 terminals A, B, C, D, using two Steiner points S_1, S_2

In the graph setting, we are given a graph $G = (V, E)$, a subset $T \subseteq V$ of terminals and a cost function $c\colon E \to \mathbb{R}_+$. We look for a cost-minimal tree in G that spans all terminals. An example can be found in Figure 1.3.4.

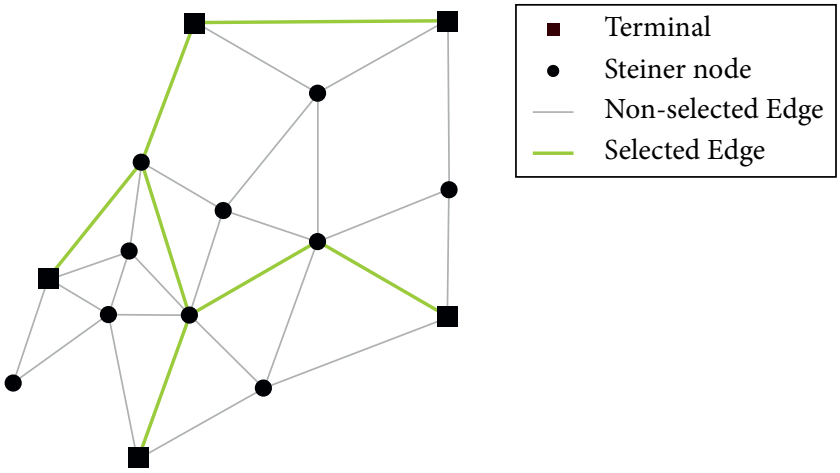


Figure 1.3.4: An instance of the Graph Steiner Tree problem. A possible solution is highlighted in green.

We now describe a standard, exponentially sized ILP model of the problem. This description is called the *undirected model*. For a node subset $S \subseteq V$, we denote by $\delta(S)$ the edges having one endpoint in S, the other in $V \setminus S$. This set of edges is also called the *cut* induced by the node subset S. A subset $S \subset V$ satisfying $S \cap T \neq T$ and $S \cap T \neq \emptyset$ is called a *terminal-separator*. We denote the set of terminals-separators as $\mathcal{S}_T$.

$$\begin{aligned} \text{(ST-UNDIR):} \quad & \min \sum_{e \in E} c_e x_e & & \\ \text{subject to} \quad & \sum_{e \in \delta(S)} x_e \geqslant 1 & S \in \mathcal{S}_T & \qquad (1.3.5) \\ & x_e \in \{0,1\} & \forall e \in E & \end{aligned}$$

We also describe two additional formulations for the problem which are stronger than the simple undirected formulation.

The first one is the so-called *partition formulation*, compare also Chopra [1989] and the exposition in Bertsimas and Weismantel [2005].

We call a collection $V_1, \ldots, V_p$ of subsets of V a *partition* of V if the following three conditions are satisfied:

1. Each V_i contains at least one terminal,
2. V_i and V_j are disjoint for $i \neq j$ and
3. all nodes of V are contained in some V_i.

We denote the set of edges whose endpoints lie in different partition sets of the partition as $\delta(V_1, \ldots, V_p)$. Then, we can write down the Steiner Partition Formulation:

$$\begin{aligned} \text{(ST-PART):} \quad & \min \sum_{e \in E} c_e x_e & & \\ \text{subject to} \quad & \sum_{e \in \delta(V_1,\ldots,V_p)} x_e \geqslant p-1 & \text{for all partitions } (V_1, \ldots, V_p) \text{ of } V & \qquad (1.3.6) \\ & x_e \in \{0,1\} & \forall e \in E & \end{aligned}$$

It can easily be seen that the partition formulation is at least as strong as the undirected formulation, as any terminal-separator also defines a partition of V.

An even stronger formulation for this problem is the *directed formulation*. For this formulation, we change the undirected graph $G = (V, E)$ into a bidirected graph $G = (V, A)$ by inserting each edge in both directions. The cost will be on both arcs. We choose an arbitrary root node $r \in T$; then, we need to connect it to all other terminals.

The formulation is as follows:

$$\begin{aligned} \text{(ST-DIR):} \quad & \min \sum_{(i,j) \in A} c_{ij} y_{ij} & & \\ \text{subject to} \quad & \sum_{(i,j) \in \delta^+(S)} y_{ij} \geqslant 1 & \forall S \subset V : r \in S, T \cap S \neq T & \qquad (1.3.7) \\ & y_{ij} + y_{ji} \leqslant 1 & \forall e = \{i,j\} \in E & \qquad (1.3.8) \\ & y_{ij} \in \{0,1\} & \forall e = \{i,j\} \in E & \end{aligned}$$

The partition inequalities can be shown to be implied by the directed formulation by summing the inequalities in ST-DIR with the proper coefficients. Therefore, this formulation is at least as strong as the partition formulation.

The best LP relaxation for the Steiner Tree problem that is known is based on directed components of the graph and has an approximation ratio of 1.55, see Byrka et al. [2013].

Connected Facility Location problem (CFLP)

A variant of the Facility Location problem is the *Connected* Facility Location problem which combines aspects of the UFL problem and the Steiner Tree problem.

In the standard formulation, we are given an undirected graph $G = (V, E)$, edge costs $c\colon E \to \mathbb{Q}^+$, a set of facilities $F \subseteq V$ with opening costs $f_i \in \mathbb{R}_+, i \in F$, a set of customers $C \subseteq V$ with demands $d(j) \in \mathbb{Q}^+$ and a parameter $M \geqslant 1$. The goal is to determine a subset of the facilities $F \subseteq V$ to be opened, to assign each customer $j \in C$ to some open facility $\sigma(j)$ and also build a Steiner tree T connecting the open facilities, while minimizing the total cost

$$\sum_{i \in F} f_i y_i + M \sum_{e \in T} c_e + \sum_{i \in F} \sum_{j \in C} d(j) \ell(j, \sigma(j))$$

where $\ell(j, k)$ denotes the length of a shortest path between the customer j and its assigned facility $k = \sigma(j)$ with respect to the edge costs c_e.

A standard ILP for CFL is the following, see also Swamy and Kumar [2004]. For the following formulation, we assume that one fixed facility v is open. As every solution has to open at least one facility and we can try all possible facilities in F as the one that is fixed to be open, we can use the model to find an optimal solution for the problem.

Note that we can modify the graph by replacing the shortest paths between facilities and customers with direct edges of the same cost. For this reason, we write c_{ij} from now instead of the explicit shortest path length $\ell(i, j)$.

$$\text{(CFLP):}\ \min \sum_{i \in F} f_i y_i + \sum_{j \in C} d_j \sum_{i \in F} c_{ij} x_{ij} + M \sum_{e \in E} c_e z_e$$

subject to

$$x_{ij} \leqslant y_i \qquad \forall i \in F, j \in C \tag{1.3.9}$$

$$\sum_{i \in F} x_{ij} \geqslant 1 \qquad \forall j \in C \tag{1.3.10}$$

$$y_v = 1 \tag{1.3.11}$$

$$\sum_{i \in S} x_{ij} \leqslant \sum_{e \in \delta(S)} z_e \qquad \forall S \subseteq V : v \notin S, \forall j \in C \tag{1.3.12}$$

$$x_{ij}, y_i \in \{0, 1\} \qquad \forall i \in F, j \in C$$

Swamy and Kumar [2004] used this formulation for an 8.55−approximation algorithm. More recently, a 4-approximation algorithm for this problem was given by Eisenbrand et al. [2010]. In chapter 5, we consider an ILP formulation of an incremental version of this problem.

Minimum (Connected) Dominating Set problem

In the Minimum Dominating Set problem (MDSP), we are given a graph $G = (V, E)$ and a cost function $c\colon V \to \mathbb{R}_+$ on the nodes of the graph. The goal is to find a node subset D such that each node of v is either in D or is adjacent to a node in D. We remind the reader that we denote the set of all nodes adjacent to v including the node v itself by Γ_v. A simple ILP formulation of the MDS problem is as follows:

$$\begin{aligned} &\text{(MDSP):} && \min \sum_{v \in V} c_v x_x && && \\ &\text{subject to} && \sum_{w \in \Gamma_v} x_w \geqslant 1 && \forall v \in V && (1.3.13) \\ & && x_v \in \{0,1\} && \forall v \in V && \end{aligned}$$

A related problem is the Minimum *Connected* Dominating Set problem (MCDSP). In that problem, we additionally require the set D to induce a connected subgraph of V. An example for this problem can be seen in Figure 1.3.5.

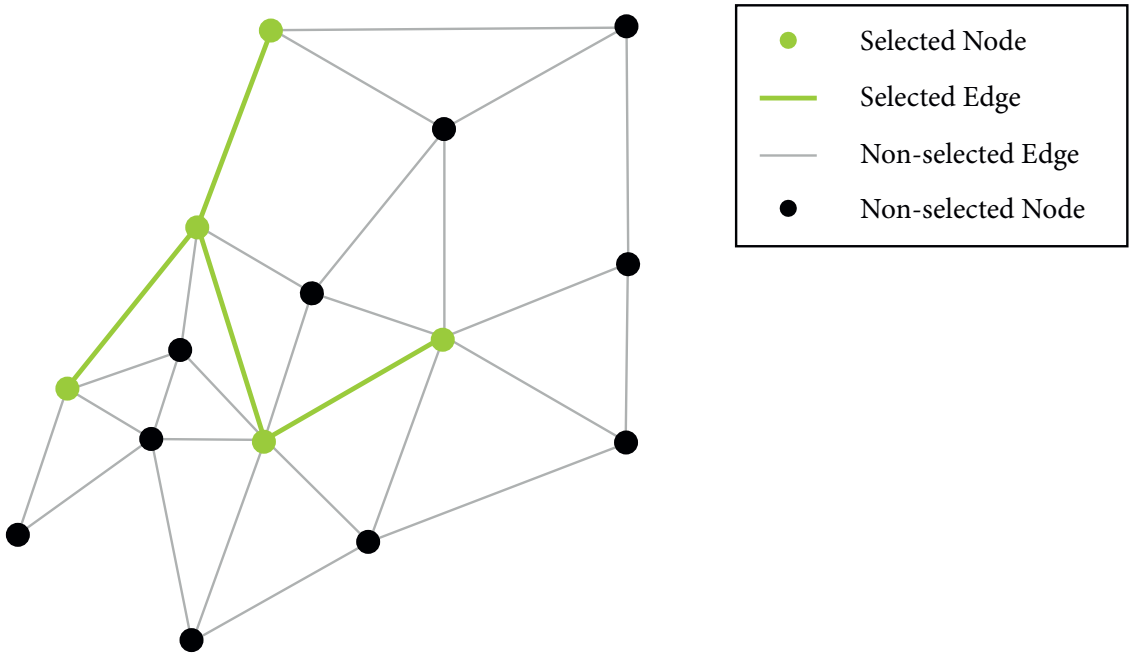


Figure 1.3.5: An instance of the MCDS problem. A possible solution is highlighted in green.

A branch-and-cut algorithm for the node-weighted MCDS problem is given by Gendron et al. [2014]. We now consider the ILP formulation their work is based on:

$$\begin{aligned}
\text{(MCDSP-ext):} \quad & \min \sum_{i \in V} y_i \\
\text{subject to} \quad & \sum_{e \in E} x_e = \sum_{i \in V} y_i - 1 && && (1.3.14)\\
& \sum_{e \in E(S)} x_e \leqslant \sum_{i \in S \setminus \{j\}} y_i && \forall S \subseteq V, j \in S && (1.3.15)\\
& \sum_{j \in \Gamma_i} y_j \geqslant 1 && \forall i \in V && (1.3.16)\\
& x_e \geqslant 0 && \forall e \in E && \\
& y_i \in \{0,1\} && \forall i \in V &&
\end{aligned}$$

A polyhedron Q is called an *extended formulation* of the polyhedron P if there is a projection mapping the variables of Q to the variables of P in such a way that feasibility is preserved in both directions. In this sense, the MCSDP-ext formulation can be considered an extended formulation of the following formulation MCDS-node that uses node variables only.

For this purpose, we call a subset S of V a *separator* if the graph induced by the node set $V \setminus S$ is disconnected. We denote the set of all separators by $\mathcal{S}$. We can then formulate the problem using node variables only:

$$\begin{aligned}
\text{(MCDS-node):} \quad & \min \sum_{i \in V} c_i y_i \\
\text{subject to} \quad & \sum_{v \in S} y_v \geqslant 1 && \forall S \in \mathcal{S} && (1.3.17)\\
& \sum_{w \in \Gamma_v} y_w \geqslant 1 && \forall v \in V && (1.3.18)\\
& y_v \in \{0,1\} && \forall v \in V &&
\end{aligned}$$

We use a node-based formulation similar to this one in chapter 3 for a mix of a Steiner Tree problem and a MCDS problem, called the Dominating Steiner problem.

2 Passive Optical Network Design

2.1 The Two-Layer FTTX Network Design problem

In this chapter, we introduce the 2FTTx-model that is used for modeling the network design problem for passive optical access networks. The work presented is joint work with Andreas Bley and Ivana Ljubić and has been published in "EURO Journal on Computation" (Bley et al. [2013]).

2.1.1 Introduction

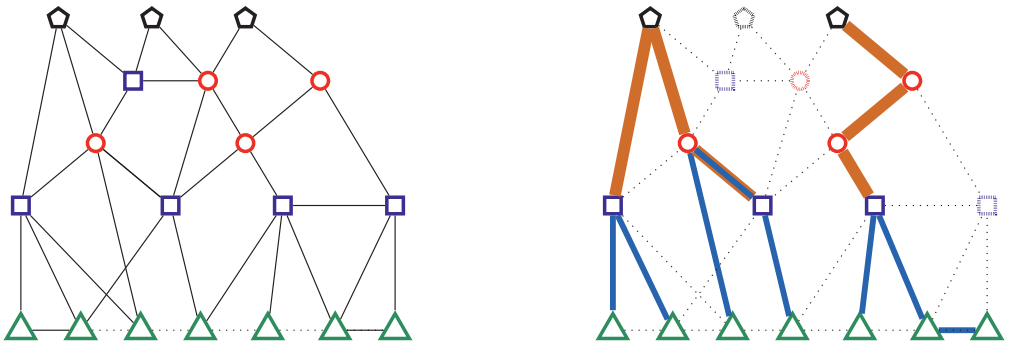
We consider a network design problem for Passive Optical Networks (PONs) which is an architecture usually reserved for access network: these kinds of networks are typically used on a metropolitan scale to build connections between households or buildings and the backbone network of the service provider, compare also Chapter 1, especially Section 1.1.2 and Section 1.1.3.

Depending on the endpoint of the fiber network and the technology used on the last mile, these networks are also called Fiber-to-the-x networks, where x can stand for home, curb, building and so on. From an optimization point of view, this specification mainly changes the size of the demands. This has no significant impact on the model we introduce and therefore, we use the same optimization model for all of these variants.

2.1.2 Problem setting

We now introduce the problem setting, beginning with a description of the network architecture we consider. The endpoints of the network which connect the PON to another, more highly aggregated network are called the *Central Offices (COs)*. Depending on the size of the PON, several central offices might be used to feed the access network. We are given some fixed CO locations and we need to decide which are the ones we open. The problem input also contains a graph where an edge corresponds to optical fibers connecting the potential CO locations to the customers. In this graph, we have intermediate aggregation points called distribution points (DPs), which are the locations in which passive optical splitters can be installed. These take the incoming signal and distribute it to several outgoing optical fibers, usually with a ratio of $(1:2)$ or $(1:16)$. This way, one optical terminal in a central office can serve multiple premises.

Optical Line Terminals (OLTs) are placed in the Central Offices and Optical Networking Units (ONUs) at the end premises which house the optical transceivers. For simplicity, we will call the end premises in this network the *customers*. We need to determine both the locations and the capacities of ONUs and DPs. Also, we need to determine which of the possible fiber connections will be built. The building of these connections incurs a very significant cost because trenching is quite expensive compared to the hardware costs.



(a) Example of the input graph (b) A feasible solution

Figure 2.1.1: 2FTTx example instance. Triangles represent customers, squares represent DPs, pentagons represent COs and circles represent remaining nodes

The model we work with is called the *Two-Level FTTx Network Design problem* (2FTTx). While capturing some important optimization aspects, we also simplify some real-world difficulties. For example, we disregard the existence of different cable types which lead to buy-at-bulk non-linear costs depending on the capacity needed. Instead, we assume that a fixed price has to be paid for each single fiber installed on an edge. Also, we ignore fiber dispersion and power budget limits.

The subnetwork containing the routing paths between COs and DPs is called the *feeder network* (FN), and the subnetwork containing the routing paths between DPs and customers is called the *distribution network* (DN).

In our setting, we assume that both the feeder network and the distribution network must have tree[1] topologies. Although there are no technological reasons for this restriction, this requirement is typically imposed by the network operator for practical reasons. With a tree-like network structure, deployment, upgrade and maintenance of a PON become much simpler and less error-prone in practice. For the same reason, operators usually also forbid to use the same cable (fiber bundle) for both feeder network and distribution network fibers, for fibers heading into different directions, or for fibers heading towards different COs.

Figure 2.1.1 illustrates an example instance.

Our main goal is to develop computational methods that enable practitioners to (approximately) solve very large instances of the 2FTTx problem with very little computing time. Such methods are of particular interest in the early stages of the long-term strategic network planning, when numerous planning scenarios with varying technological assumptions and demand, cost, or revenue predictions are evaluated. These are then used to identify the most important parameters and make the global strategic decisions concerning technology vendors, the use of existing or the building of new infrastructures, or the long-term evolution of the network, for example. These case studies require methods that are able to solve the 2FTTx network design problem for very large network regions consisting of several PON areas very fast and with a sufficiently small optimality gap, but not necessarily to optimality. The proposed Lagrangian decomposition approaches perfectly meet these requirements. They are also very useful to quickly compute

[1] or more precisely, forest

good bounds and approximate solutions in later planning stages. For the final network and hardware configuration planning, which is typically performed only once for each of the much smaller single PON areas, it is however advisable and computationally feasible to use a more accurate model of the various technical elements (such as the different duct and cable types) in order to fully exploit all potential savings.

2.1.3 Related work

There are several problems studied in the literature that deal with the design of FTTx networks. *Connected Facility Location*, for example, considers the design of a tree-star network, where facilities (e.g., splitters) are connected to customers in a star-like fashion. There are no splitter- or edge-capacities and fiber costs are discarded, so that the optimization goal consists of determining the network topology that minimizes the facility opening plus edge set-up costs. This problem has been studied in Eisenbrand et al. [2010]; Gollowitzer and Ljubić [2011]; Leitner and Raidl [2011], to mention a few recent references.

A related problem combining network design and facility location aspects is the *Two Level Network Design* (TLND) problem. There, we are given two technologies and two types of nodes that need to be served by them, and the goal consists of building a tree-tree network such that facilities are installed at the transition points between the two technologies (see Gollowitzer et al. [2013]). In Balakrishnan et al. [1994], the two-level network design problem without facilities is studied. The 2FTTx can be seen as a generalization of the TLND problem: if sufficiently large splitter- and edge-capacities are assumed, and the fiber costs are zero, the 2FTTx reduces to the TLND problem.

The *Local Access Network Design* (LAN) problem is a problem that combines the topological design of the network with the decisions on routing the fibers in order to serve customer demands, while respecting edge capacities (see, e.g. Salman [2000]; Putz [2012]). This problem captures the capacity aspects of our problem, but assumes that there are no splitters installed on the way between the COs and the end premisses.

Finally, we point out that there are other works in the literature focusing on design aspects of FTTx networks. For example, in recent works presented in Gualandi et al. [2010a,b]; Kim et al. [2011]; Chardy et al. [2012], the authors concentrate on splitter location and dimensioning aspects by assuming that the routing paths are given, and therefore can be replaced by assignment arcs. A few more problems that model FTTx deployment as variants of the facility location problems have been studied recently. In Chardy et al. [2012], the authors study a three-level FTTH network design problem in which splitters have to be located and fibers need to be routed from a CO to the end customers. Thereby, only edges of an existing infrastructure (with limited capacities) can be used, and the main optimization goal consists of placing two types of splitters at two different layers, connecting customers to a CO and routing the fibers without violating edge capacities so that customer demands are satisfied. The network topology is already given and it only remains to decide on the splitter locations, their numbers and the routings of fibers. A similar problem has been studied by Kim et al. [2011] where the authors assume that the input graph is a tree. Both single- and double-splitting schemes are studied. The goal consists of deciding on the splitter locations and their number and on the cable types to be installed along the tree edges so

that all customers are served, but not necessarily with a single-splitter assignment. The authors use Mixed-Integer Programming and heuristic techniques for solving the problems.

More practice-oriented approaches have been studied in Martens et al. [2009]. By using a two-step approach with suitable MIP formulations, it is possible to optimize fiber-optic networks in realistic scenarios (see Martens et al. [2010]). Orlowski et al. [2011] conducted various practice-oriented case studies that originated from planning scenarios by a German telecommunications provider.

Our contributions

We first propose an integrated MIP model (Section 2.2) and present families of strengthening valid inequalities for it. To handle the size and complexity of the problem, we then propose two Lagrangian decomposition approaches for solving the 2FTTx problem (Subsection 2.3.1 and Subsection 2.3.2). The first approach decomposes the problem based on the network structure and the second approach decomposes the problem based on the cost structure. The subproblems are solved using MIP techniques. A combination of heuristic ideas based on the Lagrangian Decompositions (presented in Section 2.3.3 and Section 2.3.3) and MIP techniques allows us to solve some real-world network planning instances within a few percent of optimality. Detailed computational results are shown in Section 2.4.

Follow-up Work

Since the release of the journal version, several authors have considered extensions of the model and problems. In Grötschel et al. [2013], the authors discuss modeling alternatives for different extensions and present a unified model for all the extensions. In Gouveia et al. [2015], the topology of the splitting, meaning splitting ratios and the number of splitting stages, is not fixed, but decided in the model. The authors present different formulations for this problem and provide computational experiments. In Żotkiewicz et al. [2015], an integrated platform for solving PON planning problems is presented that allows much flexibility and aims for a practically relevant implementation.

2.2 MIP model

2.2.1 Problem input and constraints

We now define the 2FTTx problem formally. In the 2FTTx problem, we are given an undirected graph $G = (V, E)$ with the set of nodes V partitioned into

- customers (V_C),
- potential distribution points (V_D),
- potential central offices (V_{CO}) and
- remaining nodes (V_O).

At least one central office has to be opened, and each customer $v \in V_C$ has to be provided with at least $d_v \geqslant 0$ fibers. Fiber connections begin at a CO, follow a path through exactly one DP and continue until they reach an end customer. Splitters are installed at DPs so that every single fiber can be split into a given ratio. The set T denotes all available splitter types, s_t is the splitter ratio for each $t \in T$ and $J_{t,v}$ is the maximal number of the splitter-type t available at the DP $v \in V_D$. Along each edge (trail) $e \in E$ at most u_e fibers can be installed in the feeder and the distribution network, independently. Finally, u_v is the capacity of a DP or CO v, that is, the maximal number of the downstream-fibers or transceivers, respectively. In total, DPs and COs are allowed to be installed in at most N_D and N_{CO} locations, respectively. The following costs are associated to the input parameters of our network:

c_v	$\forall v \in V_D \cup V_{CO}$	:	opening costs for v being a DP or CO
$c_{t,v}$	$\forall t \in T, v \in V_D$	:	cost for the installation of a splitter of type t at the DP v
c_e	$\forall e \in E$	:	installation cost for an edge e
c_e^f	$\forall e \in E$	:	cable cost along edge e in the feeder network
c_e^g	$\forall e \in E$	:	cable cost along edge e in the distribution network

The optimization goal consists of deciding which COs and which DPs to open, which splitters to install at the DPs, and how to route paths in the FN and the DN so that demands of all customers are satisfied at minimum cost. The DP and CO locations can be traversed as intermediate nodes, in which case no opening costs need to be paid for them. In addition, even if the feeder and the distribution network both use the same edge, the fixed-charge cost c_e is paid for only once. Moreover, FN and DN are usually required to have tree topologies, i.e. paths between an end-customer and a DP and between a DP and a CO are required to be unique.

We now describe how the costs and constraints are incorporated into the MIP model.

Let A be the set of arcs obtained by bidirecting edges from E. We will use binary variables $x_v \in \{0,1\}$ for each $v \in V_D$ to indicate whether splitters are installed at location v or not. Similarly, binary variables $z_v \in \{0,1\}$ will indicate whether a central office $v \in V_{CO}$ is opened or not. The number of splitters of type t installed at the DP location $v \in V_D$ will be counted using integer variables $y_{t,v}$. Finally, for each edge $e \in E$, binary variables w_e will indicate whether the edge e is used or not, and the number of fibers in the DN and the FN installed along arc $a \in A$ is counted using variables g_a and f_a, respectively. Using these variables, our objective function can be described as follows:

Objective function

$$\min \sum_{v \in V_{CO}} c_v z_v + \sum_{v \in V_D} c_v x_v + \sum_{e \in E} c_e w_e + \sum_{v \in V_D} \sum_{t \in T} c_{t,v} y_{t,v} + \sum_{a \in A} (c_a^f f_a + c_a^g g_a)$$

The first two terms are the installation costs for COs and DPs, followed by the installation costs for the edges, followed by the splitter installation costs. The last summation corresponds to the total fiber costs installed in the DN and the FN.

Bounds on splitter installations, DPs and COs The bounds regarding the total number of allowed DPs and COs are expressed using constraints (2.2.1) and (2.2.2), respectively, and the bounds regarding the maximal number of splitters of type t at the DP node v are expressed using (2.2.3):

$$\sum_{v \in V_D} x_v \leqslant N_D \tag{2.2.1}$$

$$\sum_{v \in V_{CO}} z_v \leqslant N_{CO} \tag{2.2.2}$$

$$y_{t,v} \leqslant J_{t,v} x_v \qquad \forall v \in V_D, t \in T \tag{2.2.3}$$

Before we present the remaining constraints of our MIP model, we introduce the following notation: Each splitter installed at a DP location requires a single fiber from a CO (where a transceiver needs to be placed). The overall number of splitters installed at a DP determines its *upstream-fiber demand*. The overall number of fibers available in the DN and obtained after splitting at a certain DP location determines its *downstream-fiber supply*. The following set of auxiliary variables is now introduced to simplify the notation further:

$F_v \in \mathbb{Q}$	$\forall v \in V_D$	:	the number of upstream-fibers at the DP v
$G_v \in \mathbb{Q}$	$\forall v \in V_D$	:	the number of downstream-fibers at the DP v
$H_v \in \mathbb{Q}$	$\forall v \in V_{CO}$	:	the number of transceivers installed at the CO v
$w_a^f \in \{0,1\}$	$\forall a \in A$	:	1 if the arc a is used by the f-flow (feeder)
$w_a^g \in \{0,1\}$	$\forall a \in A$	:	1 if the arc a is used by the g-flow (distribution)

Flow conservation in distribution and feeder network In order to ensure a feasible routing in the distribution network, the flow-preservation constraints (2.2.4) are used. They also state that the total customer demand has to be satisfied using the supply of downstream-fibers available at distribution points. Constraints (2.2.5) ensure that the g-flow is routed along edge e only if edge e is actually installed. Constraints (2.2.6) restrict the solution space to solutions that sent flow along arcs only in one direction. This is no restriction, since otherwise we can reduce flow in both directions simultaneously without violation of the flow conservation constraints until one of them disappears. We denote the incoming arcs of a node v by $\delta^{in}(v)$, while $\delta^{out}(v)$ denotes the outgoing arcs.

$$\sum_{a \in \delta^{in}(v)} g_a - \sum_{a \in \delta^{out}(v)} g_a = \begin{cases} d_v & v \in V_C \\ -G_v & v \in V_D \\ 0 & \text{else} \end{cases} \qquad \forall v \in V \tag{2.2.4}$$

$$g_{ij} \leqslant u_e w_{ij}^g \qquad \forall e = \{i,j\} \in E \tag{2.2.5}$$

$$w_{ij}^g + w_{ji}^g \leqslant w_e \qquad \forall e = \{i,j\} \in E \tag{2.2.6}$$

At the first glance, one might think that the variables w^g_a could be omitted. They are, however, used later on to derive cuts that will strengthen the LP relaxation. Also, they can be used in in-degree constraints to enforce tree topologies in the subnetworks. To make sure that the routing between COs and DPs is feasible, we consider constraints (2.2.7). These constraints also make sure that the overall upstream-demand at distribution points (expressed using F_v variables) is satisfied by installing a sufficient number of transceivers at the corresponding COs (H_v variables). Constraints (2.2.8) make sure that the f-flow is routed along edge e only if edge e is actually installed. Constraints (2.2.9) ensure that $f-$flow is sent along an edge in only one direction, as has been explained above for the g-flow.

$$\sum_{a\in\delta^{in}(v)} f_a - \sum_{a\in\delta^{out}(v)} f_a = \begin{cases} F_v & v \in V_D \\ -H_v & v \in V_{CO} \\ 0 & \text{else} \end{cases} \qquad \forall v \in V \tag{2.2.7}$$

$$f_{ij} \leqslant u_e w^f_{ij} \qquad \forall e = \{i,j\} \in E \tag{2.2.8}$$

$$w^f_{ij} + w^f_{ji} \leqslant w_e \qquad \forall e = \{i,j\} \in E \tag{2.2.9}$$

Note that in this model, since f and g define single-commodity flows, the flows of opposite directions cancel out. However, it might happen that along an edge $e = \{i,j\}$, f sends flow in direction (i,j) and g sends flow in direction (j,i).

Moreover, if FN and DN are required to have tree topologies, this can be enforced using the following in-degree constraints:

$$\sum_{a\in\delta^{in}(v)} w^f_a \leqslant 1 \quad \text{and} \quad \sum_{a\in\delta^{in}(v)} w^g_a \leqslant 1 \quad \forall v \in V \tag{2.2.10}$$

Upstream-demand and downstream-capacity at DPs. For each installed splitter at a DP $v \in V_D$, a single fiber in the FN is required, and the total upstream-demand at v is calculated using constraints (2.2.11). The number of downstream-fibers available at the DP v is bounded by the total number of installed splitters and their capacity, see (2.2.12). Finally, the number of available downstream fibers at DPs and COs v is also upper-bounded by u_v.

$$F_v = \sum_{t\in T} y_{t,v} \qquad \forall v \in V_D \tag{2.2.11}$$

$$G_v \leqslant \sum_{t\in T} s_t y_{t,v} \qquad \forall v \in V_D \tag{2.2.12}$$

$$G_v \leqslant u_v x_v \qquad \forall v \in V_D \tag{2.2.13}$$

$$H_v \leqslant u_v z_v \qquad \forall v \in V_{CO} \tag{2.2.14}$$

We assume that the upper bounds imposing the maximal capacity of a distribution point are non-trivial, that is

$$u_v \leqslant \sum_{t\in T} s_t J_{t,v}$$

The set of feasible 2FTTx solutions is completely described using constraints (2.2.1)–(2.2.14). This MIP model will be called the *aggregated MIP model*. It contains a large number of variables and constraints, and therefore, it is quite unrealistic that using this model one will be able to solve instances arising in the practical application. Besides, the lower bounds obtained by this model may be quite weak, due to the involved constraints of big-M type. Specifically, we mention the constraints (2.2.5), (2.2.8),(2.2.13) and (2.2.14). To overcome the problems with the weak lower bounds, in the following subsection we will first present families of strengthening valid inequalities. In the second half we will propose two Lagrangian decomposition approaches. These will enable us to solve some real-world network planning instances very close to optimality.

2.2.2 Valid inequalities for the aggregated MIP model

Connectivity constraints in the DN We already observed that the flow conservation constraints (2.2.4) together with the capacity constraints (2.2.5) define a single-commodity flow problem in the distribution network. For the problem to be feasible, for each customer $v \in V_C$, d_v units of flow need to be transported from the active ($x_k = 1, k \in V_D$) distribution points to v. This also implies that the distribution network has to be connected. The following *connectivity constraints in the DN* are therefore valid for our problem:

$$\sum_{k \in W \cap V_D} x_k + \sum_{(i,j) \in \delta^{in}(W)} w^g_{ij} \geqslant 1 \quad W \subseteq V, W \cap V_C \neq \emptyset \tag{2.2.15}$$

These constraints basically state that for each customer i there has to exist an open DP $k \in V_D$ such that they can be connected by a directed path from k to i in the subgraph of G induced by the w^g variables. These inequalities are not implied by the previous model and can be used to strengthen the LP bounds (see e.g. Ljubić et al. [2012]).

Connectivity constraints in the FN We can again observe that the flow conservation constraints (2.2.7) together with the capacity constraints (2.2.8) define a single-commodity flow problem in the feeder network. In a feasible solution, we have that for each distribution point $k \in V_D$, there need to be F_k flow units that are transported from the active central offices to k. As above, the following *connectivity constraints in the FN* are therefore valid for our problem:

$$\sum_{\ell \in W \cap V_{CO}} z_\ell + \sum_{(i,j) \in \delta^{in}(W)} w^f_{ij} \geqslant x_k \quad W \subseteq V, k \in W \cap V_D \tag{2.2.16}$$

They ensure that open COs and open DPs belong to a connected network, and even more, that for each open DP k there has to exist an open CO ℓ such that they can be connected by a directed path from ℓ to k in the subgraph of G induced by w^f variables. Also these inequalities are not implied by the previous model and can be used to strengthen the LP bounds.

Global connectivity constraints These constraints are based on the observation that the overall solution has to be connected. Furthermore, since the edges $e \in E$ can be oriented in one or the other direction, and the costs for the installation are paid only once, no matter if the same

edge is used twice (once in the FN, and once in the DN), the following directed connectivity cuts are valid and are not implied by the previously introduced cuts:

$$\sum_{\ell \in W \cap V_{CO}} z_\ell + \sum_{(i,j) \in \delta^{in}(W)} w^f_{ij} \geqslant 1 \quad W \subseteq V, W \cap V_C \neq \emptyset \tag{2.2.17}$$

These inequalities make sure that for each customer $i \in V_C$ there is a directed path between an open CO ℓ and i in the subgraph induced by w^f variables. That way, the variables w^f, originally intended to model only the arcs of the FN, are used to push up the capacities of fractional edge variables in the DN.

Lemma 2.2.1. *Constraints* (2.2.15), (2.2.16) *and* (2.2.17) *can be separated in polynomial time.*

Proof. We will explain how to separate (2.2.15), the remaining separation algorithms follow the same idea. Given the values $\tilde{x}$ and $\tilde{w}^g$ of a fractional LP solution to the aggregated MIP, we can separate constraints (2.2.15) in an auxiliary graph $G_g = (V_g, A_g)$ that is generated as follows. An additional node r is added to V and it is connected to all distribution points $v \in V_D$, i.e., $V_g = \{r\} \cup V$, $A_g = A \cup \{(r,k) \mid k \in V_D\}$. In the resulting digraph G_g, for each $v \in V_C$, there has to be a flow of value not less than 1 from r to v. We treat the values of $\tilde{w}^g_{ij}$ as the capacity on the arc (i,j) and $\tilde{x}_k$ as capacity on the arc (r,k). If the value of the maximum flow is less than one, the associated minimum cut, projected into the space of x and w^g variables, corresponds to a violated (2.2.15) inequality. □

In the following, let

$$\mathcal{G} := \left\{ z = (g, G, w^g, w) \in \mathbb{R}_+^{|A|+|V_D|} \times \{0,1\}^{|A|+|E|} \;\middle|\; z \text{ satisfies (2.2.4)-(2.2.6),(2.2.15)} \right\}$$

and

$$\mathcal{F} := \left\{ z = (f, H, F, w^f, w) \in \mathbb{R}_+^{|A|+|V_{CO}|+|V_D|} \times \{0,1\}^{|A|+|E|} \;\middle|\; z \text{ satisfies (2.2.7)-(2.2.9),(2.2.16)} \right\}$$

In Subsection 2.3.1, we illustrate how the problem can be decomposed in such a way that the subproblems associated to $\mathcal{G}$ and $\mathcal{F}$ can be treated separately.

2.3 Lagrangian Decompositions

A short introduction to Lagrangian Decomposition

We now describe the technique of Lagrangian Decomposition with a focus on solving Integer programs. The setting in which this technique is applied is usually the following. We have a model for an optimization problem that has some standard well-known combinatorial constraints, like flow constraints, knapsack constraints – we call these constraints the *simple* constraints. Additionally, we have some more constraints that destroy the simple combinatorial structure of the others – these, we call the *complicating* constraints.

Our model has the following structure:

$$\begin{aligned} (P) \qquad & \min c^T x \\ \text{subject to} \qquad & Ax \geqslant b && (2.3.1)\\ & Cd \geqslant d && (2.3.2)\\ & x_j \in \mathbb{N}_0 \qquad i \in I && (2.3.3) \end{aligned}$$

Here, $Ax \geqslant b$ will be the *complicating* constraints, while $Cx \geqslant d$ are the simple constraints. We define the Lagrangian Relaxation for some nonnegative vector $\lambda \in \mathbb{Q}^m$ by

$$\begin{aligned} (L(\lambda)) \qquad & \min c^T x + \lambda(b - Ax) \\ \text{subject to} \qquad & Cd \geqslant d && (2.3.4)\\ & x_j \in \mathbb{N}_0 \qquad i \in I && (2.3.5) \end{aligned}$$

For this decomposition to give fruitful results, $L(\lambda)$ should be much easier to solve than (P). For $\lambda = 0$, we end up with the relaxation that we get by just removing the constraints $Ax \geqslant b$. The program $L(\lambda)$ is also a relaxation for other, nonnegative values for the vector λ in the sense that for any λ, the optimal solution value for the program is a lower bound on the optimal solution value of (P).

The question is how λ should be chosen. The optimal value of λ corresponds to the optimal solution of the concave program

$$\max_{\lambda \geqslant 0} \nu(L(\lambda))$$

where ν denotes the optimal solution value. This program corresponds to the formal Lagrangian Dual of (P) with respect to $Ax \geqslant b$. Standard theorems imply that solution value for the optimal choice of Lagrangian dual multipliers λ is at least as good as the LP relaxation. The value of $L(\lambda)$ for an optimal choice of λ can be strictly better than the value of the LP relaxation of (P). It is equal to the value of the LP relaxation in the case that $L(\lambda)$ has the *integrality property*, which states that the value of an optimal solution is unchanged if we drop the integrality constraints, or in other words, if the optimal solution value of its LP relaxation is attained by an integral solution.

We now introduce our Lagrangian Decompositions. We later explain how we solve the Lagrangian dual optimization problems in Subsection 2.3.3.

2.3.1 Feeder-Distribution Decomposition

Our first Lagrangian decomposition approach decomposes the problem in the most intuitive way: the design of the FN and the design of the DN. To obtain this decomposition, we have to relax constraints that couple both networks. We proceed as follows:

1. Duplicate $y_{t,\nu}$ variables: introduce a copy of $y_{t,\nu}$ variables (denoted by $y'_{t,\nu}$) and then replace $y_{t,\nu}$ by $y'_{t,\nu}$ in equations (2.2.11).

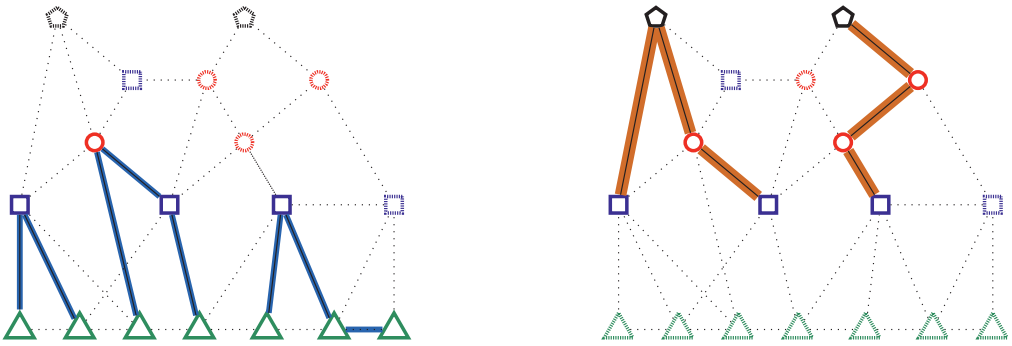
2. Duplicate x_v variables: Introduce a copy denoted by x'_v and introduce a copy of constraint (2.2.3) with both x_v and $y_{t,v}$ substituted.

3. Duplicate w_e variables: Introduce a copy denoted by w'_e and then replace w_e by w'_e in (2.2.7)-(2.2.9),(2.2.16).

4. Extend the previous model with the following inequalities, associate dual variables λ, α and β to them and relax them in a Lagrangian fashion. The corresponding dual multipliers are denoted after the inequality.

$$\begin{aligned} \sum_{t\in T} y'_{t,v} &\geqslant \sum_{t\in T} y_{t,v} && \forall v \in V_D && \dots(\lambda_v)\\ w_e &= w'_e && \forall e \in E && \dots(\alpha_e)\\ x_v &= x'_v && \forall v \in V_D && \dots(\beta_v) \end{aligned}$$

The whole problem decomposes into two subproblems that will be referred to as the *Feeder Network problem* (F) and the *Distribution Network problem* (D). This decomposition is illustrated in Figure 2.3.1.



(a) (b)

Figure 2.3.1: Lagrangian Decomposition into the Feeder and Distribution Part: (a) Distribution network (DN), and (b) Feeder network (FN). Triangles represent customers, squares represent DPs, pentagons represent COs and circles represent remaining nodes

The Feeder Network problem

After substituting $F_v := \sum_{t\in T} y_{t,v}$ for each $v \in V_D$, the feeder network subproblem is given as:

$$\begin{aligned} \text{(F)} \qquad \min \; & \sum_{v\in V_{CO}} c_v z_v - \sum_{v\in V_D} \lambda_v F_v + \sum_{a\in A} c^f_a f_a - \sum_{v\in V_D} \beta_v x'_v - \sum_{e\in E} \alpha_e w'_e \\ \text{subject to} \quad & H_v \leqslant u_v z_v && \forall v \in V_{CO} && (2.3.6)\\ & F_v \leqslant \sum_{t\in T} J_{t,v} x'_v && \forall v \in V_D && (2.3.7)\\ & \sum_{v\in V_{CO}} z_v \leqslant N_{CO} && && (2.3.8)\\ & (z, x') \in \{0,1\}^{|V_{CO}|+|V_D|}, \; (f, H, F, w^f, w') \in \mathcal{F} \end{aligned}$$

The Distribution Network problem

$$\text{(D)} \quad \min \sum_{v\in V_D}(c_v+\beta_v)x_v+\sum_{e\in E}(c_e+\alpha_e)w_e+\sum_{v\in V_D}\sum_{t\in T}(c_{t,v}+\lambda_v)y_{t,v}+\sum_{a\in A}c^g_a g_a$$

$$\text{subject to} \quad G_v \leqslant \min\{u_v x_v, \sum_{t\in T} s_t y_{t,v}\} \qquad v\in V_D \qquad (2.3.9)$$

$$y_{t,v} \leqslant J_{t,v}x_v \qquad v\in V_D \qquad (2.3.10)$$

$$\sum_{v\in V_D} x_v \leqslant N_D \qquad (2.3.11)$$

$$(x,y) \in \{0,1\}^{|V_D|}\times \mathbb{Z}_+^{|T||V_D|},\ (g,G,w^g,w)\in \mathcal{G}$$

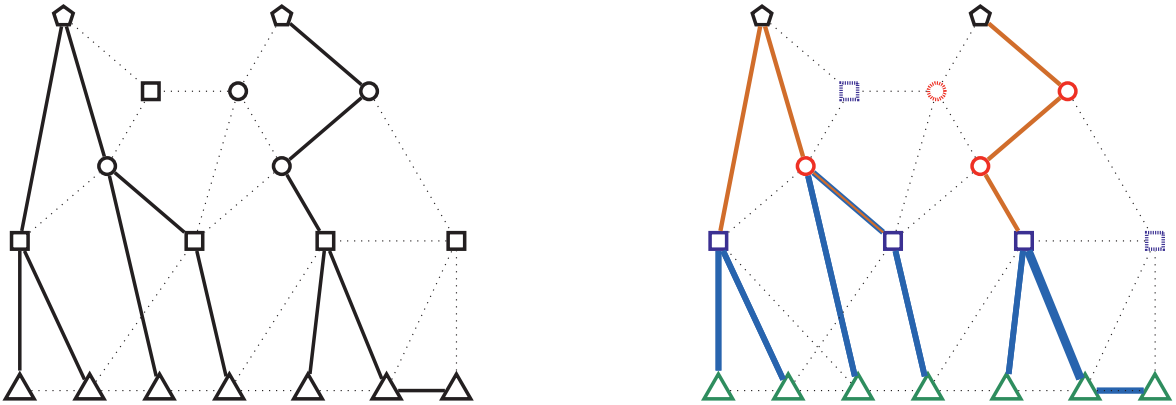
Observe that both problems, (F) and (D), are **NP**-hard. However, from a structural point of view, they are significantly simpler than the original problem, and also easier to solve from the practical perspective. The problems are generalizations of the capacitated network design problem with single source. We solve these problems using a Branch&Cut (B&C) approach whose main ingredients are outlined in Section 2.4. One of the advantages of the B&C approaches is that warm start features (i.e., initializations of upper bounds and valid inequalities) can be re-used from iteration to iteration (via solution pools and cut pools, respectively).

When solving this decomposition, we extend the model (F) with the global connectivity cuts (2.2.17) that make sure that customers are connected with open COs. These constraints restrict the set of feasible solutions for the (F) model, but they do not cut off globally optimal solutions. In addition, these constraints strengthen the Lagrangian bounds obtained by this decomposition. Finally, to make sure that global connectivity cuts are also associated to edge set-up costs, Lagrangian multipliers are initialized as $\alpha_e := -c_e$, for all $e \in E$. The remaining Lagrangian multipliers are initialized with zero values. We will refer to this decomposition as the *FD decomposition* approach or the *(F)+(D) decomposition*.

2.3.2 Fixedcharge-Flow Decomposition

In this subsection we propose an alternative Lagrangian decomposition approach in which we decompose the problem according to the classification of variables by cost types. In other words, we decompose the problem into a *Fixed-Charge* and a *Flow* subproblem. The Fixed-Charge subproblem captures all set-up costs associated to fixed-charge variables, w_e, x_v and z_v. These costs will dominate the total costs on our instances. On the other hand, the Flow subproblem captures all flows costs associated to the splitter-installation variables $y_{t,v}$ and the flow variables f_a and g_a.

To constraints that contain variables of both types we associate dual variables λ, μ and ν.



(a) (b)

Figure 2.3.2: Lagrangian Decomposition into (a) Fixed-Cost and (b) Variable-Cost. In a), only the topological considerations remain. In b), we can see the flows in the two sub-networks marked in orange and blue.

Then, we relax them in Lagrangian fashion:

$$
\begin{aligned}
y_{t,v} &\leqslant J_{t,v} x_v && t \in T, v \in V_D && \dots (\lambda_{t,v}) \\
G_v &\leqslant u_v x_v && v \in V_D && \dots (\mu_v) \\
H_v &\leqslant u_v z_v && v \in V_{CO} && \dots (\mu_v) \\
g_{ij} &\leqslant u_e w^g_{ij} && \{i,j\} \in E && \dots (\nu^g_a) \\
f_{ij} &\leqslant u_e w^f_{ij} && \{i,j\} \in E && \dots (\nu^f_a)
\end{aligned}
$$

Basically, we obtain two subproblems, one which decides which part of the network we use and another which decides how the demands are routed within the resulting network. We basically have relaxed every constraint that couples the flow variables to the network design variables.

As one easily observes, all the relaxed constraints are of big-M type, so one can hope to end up with good subproblems and fix the violation of the coupling restrictions through the Lagrangian multipliers. Interestingly, these constraints are less likely to be violated the larger the inherent constants are, so one can expect that many of them are already fulfilled even without a long search for the correct dual multipliers.

The two subproblems we obtain will be refered to as the *fixed-charge subproblem (FC)* and the *Flow subproblem (FP)*. Figure 2.3.2 illustrates the decomposition approach.

The Fixed-Charge subproblem (FC)

$$\begin{aligned}
(\mathrm{FC}) \quad \min & \sum_{v\in V_{CO}} (c_v - \mu_v u_v) z_v + \sum_{v\in V_D} (c_v - (\sum_{t\in T} \lambda_{t,v} J_{t,v} + \mu_v u_v)) x_v + \\
& + \sum_{e\in E} c_e w_e - \sum_{a\in A} u_a \left(\nu_a^f w_a^f + \nu_a^g w_a^g\right) \\
\text{subject to} \quad & \sum_{v\in V_D} x_v \leqslant N_D && (2.3.12)\\
& \sum_{v\in V_{CO}} z_v \leqslant N_{CO} && (2.3.13)\\
& w_{ij}^f + w_{ji}^f \leqslant w_e \qquad e=\{i,j\}\in E && (2.3.14)\\
& w_{ij}^g + w_{ji}^g \leqslant w_e \qquad e=\{i,j\}\in E && (2.3.15)\\
& (x,z,w,w^f,w^g) \in \{0,1\}^{|V_D|+|V_{CO}|+|E|+2|A|}
\end{aligned}$$

The Flow subproblem (FP)

$$\begin{aligned}
(\mathrm{FP}) \quad \min & \sum_{v\in V_D}\sum_{t\in T} (c_{t,v} + \lambda_{t,v}) y_{t,v} + \sum_{v\in V_D} \mu_v G_v + \sum_{V\in V_{CO}} \mu_v H_v + \\
& + \sum_{a\in A} (c_a^f + \nu_a^f) f_a + \sum_{a\in A} (c_a^g + \nu_a^g) g_a
\end{aligned}$$

subject to

$$\sum_{a\in\delta^{in}(v)} g_a - \sum_{a\in\delta^{out}(v)} g_a = \begin{cases} d_v & v\in V_C \\ -G_v & v\in V_D \\ 0 & v\in V_{CO}\cup V_O \end{cases} \qquad v\in V \qquad (2.3.16)$$

$$\sum_{a\in\delta^{in}(v)} f_a - \sum_{a\in\delta^{out}(v)} f_a = \begin{cases} F_v & v\in V_D \\ -H_v & v\in V_{CO} \\ 0 & v\in V_C\cup V_O \end{cases} \qquad v\in V \qquad (2.3.17)$$

$$F_v = \sum_{t\in T} y_{t,v} \qquad v\in V_D \qquad (2.3.18)$$

$$G_v \leqslant \sum_{t\in T} s_t y_{t,v} \qquad v\in V_D \qquad (2.3.19)$$

$$(f,g,F,G,H,y) \in (R_+^{2|A|+2|V_D|+|V_{CO}|}, Z_+^{|T||V_D|})$$

Observe that (FC) is a trivial subgraph selection problem: edge selections can simply be made by looking at the sign of the coefficient. Similarly, we can select the nodes with the largest negative coefficients greedily until we reach the bounds on the total number of DPs

and COs. Intuitively, there is not much improvement if the problem is decomposed into one hard and one trivial subproblem, as either the hard problem is as hard as the original one or the easy problem does not capture the difficulties properly, which leads to either many dual steps or a weak relaxation. To strengthen the models, we insert the connectivity constraints (2.2.15)-(2.2.17) in the original MIP model. After the Lagrangian relaxation, these end up in the (FC) subproblem. For this reason, we can strengthen the fixed-charge subproblem using valid inequalities which ensure connectivity. Due to the large number of constraints, these are separated as lazy constraints, meaning that we only add them to the model when our current solution violates them. That way, we end up with a non-trivial FC-subproblem which makes this decomposition approach useful.

Despite the fact that also in this decomposition both subproblems, (FC) and (FP), are **NP**-hard, they are significantly simpler than the original problem. The problem (FP) is **NP**-hard since the packing subproblem has to be solved at each of the installed distribution points. After adding the connectivity constraints, the problem (FC) becomes **NP**-hard, since it combines the structure of the cardinality constrained Steiner arborescence problem with node and arc weights. To solve the subproblem (FC), we develop a Branch&Cut algorithm (see Section 2.4) in which solution- and cut-pools are used as warm start features. The subproblem (FP) is solved as a compact MIP model by a black-box MIP solver with the additional advantage that solution pools are used to initialize starting solutions in each Lagrangian iteration. We will refer to this decomposition as the Fixedcharge-Flow decomposition approach or the *(FC)+(FP) decomposition*.

Valid Inequalities

When solving the (FC) subproblem, notice that the only way more than one DP/CO is opened is if the costs of the DPs/COs become negative due to the setting of the corresponding dual multipliers. Therefore, the bounds obtained by solving the (FC) subproblem can further be strengthened by additional inequalities that make sure that the capacity of open DPs/COs is sufficient to service customer demands. These inequalities can also reduce the number of iterations of the Lagrangian decomposition. The following inequalities are used for this purpose:

$$\sum_{v \in V_D} u_v x_v \geqslant \sum_{v \in V_C} d_v \tag{2.3.20}$$

$$\sum_{v \in V_{CO}} \left\{ \max_{t \in T} s_t \right\} u_v z_v \geqslant \sum_{v \in V_C} d_v \tag{2.3.21}$$

$$\sum_{v \in V_{CO}} u_v z_v + \sum_{v \in V_D} x_v \sum_{t \in T} (s_t - 1) J_{t,v} \geqslant \sum_{v \in V_C} d_v \tag{2.3.22}$$

Constraints (2.3.20) make sure that a sufficient number of downstream fibers is provided, and the slightly weaker constraints (2.3.21) make sure that a sufficient number of transceivers is installed at COs, so that customer demands can be satisfied (assuming the highest possible splitting ratio). Finally, constraints (2.3.22) combine the latter two using the fact that each splitter requires exactly one fiber coming from a CO (and therefore we have term $(s_t - 1)$ in the second summation).

Strengthening by Splitter-Counting One can aggregate variables $y_{t,v}$ as follows:

$$\sum_{v \in V_D} y_{t,v} = y_t \qquad t \in T$$

These newly introduced integer variables y_t count the number of splitters of type $t \in T$ installed across all DPs. Adding the latter equality into the aggregated MIP yields no benefits. However, in the context of our second Lagrangian relaxation, it helps in solving the (FC) subproblem. After associating dual multipliers π_t and relaxing these constraints, the new variables are added to (FC) and an additional term of $-\sum_{t \in T} \pi_t y_t$ is added in the objective function.

Then, the following inequalities are used to strengthen the (FC) subproblem:

$$\sum_{t \in T} s_t y_t \geqslant \sum_{v \in V_C} d_v \tag{2.3.23}$$

$$\sum_{v \in V_{CO}} u_v z_v + \sum_{t \in T} (s_t - 1) y_t \geqslant \sum_{v \in V_C} d_v \tag{2.3.24}$$

$$\sum_{v \in V_D} J_{t,v} x_v \geqslant y_t \qquad \forall t \in T \tag{2.3.25}$$

$$\sum_{v \in V_{CO}} u_v z_v \geqslant \sum_{t \in T} y_t \tag{2.3.26}$$

2.3.3 Generic Lagrangian Framework

We now describe the generic Lagrangian decomposition framework that is applied to both approaches. In this framework, lower bounding and upper bounding procedures are incorporated. Lower bounding procedures are based on solving lower bounds of associated MIP models, and upper bounding procedures are heuristics that we describe below. For each of the proposed decomposition approaches, we develop appropriate heuristics. They solve each of the subproblems independently, using the current Lagrangian multipliers in the objective function. Hence, in all the following heuristics, solving the subproblem always refers to the Lagrangian-modified objective functions, unless it is stated differently.

Relaxing the constraints linking the feeder and the distribution network part as described in Subsection 2.3.1, we obtain the Lagrangian function

$$L(\Lambda) := L_{F+D}(\Lambda) = L_F(\Lambda) + L_D(\Lambda),$$

where $\Lambda := (\lambda, \alpha, \beta) \in \mathbb{R}_+^{V_D} \times \mathbb{R}^E \times \mathbb{R}^{V_D}$ is the vector of Lagrangian dual multipliers for the (in)equalities linking the variables y, w, and x to their copies y', w', and x', respectively. The two functions $L_F(\Lambda)$ and $L_D(\Lambda)$ yield the optimal solution value of the two integer linear problems (F) (augmented with global connectivity cuts (2.2.17)) and (D), respectively, for the given dual multipliers Λ.

Analogously, the decomposition into fixed-charge and flow cost dependent variables described in Subsection 2.3.2 yields the Lagrangian function

$$L(\Lambda) := L_{FC+FP}(\Lambda) = L_{FC}(\Lambda) + L_{FP}(\Lambda),$$

with dual multipliers $\Lambda := (\lambda, \mu, \nu, \pi) \in \mathbb{R}_+^{2|V_D|+2|T|+|V_{CO}|+2|E|}$ (assuming we introduce the extra splitter count variables y_t as described in Section 2.3.2). Here, $L_{FC}(\Lambda)$ represents the optimal solution value of (FC) after adding constraints (2.2.15)-(2.2.17) and (2.3.20)-(2.3.26), while $L_{FP}(\Lambda)$ represents the optimal solution value of (FP).

It is well known that for each dual vector Λ the value $L(\Lambda)$ is a lower bound for the optimal value of original aggregated model (2.2.1)–(2.2.14) and, hence, also $L^* := \max_\Lambda L(\Lambda)$ is a valid lower bound. As there are only finitely many (basic) solutions to the original model and to each of the subproblems (D), (F), (FC), and (FP), the corresponding dual functions L_F, L_D, L_{FC} and L_{FP} are piece-wise linear and concave in Λ. Hence, convex function optimization techniques can be applied in order to find dual multipliers Λ^* that yield the best possible lower bound L^*.

Finding the dual vectors

In our implementation, we employ a bundle method to determine the dual vectors.

Bundle methods typically converge relatively fast requiring only a few evaluations of the dual function(s), which is very attractive in our application, where each evaluation (in principle) requires the solution of an integer linear program. Furthermore, they permit the use of an independent bundle of subgradients for each of the two sub-functions L_F and L_D or L_{FC} and L_{FP} involved in the respective Lagrangian function, potentially leading to a further reduction in (sub-)function evaluations. Finally, general purpose implementation of these methods are available, such as ConicBundle (Helmberg and Kiwiel [2002]; Helmberg [2012]), which has already proved its practicability and efficiency in the solution of large-scale problems (see e.g. Helmberg [2009]).

The basic theory of bundle methods is explained in Hiriart-Urruty and Lemaréchal [1993] and Bonnans et al. [2003]. In principle, given a starting point for the dual multipliers, the bundle method iteratively determines the next candidate as an optimizer of a quadratic model with the current point as a stability center and dual constraints stemming from a set (bundle) of previous optimal solutions. If the value of the optimal solution of this quadratic model improves sufficiently over the value at the stability center, the method performs a descent step and proceeds. Otherwise, a null step not changing the stability center but improving the quadratic model with the new subgradient is performed.

For the initial dual multipliers and after each descent step of the bundle algorithm, we apply one of the heuristics described in the following sections to compute feasible primal solutions for the overall problem.

Solving the evaluation subproblems

In order to reduce the run time of the two proposed Lagrangian relaxation approaches in practice, we also decided to avoid the solution of the integer linear programs (F), (D), (FC), and (FP) via Branch&Bound in the evaluation of the corresponding (sub-)functions. Instead, we stop after processing the root node of the corresponding Branch&Bound trees. Conceptually, this also can be regarded as the solution of an (appropriately defined) linear relaxation of the respective subproblems, namely a relaxation that includes all those constraints that are implicitly enforced via the simple preprocessing techniques and cutting planes of the ILP solver at the root node

and that relaxes all other integrality constraints. Unfortunately, however, it was necessary to disable most of the heuristic preprocessing and cutting plane generation techniques that are implemented in the ILP solver in order to avoid inconsistencies in their application and resulting numerical instabilities in the bundle algorithm. The resulting bounds will, of course, be weaker than those that can be obtained by optimally solving the integer programming subproblems. Yet, the bounds are very satisfactory from a practical point of view and much faster to compute.

FDF Heuristic

For the first decomposition approach, we develop a heuristic that we refer to as the *Feeder-Distribution-Feeder (FDF) Heuristic*. Pseudo-code of this heuristic is given in Algorithm 1 on the following page. The heuristic consists of three stages: in the first stage we solve the feeder subproblem (F) extended by global connectivity cuts (2.2.17). That way, we obtain a preliminary topology of our network that makes sure that all customers are connected to each other and to at least one open CO. In the second stage, the edges of this network are used without cost (compare Step 7) for solving the distribution subproblem. The last stage is a transition from the distribution subproblem into the feeder subproblem. The Solution of (D) is denoted by S_D. Edges that belong to S_D are now taken in the solution of (F). In addition, open DPs and their demands are uniquely determined by S_D and the values for F_v and x'_v are used as inputs for the feeder subproblem (F). If the last stage returns a feasible solution for (F), after merging it with S_D, we obtain a feasible 2FTTx solution. The advantage of calling the (F) subproblem at the beginning is that by incorporating the global connectivity requirements, it provides a global view of the problem.

Each of the subproblems in this procedure is solved as a Branch&Cut reusing the cuts and feasible solutions from the previous iterations. Since these B&C algorithms are called as heuristic procedures, we do not search for the optimal solution, but we stop the execution of these frameworks as soon as two feasible solutions are found.

FCF Heuristic

Pseudo-code of this heuristic is given in Algorithm 2 on page 68. This heuristic first solves the fixed-charge subproblem (FC), including all connectivity constraints (2.2.15)-(2.2.17), and inequalities (2.3.20)-(2.3.26). That way, the topology of the network is determined, and it only remains to make capacity and routing decisions. In order to do so, we create a subgraph $\tilde{G}$ of G, induced by the given topology, and resolve the whole aggregated MIP on it. This aggregated MIP also contains constraints (2.2.15)-(2.2.17) and (2.3.20)-(2.3.22), but solving it is usually much faster than solving the original aggregated MIP, due to the fixing of variables.

2.4 Computational results

2.4.1 Branch&Cut algorithms

The aggregated MIP with connectivity constraints (2.2.15)-(2.2.17), the distribution subproblem (D), the feeder subproblem (F) and the fixed-charge subproblem (FC) are all solved using

Algorithm 1 Feeder-Distribution-Feeder Heuristic.

Solve the feeder subproblem (F) (incl. global connectivity cuts (2.2.17))
if (F) has no incumbent **then**
  End heuristic
else
  Let S_F be the best solution of (F)
end if
Initialize distribution subproblem (D) with edge costs set to zero for all $e \in S_F$
Solve the distribution subproblem (D)
if (D) has no incumbent **then**
  End heuristic
else
  Let $S_D := (\tilde{g}, \tilde{G}, \tilde{w}^g, \tilde{w}, \tilde{y}, \tilde{x})$ be the best solution of (D)
end if
Solve the feeder subproblem (F) with edge costs set to c_e for all $e \in E$ and:
  $w'_e \geqslant \tilde{w}_e, e \in E, \quad F_v := \sum_{t \in T} \tilde{y}_{t,v}$ and $x'_v := \tilde{x}_v, v \in V_D$.
if (F) has no incumbent **then**
  End heuristic
else
  Let S_{F2} be the best solution of (F)
end if
Return merged solution: S_{F2} plus S_D.

Branch&Cut algorithms. In this section we explain the main ingredients of these algorithms.

Separation of Connectivity cuts

Connectivity constraints are separated using maximum flows, as explained in the proof of Lemma 2.2.1. The maximum flow is calculated using the push-relabel procedure (see, for example Cherkassky and Goldberg [1997]). To speed-up the separation, we exploit the idea of backward cuts in order to detect more diverse cuts, further away from the artificial root node. The idea, applied to constraints (2.2.15), for example, is as follows:

In a first step, the arcs of the original graph are reversed. Then, the maximum flow from a customer towards the artificial root node is calculated. If violated, the arcs of the associated minimum cut are reversed and the corresponding connectivity cut is added to the model. We enforce generation of sparse cuts by adding an ε value to each edge, and use nested cuts to generate more cuts in fewer iterations (see e.g. Ljubić et al. [2006]). At each call of the separation callback, we generate a new random ordering of the customers in order to avoid separating the same customers over and over. The cut separation is not executed for another customer during a run if the number of already generated cuts exceeds 100. The cut separator is called at the root node of the Branch&Bound tree and at every further node with quadratic index. It is also used to check feasibility of integral solutions, in the course of which lazy constraints are generated.

Algorithm 2 FixedCharge-Flow Heuristic

1: Solve the Fixed-Charge subproblem (FC)
2: **if** (FC) has no incumbent **then**
3: End heuristic
4: **else**
5: Let S_{FC} be the best solution of (FC)
6: **end if**
7: Let $\tilde{G}$ be a subgraph of G induced by the edges of S_{FC}
8: Solve aggregated MIP on $\tilde{G}$ with $w_e := 1$, for all $e \in S_{FC}$.
9: **if** aggregated MIP has no incumbent **then**
10: End heuristic
11: **else**
12: **Return** the best aggregated MIP solution
13: **end if**

Cut pools and warm start

Since B&C algorithms are called in each iteration of the Lagrangian decomposition approach, for the implementations of the (F), (D) and (FC) model we use cut pools to store previously detected violated cuts and reuse them in each new iteration in a warm-start fashion. This is possible because from iteration to iteration, only the objective function changes due to the new dual multipliers, and the polytopes associated to feasible solutions remain the same. Hence, connectivity cuts (2.2.15)-(2.2.17) detected in earlier iterations are still valid.

Regarding the initialization of upper bounds at the beginning of the B&C execution, we use the CPLEX `MIPstart` feature - the best solution among the ones found in previous iterations is set as the initial feasible solution.

Primal heuristics

CPLEX default heuristics were activated. For the Branch&Cut calls within the Lagrangian heuristics, the emphasis is set to finding feasible solutions. In addition, we enhance the search for upper bounds of the (FC) model by our own upper bounding procedure. This procedure is a LP-rounding heuristics based on the following idea:

i) A set of *terminals* is determined, depending on the fractional values of the x- and z-variables.

ii) A Steiner tree is built to connect those terminals.

iii) The values of the remaining variables are calculated using an auxiliary MIP in which the Steiner tree edges are fixed to one ($w_e := 1$), and the remaining edges are fixed to zero ($w_e := 0$).

To determine the subset of terminals, we apply an LP-rounding technique: Terminals are customers plus all nodes $v \in V_D \cup V_{CO}$ whose corresponding LP-values x_v and z_v are greater or equal to a given threshold π. In the default implementation, π is set to $1/2$.

To calculate the Steiner tree on a given set of terminals, we apply the *distance network heuristic* (see e.g. Mehlhorn [1988]): First, a distance network is built which is a complete graph containing terminals and whose edge weights are the lengths of the shortest paths between them. These shortest paths are calculated with respect to the weights associated to w_e variables. Then, a minimum spanning tree (MST) on the distance network is found, the paths are mapped back in the original network, one more MST is built to avoid cycles, and finally leaves that are not terminals are pruned from the tree.

MIP initialization

All valid inequalities mentioned for models (F), (D), (FC), and the aggregated MIP, are added at the very beginning to the MIP, except the connectivity constraints (2.2.16)-(2.2.17) that are dynamically separated. In addition, to speed-up the separation, we add some more valid inequalities. The models (D) and the aggregated MIP models are additionally initialized with:

$$\sum_{(i,j)\in A, i\neq k} w^g_{ij} + x_j \geqslant w^g_{jk}, \text{ for all } (j,k) \in A,\ j \in V_D$$

$$\sum_{(i,j)\in A, i\neq k} w^g_{ij} \geqslant w^g_{jk}, \text{ for all } (j,k) \in A,\ j \notin V_D$$

$$\sum_{(i,j)\in \delta^{in}(j)} w^g_{ij} \geqslant 1, \text{ for all } j \in V_C$$

The models (F), (FC) and the aggregated MIP models are additionally initialized with:

$$\sum_{(i,j)\in A, i\neq k} w^f_{ij} + z_j \geqslant w^f_{jk}, \text{ for all } (j,k) \in A,\ j \in V_{CO}$$

$$\sum_{(i,j)\in A, i\neq k} w^f_{ij} \geqslant w^f_{jk}, \text{ for all } (j,k) \in A,\ j \notin V_{CO}$$

$$\sum_{(i,j)\in \delta^{in}(j)} w^f_{ij} \geqslant x_j, \text{ for all } j \in V_D$$

$$\sum_{(i,j)\in \delta^{in}(j)} w^f_{ij} \geqslant 1, \text{ for all } j \in V_C$$

Implementation details

The B&C algorithms were implemented in C++ using the CPLEX™ 12.4 callable library. All the experiments were performed on AMD 6-core-machines with 8GB RAM. Our cut separators are thread-safe and we run CPLEX in a multi-threaded way to exploit the parallel computational power of newer processors. For the separation of constraints (2.2.15)-(2.2.17), we use the max-flow implementation by Goldberg [2012]. The ConicBundle algorithm of Helmberg and Kiwiel [2002], available at Helmberg [2012], is used to solve the convex optimization problem of finding the optimal Lagrangian dual multipliers. In both decomposition approaches, we use two independent bundles of subgradients to describe the dual functions corresponding either to the two problems (F) and (D) in the (F)+(D)-Decomposition or to the two problems (FC) and (FP)

in the (FC)+(FP)-Decomposition. In order to keep the number of function evaluations small, we use a large maximum bundle size of 100 for both dual functions in both decomposition approaches.

2.4.2 Benchmark instances

We consider a set of nine benchmark instances of different sizes originating from the German research project FTTx-Plan [2012]. These instances correspond to typical regional fiber deployment planning problems – both FTTH and FTTB – in mostly urban regions that can be covered by 1 to 15 central offices. Some of our benchmark instances correspond to real-world planning problems provided by industry partners. For the other instances, the underlying networks are generated from publicly available street network information by considering realistic scenarios of potential customers, distribution points, and central offices in a region of typical size and creating potential connections along one or both sides of street segments depending on the street type, from the customer locations to the closest streets, and appropriate interconnection points and edges at crossings and joins. Figure 2.4.1 and Figure 2.4.2 on page 76 show exemplary instances E_1 and E_3 embedded in Google Maps. Costs and capacities are obtained by mapping the very complex real-world network component costs, parameters, and installation costs of a typical PON system to the simpler cost and capacity model used in our optimization model.

Figure 2.4.1: An example instance. Squares, triangles and circles represent potential COs, potential DPs and customers, respectively. Maps courtesy of Google Maps.

Table 2.1 on the following page provides the most important parameters of these benchmark instances. The number of customer locations to be served ranges from 36 in the smallest instance to 3862 in the largest one. The average fiber demand per customer location ranges from 2.0 to 13.5. The total number of edges, which correspond to the trails or street segments that may be

inst	$\|V\|$	$\|E\|$	$\|V_D\|$	$\|V_{CO}\|$	$\|V_C\|$	$\sum d_v$	$\overline{c_e}$	$\overline{c_{DP}}$	$\overline{c_{CO}}$
E_1	637	826	97	4	36	488	5878	4600	418670
E_2	1315	1434	143	5	88	278	5341	4576	509750
E_3	1675	1730	99	5	552	2290	637	3433	413156
E_4	2271	1419	494	4	349	717	1039	1500	418670
E_5	6750	7352	520	11	571	5006	672	3186	305454
E_6	6750	7352	520	11	571	5006	672	3186	300000
E_7	4110	4350	224	6	1072	4164	635	3512	466927
E_8	4227	4484	314	5	1379	5542	3483	3417	477505
E_9	11544	12478	875	15	3862	14088	3326	3274	522729

Table 2.1: Overview of basic instance properties.

used by the network, ranges from approximately 800 in the smallest instance to approximately 12,500 in the largest one.

The benchmark instances used in our experiments correspond to so-called greenfield planning problems, where edges, DPs, and COs (mostly) need to be build from scratch. Accordingly, the fixed-charge costs associated with the installation of an edge, of DPs, or of COs also contain the cost for trenching and installing ducts, cabinets, or underground closures. Table 2.1 also shows the average values of these fixed setup costs in our instances. The fixed setup costs for COs and DPs depend on the device type (underground closure vs. street cabinet DP, for example) and on its location and vary only moderately among the different potential CO and DP locations. The fixed setup cost of the edges, on the other hand, also depends linearly on the length of the edges and varies a lot within each instance, ranging from 0 for edges connecting co-located nodes to 45 times the average (instance E_4) or 20% of the average setup cost a CO (instances E_8, E_9). In general, the average fixed-charge setup cost of an edge is of the same order as the average setup cost of a DP.

The fiber installation costs depend linearly on the length of the edges in all instances. In the distribution network a fiber installation typically uses a larger number of smaller cables and ducts with a higher fraction of dead (i.e., unused) fibers than this is the case in the feeder network. In order to account for this fact, the fiber installation costs in the distribution network are larger than those in the feeder network in our instances. For the smaller instances E_1, E_2 and E_4, distribution fibers cost approximately 5.3 times as much as feeder fibers per kilometer. In the other instances, they cost approximately 1.3 times as much. The fixed-charge setup costs of the edges, however, highly dominate the costs for installing fibers along the edges in our greenfield planning problems. The fixed setup cost of an edge is approximately 3,000 times the cost of a single feeder fiber installed on this edge.

In all instances, we consider the same 5 splitter types with splitting ratios of 2, 4, 8, 16, and 32, and cost 161, 272, 352, 427, and 890 per device, respectively. Thus, the cost of a 1:32 splitter ranges between 450 and 4,000 times the average feeder fiber cost and between 20% and 60% of the average DP setup cost.

In the solutions found by our algorithms, the total fixed-charge setup costs for the edges, DPs, and COs clearly dominate the total flow dependent cost for installing feeder and distribution

fibers and splitters. The ratio between fixed-charge costs to flow dependent costs ranges from approximately 18:1 in instances E_5 and E_6 to 150:1 in instance E_2. More details on the considered technical and managerial aspects and the methodology for the generation of the original benchmark instances can be found in Martens et al. [2009, 2010] and Orlowski et al. [2011].

2.4.3 Computations

In Table 2.2 on the following page, we provide a comparison of the two decomposition approaches against the two variants of the aggregated model, one that is implemented as a Branch&Cut approach with cuts (2.2.15)-(2.2.17), denoted by *Aggr.MIP+Cuts*, and one that is a compact aggregated MIP formulation, denoted by *Aggr.MIP*. A time limit of two hours has been imposed to all four approaches. However, the Lagrangian decomposition approaches typically converged much faster. The column *Best UB* shows the objective value of the best solution found among all four approaches. For the two decomposition approaches (denoted by (F)+(D) Decomp. and (FC)+(FP) Decomp., resp.), we report the final gap obtained at the end of the last iteration (*gap*[%]), the gap of the final lower bound with respect to the global upper bound reported as Best UB (gap_{UB}[%]), the total number of Lagrangian iterations (#It), the total number of subproblem evaluations within the bundle method (#Ev), and the total running time (t [s]). Asterisk next to the running time denotes that the approach did not converge within two hours, and the reported values are obtained in the last iteration within this time limit. For the two variants of the aggregated model, instead of the number of iterations, we report the total number of Branch&Cut nodes explored within the given time limit (#Nodes). For comparison, we also report in the column gap_L the relative gap obtained with the Aggr.MIP+Cuts-approach within the time required by (FC)+(FP) Decomp. to terminate and in column t_L the time needed by Aggr.MIP+Cuts to reach the same gap as the (FC)+(FP) decomposition at termination.

Comparing the results provided in Table 2.2 on the following page, we find that none of the two aggregated MIP approaches completed within two hours. For instance E_8, the aggregated MIP approach without cuts even failed exceeding the available memory. Furthermore, we observe that the aggregated MIP approach without cuts exhibits the worst performance. For three out of nine instances, no feasible solution is found within two hours, and for three out of remaining six, the upper bounds were above 30%. We find that in these cases the gaps are mainly caused by the poor quality of the bounds produced by this approach. For all instances, the number of explored Branch&Bound nodes is at least a six-figure number (except for the instance E_8 that exceeds the memory limit).

The addition of connectivity cuts clearly improves the performance of the aggregated MIP model: The number of explored Branch&Bound nodes reduces by one to two orders of magnitude, and both the gaps and the solutions obtained after two hours are significantly improved. However, for the large instances E_5, E_6 and E_9 with more than 5,000 edges, no feasible solutions are found using the approach Aggr.MIP+Cuts.

In contrast to the aggregated MIP approaches, both decomposition approaches terminate much earlier with strong lower and upper bounds. The overall gaps remain below 4% in all cases. Comparing the running times of the two decomposition approaches, we observe that (FC)+(FP) performs slightly better. Its average resp. median running time is 2369.89 resp. 1779 seconds,

Inst	Best UB	(F)+(D) Decomp. *gap*[%]	gap_{UB}[%]	#It	#Ev	t [s]	(FC)+(FP) Decomp. *gap*[%]	gap_{UB}[%]	#It	#Ev	t [s]
E_1	1638530.00	0.98	0.98	1	10	444	0.97	0.97	8	10	576
E_2	2104409.50	0.65	0.65	8	78	*6872	0.51	0.51	4	12	430
E_3	1321696.90	1.67	1.67	5	49	4888	1.65	1.65	6	10	512
E_4	741505.59	1.31	1.27	2	8	917	1.23	1.18	7	10	273
E_5	2153458.28	3.57	3.33	1	10	2365	3.34	3.34	9	19	4776
E_6	2173096.57	3.37	3.15	1	16	4150	3.25	3.25	5	13	3362
E_7	2299736.60	2.14	2.14	3	20	2160	2.36	2.35	8	10	2435
E_8	10857507.77	0.63	0.53	1	10	1478	0.56	0.56	6	10	1779
E_9	29992128.83	0.97	0.83	2	12	*7149	0.73	0.73	3	4	*7186

Inst	Best UB	Aggr.MIP+Cuts *gap*[%]	gap_{UB}[%]	#Nodes	gap_L[%]	t_L[s]	Aggr.MIP *gap*[%]	gap_{UB}[%]	#Nodes
E_1	1638530.00	0.24	0.23	51283	0.72	206	33.73	33.73	1294596
E_2	2104409.50	0.05	0.05	75182	0.34	107	35.46	35.39	1225815
E_3	1321696.90	1.15	1.05	124355	1.46	238	3.07	2.74	989764
E_4	741505.59	0.93	0.93	103538	1.10	166	5.35	4.54	712821
E_5	2153458.28	—	2.88	2952	—	>30h	—	26.52	214674
E_6	2173096.57	—	2.76	2199	—	>30h	33.01	26.38	190280
E_7	2299736.60	2.70	1.75	35673	2.92	13393	8.20	6.66	367052
E_8	10857507.77	0.54	0.44	35938	—	3478	—	12.52	*33733
E_9	29992128.83	—	3.13	0	—	>30h	—	10.45	120503

Table 2.2: Comparison of the two decomposition approaches, the Branch&Cut for the aggregated model and the compact aggregated model.

compared to an average (median) running time of 3380.33 resp. 2365 seconds for (F)+(D). The quality of the solutions obtained with the two approaches is similar.

For the smaller instances, where the Aggr.MIP+Cuts-approach was able to find feasible solutions, the final gaps obtained with this approach are slightly smaller than the ones obtained by the decomposition approaches. In these cases, the aggregated approach benefits from solving a single global model of the problem, which permits to fully exploit all optimization potentials via branching, while both decomposition approaches operate on pairs of two independent submodels that are coupled only rather loosely via Lagrangian multipliers. However, we emphasize that the main purpose of the proposed Lagrangian decomposition approaches is to compute strong valid lower and upper bounds for large problem instances. This means that embedding these decomposition approaches into a (coordinated) Branch&Bound framework would further improve the obtained bounds and solutions.

To take a closer look at the performance of the proposed approaches, Figure 2.4.3 on page 77 and Figure 2.4.4 on page 78 and Figure 2.4.5 on page 78 show the progress of the lower and the upper bounds for instances E_4, E_5 and E_6, respectively. We observe that both decomposition approaches already reach very strong lower and upper bounds within only several minutes. In the remaining time, there is only a little progress in these values until the ConicBundle method converges (in which case the Lagrangian multipliers remain unchanged and the algorithm

terminates). The two aggregated MIP models, on the other hand, improve the bounds constantly but very slowly as more Branch&Cut nodes are explored. Also, we observe that there is a significant improvement in the quality of both lower and upper bounds when adding connectivity constraints. If Aggr.MIP+Cuts happens to find feasible solutions, it finds them relatively early in the exploration in the Branch&Bound tree. For the aggregated MIP approach without cuts, on the other hand, good solutions are found either relatively late in the Branch&Cut process, or are not found at all.

When analyzing the performance of the (F)+(D) decomposition, we came to the conclusion that its convergence and the overall performance strongly depend on the initial values of Lagrangian multipliers. This can be seen from Figure 2.4.6 on page 79, which shows the progress of lower and upper bounds for the instance E_1 with three different initializations of Lagrangian multipliers α_e:

i) the edge installation costs are fully charged to feeder subproblem ($\alpha_e = -c_e$, $e \in E$),

ii) the edge installation costs are fully charged to the distribution subproblem ($\alpha_e = 0$, $e \in E$),

iii) half of the edge installation costs charged to both the feeder and to the distribution subproblem ($\alpha_e = -c_e/2$, $e \in E$).

All other Lagrangian multipliers are initialized to zero. The presented results indicate that a bad initialization of the Lagrangian multipliers can drastically slow down the overall performance.

The success of the (FC)+(FP) decomposition over the (F)+(D) decomposition can be explained by the global connectivity constraints added to the fixed-charge subproblem. This can be seen from Figure 2.4.7 on page 79, where we show the progress of lower and upper bounds of the (FC)+(FP) decomposition with and without adding global connectivity constraints to the fixed-charge subproblem.

The results indicate that the global cuts are not only crucial for obtaining high quality lower bounds, but also for obtaining feasible solutions. When global cuts are turned off, no upper bound was found within 800 seconds, whereas a high quality solution is obtained in less than 100 seconds, otherwise. Similar behaviors to the ones reported in Figure 2.4.6 on page 79 and Figure 2.4.7 on page 79 were also observed for the remaining instances.

Recall at this point that all benchmark instances considered in this study stem from greenfield planning problems, where the setup costs of the edges include trenching costs and, thus, constitute the dominant share of the overall network cost. The impact of the global connectivity constraints may be smaller for instances stemming from planning problems where (mostly) existing edges can be used without or with only very small setup costs. Thus, the overall efficiency of the (FC)+(FP) decomposition approach observed in our experiments for greenfield planning instances may deteriorate for instances where the fixed setup cost incurred by trenching, placing closures and cabinets, opening central offices, and performing the setup activities no longer dominate the flow dependent hardware costs for fibers and splitters in the networks.

2.5 Conclusions

In this chapter, we have proposed a new combinatorial optimization problem that models a detailed deployment of passive optical networks. To solve the problem, four mixed-integer-

programming approaches were proposed: two of them consider a MIP model and solve it either as a compact MIP, or by a Branch&Cut algorithm (by adding additional valid inequalities to model connectivity). The remaining two approaches are Lagrangian decompositions whose subproblems are still **NP**-hard to solve, but can be efficiently tackled by Branch&Cut approaches. Our computational study has shown that the decomposition approaches outperform the aggregated MIP approaches, both with respect to the running time, and with respect to the quality of the obtained lower and upper bounds.

Among the two decomposition approaches, a slight preference is given to the one that decomposes the problem according to its cost structure, into the *fixed-charge* and *variable-cost subproblems*. The reason for this is the global view of this approach, which is ensured by global connectivity constraints added into the fixed-charge subproblem. These constraints guide the topology of the network throughout Lagrangian iterations. Both decomposition approaches are capable of solving realistic instances (with almost 5000 nodes and 12500 edges) with final gaps of only a few percents. The obtained results indicate that these decomposition approaches could be even further improved by embedding them into a Branch&Bound framework. It might also be interesting to compare other techniques of solving the 2FTTx, like Benders decomposition, column generation, or to study heuristic approaches or approximation algorithms.

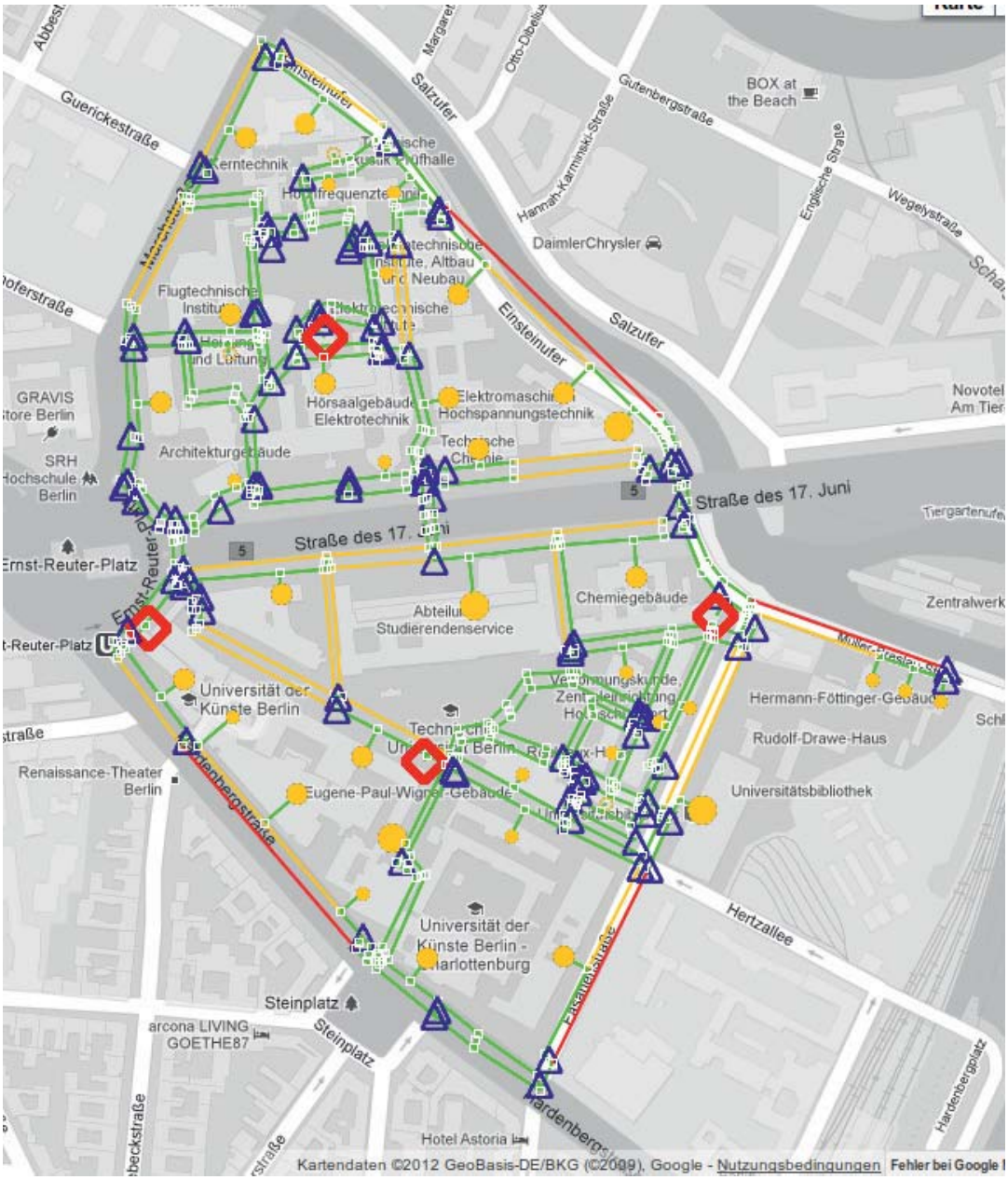
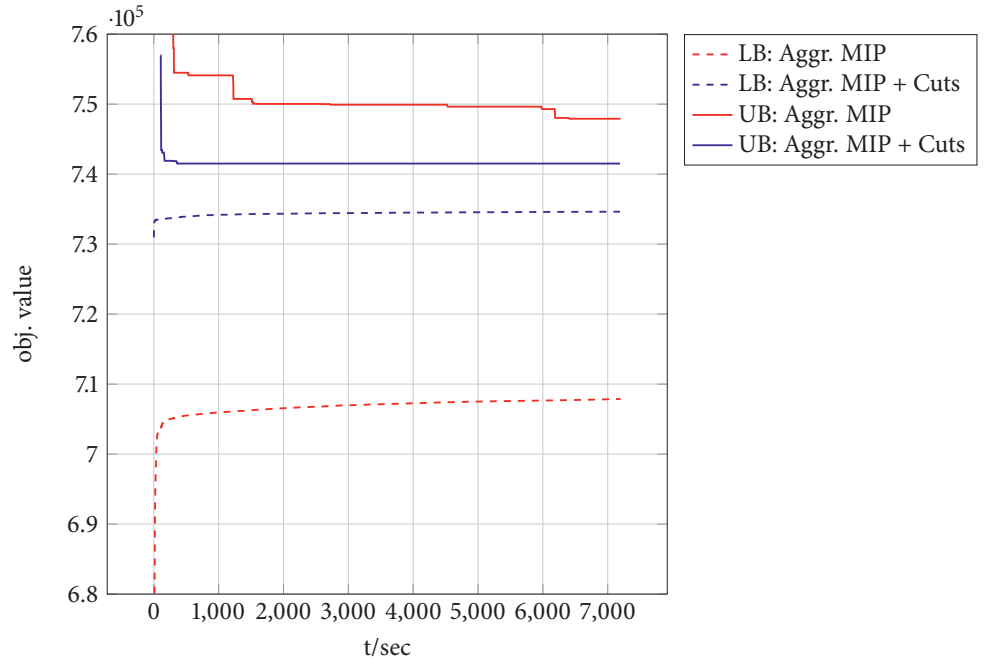
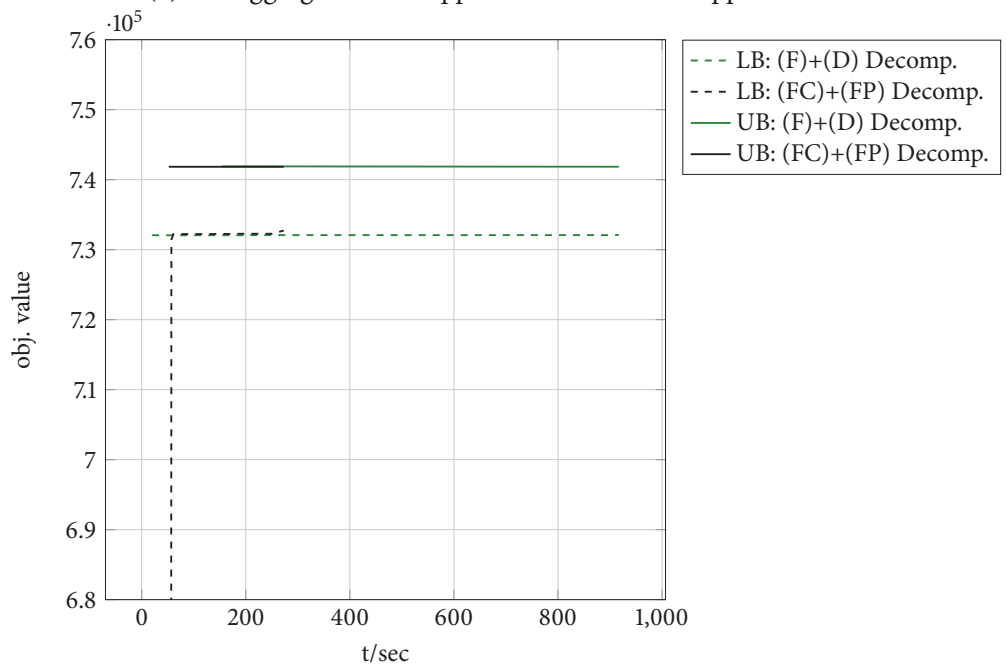


Figure 2.4.2: An example instance: squares, triangles and circles represent potential COs, potential DPs and customers, respectively. Maps courtesy of Google Maps.



(a) two aggregated MIP approaches: lower and upper bounds



(b) two decomposition approaches: lower and upper bounds

Figure 2.4.3: Progress of lower and upper bounds for the instance E_4

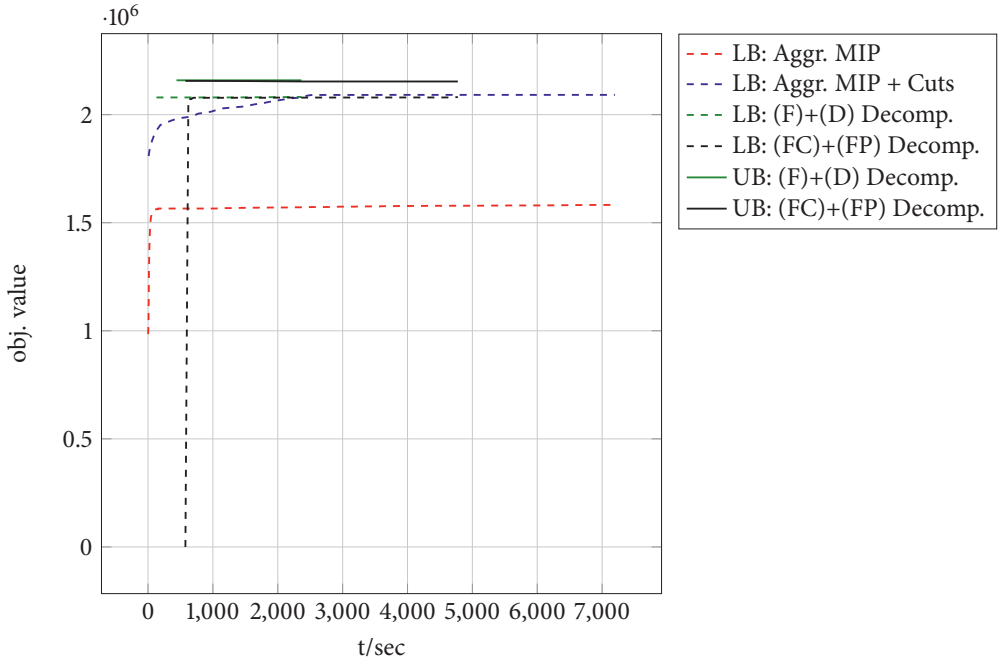


Figure 2.4.4: Progress of lower and upper bounds of four approaches on the instance E_5.

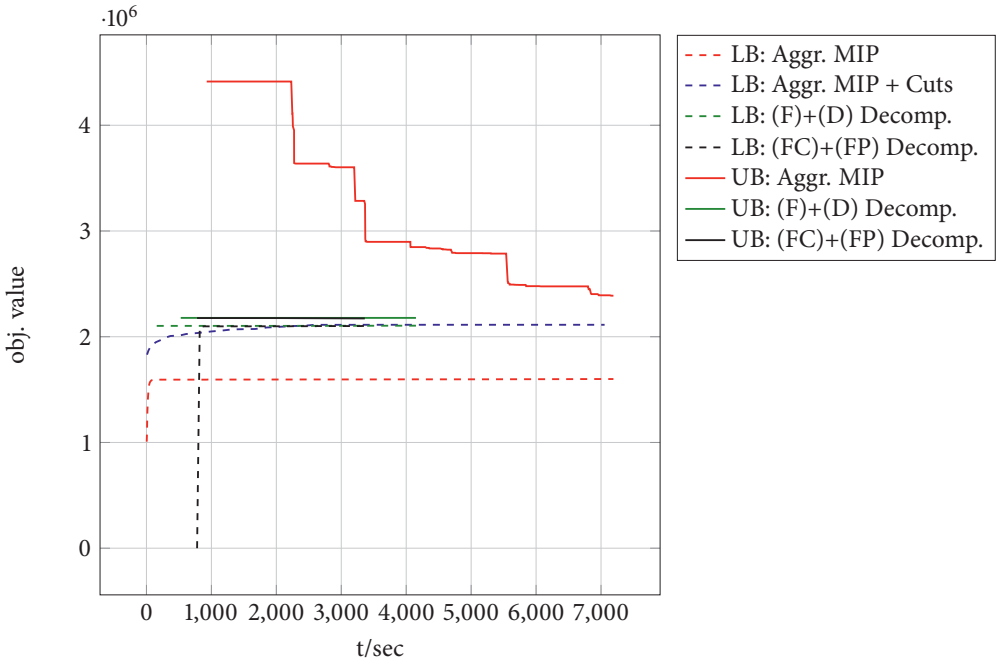


Figure 2.4.5: Progress of lower and upper bounds of four approaches on the instance E_6.

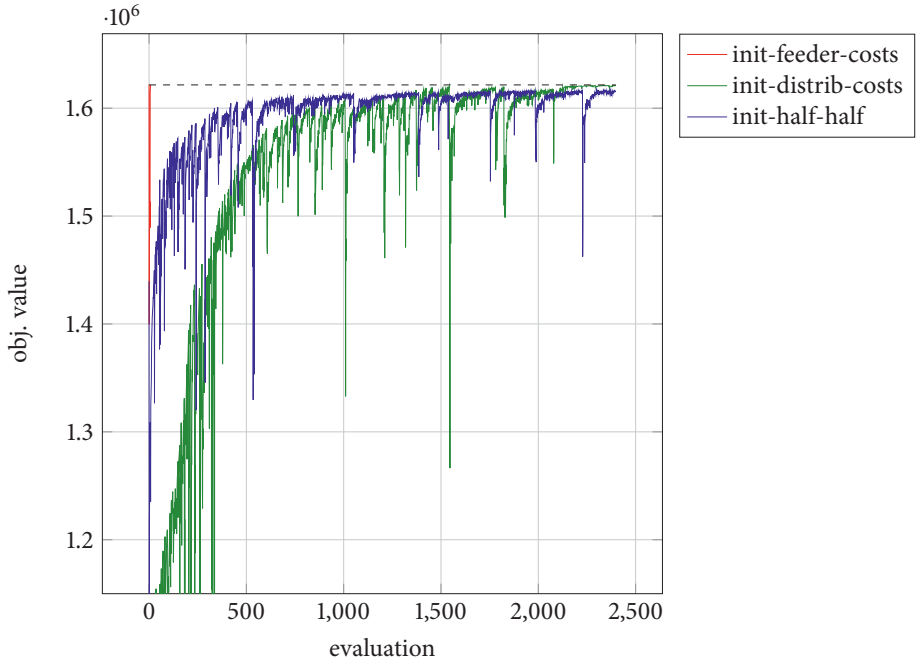


Figure 2.4.6: Comparison of subproblem evaluations for different settings of Lagrangian multipliers for the (F)+(D) decomposition in instance E_1.

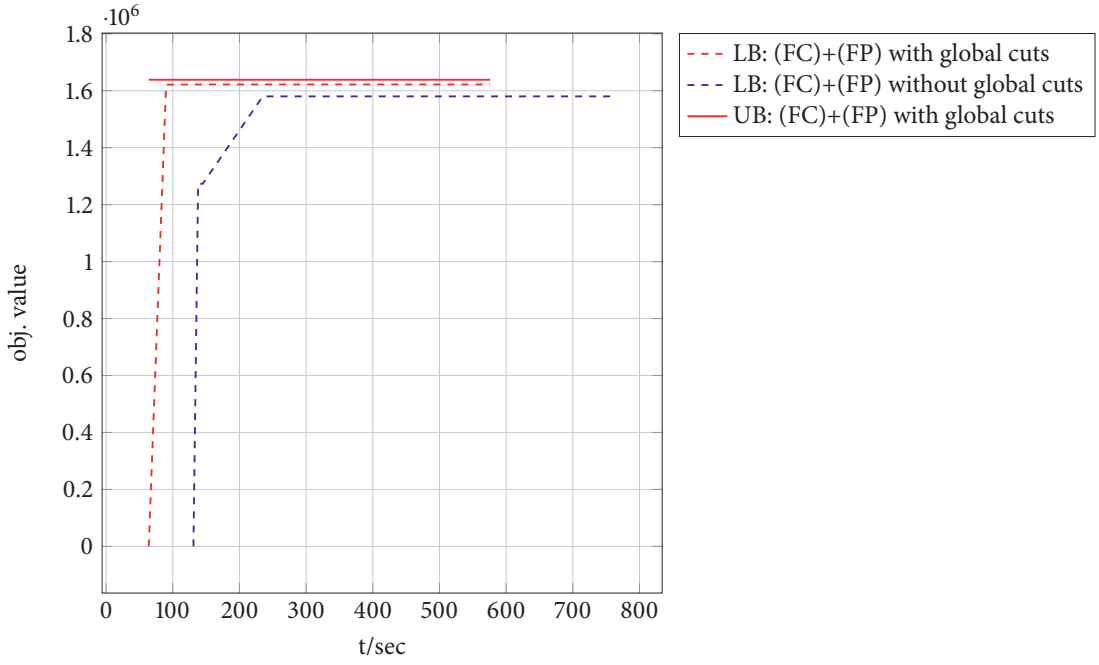


Figure 2.4.7: Comparison of the (FC)+(FP) decomposition with and without global cuts in instance E_1.

3 Node-Weighted Steiner Problems

3.1 The Node-Weighted Dominating Steiner problem

In this chapter, we introduce the *Node-weighted Dominating Steiner problem* (NWDSTP). The work presented is joint with Andreas Bley and Ivana Ljubić and has been submitted to "Networks", where it is currently pending review.

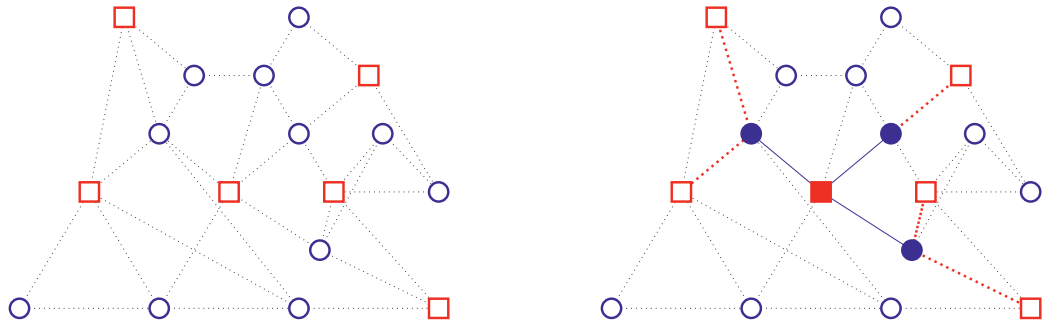
3.1.1 Introduction

The NWDST problem arises in several practical applications related to the design of telecommunication or logistics networks. Such networks typically consist of two or more administrative or technology levels representing a core part of higher aggregation and an access network part that is less aggregated. Nodes of the higher-level network must be equipped with higher-level technology or provide some features which require additional setup costs. Customer traffic enters the network at the lower access network level and is sent to a core node, where it is aggregated with the traffic of other customers. This is done because it can be transported through the core network more efficiently this way. After traversing the core network, the traffic is disaggregated and sent to its destination through the access network level.

In many applications, transportation within the access network is restricted to use direct connections between customers and core/hub nodes. In situations where the overall network costs consist only of or are strongly dominated by the costs of setting up core/hub nodes, the task of finding a minimum-cost network naturally leads to the Node-weighted Dominating Steiner problem. In the area of telecommunications, this cost structure appears in the planning of virtual networks, in virtual function placement in the context of cloud services and in the planning of optical overbuilds for existing copper-based access networks. Customer nodes then represent the set of terminals T in the NDWSTP and the node subset corresponding to the optimal NWDSTP solution provides the optimal location for core/hub nodes.

3.1.2 Problem setting

We consider the Node-Weighted Dominating Steiner problem, denoted by NWDSTP, which is defined as follows. Given a connected graph $G = (V, E)$, a set of terminal nodes $T \subseteq V, |T| \geqslant 2$ and a weight function $c\colon V \to \mathbb{R}$ on the nodes of G, we seek a connected subgraph of minimum weight such that each terminal is contained in the chosen subgraph or adjacent to a node in the subgraph. Another way to express this problem is to say that we look for a Steiner tree that minimizes the weight of the inner nodes, as in a tree (with $|V| \geqslant 3$), any leaf of the tree has an edge to an inner node.



(a) Input Graph (b) A feasible solution

Figure 3.1.1: An exemplary instance. Terminal nodes are shown as squares, steiner nodes as circles. Filled nodes are selected to be part of the T-dominating set.

We use the notation $n := |V|$, $m := |E|$ and $k := |T|$. The nodes of $G = (V, E)$ that are *dominated* by a node $v \in V$ are defined as the node v itself and all of its neighbors. Hence, we can also express the NWDST problem as the search for a least expensive connected subset of V that dominates T. We will also refer to a solution of the NWDST problem as a connected T-dominating subset.

We assume implicitly that the input graph G is connected throughout the rest of this chapter.

Note that, due to the nature of the objective function, we are only interested in the node set of the optimal solution. In terms of the characteristics of this node set, it is sufficient to look for a subgraph induced by these nodes that satisfies the following side constraints: (1) the induced subgraph must be connected and (2) it has to dominate the set of terminals. Figure 3.1.1 illustrates an input graph and its feasible solution.

3.1.3 Related work

There are several related problems studied in the literature that deal with related questions.

Group Steiner Tree problem In the Group Steiner Tree problem, we are given several groups of nodes and seek for a tree that contains at least one node from each group, minimizing the total edge weight. One easily verifies that the node-weighted variant of the Group Steiner Tree problem, where we seek for a tree that minimizes the weight of the nodes contained in the tree, is polynomially equivalent to the NWDST problem. Any given instance of the NWDST problem can be transformed into an equivalent instance of the Group Steiner Tree problem by replacing each terminal node t by the terminal group consisting of t and all its neighbors. Reversely, an instance of the Group Steiner Tree problem can be reduced to an NWDSTP instance which is obtained by adding for each terminal group T_i one additional node i connected to all nodes in T_i to the graph, replacing the requirement to contain one node from the terminal group T_i by the requirement to dominate this individual node i, and setting the cost of the added nodes to a value that is larger than the sum of the costs of all original nodes. This ensures that no optimal solution will actually contain any of the added nodes. Instead, they will be dominated by one of their neighbors from the given terminal group and, hence, the solution contains a valid

solution for the Group Steiner tree problem within the original graph. Garg et al. [2000] gave a polylogarithmic approximation algorithm for the problem. Demaine et al. [2013] improved the ratio in the special case where the graph is planarly embedded and each group is the set of nodes on a face. A fault-tolerant version of the problem is considered by Khandekar et al. [2012]. Lower bounds for the approximability of the problem are studied by Halperin and Krauthgamer [2003].

Minimum (Connected) Dominating Set problem Another problem closely related to NWDSTP is the Minimum Dominating Set problem (MDS). In this problem, the goal is to find a minimum-cost node set that dominates the entire graph. This problem is one of the classical problems considered in Garey and Johnson [1979]. There, the authors show the MDS problem as well as its connected version MCDS, where the dominating set must induce a connected subgraph, to be **NP**-complete. Furthermore, Bar-Yehuda and Moran [1984] showed that the MDS problem is polynomially equivalent to the *Set Cover problem*. Thus, the strong logarithmic inapproximability threshold for the Set Cover problem shown by Feige [1998] carries over in a straightforward way to the MDS problem. Finally, Hedetniemi et al. [1986] proposed a linear-time algorithm for the special case where G is a cactus graph, which is a graph in which any two simple cycles have at most one node in common.

In the Minimum Connected Dominating Set problem (MCDS) we seek for a least-cost connected subset of nodes that dominates the whole graph. Therefore, the MCDS is the special case of the NWDSTP where all nodes in the graph are terminals. Recently, Gendron et al. [2014] proposed a Branch&Cut algorithm, a Benders decomposition approach, and a hybridization of the two for solving the problem to optimality. Strong logarithmic inapproximability bounds hold for the MCDS as well. Guha and Khuller [1998] presented a $(3 \ln n)$-approximation algorithm for the node-weighted MCDS, which they improved in Guha and Khuller [1999b] to a $((1.35 + \varepsilon) \ln k)$-approximation algorithm for any constant $\varepsilon > 0$. A $(\ln \Delta + 2)$-approximation algorithm, where Δ denotes the maximum node degree in the graph, is presented in Ruan et al. [2004]. For a more comprehensive literature overview on the MCDS and its relation to the Maximum Leaf Spanning Tree problem—a useful relation which unfortunately does not carry over to the NWDSTP— see Gendron et al. [2014].

The classical and the node-weighted Steiner Tree problem Finally, the classical Steiner Tree problem and its node-weighted variant are very closely related to NWDSTP. In these problems, we are given a graph $G = (V, E)$ and a subset $T \subseteq V$ of terminals and wish to find a tree of minimum weight that includes all terminals. The classical edge-weighted problem version, where the weight of an edge is defined by its metric length, is known to be **NP**-hard for many metrics. The node-weighted version was considered by Klein and Ravi [1995], where the authors developed a $(2 \ln k)$-approximation algorithm and proved that the problem is as hard to approximate as the Set Cover problem. Finally, a $(1.5 \ln k)$-approximation for the problem was described in Guha and Khuller [1999b].

Our contributions

We are interested in Integer Linear Programming (ILP) formulations for the NWDST problem that use only $O(n)$ variables, where $n = |V|$. Many well established ILP formulations for Steiner trees and related problems are based on variables representing undirected or directed edges and constraints modelling a multi-commodity flow or ensuring connectivity. Such models typically lead to a very large number of variables in the resulting formulations. However, for applications where costs arise only at the nodes or even only at the internal transit nodes contained in the solutions (as it is the case for NWDST problem), edge variables introduce an unnecessary modeling overhead that may harm the computational performance. These variables are not necessary for modelling the problem, although they are quite convenient from a modelling perspective.

We propose a formulation that uses node variables only and that models connectivity through node-cut inequalities that can be separated in polynomial time. The resulting model contains a substantially smaller number of variables than the commonly used edge based models. Thus, our model is expected to lead to a better computational performance when solving very large problem instances using cutting plane approaches. We should mention that only very recently, node-based ILP models gained in popularity for modeling Steiner trees and related problems.

One example where this phenomenon can be observed is the so-called *Prize-Collecting Steiner Tree problem*. The problem is similar to the basic Steiner Tree problem, but in this variant, there are no terminals, but the cost-function is allowed to take negative values, enabling so-called *prizes* for including certain nodes in the resulting solution. Node-based models for the prize-collecting Steiner Tree problem where were one of the most important ingredients of the implementation of Fischetti et al. [2015]. There, the authors managed to solve some long-standing unsolved benchmark instances from the publicly available instance libraries to provable optimality. Node-based models have also been used in forestry applications (Carvajal et al. [2013]) and in bioinformatics (Álvarez-Miranda et al. [2013]). Finally, a polyhedral study for the related connected subgraph polytope based on node-variables is given in Wang et al. [2015].

The remainder of this chapter is organized as follows. In Section 3.2 we introduce the basic notation, our node variable based integer linear programming formulation of NWDSTP, and the polyhedron P defined by all feasible solutions. The fundamental properties of feasible solutions and the NWDSTP polyhedron P are discussed in Subsection 3.3.1. In the rest of Section 3.3, we study under which conditions the original model inequalities define facets of P. In Section 3.4, we introduce and analyze partition inequalities based on node-separators, which can be added to the original model in order to strengthen its linear relaxation. We show that these inequalities are valid for P and in fact do strengthen the linear relaxation of the model. For the special case where the underlying graph is a cycle and no two terminals are adjacent, we prove that a formulation containing all partition inequalities yields an exact description of P, that is, already the linear programming relaxation of such a formulation yields integer optimal solutions.

In Section 3.5, we evaluate the effectiveness of the presented partition inequalities. Finally, we draw our conclusions in Section 3.6.

3.2 Integer programming formulation

Before we can formally define the model considered in this chapter, we need to introduce some basic notation.

For a node set $U \subseteq V$, we denote by $G[U]$ the *subgraph induced by* U. A subset $S \subset V$ is called a *separator* of G if $G[V \setminus S]$ is disconnected. A node subset $S \subset V$ is called a k,ℓ*-separator* if $k, \ell \in V \setminus S$ and the nodes k and ℓ lie in different components of $G[V \setminus S]$. The set of all k,ℓ-separators is denoted by $\mathcal{S}_{k\ell}$.

For notational simplicity, we typically write $G \setminus S$ for $G[V \setminus S]$. A separator or a k,ℓ-separator S is called *minimal* if no proper subset $S' \subsetneq S$ is a separator or k,ℓ-separator, respectively. Given a separator S and $v \in V \setminus S$, we denote by $C_v \subseteq V$ the nodes of the connected component of $G \setminus S$ that contains v.

We repeat some notation from Section 1.3.6 for the convenience of the reader. For a node $v \in V$, we denote by Γ^*_v the set of all nodes adjacent to v. Furthermore, we let

$$\Gamma_v := \Gamma^*_v \cup \{v\}$$

be the set of all neighbors of v and v itself.

We are now ready to introduce an integer programming formulation for NWDSTP. Our model uses only the binary node variables $y_v \in \{0,1\}$, $v \in V$. These are interpreted as

$$y_v = \begin{cases} 1 & \text{if } v \text{ is contained in the dominating Steiner tree} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the set $I^y := \{v \in V \mid y_v = 1\}$ of nodes contained in the chosen dominating Steiner tree is not required to contain all terminals in T, but only to be a dominating set for T. Given a vector $y \in \mathbb{R}^V$ and a set $S \subseteq V$, we typically write $y(S) := \sum_{v \in S} y_v$ in order to simplify notation.

Using these variables, we can formulate NWDSTP as follows:

$$\begin{aligned}
\text{(NWDSTP):} \quad & \min \sum_{v \in V} c_v y_v \\
\text{subject to} \quad & y(S) \geqslant y_k + y_\ell - 1 && \forall k,\ell \in V \setminus T,\ k \neq \ell,\ S \in \mathcal{S}_{k\ell} && (3.2.1)\\
& y(S) \geqslant y_k && \forall k \in V \setminus T,\ \ell \in T,\ S \in \mathcal{S}_{k\ell} && (3.2.2)\\
& y(S) \geqslant 1 && \forall k,l \in T,\ S \in \mathcal{S}_{kl} && (3.2.3)\\
& y(\Gamma_v) \geqslant 1 && \forall v \in T && (3.2.4)\\
& y_v \in \{0,1\} && \forall v \in V
\end{aligned}$$

One easily verifies that (NWDSTP) is a correct integer linear programming model for the node-weighted dominating Steiner tree problem. Clearly, all inequalities (3.2.1)–(3.2.4) are valid for all incidence vectors of T-dominating Steiner trees and the objective function properly models the node weight. To see that the given constraints are sufficient, let $I = I^y := \{v \in V \mid y_v = 1\}$ be the set of nodes defined by a solution of (NWDSTP). Inequalities (3.2.3) ensure that for each terminal node pair $k,\ell \in T$ and each k,ℓ-separator S with k and ℓ in different components of

(a) feasible solution without (3.2.1)

(b) feasible solution without (3.2.2)

(c) feasible solution without (3.2.3)

(d) feasible solution without (3.2.4)

Figure 3.2.1: Necessity of model inequalities. Red circles are Steiner nodes that are part of the solution, green triangles are terminals, filled nodes are terminals that are part of the solution.

$G \setminus S$ at least one node from S is contained in I. This implies that I intersects each terminal separator. So, for all $k, \ell \in T$ that are not direct neighbors, I contains a neighbor of k, a neighbor of ℓ, and both are connected within I. Similarly, inequalities (3.2.1) and (3.2.2) require that I intersects each k,ℓ-separator for any node pair $k, \ell \in T \cup I$. Together with inequalities (3.2.3) this implies that the chosen nodes I induce a connected subgraph of G. Finally, inequalities (3.2.4) imply that each terminal node $i \in T$ is itself contained or has a neighbor in I. Thus, I forms a connected set dominating T, a dominating Steiner tree. Figure 3.2.1 illustrates infeasible cases that may occur by leaving out some of the constraints (3.2.1)–(3.2.4).

The convex hull of all integer solutions of (NWDSTP) defines the polyhedron

$$P := \operatorname{conv}\left\{ y \in \{0,1\}^V \;\middle|\; y \text{ satisfies (3.2.1)–(3.2.4)} \right\}.$$

Clearly, P is nothing but the convex hull of the characteristic vectors of all connected sets $I \subseteq V$ that dominate the terminal set T. We call P the NWDSTP polyhedron.

3.3 Polyhedral investigations

3.3.1 Basic properties

First, we discuss some basic properties of P and of separators which will be helpful for the polyhedral studies that follow.

We begin with a simple observation on the dimension of the polytope P:

Lemma 3.3.1. *If G is 2-connected, then* $\dim(P) = n$.

Proof. Obviously, we have $\dim(P) \leqslant n$. To see that $\dim(P) \geqslant n$, we now construct $n+1$ affinely independent vectors in P.

Let $v \in V$ be arbitrary. As G is 2-connected, $G \setminus \{v\}$ is connected. Clearly, all nodes in V are either contained in $V \setminus \{v\}$ or adjacent to a node in $V \setminus \{v\}$. Thus, for each $v \in V$, the vector x^v with

$$x^v_u := \begin{cases} 1 & u \neq v \\ 0 & u = v \end{cases}, \quad u \in V$$

is a feasible solution of (NWDSTP) and hence contained in P. The same trivially holds for the vector $\mathbb{1}_n$ of all-ones. Since $\mathbb{1}_n - x^v$ is the characteristic vector of the node v, the $n+1$ vectors $\mathbb{1}_n$ and x^v, $v \in V$, are affinely independent. Thus $\dim(P) \geqslant n$. □

We assume throughout the remainder of this chapter that G is 2-connected. This can be done without loss of generality. If the underlying graph is not 2-connected, we can easily decompose the problem on the overall graph into the corresponding problems on its 2-connected components, the blocks, and consider each of them separately.

Next, assume we are given a separator or a k, ℓ-separator S that is not minimal and consider the corresponding separator inequality (3.2.1)-(3.2.3). As S is not minimal, there exists a smaller (in fact, even a minimal) separator or k, ℓ-separator $S' \subsetneq S$. Obviously, the corresponding separator inequality (3.2.1)-(3.2.3) for S' dominates the one for S: The inequality for S can be obtained by the inequality for S' by adding the nonnegativity constraints for all $v \in S \setminus S'$. Hence, only minimal separators can induce facet-defining inequalities of P.

In order to characterize which minimal separators actually do induce facets of P, we need some further properties.

Lemma 3.3.2. *Let G be 2-connected, S be a minimal separator and $v \in S$. Then there is a spanning tree B on G such that v is the only inner (i.e. non-leaf) node of B that is contained in S.*

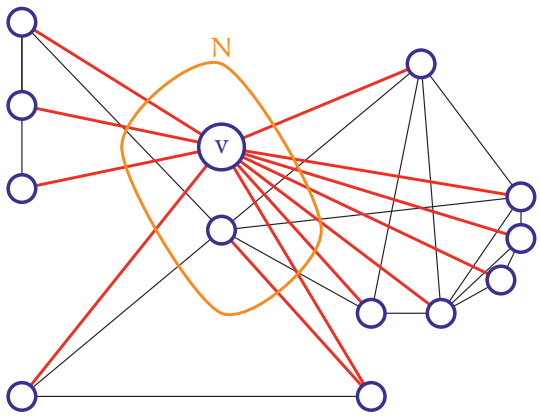


Figure 3.3.1: Illustration for Lemma 3.3.2: A possible tree B' is shown in red

Proof. As S is minimal, $G' := G \setminus (S \setminus \{v\})$ is connected, while $G \setminus S$ is not. Let B' be a spanning tree in G', compare also Figure 3.3.1. Since $G \setminus S$ is not connected, v cannot be a leaf node of B' but must be an inner node. Otherwise $B' - v$ would be a spanning tree of $G \setminus S$.

As S is a minimal separator, each $i \in S \setminus \{v\}$ is contained in an edge leaving S. Now pick one such edge for each $i \in S \setminus \{v\}$ and denote it by e_i. Finally, set

$$B := B' \cup \bigcup_{i \in S \setminus \{v\}} e_i.$$

With this construction, every node $i \in S \setminus \{v\}$ has degree 1 with respect to B and, as B' spans $G \setminus (S \setminus \{v\})$, B also spans G. Hence, B has the claimed properties. □

Finally, we observe that *each* node in a minimal separator S contains edges to *all* connected components of $G \setminus S$.

Lemma 3.3.3. *Let S be a minimal separator and let $C_1, \dots, C_k$ be the connected components of $G \setminus S$. Then, for each $v \in S$ and each $j \in \{1, \dots, k\}$, there is an edge uv with $u \in C_j$.*

Proof. Assume the claim is wrong. Then there is a node $v \in S$ and a component C_j of $G \setminus S$ such that $uv \notin E$ for all $u \in C_j$. Let $w \in C_j$. As C_j is a component of $G \setminus S$ with $w \in C_j$, $v \notin C_j$, and $uv \notin E$ for all $u \in C_j$, the set $S' := S \setminus \{v\}$ is a w,v-separator. This contradicts the minimality of S. □

3.3.2 Model inequalities

We now investigate under which conditions the model inequalities (3.2.3) and (3.2.4) define facets of P. Note that inequalities of type (3.2.2) and (3.2.1) are lifted variants of inequalities of the same type as (3.2.3), but for the case where only one or none of the components of $G \setminus S$ contains a terminal node, with corresponding right-hand side constants of 0 and -1, respectively. Conditions for these inequalities to be facet-defining can be derived from those for inequality (3.2.3) for the same separator S and terminal set $T' = T \cup \{k\}$ or $T' = T \cup \{k, \ell\}$, respectively.

In the case that every node in G is a terminal, conditions for (3.2.3) to be facet-defining were given in Theorem 3.9 of Fujie [2004]. We now discuss the conditions for the general case.

Recall that, given a separator S and $v \in V \setminus S$, C_v denotes the connected component of $G \setminus S$ that contains v. A node $v \in V \setminus S$ is called S-*replaceable* if there exists a node $j \in S$ such that the subgraph $G[(C_v \setminus \{v\}) \cup \{j\}]$ induced by nodes of C_v without v but with j added is connected. In other words, the connected component C_v remains connected if we replace node $v \in C_v$ by $j \in S$.

Theorem 3.3.4. *Let G be 2-connected and S be a minimal separator. Furthermore, let each $v \in V \setminus S$ be S-replaceable. Then,* (3.2.3) *is a facet of* P.

Proof. Let $F_S := \{y \in P \mid y(S) = 1\}$ be the face of P that is induced by the inequality (3.2.3) for S. Assume that F_S is contained in a facet F induced by some valid inequality $\sum_{i \in V} \alpha_i y_i \geqslant \alpha_0$ for P, i.e.,

$$F_S \subseteq F := \left\{ \sum_{i \in V} \alpha_i y_i = \alpha_0 \right\}.$$

We will show that this implies

$$\alpha_v = \begin{cases} \alpha_0 & v \in S \\ 0 & v \notin S \end{cases},$$

which, in turn, implies that $F_S = F$ and F_S is a facet.

Our proof consists of two parts: First, we show that $\alpha_v = 0$ for all $v \notin S$. Then, in the second part, we show that for all $v \in S$ we have $\alpha_v = \alpha_0$. Throughout this proof we will often interpret binary vectors $y \in \{0,1\}^V$ as node sets $I^y := \{v \in V \mid y_v = 1\} \subseteq V$ and node sets $I \subseteq V$ as their characteristic vectors $y^I \in \{0,1\}^V$. Given a binary vector $y \in P$, the set I^y then corresponds to the chosen connected dominating set. For a given binary vector y that is not necessarily in P, we say that y or its node set I^y dominates all nodes in the neighborhood of I, i.e., all nodes in $\{v \in V \mid v \in I \text{ or } uv \in E \text{ for some } u \in I\}$.

Clearly, for any spanning tree $B \subseteq E$ of G, the set $I(B) \subseteq V$ of *inner* (non-leaf) nodes of B defines a connected dominating set. Using this observation, we will now construct different spanning trees in G, whose inner node sets define vectors proving our claims.

To show the first claim, namely that $\alpha_v = 0$ for all $v \notin S$, it suffices to construct a spanning tree that has exactly one inner node in S and node v as a leaf. So, let $v \in V \setminus S$. We denote the node sets of the connected components of $G \setminus S$ by $C_1, \dots, C_r$. Without loss of generality we may assume $v \in C_1$. Let $j \in S$ be a replacement node for v, i.e., a node $j \in S$ such that $G[(C_v \cup \{j\}) \setminus \{v\}]$ is connected. Let $M := |S| - 1$.

We choose an arbitrary spanning tree $B_1 = (V_1, E_1)$ within $G[C_1 \cup \{j\} - \{v\}]$ and arbitrary spanning trees $B_2 = (V_2, E_2), \dots, B_r = (V_r, E_r)$ within the connected components $C_2, \dots, C_r$, respectively. Lemma 3.3.3 implies that there exists an edge from j to each component C_k, $k = 2, \dots, r$, because S is minimal. For each component C_k, $k = 2, \dots, r$, we choose one of these edges and denote it by c_k.

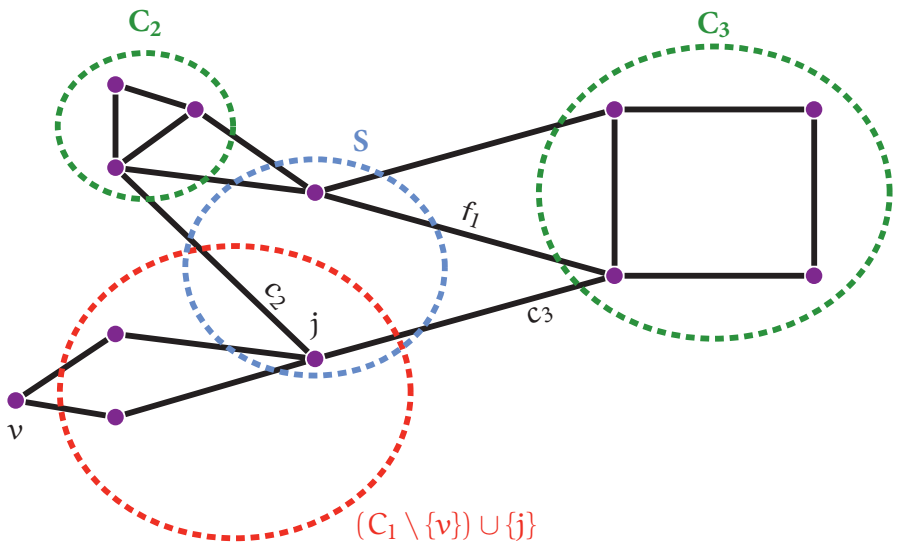


Figure 3.3.2: Illustration for proof of Theorem 3.3.4

Next, we choose edges $f_2, \dots, f_M$ such that each $n \in S \setminus \{j\}$ is part of at least one edge to some component C_i. Finally, we set

$$E' := \bigcup_{i=1}^{r} E_i \cup \{c_i \mid i = 1 \dots r\} \cup \{f_i \mid i = 2 \dots M\}.$$

Figure 3.3.2 illustrates this construction.

One easily verifies that E' is a tree that spans all nodes in V except for v. Hence, the set $I = I(E') \cup \{V_1 \setminus v\}$ consisting of all inner nodes $I(E')$ of E' plus all nodes in C_1 except v is a connected dominating set. Consequently, its characteristic vector y^I belongs to P. As all nodes in $S \setminus \{j\}$ are leaf nodes in E', y^I also satisfies the equality $y^I(S) = 1$, and hence $y^I \in F_S$.

As v is adjacent to a node in I, also $J := I \cup \{v\}$ is a connected dominating set and its characteristic vector y^J satisfies $y^J \in P$ and $y^J(S) = 1$.

Consequently, both vectors y^I and y^J are contained in $F_S \subseteq F$, which implies

$$\sum_{i \in V} \alpha_i (y^I_i - y^J_i) = \alpha_v = 0 \,.$$

This concludes the first part of the proof.

To prove the second claim, namely $\alpha_v = \alpha_0$ for all $v \in S$, let $v \in S$. Note that S is inclusion-wise minimal. Thus, Lemma 3.3.2 implies that there exists a spanning tree B of G such that v is the only inner node of B within S. Let y^I be the characteristic vector of the inner nodes $I := I(B)$ of B. Clearly, I defines a connected dominating set, so we have $y^I \in P$. Since there is only one inner node within S, we have $\sum_{i \in S} y^I_i = 1$, which implies $y^I \in F_S \subseteq F$. Finally, $I \cap S = \{v\}$ implies $\sum_{i \in S} \alpha_i y^I_i = \alpha_v = \alpha_0$, which concludes the proof of the second claim.

Consequently, any inequality inducing F has the form $\sum_{i \in S} \alpha_0 y_i = \alpha_0$ and, hence, $F_S = F$ is a facet. □

Next, we study the inequalities of type (3.2.4).

Consider inequality (3.2.4) for some terminal node $r \in T$ and assume there exists another terminal node $j \in V \setminus \Gamma_r$. If Γ^*_r is not a minimal separator, then inequality (3.2.4) for r is trivially dominated by the separator inequality (3.2.3) for any minimal separator S which is a (proper) subset of Γ^*_r and, hence, (3.2.4) cannot define a facet of P. If, on the other hand, Γ^*_r is a minimal separator, a sufficient condition for inequality (3.2.3) and thus also for (3.2.4) to be facet-defining is given in Theorem 3.3.4.

In the following, we thus assume that $V \setminus \Gamma_r$ does not contain any terminal. This case may occur if the set of terminals forms a highly connected cluster in the underlaying graph, for example. For this special case, a complete characterization of the cases when inequalities of type (3.2.4) define facets of P is given by the following two theorems. In the first theorem, we define the conditions that must be satisfied and prove that these are sufficient for inequality (3.2.4) to be facet-defining (Theorem 3.3.5). The necessity of the constraints is then shown in Theorem 3.3.6.

Theorem 3.3.5. *Let G be 2-connected, $r \in T$, and $T \subseteq \Gamma_r$. For each node $j \in V \setminus \Gamma_r$, let C_j denote the component of $G \setminus \Gamma_r$ that contains node j. Then inequality $y(\Gamma_r) \geqslant 1$ defines a facet of P if both conditions (C1) and (C2) hold:*

(C1) For all $j \in V \setminus \Gamma_r$, the graph G contains a tree B that

- *(i) spans all terminals,*
- *(ii) contains at least one node from the component C_j,*
- *(iii) does not use j as an inner node, and*
- *(iv) uses exactly one node from Γ_r as an inner node.*

(C2) For all $j \in \Gamma_r$, the graph G contains a tree B that

- *(i) spans all terminals,*
- *(ii) contains j, and*
- *(iii) uses j and only j as an inner node in Γ_r.*

Note that condition (C1-ii) is equivalent to the condition that node j itself is used as a leaf-node in the tree B.

Proof. Let $F_r = \{y \in P \mid y(\Gamma_r) = 1\}$ be the face of P induced by the inequality $y(\Gamma_r) \geqslant 1$. Let furthermore F be a facet of P containing F_r and assume that F is induced by the inequality $\sum_{i \in V} \alpha_i y_i \geqslant \alpha_0$, i.e., $F_r \subseteq F = \left\{y \in P \mid \sum_{i \in V} \alpha_i y_i = \alpha_0\right\}$. We show that the two conditions (C1) and (C2) imply

$$\alpha_j = \begin{cases} \alpha_0 & j \in \Gamma_r \\ 0 & j \notin \Gamma_r \end{cases},$$

which, in turn, implies that inequality $\sum \alpha_i y_i \geqslant \alpha_0$ is a multiple of inequality $y(\Gamma_r) \geqslant 1$. As P is bounded and full-dimensional, it then follows that $F_r = F$ is a facet.

The proof consists of two parts: In the first part, we show that $\alpha_j = 0$ for $j \in V \setminus \Gamma_r$. Then, in the second part, we show that $\alpha_j = \alpha_0$ for each $j \in \Gamma_r$.

For the first part, we let $j \in V \setminus \Gamma_r$ and choose a tree $B = (V_j, E_j)$ that satisfies condition (C1).

With (C1-i), we clearly have $T \subseteq V_j$. Due to (C1-iii), B does not use j as an inner node. If $j \in B$, then node j is a leaf of B. Otherwise, we need to modify the tree B. Let $R = (r_1, f_1, r_2, f_2, \ldots, f_{k-1}, r_k = j)$ be a shortest path within component C_j that connects some arbitrary node of B to j, where r_i denote the nodes and f_i denote the edges of R. Such a path exists, because (C1-ii) implies that B contains at least one node from C_j, and C_j is a connected component. As we chose a shortest path, $r_1 \in B$ and $r_2, \ldots, r_k \notin B$. Now we add the path R to B. Clearly, this yields a new tree B that contains j as a leaf node. Also note that we did not add any node from Γ_r to B.

Next, let $I = I(B)$ be the set of all internal nodes of B. Because B spans all terminals, I is a connected dominating set for T and its characteristic vector y^I defines a feasible point in the polyhedron P. Because B uses exactly one node in Γ_r as an inner node, we also have $|\Gamma_r \cap I| = 1$. Thus, $y^I \in F_r \subseteq F$.

Note that $y^I_j = 0$, because j was a leaf node in the tree B. Thus, j is adjacent to some node in I and $J := I \cup \{j\}$ defines a connected dominating set for T as well. Consequently, also the characteristic vector y^J is contained in P. As $j \notin \Gamma_r$, y^J also satisfies $y^J(\Gamma_r) = 1$ and, consequently, also $\sum_{i \in V} \alpha_i y^J_i \geqslant \alpha_0$.

Subtracting the two equalities for y^I and y^J, we obtain our first claim

$$\sum_{i \in V} \alpha_i (y^J_i - y^I_i) = \alpha_j = 0.$$

To prove the second claim, let $j \in \Gamma_r^*$. By condition (C2), there exists a tree B spanning j and all terminals, whose only inner node in Γ_r is j. We choose one such tree B and let $I := I(B)$ again be the set of its inner nodes. As B spans all terminals, I is a connected dominating set for T and its characteristic vector y^I is contained in P. As j is the only inner node of B within Γ_r, we also have $y^I \in F_r \subseteq F$, which implies our second claim $\alpha_j = \alpha_0$ and concludes the proof. □

We now show that the conditions (C1) and (C2) of Theorem 3.3.5 are also necessary.

Theorem 3.3.6. *Let* G *be 2-connected,* $r \in T$, *and* $T \subseteq \Gamma_r$. *If the inequality* $y(\Gamma_r) \geqslant 1$ *defines a facet of* P, *then both conditions (C1) and (C2) must hold.*

Proof. Assume that there exists a node j that violates either condition (C1) or condition (C2), depending on whether $j \in \Gamma_r$ or not. We show that then the stronger inequality

$$y(\Gamma_r) \geqslant 1 + y_j \tag{3.3.1}$$

holds for all $y \in P$. As the polyhedron P is full-dimensional and inequality (3.3.1) clearly dominates $y(\Gamma_r) \geqslant 1$, this implies that the latter inequality is not a facet of P.

Since P is defined as the convex hull of all integer solutions of (NWDSTP), it suffices to show that (3.3.1) holds for all integer solutions y of (NWDSTP), that is, for all characteristic vectors of connected dominating sets.

Clearly, (3.3.1) holds for all integer solutions $y \in P$ with $y_j = 0$, for which it is equivalent to the model inequality $y(\Gamma_r) \geqslant 1$. So, we may restrict our attention to integer solutions $y \in P$ with $y_j = 1$.

First, we consider the case $j \in \Gamma_r$, which means that condition (C2) is violated for j. If there was a connected dominating set I with $j \in I$ and $I \cap \Gamma_r = j$, one easily obtains a tree B with internal nodes I that spans T and j and contains only j as an internal node within Γ_r by extending a spanning tree on I with edges from I to the terminals T. As (C2) is violated and, thus, such a tree does not exist, there exists no connected dominating set I with $j \in I$ and $I \cap \Gamma_r = j$. Hence, any connected dominating set I with $j \in I$ contains at least two nodes in Γ_r. Consequently, any integer solution $y \in P$ with $y_j = 1$ for $j \in \Gamma_r$ satisfies $y(\Gamma_r) \geqslant 2$ and, thus, (3.3.1).

Now, assume $j \in V \setminus \Gamma_r$, which means that condition (C1) is violated for j. Let $N := \{v \in V \mid uv \in E \text{ for some } u \in C_j\}$ be the set of nodes that are adjacent to some node in C_j. Note that any connected dominating set that also dominates the node j must contain at least one node of N. Otherwise, N would separate j from r.

Analogously to the previous case we now observe that there exists no connected dominating set I with $j \notin I$, $I \cap N \neq \emptyset$, and $|I \cap \Gamma_r| = 1$, because any such set I would result in a tree B that meets the requirements of constraint (C1). Hence, any connected dominating set I with $j \notin I$ and $I \cap N \neq \emptyset$ contains at least two nodes in Γ_r, implying that any integer solution $y \in P$ with $y_j = 1$ for $j \in V \setminus \Gamma_r$ satisfies $y(\Gamma_r) \geqslant 2 = 1 + y_j$.

Hence, if node j violates the corresponding constraint (C1) or (C2), then all integer solutions $y \in P$ satisfy (3.3.1). As discussed above, this conflicts with the assumption that the inequality $y(\Gamma_r) \geqslant 1$ defines a facet of P. □

3.4 Partition inequalities

In this section, we show how the well-known Steiner partition inequalities studied by Chopra [1989]; Chopra and Rao [1994] for edge-based formulations of the classical Steiner tree problem can be adapted to our setting. The classical edge-based Steiner partition inequalities are based on the observation that, given a partition of the node set of the graph, any feasible solution must contain at least as many edges between different components of the partition as are needed to connect all those components that contain terminals.

In order to apply this observation to our node-based setting, we consider separator node sets whose removal partitions the remaining graph into several components, instead of edge

sets. Given a graph $G = (V, E)$ and a subset $S \subseteq V$, we denote by $C(S)$ the set of connected components of $G \setminus S$. This set consists of the subsets

$$C(S) = T(S) \dot\cup N(S),$$

where $T(S)$ denotes the set of components containing terminals and $N(S)$ denotes the subset of components without terminals. For simplicity, we will refer to the components in $T(S)$ as *terminal-components* and to those in $N(S)$ as *terminal-free components*. We also let $t_S := |T(S)|$ and $n_S := |N(S)|$.

Instead of simply counting the number of edges between different components, as in the classical edge-based Steiner partition inequalities, we must take care of the fact that the nodes in S may actually connect more than two components of $G \setminus S$ and S in a solution. Clearly, any Steiner tree B that spans all terminals must contain at least $t_S - 1$ many edges between S and the different components of $G \setminus S$ in order to connect all terminal-components. Each node $v \in S$, however, is adjacent only to a certain number of components, and it provides connectivity to those components only if it is chosen to be in the solution. Furthermore, any Steiner tree B defines a forest within $G[S]$. Hence, the number of edges that B contains within each component of $G[S]$ is bounded by the size of this component minus one, and this bound reduces if some nodes of the component are not contained in the solution. The latter two observations allow us to bound the total contribution of a single node $v \in S$ to the overall connectivity of any Steiner tree B and motivate the following definitions.

Definition 3.4.1. Let $G = (V, E)$, $S \subset V$, and $s \in S$. Denoting the connected component of s in $G[S]$ by C_s, we call

$$q_S(s) := \frac{|C_s| - 1}{|C_s|}$$

the *tree quotient* of the node s with respect to the separator S.

Definition 3.4.2. Let $G = (V, E)$, $S \subset V$, and $C(S) = T(S) \dot\cup N(S)$ as defined above. For $v \in S$, we define its S-*degree* as

$$\delta_S(v) := |\{C \in C(S) \mid uv \in E \text{ for some } u \in C\}| + q_S(v)$$

Note that $\delta_S(v)$ is exactly the number of components of $G \setminus S$ that v is adjacent to if v is not adjacent to any other node in S. Otherwise, if v also has neighbors in S, its S-degree is the number of adjacent components in $G \setminus S$ plus a certain fraction pertaining to the number of edges it needs to use to build connections inside S.

Using this notation, we can formulate the basic Steiner partition inequalities for our node-based ILP model.

Theorem 3.4.3. *For each subset $S \subset V$, the following Steiner partition inequality is valid for P:*

$$\sum_{v \in S} (\delta_S(v) - 1)\, y_v \geqslant t_S - 1 \tag{3.4.1}$$

Proof. Consider any subset $S \subset V$ and a feasible solution y. We would like to show that y satifies inequality (3.4.1).

In any feasible solution y, the set $I^y := \{v \in V \mid y_v = 1\}$ forms a single connected component dominating all terminals. Let now B be a Steiner tree that spans the nodes of I^y and all terminal nodes. To simplify the calculations, we assume w.l.o.g. that any terminal component of $G \setminus S$ only contains the terminal. Otherwise, we can contract these components to single terminals.

The tree B then contains at least $|T| + \sum_{i \in S} y_i$ many nodes. Now, we bound the number of edges of B.

For $s \in S$, let $\delta_S^+(s) := |\{ (s,t) \in E \mid t \notin S \}|$ denote the out-degree of the node s with respect to S, that is, the number of edges whose other endpoint lies outside S. The number of edges in B between S and components of $G \setminus S$ clearly is upper-bounded by

$$\sum_{i \in S} \delta_S^+(i) y_i.$$

Furthermore, a tree or a forest on n nodes can contain at most $n-1$ edges. Denoting the connected components of $G[S]$ by $S_1, \ldots, S_k$, this implies that B contains at most $|S_i| - 1$ edges within each S_i. Hence, the total number of edges of B with both end-nodes in S cannot exceed

$$\sum_{i \in S} q_S(i) y_i.$$

As the number of nodes cannot be higher than the number of edges plus one, we get

$$\sum_{i \in S} (\delta_S^+(i) + q_S(i)) y_i \geqslant -1 + |T| + \sum_{i \in S} y_i \,,$$

which implies

$$\sum_{i \in S} (\delta_S(i) - 1) y_i \geqslant |T| - 1 \,.$$

Thus, any $y \in P$ must satisfy the Steiner partition inequality (3.4.1). □

Note that, given a minimal k,ℓ-separator $S \in \mathcal{S}_{k,\ell}$ for arbitrary distinct terminals $k, \ell \in T$, its associated Steiner partition inequality (3.4.1) is implied by the minimal separator inequality (3.2.3): Due to minimality of S, Lemma 3.3.3 implies that all nodes in S are adjacent to all components in $C(S)$ and, hence, $(\delta_S(v) - 1) \geqslant t_S - 1$. Multiplying (3.2.3) with $t_S - 1$, one obtains a stronger inequality than (3.4.1).

However, there are graphs and families of separators $S \subset V$ for which the Steiner partition inequalities are not implied by the inequalites of (NWDSTP) and where they actually strengthen the linear relaxation of this formulation.

Example 3.4.4. Consider the example shown in Figure 3.4.1 on the following page. The values shown next to the nodes correspond to the LP relaxation values of the model (3.2.1)-(3.2.4) for the objective function $c_v = 1$ for all $v \in V$. Notice that each minimal separator is a subset $S' \subset V$ of cardinality two, hence all minimal separator inequalities are satisfied. Consider now an arbitrary subset S of three non-terminal nodes such that the number of terminal-components in $G \setminus S$ is $t_S = 3$, for example $S = \{2, 4, 6\}$. Then the Steiner partition inequality (3.4.1) associated to S is violated by the depicted fractional solution.

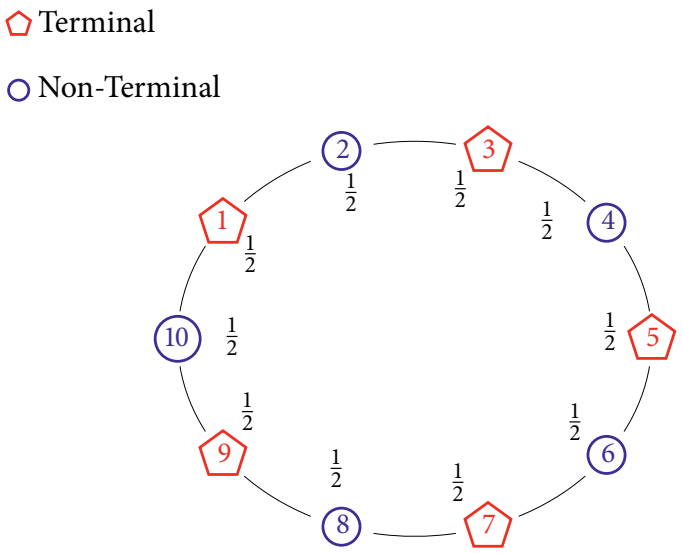


Figure 3.4.1: A valid solution for the NWDSTP relaxation (3.2.1)-(3.2.4) that is cut off by the Steiner partition inequalities (3.4.1).

Increasing the length of the cycle graph in Example 3.4.4 to $2k$ nodes, one easily verifies that the ratio between the linear relaxation value obtained with the original formulation (NWDSTP) and the formulation with additional Steiner partition inequalities can be as large as $2 - \frac{1}{k}$ for any $k > 2$. More precisely, the solution y' with $y'_v = 1/2$ for all $v \in V$ is an optimal solution for the LP relaxation of (NWDSTP) for the objective function $c = 1$, with objective value equal to k. On the other hand, the solution $\tilde{y}$ with $\tilde{y}_v = (k-1)/k$ is an optimal LP-solution for the relaxation including the Steiner partition inequalities, with the objective value of $2(k-1)$.

In those cases where some components of $G \setminus S$ contain no terminals, the corresponding Steiner partition inequality can be strengthened easily.

3.4.1 Lifted Partition inequalities

Let $N(S) = \{N_1(S), \ldots, N_{n_S}(S)\}$, $N_i(S) \subset V$ be the set of terminal-free components of $G \setminus S$ and consider ordered n_S-tuples $(v_1, \ldots, v_{n_S})$ of nodes such that $v_i \in N_i(S)$ for $i = 1, \ldots, n_S$. The set of all such n_S-tuples is denoted by $\mathcal{N}(S)$. Lifting the variables corresponding to one such n_S tuple into the Steiner partition inequality for S, we obtain a stronger inequality.

Theorem 3.4.5. *For each subset $S \subset V$ and each tuple $(v_1, \ldots, v_{n_S}) \in \mathcal{N}(S)$, the following lifted Steiner partition inequality is valid for P:*

$$\sum_{v \in S} (\delta_S(v) - 1)\, y_v - \sum_{i=1}^{n_S} y_{v_i} \geqslant t_S - 1 \tag{3.4.2}$$

Proof. The validity of lifted Steiner partition inequalities (3.4.2) can be shown analogously to the validity of the Steiner partition inequalities (3.4.1). Note that the tuple $(v_1, \ldots, v_{n_S})$ contains one representative node for each terminal-free component $N_i(S)$ of $G \setminus S$. If $y_{v_i} = 1$ for one

such representative node v_i, then the component containing v_i needs to be connected to the rest of the solution. In this case, this component can be treated as if it contained a terminal. Consequently, the number of components that need to be connected by the solution y increases to t_S for the terminal-components plus $\sum_{i=1}^{n_S} y_{v_i}$ for the terminal-free components that contain chosen nodes. Hence, inequality (3.4.2) holds. □

There exist problem instances where the addition of lifted Steiner partition inequalities (3.4.2) strengthens the linear relaxation of (NWDSTP), even when compared to the relaxation obtained after adding the basic Steiner partition inequalities (3.4.1).

Example 3.4.6. Consider the example shown in Figure 3.4.2. In the figure, we see an example of a valid solution for the NWDSTP relaxation. As one easily verifies, this solution is not cut away by the inequalities given in the (NWDSTP) formulation including Steiner partition inequalities (3.4.2). Consider now the separator $S = \{1, 3, 6, 9\}$. The lifted partition inequality for this separator is

$$y_1 + y_3 + y_6 + y_9 - y_{10} - y_2 \geqslant 1,$$

which the given solution violates.

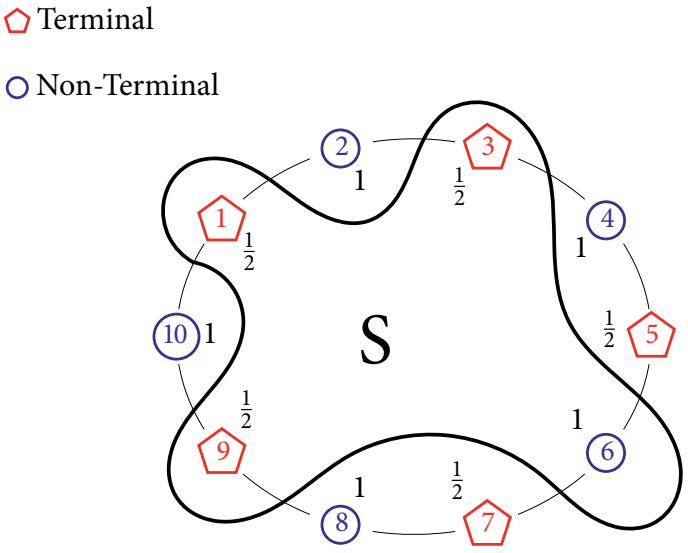


Figure 3.4.2: A valid solution for the NWDSTP relaxation. The solution values are marked inside the circle, the separator $S = \{1, 3, 6, 9\}$ is depicted in black.

In the remainder of this section we show that in the special case where G is a cycle, the lifted Steiner partition inequalities (3.4.1) combined with the model constraints (3.2.1)-(3.2.4) are sufficient to describe the dominant of P.

3.4.2 A complete description of P on a cycle

Throughout this section we assume that $G = (V, E)$ is a cycle.

Definition 3.4.7. Let $G = (V, E)$ be a cycle, $S \subset V$ and $C(S) = T(S) \dot\cup N(S)$ as defined above. For $v \in S$, we define its *circular* S*-degree* as

$$\delta_S^C(v) := |\{C \in C(S) \mid uv \in E \text{ for some } u \in C\}| + \begin{cases} 1 & \text{if } \{u \in S \mid uv \in E\} \neq \emptyset \\ 0 & \text{otherwise} \end{cases}.$$

We switch to this weaker definition instead of the S-degree because this simplifies the proofs and yields the same results in cycle graphs.

We begin by studying the case where G consists of alternating terminal and non-terminal nodes. So, in the following let $V = \{0, \dots, 2k-1\}$, $E = \{\{i, i+1\} \mid i = 0, \dots, 2k-1\}$, and $T = \{v_0, v_2, \dots, v_{2k-2}\}$ for $k \geqslant 2$.

To simplify notation, we assume that the node index $2k$ is equivalent to the node index 0, or in other words, all calculations with node indices are to be understood as being executed in $\mathbb{Z}/2k\mathbb{Z}$.

We consider the linear programming relaxation of (NWDSTP) that is defined by the set of all lifted Steiner partition inequalities (3.4.2) and the variable boundary constraints only, i.e.,

$$\begin{aligned} &\text{(PART*):} && \min \quad c^T y \\ &\text{subject to} && \sum_{v \in S} y_v - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1 && \forall\, S \in \mathcal{S},\ (n_1, \dots, n_{n_S}) \in \mathcal{N}(S) \\ & && 1 \geqslant y_v \geqslant 0 && \forall\, v \in V \end{aligned}$$

It is easy to verify that all inequalities (3.2.1)-(3.2.4) of (NWDSTP) are implied by the inequalities of (PART*) if G is a cycle of alternating terminals and non-terminals. Hence, the linear relaxation of (NWDSTP) is a relaxation of (PART*). On the other hand, the (PART*) is a relaxation of P, because all lifted Steiner partition inequalities (3.4.2) are valid for P.

In the following, we show that the polytope defined by (PART*) is integral, that is, for each objective c there exists an integer optimal solution y^* for (PART*). This then directly implies that (PART*) is a complete description of P.

First, we observe that the lifted partition inequalities corresponding to non-stable sets S are redundant.

Lemma 3.4.8. *Let* $S \in \mathcal{S}$ *and* $(n_1, \dots, n_{n_S}) \in \mathcal{N}(S)$. *If* S *is not a stable set, then inequality* (3.4.2) *is redundant in (PART*).*

Proof. Since S is a separator and G is a cycle, S consists of at least two connected components. If S is not a stable set, then at least one of these components contains more than one node. As G is a circle, this component is a path P in G. Without loss of generality we assume that the nodes are numbered in such a way that $P = (i, i+1, \dots, i+\ell)$. Let $S' := S \setminus \{i, \dots, i+\ell-1\}$. In words, S' is the separator obtained from S by replacing all nodes of P with the single node $i+\ell$.

Clearly, $G \setminus S'$ contains exactly as many components as $G \setminus S$. Furthermore, all components of $G \setminus S$ that are not adjacent to i are also components of $G \setminus S'$. Only the one component C of $G \setminus S$ that is adjacent to i increases in $G \setminus S'$ to a component $C' = C \cup \{i, \dots, i+\ell-1\}$.

If C is a terminal component, then C' is a terminal component too. In this case, consider the lifted Steiner partition inequality (3.4.2) defined by S' and $(n_1, \dots, n_{n_S})$. As the terminal-free

components of $G \setminus S$ are exactly the terminal-free components of $G \setminus S'$, we have $(n_1, \ldots, n_{n_S}) \in \mathcal{N}(S) = \mathcal{N}(S')$. Hence, the lifted partition inequality defined by S' and $(n_1, \ldots, n_{n_S})$ is valid and contained in the system (PART*). As $S' \subsetneq S$, the inequality for S' clearly dominates the one for S.

If C is a terminal-free component, then the larger C' must be a terminal component. In this case, we have $n_{S'} = n_S - 1$ and $t_{S'} = t_S + 1$. Let w.l.o.g. C denote the last terminal-free component of $G \setminus S$, i.e., $n_{n_S} \in C$. As all other terminal-free components of $G \setminus S$ remain terminal-free in $G \setminus S'$, we have $(n_1, \ldots, n_{n_S - 1}) \in \mathcal{N}(S')$. Hence, the lifted Steiner partition inequality (3.4.2) defined by S' and $(n_1, \ldots, n_{n_S - 1})$ is valid and contained in the system (PART*). Together with the inequalities $y_{n_{n_S}} \leqslant 1$ and $y_v \geqslant 0$ for $v = i, \ldots, i + \ell - 1$, this inequality implies the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$.

Consequently, inequality (3.4.2) is redundant if S contains a path. □

Due to Lemma 3.4.8, we may instead of PART* consider the formulation PART, which contains lifted Steiner partition inequalities only for stable sets S. This relaxation then defines the same polytope as PART*.

$$
\begin{aligned}
&\text{(PART):} && \min \quad c^T y && \\
&\text{subject to} && \sum_{v \in S} y_v - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1 && \forall \text{ stable } S \in \mathcal{S},\ (n_1, \ldots, n_{n_S}) \in \mathcal{N}(S) \\
& && 1 \geqslant y_v \geqslant 0 && \forall\, v \in V
\end{aligned}
$$

In our next step, we show that, for any nonnegative objective function c, there exists an optimal solution for (PART) that satisfies the additional equalities

$$y_i = y_{i-1} + y_{i+1} - 1 \qquad \forall\, i \in T, \tag{3.4.3}$$

To facilitate this, we introduce the notion of an irreducible solution.

Definition 3.4.9. A solution $y \in \mathbb{R}^V$ of (PART) is called *irreducible* if there is no solution $y' \in \mathbb{R}^V$ of (PART) with $y' \leqslant y$ and $y' \neq y$.

Clearly, for any nonnegative objective function c, an optimal and irreducible solution y of (PART) exists. In the following two lemmas we show that irreducible solutions satisfy (3.4.3).

Lemma 3.4.10. *Let $c \in \mathbb{Q}^V_{\geqslant 0}$ be a nonnegative objective function and $y \in \mathbb{R}^V$ be an irreducible, optimal solution of (PART). Then we have*

$$y_i \geqslant y_{i-1} + y_{i+1} - 1 \qquad \forall\, i \in T\,. \tag{3.4.4}$$

Proof. Assume the claim was wrong. We have $y_i < y_{i-1} + y_{i+1} - 1$ for some $i \in T$. To simplify notation, let $u := i - 1$, $v := i$, and $w := i + 1$. Then

$$\varepsilon := y_u + y_w - 1 - y_v > 0\,. \tag{3.4.5}$$

We will show that $\bar{y} \in \mathbb{R}^V$ defined as

$$\bar{y}_i := \begin{cases} y_i - \frac{\varepsilon}{2} & \text{if } i \in \{u, w\} \\ y_i & \text{otherwise} \end{cases} \qquad \text{for } i \in V \tag{3.4.6}$$

satisfies (PART). As $\bar{y} \leqslant y$ and $\bar{y} \neq y$, this implies that y is not irreducible for (PART), which contradicts the preconditions of the lemma.

Obviously, $\bar{y}$ satisfies all boundary constraints $0 \leqslant y_i \leqslant 1$ of (PART). It remains to show that $\bar{y}$ also satisfies all non-redundant lifted Steiner partition inequalities (3.4.2).

So, let $S \in \mathcal{S}$, $(n_1, \ldots, n_{n_S}) \in \mathcal{N}(S)$, and consider the lifted Steiner partition inequality defined by S and $(n_1, \ldots, n_{n_S})$.

We now distinguish several cases, depending on which nodes belong to S.

Case 1: $u, w \in S$

Due to Lemma 3.4.8, we may assume $v \notin S$. Otherwise, the lifted Steiner partition inequality defined by S and $(n_1, \ldots, n_{n_S})$ is redundant (as S would then contain a path between u and w).

Let $S' := (S \setminus \{u, w\}) \cup \{v\}$, as illustrated in Figure 3.4.3.

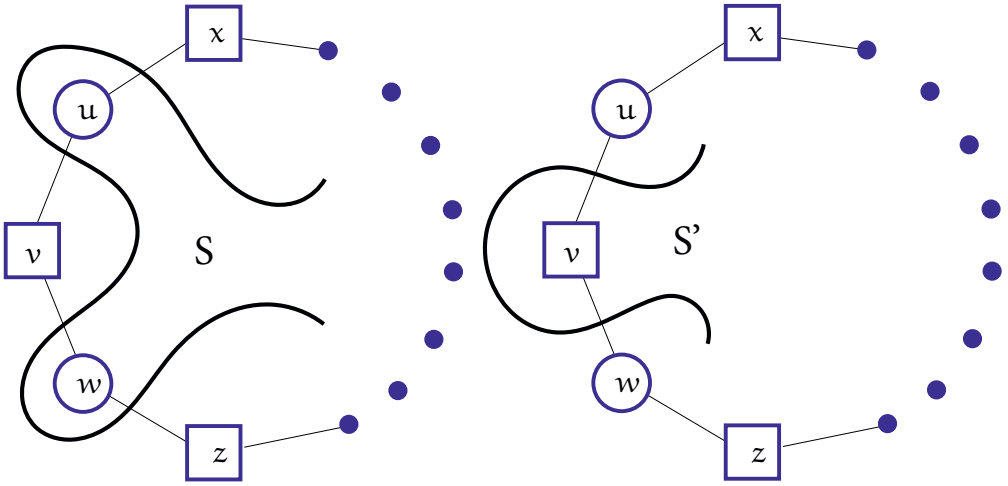


Figure 3.4.3: Lemma 3.4.10, Case 1

If S' is not a separator, we must have $S = \{u, w\}$. In this case $G \setminus S$ contains two terminal-components (one is $\{v\}$, and the other contains all the remaining terminals of $T \setminus \{v\}$). The (lifted) Steiner partition inequality defined by S then reads

$$y_u + y_w \geqslant 1\,.$$

With (3.4.5) and (3.4.6), this inequality is trivially satisfied by $\bar{y}$.

If S' is a separator, the terminal-free components in $G \setminus S'$ and $G \setminus S$ are equal. Therefore, also the lifted Steiner partition inequality for S' and $(n_1, \ldots, n_{n_S}) \in \mathcal{N}(S) = \mathcal{N}(S')$ is part of the system (PART) and holds for y, i.e.,

$$\sum_{x \in S'} y_x - \sum_{i=1}^{n_S} y_{n_i} \geqslant t'_S - 1\,.$$

On the other hand, the number $t_{S'}$ of terminal-components in $G \setminus S'$ is one less than the number t_S of terminal-components in $G \setminus S$, as the terminal component $\{v\}$ disappears. This implies

$$y_u + y_w - \varepsilon + \sum_{i \in S', i \neq v} y_i - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1$$

and, together with (3.4.5) and (3.4.6),

$$\sum_{i \in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1 \,.$$

Hence, $\bar{y}$ fulfills the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$.

Case 2: $u \notin S, w \notin S$

As u, w are not in S, the only way they can be involved in the lifted Steiner partition inequality is as one of the lifted non-terminal variables y_{n_i}. As these variables have a negative coefficient, however, the inequality trivially holds for $\bar{y}$ if it holds for y.

Case 3: $u \in S, w \notin S$ (or, analogously, $u \notin S, w \in S$)

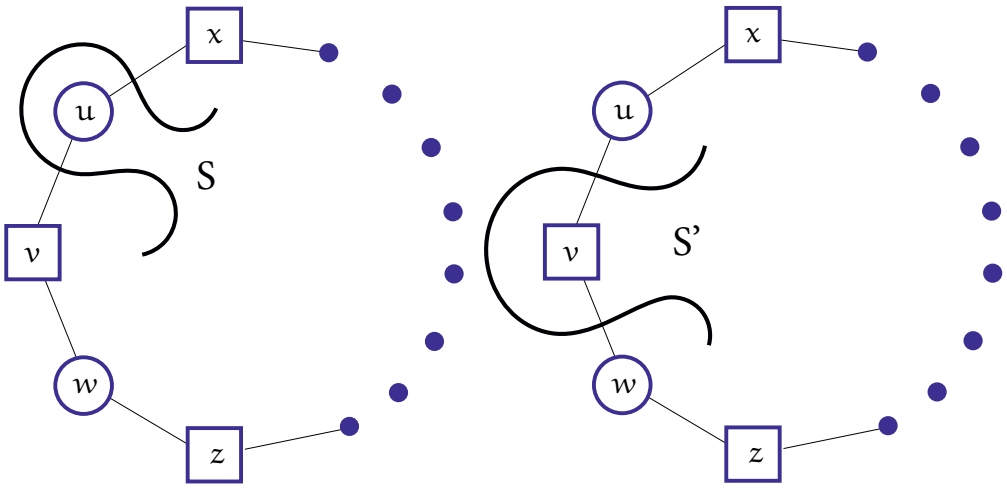


Figure 3.4.4: Lemma 3.4.10, Case 3

Let z be the neighbor of w which is not v and x be the neighbor of u which is not v, as illustrated in Figure 3.4.4. By Lemma 3.4.8 we may assume that $v \notin S$ and $x \notin S$.

We consider the separator $S' := (S \setminus \{u\}) \cup \{v\}$ obtained by replacing node u by node v in S. If S is a separator, then S' is as well, because $w \notin S$. Replacing u by v enlarges the component containing x, reduces the component containing w, and leaves all other components of $G \setminus S$ unchanged. While the component containing x will be a terminal component in both $G \setminus S$ and $G \setminus S'$, the component containing w is a terminal component in $G \setminus S$, but the reduced component in $G \setminus S'$ might be a terminal-free component.

If $z \notin S$, then the component that contains w also contains the terminal x and is a terminal-component in both $G \setminus S$ and $G \setminus S'$. Thus, the number of terminal-components and node sets of terminal-free components are the same for $G \setminus S$ and $G \setminus S'$. In this case, also the lifted Steiner partition inequality for S' and $(n_1, \dots, n_{n_S}) \in \mathcal{N}(S) = \mathcal{N}(S')$ is part of the system (PART) and holds for y, i.e.,

$$\sum_{i \in S'} y_i - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_{S'} - 1 = t_S - 1 \,.$$

Adding (3.4.5) and $1 \leqslant y_w$ to this inequality, we obtain

$$\sum_{i \in S} y_i - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1 + \varepsilon \,,$$

and, with (3.4.6),

$$\sum_{i \in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1 \,.$$

Otherwise, if $z \in S$, then the number $t_{S'}$ of terminal-components in $G \setminus S'$ is one less than the number t_S of terminal-components in $G \setminus S$. Also, $G \setminus S'$ contains the terminal-free component $\{w\}$, which has not been a terminal-free component in $G \setminus S$. In this case, the lifted Steiner partition inequality for S' and $(n_1, \dots, n_{n_S}, w) \in \mathcal{N}(S) \times \{w\} = \mathcal{N}(S')$, namely

$$\sum_{i \in S'} y_i - \sum_{i=1}^{n_S} y_{n_i} - y_w \geqslant t_{S'} - 1 = t_S - 2$$

is part of the system (PART) and holds for y. Together with (3.4.5) this implies

$$\sum_{i \in S'} y_i + y_u - y_v - \sum_{i=1}^{n_{S'}} y_{n_i} - y_w + y_w \geqslant t_S - 2 + 1 + \varepsilon \,,$$

and with (3.4.6) finally

$$\sum_{i \in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1 \,.$$

Hence, $\bar{y}$ fulfills the lifted Steiner partition inequality for S and $(n_1, \dots, n_{n_S})$ also in this case, which concludes the proof. □

The validity of the reverse inequalities is shown in the following lemma.

Lemma 3.4.11. *Let $c \in \mathbb{Q}^V_{\geqslant 0}$ be a nonnegative objective function and $y \in \mathbb{R}^V$ be an irreducible optimal solution of (PART). Then we have*

$$y_i \leqslant y_{i-1} + y_{i+1} - 1 \qquad \forall\, i \in T \,. \tag{3.4.7}$$

Proof. Assume the claim was wrong and that there is some $i \in T$ with $y_i > y_{i-1} + y_{i+1} - 1$. Denoting again $u := i-1$, $v := i$, and $w := i+1$, this means

$$\varepsilon := y_v - y_u - y_w + 1 > 0\,. \tag{3.4.8}$$

Similar to the proof of Lemma 3.4.10, we will show that $\bar{y} \in \mathbb{R}^V$ defined as

$$\bar{y}_i := \begin{cases} y_i - \varepsilon & \text{if } i = v \\ y_i & \text{otherwise} \end{cases} \qquad \text{for } i \in V \tag{3.4.9}$$

satisfies (PART), which cannot be the case if y is an irreducible solution of (PART).

Note that (PART) contains the lifted Steiner partition inequality $y_u + y_w \geqslant 1$ corresponding to $S = \{u, w\}$. With (3.4.8) this implies $1 \geqslant y_v - \varepsilon = y_u + y_w - 1 \geqslant 0$ and thus $1 \geqslant \bar{y}_v = y_v - \varepsilon \geqslant 0$. Hence, $\bar{y}$ satisfies all boundary constraints $0 \leqslant y_i \leqslant 1$ of (PART).

To show that $\bar{y}$ also satisfies all lifted Steiner partition inequalities (3.4.2) of (PART), let $S \in \mathcal{S}$ and $(n_1, \ldots, n_{n_S}) \in \mathcal{N}(S)$ define a non-redundant lifted Steiner partition inequality. We may assume that S is a stable set, because otherwise the lifted Steiner partition inequality would be redundant.

If $v \notin S$, then the lifted Steiner partition inequality trivially holds for $\bar{y}$, because it holds for y and, as $v \in T$, the coefficient of variable y_v is 0. Hence, we may assume $v \in S$ for the rest of the proof.

Again, we distinguish several cases depending on S.

Case 1: $x, z \in S$

Let $S' := S - \{v\}$, as illustrated in Figure 3.4.5. Note that $u, w \notin S$, because S is assumed to be a stable set.

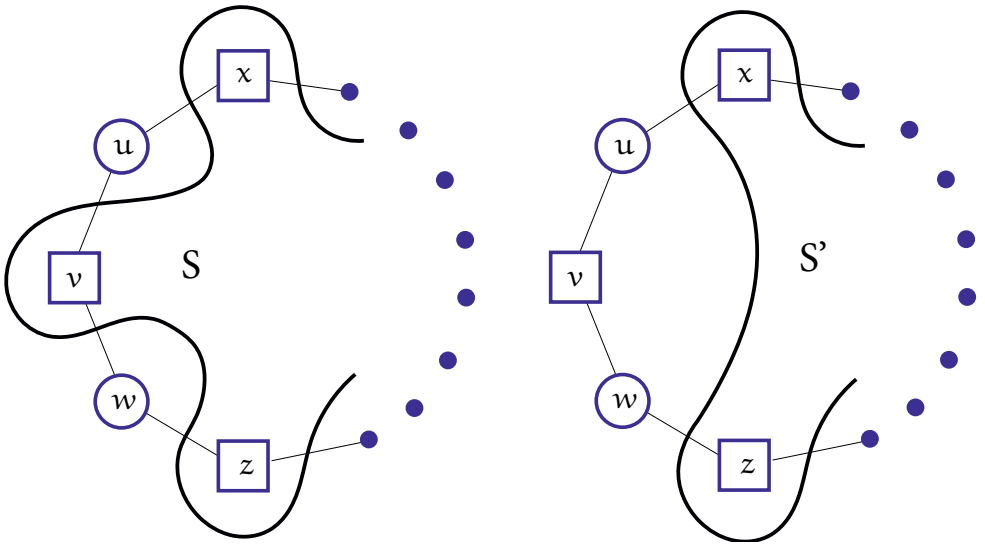


Figure 3.4.5: Lemma 3.4.11, Case 1

If S' is not a separator, we must have $x = z$, $S = \{x, v\}$, and $V = \{x, u, v, w\}$. Otherwise, S would not be a stable set or not be a separator. In this case, the inequality belonging to S is

$$y_v + y_x - y_u - y_w \geqslant -1\,.$$

which is trivially satisfied by $\bar{y}$.

If S' is a separator, then $G \setminus S'$ contains one more terminal component than $G \setminus S$, namely the component $\{u, v, w\}$, while the two terminal-free components $\{u\}$ and $\{w\}$ of $G \setminus S$ are no longer components of $G \setminus S'$. All other components of $G \setminus S$ are also components of $G \setminus S'$. As the two components $\{u\}$ and $\{w\}$ of $G \setminus S$ contain only a single node each, the two non-terminal variables y_u and y_w necessarily occur in the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$ with a coefficient of -1. We may assume w.l.o.g. that the terminal-free components of $G \setminus S$ are numbered in such a way that $u = n_{n_S-1}$ and $w = n_{n_S}$. As all other terminal-free components remained unchanged, we have $\mathcal{N}(S) = \mathcal{N}(S') \times \{u\} \times \{w\}$. Thus, S' and $(n_1, \ldots, n_{n_S-2}) \in \mathcal{N}(S')$ define the lifted Steiner partition inequality

$$\sum_{i \in S'} y_i - \sum_{i=1}^{n_{S'}} y_{n_i} \geqslant t_{S'} - 1\,,$$

which is part of the system (PART) and hence valid for y. Adding (3.4.8), we get

$$\sum_{i \in S'} y_i + y_v - \sum_{i=1}^{n_{S'}} y_{n_i} - y_u - y_w \geqslant t_{S'} - 2 + \varepsilon = t_S - 1 + \varepsilon\,.$$

With (3.4.9) this implies

$$\sum_{i \in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1\,.$$

In other words, $\bar{y}$ satisfies the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$.

Case 2: $z \in S$, $x \notin S$ (or, analogously, $z \notin S$, $x \in S$)

In this case, we let $S' := S \setminus \{v\} \cup \{u\}$, as shown in Figure 3.4.6 on the following page. Since S is assumed to be a stable set, we have $u, w \notin S$.

Replacing v by u in the separator, the terminal-free component $\{w\}$ of $G \setminus S$ becomes a terminal component $\{v, w\}$ in $G \setminus S'$, and the terminal component containing x in $G \setminus S$ becomes smaller in $G \setminus S'$, as node u is removed from this component. All other components of $G \setminus S$ remain unchanged in $G \setminus S'$. As $\{w\}$ is a single node terminal-free component of $G \setminus S$, the variable y_w must occur in the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$ with a coefficient of -1. Assuming that $w = n_{n_S}$ corresponds to the last terminal-free component in $N(S)$, we have $(n_1, \ldots, n_{n_S-1}) \in \mathcal{N}(S')$. Thus, lifted Steiner partition inequality for S' and $(n_1, \ldots, n_{n_S-1})$

$$\sum_{i \in S'} y_i - \sum_{i=1}^{n_S-1} y_{n_i} \geqslant t_{S'} - 1$$

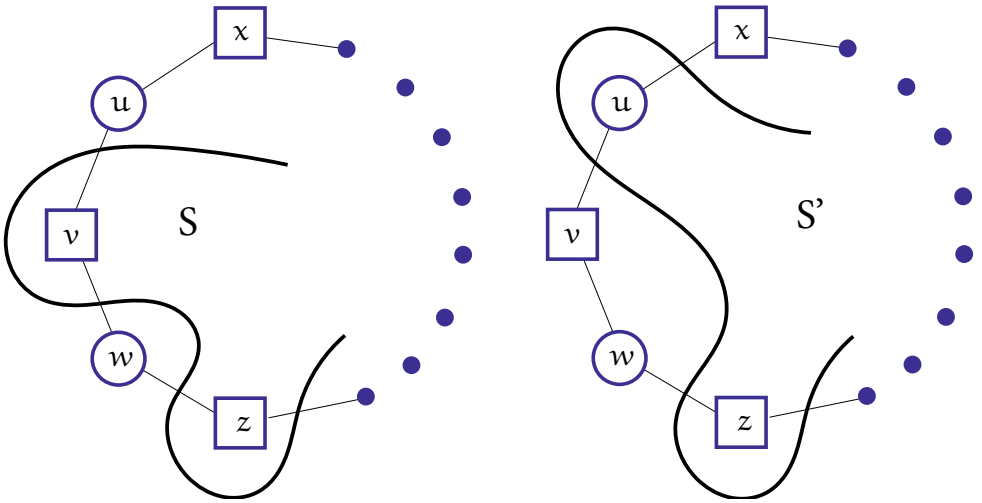


Figure 3.4.6: Lemma 3.4.11, Case 2

is part of (PART) and holds for y. Adding (3.4.8), we get

$$\sum_{i \in S'} y_i + y_v - y_u - \sum_{i=1}^{n_S - 1} y_{n_i} - y_w \geqslant t_{S'} - 2 + \varepsilon = t_S - 1 + \varepsilon \,.$$

With (3.4.9) this implies

$$\sum_{i \in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1 \,,$$

so the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$ holds for $\bar{y}$.

Case 3: $x, z \notin S$

Because S is a stable set, we have $u, w \notin S$ in this case. We let $S' := (S \setminus \{v\}) \cup \{u, w\}$, as illustrated in Figure 3.4.7 on the following page. Note that S' must be a separator in this case, because x and z must belong to different components of $G \setminus S'$ as S is a separator.

Note that $G \setminus S'$ contains one terminal component more than $G \setminus S$, namely the component $\{v\}$, and that the terminal-free components of $G \setminus S$ and of $G \setminus S'$ are exactly the same. Hence, the lifted Steiner partition inequality for S' and $(n_1, \ldots, n_{n_S})$

$$\sum_{i \in S'} y_i - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_{S'} - 1 = t_S$$

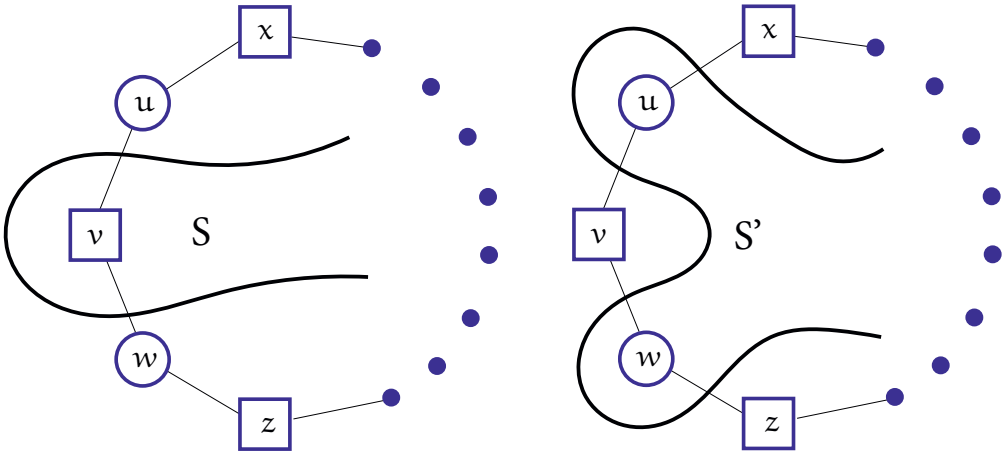


Figure 3.4.7: Lemma 3.4.11, Case 3

is contained in (PART) and satisfied by y. Adding (3.4.8) and plugging in (3.4.9) we obtain first

$$\sum_{i\in S'} y_i + y_v - y_u - y_w - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1 + \varepsilon \quad \text{and then}$$

$$\sum_{i\in S} \bar{y}_i - \sum_{i=1}^{n_S} \bar{y}_{n_i} \geqslant t_S - 1\,.$$

Hence, $\bar{y}$ fulfills the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$ also in this case, which concludes the proof. □

Together, Lemma 3.4.10 and Lemma 3.4.11 imply that the equalities (3.4.3) hold for all irreducible optimal solutions of (PART) for all nonnegative objective functions. This now allows us to prove the main result of this section.

Theorem 3.4.12. *Let* $G = (V, E)$ *be a cycle with* $V = \{v_0, \ldots, v_{2k-1}, v_{2k} = v_0\}$, $E = \{v_i v_{i+1} \mid i = 0, \ldots, 2k-1\}$, *and* $T = \{v_0, v_2, \ldots, v_{2k-2}\}$ *for* $k \geqslant 2$. *Then the lifted Steiner partition inequalities* (3.4.2) *and the nonnegativity constraints* $y_v \geqslant 0$ *for all* $v \in V$ *completely describe the dominant of* P. *In other words, for each nonnegative objective function* $c \in \mathbb{R}^V_{\geqslant 0}$ *there exists an optimal solution of (PART) that is integer.*

Proof. For each nonnegative objective function c, there exists an optimal solution of (PART) that is irreducible. As we have seen in the previous two lemmata, any irreducible optimal solution satisfies $y_{v_i} = y_{v_{i-1}} + y_{v_{i+1}} - 1$ for all terminal nodes $v_i \in T$. Hence, it suffices to show that the linear program (PARTE) obtained by adding these equalities to (PART) has integer optimal solutions. This system reads

$$\begin{aligned}
&\text{(PARTE):} && \min c^T y && && \\
&\text{subject to} && \sum_{v\in S} y_v - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1 && \forall\, S \in \mathcal{S},\ (n_1,\dots,n_{n_S}) \in \mathcal{N}(S) && (3.4.10)\\
& && y_t - \sum_{v\in\Gamma_t^*} y_v = -1 && \forall\, t \in T && (3.4.11)\\
& && 1 \geqslant y_v \geqslant 0 && \forall v \in V && (3.4.12)
\end{aligned}$$

To prove that this formulation has integer optimal solutions, we show how the associated polyhedron is obtained from the minimum spanning tree polytope of a smaller graph. To this end, we consider the graph $G' = (V', E')$ with

$$\begin{aligned}
V' &:= \{v_0, v_2, \dots, v_{2k-2}\} = T \quad \text{and}\\
E' &:= \{e_1, e_3, \dots, e_{2k-1}\} \quad \text{with} \quad e_i = v_{i-1}v_{i+1} \text{ for } i = 1,\dots,2k-1.
\end{aligned}$$

This graph G' is a cycle on the k terminals of G, whose edges correspond to the non-terminal nodes. This allows us to associate each edge e_i with the corresponding non-terminal v_i. On this graph G', we define the edge-based objective function $c' \in \mathbb{R}^{E'}$ as

$$c'_{e_i} := c_{v_{i-1}} + c_{v_i} + c_{v_{i+1}} \quad \text{for all } e_i \in E'.$$

Obviously, $c' \geqslant 0$.

It is not hard to see that there is a simple correspondence between irreducible solutions for the NWDST problem on G and minimum spanning trees in G': Any given spanning tree $B' \subseteq E'$ in G' defines a connected dominating Steiner tree $I = I(B') := \{v_i \mid e_i \in B'\} \cup \{v_i \mid \text{both } e_{i-1}, e_{i+1} \in B'\}$ with objective $c(I) = c'(B') - c(T)$. Reversely, the set $I \subset V$ of internal nodes of any irreducible connected dominating Steiner tree in G defines a spanning tree $B' = B'(I) := \{e_i \mid v_i \in I \setminus T\}$ with objective $c'(B') = c(I) + c(T)$. Note that the objective values $c(I)$ and $c'(B')$ of a connected dominating Steiner tree and its corresponding spanning tree differ by $c(T)$, which is constant for any given problem instance. This allows us to derive the polyhedral description of the irreducible connected dominating Steiner trees in G from the polyhedral description of the minimum spanning trees in G'.

Given a partition $V_1 \dot\cup \dots \dot\cup V_k = V$ of the nodes of some graph $G = (V, E)$, we denote by $\delta(V_1,\dots,V_k) := \{uv \in E \mid u \in V_i,\ v \in V_j \text{ for } i \neq j\}$ the set of edges between different node sets of the partition. It has been shown in Chopra [1989] that the minimum spanning tree polytope for any graph G is completely described by the boundary constraints $1 \geqslant x_e \geqslant 0$ for all $e \in E$ and the Steiner partition inequalities $\sum_{e\in\delta(V_1,\dots,V_k)} x_e \geqslant k-1$ for all partitions $V_1 \dot\cup \dots \dot\cup V_k = V$. In fact, it is sufficient to consider so-called valid partitions, where each component V_i is connected. Partition inequalities corresponding to non-valid partitions are redundant. Applying this result to our graph G', we find that the following linear program is integral, i.e., has an integer optimal solution for each objective function:

$$\begin{aligned}
\min \sum_{e\in E'} c'_e y_e & \\
\sum_{e\in\delta(V'_1,\dots,V'_k)} y_e &\geqslant k-1 && \forall \text{ valid partitions } V'_1 \dot\cup \dots \dot\cup V'_k = V' \\
1 \geqslant y_e &\geqslant 0 && \forall\, e \in E'
\end{aligned}$$

Interpreting the edges e_i of G' as non-terminal nodes v_i in G, valid partitions $V'_1 \dot\cup \dots \dot\cup V'_k$ of the nodes of G' can be interpreted as separator node sets $S \subseteq V \setminus T$ in G. Given a valid partition $V'_1 \dot\cup \dots \dot\cup V'_k = V'$, the corresponding separator is $S := \{v_i \mid e_i \in \delta(V'_1, \dots, V'_k)\}$ and the number of resulting components is k. Note that each of the resulting components is a terminal component and that each (terminal) node set V'_i is fully contained in one of these components. Reversely, any subset $S \subseteq V \setminus T$ defines a partition $V'_1 \dot\cup \dots \dot\cup V'_k$ of V' with $k = t_S$. Hence, we can equivalently write the above linear program as follows:

$$\begin{aligned}
\min \sum_{v\in V\setminus T} c'_v y_v & \\
\sum_{v\in S} y_v &\geqslant t_S - 1 && \forall\, S \subseteq V \setminus T \\
1 \geqslant y_v &\geqslant 0 && \forall\, v \in V \setminus T
\end{aligned}$$

Clearly, also this linear program is integer. However, it only contains the variables for the non-terminal nodes $v \in V \setminus T$. Introducing the missing variables y_t for the terminal nodes $t \in T$ together with the equalities $y_t - \sum_{v\in\Gamma^*_t} y_v = -1$ linking them to the variables of their neighboring non-terminal nodes and adding the constant $-c(T)$ to the objective, we obtain the following linear program:

$$\begin{aligned}
\text{(MT)} \qquad \min \sum_{v\in V\setminus T} c'_v y_v - c(T) \quad \Big(= \sum_{v\in V} c_v y_v\Big) & && && (3.4.13)\\
\sum_{v\in S} y_v &\geqslant t_S - 1 && \forall\, S \subseteq V \setminus T && (3.4.14)\\
y_t - \sum_{v\in\Gamma^*_t} y_v &= -1 && \forall\, t \in T && (3.4.15)\\
1 \geqslant y_v &\geqslant 0 && \forall\, v \in V &&
\end{aligned}$$

Note that the newly introduced variables y_t for $t \in T$ neither occur in the objective function (with respect to c') nor in the partition inequalities (3.4.14). In fact, the variable y_t for terminal node $t \in T$ only occurs in its boundary constraints and in the equality (3.4.15) linking it to its two neighboring non-terminals. Given integer values for the variables y_v for $v \in V \setminus T$, one can thus set the values of the y_t for $t \in T$ in such a way that (3.4.15) and the boundary constraints are satisfied. Hence, also the linear program (MT) has an integer optimal solution for each objective function c'.

For nonnegative objective functions c, also c' is nonnegative. Hence, as shown in the previous lemmata, the irreducible solutions y of (PARTE), which coincide with the irreducible solutions of (MT), satisfy the additional equalities (3.4.3), i.e., we have $y_v = y_{v-1} + y_{v+1} - 1$ for all $v \in T$. This immediately implies that these solutions y also satisfy $\sum_{v \in V \setminus T} c'_v y_v = \sum_{v \in V} c_v y_v + c(T)$. Hence, (MT) has integer optimal solutions also for the objective $c^T y$ if c is nonnegative.

To conclude the proof, we now show that (MT) and (PARTE) are equivalent and define the same set of feasible solutions. Obviously, both models contain the same boundary constraints and the same equalities (3.4.11) and (3.4.15), respectively. Furthermore, the inequalities of type (3.4.14) are a subset of the larger class of lifted Steiner partition inequalities (3.4.10), which implies that (MT) is a relaxation of (PARTE). Hence, it suffices to show that all lifted Steiner partition inequalities (3.4.10) are implied by the constraints of (MT).

So, let $S \in \mathcal{S}$ and $(n_1, \ldots, n_{n_S}) \in \mathcal{N}(S)$ and consider the lifted Steiner partition inequality (3.4.10) defined by S and $(n_1, \ldots, n_{n_S})$. Due to Lemma 3.4.8 we may assume without loss of generality that S is a stable set, as otherwise the inequality would be redundant.

Let $S^+ := \{v \in \Gamma^*_t \mid t \in S \cap T\}$ be the set of nodes that are adjacent to a terminal node in S. Since S is stable, we have $S^+ \cap S = \emptyset$. Because G is a cycle, nodes in S^+ may be neighboring either one or two terminal nodes in S. For simplicity, we denote by $S^+_2 := \{v_i \in S^+ \mid v_{i-1}, v_{i+1} \in S\}$ the set of nodes that are adjacent to two terminal nodes is S.

Finally, let $S' := S \setminus (S \cap T) \cup S^+$ be the separator obtained by replacing all terminals in S by their two neighbors. Obviously, we have $t'_S = t_S + |S \cap T|$. Also, as terminals and non-terminals alternate on the cycle G, the terminal-free components of $G \setminus S$ must be exactly the single nodes in S^+_2, i.e., $(n_1, \ldots, n_{n_S})$ is just some ordering of the nodes in S^+_2. Since $S' \subseteq V \setminus T$, the corresponding partition inequality (3.4.14)

$$\sum_{v \in S'} y_v \geqslant t'_S - 1$$

is part of (MT). Adding the equalities (3.4.15) for all $t \in S \cap T$, one obtains

$$\sum_{v \in S'} y_v - \sum_{v_i \in S \cap T} (y_i - y_{i-1} - y_{i+1}) \geqslant t'_S - 1 - |S \cap T| = t_S - 1\,.$$

Rearranging terms, we get

$$\Big(\sum_{v \in S'} y_v + \sum_{v \in S^+} y_v - \sum_{v \in S \cap T} y_v\Big) - \sum_{v \in S^+_2} y_v \geqslant t_S - 1\,,$$

which finally leads to

$$\sum_{v \in S} y_v - \sum_{i=1}^{n_S} y_{n_i} \geqslant t_S - 1\,.$$

Hence, the lifted Steiner partition inequality for S and $(n_1, \ldots, n_{n_S})$ is implied by the constraints of (MT).

Consequently, both (MT) and (PARTE) describe the same set of solutions and, as (MT) has integer optimal solutions, so does (PARTE). □

We can extend the characterization to instances in circles where more than one non-terminals form a path. We say that a non-terminal v *lies between* the terminals t_1 and t_2 if walking along the circle in both possible diretions, beginning at the node v, these are the first terminals we encounter.

Lemma 3.4.13. *Let $G = (V, E)$ be a cycle and let $T \subset V$ be a terminal set that forms a stable set in G, that is, no two terminals are adjacent. Let $c \in \mathbb{Q}^V_{\geqslant 0}$ be nonnegative. Then, for any component C of $G \setminus T$ and any irreducible integer solution y, all nodes $v \in C$ have the same value y_v.*

Proof. Note that the components of $G \setminus T$ consist of non-terminal nodes only. Furthermore, any terminal node $t \in T$ is adjacent to exactly two components of $G \setminus T$. Clearly, an integer solution y must choose all nodes of all but one component of $G \setminus T$ in order to be feasible. Otherwise, at least two nodes of two different components of $G \setminus T$ would not be chosen, and these non-chosen nodes would define a terminal separator. With this observation, one easily verifies that an integer solution y is irreducible if and only if it chooses all nodes of G except for those of one component C of $G \setminus T$ and the two terminal nodes adjacent to this component C. Hence, the variables of all non-terminals in the same component are either all equal to 1 or all equal to 0. □

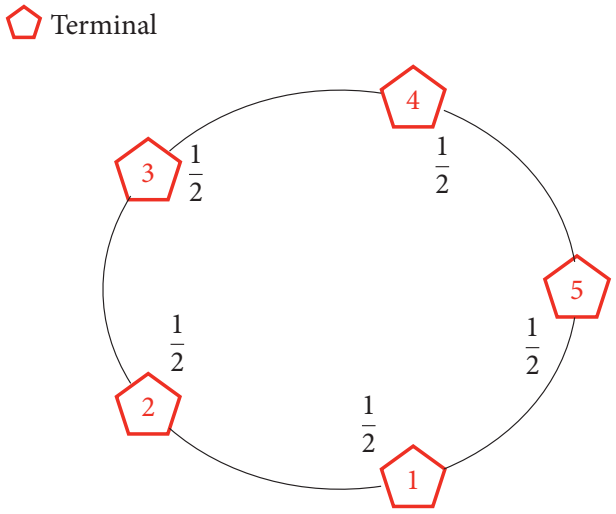


Figure 3.4.8: A valid basic solution for P that is not cut away by the lifted Steiner partition inequalities

In cycles where terminals are adjacent, the lifted partition inequalities are insufficient to fully describe the dominant of P. A simple example is given in Figure 3.4.8. In this example, the inequality $\sum_{x \in V} x_i \geqslant 3$ is needed in the description of the dominant of P, but it is not implied by the (lifted) Steiner partition inequalities.

However, for instances where terminals are adjacent, the solutions do not change structurally if we insert non-terminals with zero cost between every adjacent terminal-pair. One easily verifies that the optimal irreducible solutions for the modified instance including these extra

non-terminals exactly correspond to optimal irreducible solutions of the original problem: An artificial non-terminal will be chosen if and only if one of its adjacent terminals is chosen.

Hence, we immediately get a complete description for the case with adjacent terminals using an extended formulation with at most twice as many variables.

Theorem 3.4.14. *The dominant of* P *is completely described by the lifted partition inequalities and the variable bounds in the case where the underlying graph* G *is a cycle and no two terminals are adjacent.*

In the case where adjacent terminals exist, these inequalities yield a complete description of an extended formulation, where we lift in one artificial non-terminal variable for each adjacent terminal pair.

In the case where adjacent terminals exist, one can construct a complete description in the original variable space by projecting this extended description back onto the original variable space in a straightforward way, using Fourier-Motzkin-Elimination for example.

3.4.3 Indegree inequalities

In this subsection, we will derive a connection between the lifted Steiner partition inequalities and the indegree inequalities. The well-known indegree inequalities are valid for the description of connected subgraphs. Let $V = \{1, \dots, n\}$ and $d \in \mathbb{R}^n$. We call d an *indegree-vector* if there is an orientation O of G such that for this orientation, d_i is the vector of indegrees of G oriented by O. For each such vector d, the corresponding indegree inequality

$$\sum_{i \in V} (1 - d_i) y_i \leqslant 1$$

is then valid. Korte et al. [1991] have shown that these inequalities induce all nontrivial facets of the connected subgraph polytope when G is a tree. Further conditions under which these inequalities are facet-defining for the connected subgraph polytope have been studied by Wang et al. [2015].

For the NWDSTP polytope P on a cycle graph, we will now show that the indegree inequalities are implied by lifted Steiner partition inequalities.

Theorem 3.4.15. *If* $G = (V, E)$ *is a simple cycle, the indegree inequalities are implied by the lifted Steiner partition inequalities* (3.4.2).

Proof. Let $G = (V, E)$ denote a simple cycle and let O be an orientation of G. Let d denote the indegree vector corresponding to O. Let $V := V_0 \dot\cup V_1 \dot\cup V_2$, where $V_i := \{ v \in V \mid d_v = i \}$. We set $S := V_2$, and, as above, let t_S be the number of terminal-components in $G - S$. Because G is a simple cycle, every component of $G - S$ contains exactly one node from V_0. We set these nodes as the n_i. Let $T(S)$ be the terminal-components from $G \setminus S$. The resulting lifted partition inequality is given as:

$$\sum_{i \in V_2} y_i \geqslant \sum_{i \in V_0 \setminus T(S)} y_{n_i} + t_S - 1.$$

After adding $\sum_{i \in V_1 \cup V_2} y_i$ to both sides, we obtain:

$$\sum_{i \in V_2} 2y_i + \sum_{i \in V_1} y_i \geqslant \sum_{i \in V_1 \cup V_2} y_i + \sum_{i \in V_0 \setminus T(S)} y_{n_i} + t_S - 1.$$

Finally, after rewriting, we have:

$$\sum_{i \in V} d_i y_i = \sum_{i \in V_2} 2y_i + \sum_{i \in V_1} y_i + \sum_{i \in V_0} 0 \cdot y_i \geqslant \sum_{i \in V_1 \cup V_2 \cup (V_0 \setminus T(S))} y_i + t_S - 1 \geqslant \sum_{i \in V} y_i - 1$$

which concludes the proof. □

3.5 Computational experiments

In this section we evaluate the effectiveness of the partition inequalities in comparison to the pure model inequalities. We solve the corresponding linear programs and then compare their values to the optimal solution determined by a branch-and-cut approach.

3.5.1 Instances

Our test instances are based on benchmark instances for the traditional Steiner tree problem taken from the well-known SteinLib (Koch et al. [2000]) library. To create the node costs for our models, we take for every node the average of the costs of the adjacent edges and scale and round them to integer values. Because the partition inequalities are expected to be especially effective in highly sparse graphs with lots of terminals, we also generate some extended instances based on the SteinLib instances by replacing every original edge with a path of length 4 introducing two terminals and one non-terminal intermediate nodes and distributing the original edge costs evenly to these new nodes. Additionally, we also included the instance alt10 defined on a cycle of length 10 with alternating terminals and non-terminals to verify that the partition inequalities indeed close the integrality gap on such instances.

3.5.2 Implementation

We implemented two cutting plane approaches, one which generates the model inequalities and another one in which we also separate the lifted partition inequalities described above. Furthermore, we implemented a straightforward single-commodity arc-variable based flow formulation to verify the optimal integer solution values. We omit the model details, but mention that it is very similar to models used in Fernandes and Gouveia [1998]. All linear and integer programs are solved using Gurobi. The optimal solution values are then used to evaluate the quality of the LP relaxation of the node-based formulation with and without lifted partition inequalities.

To separate the model inequalities, we use the well-known HIPR code for max-flow problems by Andrew Goldberg (Goldberg [2012]) and employ the standard transformation from a max-flow solution to find a $s - t$—minimum cut by performing a graph search like BFS along non-saturated edges, taking all reached as one part of the cut.Note that the separating set in

inequalities (3.2.1), (3.2.2) and (3.2.3) is not an $s-t$—cut, but a separating set disjoint from $\{s,t\}$. To find this set using $s-t$—cuts, we solve the min $s-t$—cut problem in in a transformed instance, where every node v is replaced by two nodes v^+, v^- connected by an arc from v^+ to v^- with a capacity coming from equal to the fracional value of the corresponding binary node variable in the LP solution. An arc in the original graph from some v to some w becomes an arc from v^- to w^+ of infinite capacity. As the capacity of these arcs is infinite, these will not be part of the min-cut, resulting in a separating set corresponding to a node set in the original graph.

As the partition inequalities turn out to be computationally difficult to separate, we use a greedy heuristic approach to separate them instead of an exact separation routine. Thus, the presented improvements by using the partition inequalities are only lower bounds with respect to their true effect.

Our heuristic works as follows: First, given an LP solution, we consider the set $S^* := V \setminus T$. We remove from S^* all nodes whose fractional value in the LP solution is equal to 1. The resulting set induces a (lifted) partition inequality. We define the *neighborhood* of a separator S as the set of subsets of V that result from either removing a non-terminal node from S or adding a terminal node to S. To strengthen the inequality, we then generate every separator in its neighborhood and check how much the corresponding partition inequality is violated. We then continue with the most violated inequality we found and the separator belonging to it and again try to strengthen it by enumerating its neighborhood. This process is then continued until the separator is the set T.

We then add the most violated of the considered partition inequalities to the LP and resolve. If there are several partition inequalities with the same violation, we use one with minimal support.

3.5.3 Results

We present the results of our computational experiments in Table 3.1. The columns contain the instance name ("Instance"), the number of nodes ("n"), the number of edges ("m"), the average cycle length ("cycle"), the gap for the Steiner relaxation ("gap St") and the gap for Steiner+partition relaxation ("gap ST+P"). The average cycle length is the length of a shortest cycle containing a particular edge e, averaged over all edges e of the graph.

The instances b01, b02, b03, b05, b08 and b10 were already completely solved by the LP relaxation without the partition inequalities. We therefore omitted the computational results for these instances.

We remark that we use the computational experiments only to estimate the qualitative effect of the partition inequalities on the LP bounds and gaps for some benchmark instances. We believe that it is possible to extend the computational results to a full-fledged computational study employing many algorithmic tools to make the separation of our inequalities effective in a Branch&Cut-framework, but this will not be done here.

As can be seen in Table Table 3.1, the partition inequalities are useful for improving some of the gaps in the original instances. For the extended instances, the rather large gaps of the Steiner-only relaxation are almost or completely closed in all test cases. As expected, the lifted partition inequalities are rather effective in highly sparse graphs, especially with many terminals, while they are much less effective in graphs graphs with higher density or fewer terminals.

This might be because extremely sparse graphs with many terminals very likely contain large cycles with many terminals, on which the lifted partition inequalities have been proven to be effective. Note that the extended instances we created by replacing every original edge with a path of length 4 have these special properties by construction. Graphs with these properties also arise in several real-world applications, for example, when many clients distributed along street networks need to be connected.

3.6 Conclusions

We introduced the Node Weighted Dominating Steiner Tree problem, a generalization of the Minimum Connected Dominating Set problem in graphs, in which a least-cost subset of nodes is to be found such that the given set of terminals is dominated. We provided an integer programming model that uses node variables only and that requires polynomial separation of an exponential number of connectivity cuts. For the underlying polytope, we gave the conditions under which these inequalities are facet-defining. In the second half, we introduced a new family of Steiner partition inequalities and showed how to lift them to provide stronger linear programming relaxation bounds. For the special case when the input graph is a cycle, we showed that the dominant of the underlying polytope is either integral, or can be lifted and projected to easily obtain integral solutions.

Finally, we believe that for this new and challenging problem which is of particular importance in the design of communication networks dealing with cloud services, much more studies need to be done. On the one hand, exact methods for handling the large-scale instances are needed. The practical relevance and computational strength of the minimal node-separator inequalities studied in this chapter have been demonstrated in Fischetti et al. [2015], where these inequalities are shown to play an important role for the related Steiner Tree problems. However, for the new families of (lifted) Steiner partition inequalities, it remains an open question how they can influence the performance of Branch&Cut algorithms. On the other hand, further polyhedral studies could support the development of exact methods by the investigation of other families of valid inequalities that strengthen the LP relaxation bounds.

Table 3.1: Comparison of the LP relaxation gaps for the model with Steiner inequalities only and the model with both Steiner and lifted partition inequalities.

Instance	n	m	t	cycle	gap St[%]	gap St+P[%]
alt10	10	20	5	10.00	28.57	0.00
b04	50	200	9	4.00	3.75	3.75
b06	50	200	25	3.99	2.25	2.25
b07	75	188	13	4.15	0.40	0.27
b09	75	188	38	4.07	0.97	0.97
b11	75	300	19	4.58	2.42	2.42
b12	75	300	38	4.49	8.50	8.50
b13	100	250	17	4.14	0.92	0.92
b14	100	250	25	4.14	1.63	1.25
b15	100	250	50	4.98	1.36	0.44
b16	100	400	17	4.66	6.63	6.63
b17	100	400	25	4.49	4.52	4.52
b18	100	400	50	4.27	3.41	3.41
c01	500	1250	5	5.68	8.69	7.62
c02	500	1250	10	5.42	12.30	12.05
c03	500	1250	83	5.54	6.55	5.42
c04	500	1250	125	5.78	2.91	2.19
c05	500	1250	250	5.22	1.32	0.71
c06	500	2000	5	5.53	0.57	0.57
c08	500	2000	83	5.58	4.76	4.76
c10	500	2000	250	5.52	1.70	1.70
b01-ext	239	504	126	11.30	11.05	0.00
b02-ext	239	504	126	18.35	29.11	0.00
b03-ext	239	504	126	14.29	11.80	0.00
b04-ext	350	800	200	16.00	15.57	0.00
b05-ext	350	800	200	15.68	10.40	0.00
b06-ext	350	800	200	15.96	19.28	0.00
b07-ext	357	752	188	16.60	18.98	0.00
b09-ext	357	752	188	16.30	8.27	0.06
b11-ext	525	1200	300	18.32	14.64	0.00
b12-ext	525	1200	300	17.95	3.46	0.00
b13-ext	475	1000	250	16.58	17.42	0.00
b14-ext	475	1000	250	16.54	13.68	0.00

4 Incremental Facility Location

4.1 The incremental UFL problem

In this chapter, we consider the incremental UFL problem. The research presented here is joint work with Ashwin Arulselvan and Martin Skutella and has and has appeared in "Networks" (Arulselvan et al. [2015]).

4.1.1 Introduction

In this chapter, we consider a problem concerning the incremental building of networks, that is, the network is built in a longer some time period due to financial, technical or other resource constraints. The overarching goal is to have a good network not just when it is completely built, but also at intermediate points in time. This involves a decision which customers to ignore at a certain point in time and which ones to service and then how to keep incrementally building these sets. We consider what could be seen as a more abstract version of the 2FTTx-problem from Chapter 2, called the *Uncapacitated Facility Location problem* (UFLP). We introduced this problem earlier in Section 1.3.6. In the context of optical networks, the facility opening cost can be understood as the cost of installing physical devices at certain junction points, while the assignment cost could be seen as the installation cost of optical cables.

Given an UFLP instance, we can also ask for a cost-minimal solution that only covers $k < n$ of the original customers. This problem has been introduced in Charikar et al. [2001], they call it the *Robust Facility Location problem* (RFLP). The robustness here does not refer to robustness in the sense of Bertsimas and Sim [2003]; it is meant in the sense that optimality of a solution is quite indifferent to the shifting of outlier customers that lie very far away, as these can be chosen to be ignored.

We consider a variant of the RFLP called the ℓ-RFLP. In this variant, we are given an additional input $\ell \leqslant n$ and we would like to construct a cost-minimal solution of the UFLP that connects an arbitrary subset of size ℓ of the n customers to open facilities such that the total cost is minimized.

Because we are allowed to ignore the remaining $n - \ell$ customers, the presence of a very small fraction of customers who are far away has no effect on the solution.

As already stated above, we study here an incremental version of the aforementioned problem. To be more specific, we study the Incremental Facility Location problem (IncFLP). A solution of the problem amounts to a sequence of facilities and customers and an assignment of customers to facilities. Here, the customer sequence defines the order in which the customers will be served and the facility sequence defines the order in which the facilities will be opened. We can understand the sequence as consecutive events happening in time, either the opening of a facility or the servicing of a new customer. We disallow assigning customers to non-open

facilities and also forbid reassignment of customers. In this sense, the solution we seek for will be strictly incremental.

For every intermediate time point ℓ in the sequence, we compare the cost of the solution incurred by the incremental sequence up to this point to the optimal cost of the ℓ-RFLP. We will make these definitions more precise in the next section.

A planner or network builder could choose a specific point in the sequence depending on the resource availability and choose to serve that many customers in the sequence at that point. Subsequent service could be provided for the remaining customers in the sequence at later time periods. Because we compare the cost of serving the first ℓ customers in the sequence with the cost of optimally serving any ℓ customers, we make sure that the sequence will be cost effective at every intermediate point.

Related work

The Uncapacitated Facility Location problem (UFLP) with metric assignment cost has a long line of research (see e.g. Byrka and Aardal [2010]; Chudak and Shmoys [2003]; Cornuéjols et al. [1990]; Shmoys et al. [1997]). The best approximation guarantee known for this problem is 1.488 (Li [2011]) and there can be no polynomial time algorithm to approximate it within a ratio of 1.463 unless $\mathbf{NP} \subseteq \mathrm{DTIME}(n^{O(\log\log n)})$ (Guha and Khuller [1999a]). The problem has also been studied in incremental settings (Lin et al. [2010]; Plaxton [2003]) but prior work involves different settings than the one considered in our work.

Plaxton [2003] is interested in a nested sequence of facilities and a threshold sequence for the scaling of the assignment costs. A solution in the sequence is a solution corresponding to a specific scaling factor, which could be inferred from the threshold sequence. Each solution in the sequence serves all customers. The nested sequence of facilities gives a near optimal solution for every possible scaling factor. Lin et al. [2010] provided a framework for solving several incremental problems and solved the problem studied in Plaxton [2003] under the same settings. The framework proposed by them uses a black box algorithm to iteratively augment partial competitive solutions to reach subsequent competitive solutions that eventually become complete. Chrobak et al. [2008] obtained competitive deterministic and randomized algorithms for the Incremental k-median problem through a reduction to the Online Bidding problem. We follow the doubling technique proposed in the above works. This involves incrementally building a solution by comparing it with the optimal solution to ensure competitiveness and refining it when it is necessary. Since we are required to produce a sequence of customers and facilities in our solution, we deviate slightly by saving the partially refined solutions and delay the construction of incremental solutions. We follow up with a second phase to construct the incremental solution from the partially saved refined solutions.

In Fotakis [2006], the authors are interested in an incremental solution when the demand points arrive one at a time assuming we have complete knowledge of the network. A constant factor competitive algorithm is proposed assuming uniform facility cost. In this algorithm, merging of existing clusters is allowed. This is in contrast to an online algorithm in Meyerson [2001], where the decision to be served by a facility is irrevocable. An $O(\log n)$-competitive algorithm was proposed for this version of the online problem. This was later improved to $O(\frac{\log n}{\log\log n})$ in Fotakis [2008]. In both works, the algorithm is compared against an offline

algorithm that is aware of the arrival order of the customers and the competitive ratio is calculated for all possible arrival orders.

The robust version of the facility location problem (RFLP) was first studied by Charikar et al. [2001]. They gave a 3-approximation algorithm using the primal-dual technique. This approximation guarantee was later improved to 2 by Jain et al. [2003]. We will be using their algorithm as a black box algorithm in our framework.

Our contributions

We provide an algorithm that produces a sequence that is within a factor of 8 from the optimal ℓ-RFLP for $\ell = 1, \ldots, |R|$, where $|R|$ denotes the total number of customers in the given UFLP instance. We then present the computational results of our implementation of the framework on a set of benchmark instances for the UFLP. For each instance, we give the worst-case and average-case bounds of our incremental sequence (for $\ell = 1, \ldots, n$) constructed by the algorithm when compared against the optimal cost of the ℓ-RFLP.

We begin in Subsection 4.1.2 with some basic notation and the formal problem definition. In Subsection 4.1.3, we provide the algorithmic framework and explain it. The analysis of the algorithm which shows it to be 8-competitive is given in Section 4.2. Concerning lower bounds, we present an example restricting the best possible factor that could be achieved by any algorithm in Section 4.3. In Section 4.4, we present our computational results. Finally, we conclude with a summary and notes on future extensions in Section 4.5.

4.1.2 Problem setting

We are given a set F of potential facility locations and a set R of customers. Also, we are given metric service costs $c_S \colon F \times R \to \mathbb{Q}_+$ and facility opening costs $c_F \colon F \to \mathbb{Q}_+$. Let $m = |F|$ be the number of potential facilities and $n = |R|$ the number of customers. We seek nested sets of facilities, customers and assignment edges between them for $\ell = 1, \ldots, n$. We call an approximation algorithm to this problem k-*competitive* if the costs of the induced solutions of the ℓ-RFLPs are not more than a factor of k away from the costs of optimal solutions to the ℓ-RFLPs, where k does not depend on ℓ and also not on the instance.

We need additional notation to make this precise. For a subset of facilities $F' \subseteq F$, we denote the total facility cost of this set by $c_F(F')$. We denote the service cost incurred by serving a set of customers $R' \subseteq R$ by a set of facilities $F' \subseteq F$ by $c_S(F', R') = \sum_{j \in R'} c_S(F', j)$, where $c_S(F', j)$ is the cost of the cheapest assignment of customer j to a facility in F'. For a number r less than $|R'|$, let $c_S(F', R', r)$ denote the cost of the cheapest assignment of r customers from R' to a facility in F'. To provide shorter notation, we also set $c(F', R') := c_F(F') + c_S(F', R')$. In the IncFLP, we seek a sequence to serve customers

$$\emptyset = R_0 \subsetneq R_1 \subsetneq R_2 \subsetneq \cdots \subsetneq R_n = R$$

where $|R_\ell| = \ell$ and a sequence to open facilities

$$F_1 \subseteq F_2 \subseteq \cdots \subseteq F_n$$

such that the customers in $R_\ell \backslash R_{\ell-1}$ can be assigned to facilities in F_ℓ. The objective is to minimize the competitive ratio of the sequence. The competitive ratio of a sequence is defined as

$$\max_{\ell=1,\ldots,n} \frac{c_F(F_\ell) + \sum_{k=1}^{\ell} c_S(F_k, R_k \backslash R_{k-1})}{OPT_\ell} ,$$

where OPT_ℓ is the optimal cost of the ℓ-RFLP.

4.1.3 Two-phase algorithm

Let A be any approximation algorithm for the RFLP. We will use A as a black-box. We write $(Z, M) = A(F, R, \ell)$ and mean that A takes as input the set of potential facility locations F, the set of customers R and an integer $\ell \leqslant n$, where ℓ is the number of customers to be served. It produces the output (Z, M), where $Z \subseteq F$ is the set of facilities opened and $M \subseteq R$, $|M| = \ell$, is the set of customers served by the facilities in Z. Given a set of customers and the set of open facilities, an optimal assignment could be easily computed greedily by assigning each customer in the given set to its closest open facility. We now provide our incremental framework $FacInc(F, R)$.

We start with a solution to the UFLP obtained from A with $\ell := n$. Let $F' \subseteq F$ be the set of facilities opened in this solution. Our algorithm runs in two phases, the refinement phase and the incremental phase. The refinement phase constructs and collects solutions to the ℓ-RFLP. These solutions are not necessarily incremental. In the incremental phase, these partial solutions are glued together to construct an incremental solution. The point at which we collect a new partial solution in the refinement phase is called a refinement point. The initial complete solution for the UFLP will be the first refinement point solution we collect. We iteratively reduce the value of ℓ from $|R|$ to 1 and check if the last collected refinement point solution cost exceeds twice the cost of the solution $A(F, R, \ell)$. If it does, then we collect the $A(F, R, \ell)$ as a new refinement point solution.

Algorithm 3 Refinement phase

```
1: Initialize: F_c := F', R_c := R, P := {n}
2: for ℓ = (n − 1) to 1 do
3:     (F_A, R_A) := A(F, R, ℓ)
4:     if c(F_c, R_c) ⩾ 2(c_F(F_A) + c_S(F_A, R_A)) then
5:         F_ℓ := F_c := F_A
6:         R_ℓ := R_c := R_A
7:         P := P ∪ {ℓ}
8:     end if
9: end for
```

The elements in the set P correspond to the points at which refinement took place. Let their total number $|P| = K$. We shall now index the refinement point solutions from 1 to K in the increasing order of the number of customers they are serving. This helps us in presenting the analysis with clarity. Let us denote the solutions as (F_1, R_1), $(F_2, R_2), \ldots, (F_K, R_K)$; notice that $F_K = F'$ and $R_K = R$.

For $k = 1, \ldots, K$, let $r_k := |R_k|$ be the number of customers served in the k^{th} refinement point solution. We call the time between the service commencement of customer $(r_{k-1}+1)$ and r_k as the k^{th} period (with $r_0 := 0$). We also say this period *belongs* to the solution (F_k, R_k). From a solution (F_k, R_k) at a refinement point, we can construct solutions with a constant approximation ratio for its period.

In the incremental phase, we glue these partial refined solutions together to construct an incremental solution. We will do this by opening all facilities belonging to a period at its beginning.

Algorithm 4 Incremental phase

```
1: Initialize: F̃_1 := F_1, k := 2
2: for ℓ = 2 to n do
3:    if (ℓ − 1) ∈ P then
4:       F̃_ℓ := F̃_{ℓ−1} ∪ F_k
5:       k := k + 1
6:    else
7:       F̃_ℓ := F̃_{ℓ−1}
8:    end if
9: end for
```

Notice that $\tilde{F}_{r_k} = \bigcup_{j=1}^{k} F_j$ and we now have our incremental sequence of facilities:

$$\tilde{F}_1 \subseteq \tilde{F}_2 \subseteq \cdots \subseteq \tilde{F}_{n-1} \subseteq \tilde{F}_n = \bigcup_{k=1}^{K} F_k \ .$$

As for customers, we can pick an incremental sequence greedily by adding in every step the current non-served customer with cheapest service cost with respect to the currently open facilities. This yields a sequence

$$\emptyset = \tilde{R}_0 \subsetneq \tilde{R}_1 \subsetneq \cdots \subsetneq \tilde{R}_{n-1} \subsetneq \tilde{R}_n = R \ .$$

4.2 Analysis

We first show that, at refinement points, the cost of our incremental solution is at most twice the cost of the solution provided by the black-box Algorithm A.

Theorem 4.2.1. *For all refinement points* $k = 1, \ldots, K$*, the cost of the incremental solution for* r_k *customers that is found by our algorithm is at most* $2c_F(F_k) + 2c_S(F_k, R_k)$.

To this end, we first show the following lemma, which bounds the increase in cost from one refinement point to the next.

Lemma 4.2.2. *For all refinement points* $k = 2, \ldots, K$,

$$2c(F_{k-1}, R_{k-1}) + c_F(F_k) + c_S(F_k, R_k \setminus \tilde{R}_{r_{k-1}}, r_k - r_{k-1}) \leqslant 2c(F_k, R_k).$$

Proof. Our refinement condition at step 4 of the incremental phase implies

$$2c(F_{k-1}, R_{k-1}) < c(F_k, R_k).$$

This yields

$$\begin{aligned} 2c(F_{k-1}) + 2c_S(F_{k-1}, R_{k-1}) + c_F(F_k) + c_S(F_k, R_k \setminus \tilde{R}_{r_{k-1}}, r_k - r_{k-1}) \\ < 2c_F(F_k) + c_S(F_k, R_k \setminus \tilde{R}_{r_{k-1}}, r_k - r_{k-1}) + c_S(F_k, R_k) \\ \leqslant 2c_F(F_k) + 2c_S(F_k, R_k) \ . \end{aligned}$$

□

We are now ready to bound the additional cost incurred by the solution being incremental.

Lemma 4.2.3. *For all refinement points* $k = 1, \ldots, K$,

$$2c(F_1, R_1) + \sum_{j=2}^{k} \left(c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{r_{j-1}}, r_j - r_{j-1}) \right) \leqslant 2c(F_k, R_k).$$

Proof. For $k = 1$, the statement is trivial. For $k = 2$, the statement follows directly from Lemma 4.2.2. We proceed by induction on k. Assume the statement is true for $k-1$. Then we get

$$\begin{aligned} &2c_F(F_1) + 2c_S(F_1, R_1) + \sum_{j=2}^{k} \left(c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{r_{j-1}}, r_j - r_{j-1}) \right) \\ &= 2c_F(F_1) + 2c_S(F_1, R_1) + \sum_{j=2}^{k-1} \left(c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{r_{j-1}}, r_j - r_{j-1}) \right) \\ &\qquad + c_F(F_k) + c_S(F_k, R_k \setminus \tilde{R}_{r_{k-1}}, r_k - r_{k-1}) \\ &\leqslant 2c_F(F_{k-1}) + 2c_S(F_{k-1}, R_{k-1}) + c_F(F_k) + c_S(F_k, R_k \setminus \tilde{R}_{r_{k-1}}, r_k - r_{k-1}) \\ &\leqslant 2c_F(F_k) + 2c_S(F_k, R_k) \ , \end{aligned}$$

where the last inequality follows from Lemma 4.2.2 again. □

We are now ready to prove Theorem 4.2.1.

Proof of Theorem 4.2.1. Since $\tilde{F}_{r_k} = \bigcup_{j=1}^{k} F_j$ we can bound the facility opening costs by

$$c_F(\tilde{F}_{r_k}) \leqslant \sum_{j=1}^{k} c_F(F_j) \leqslant 2c_F(F_1) + \sum_{j=2}^{k} c_F(F_j) \ .$$

Moreover, by the greedy construction of the sequence of customers, we can bound the connection cost by

$$c_S(F_1, R_1) + \sum_{j=2}^{k} c_S(F_j, R_j \setminus \tilde{R}_{r_{j-1}}, r_j - r_{j-1}) \ .$$

The desired bound thus follows from Lemma 4.2.3. □

It remains to analyze the performance of our incremental solution for an arbitrary number of customers ℓ, where ℓ is not necessarily a refinement point. The next theorem is an easy consequence of Theorem 4.2.1 and the refinement condition at Step 4 of the refinement phase.

Theorem 4.2.4. *For $\ell = 1, \ldots, n$, the cost of the incremental solution for ℓ customers is at most*

$$4c(A(F, R, \ell)),$$

where $c(A(F, R, \ell))$ denotes the cost of the solution found by Algorithm A.

Theorem 4.2.4 implies that our algorithm yields an 8-competitive incremental solution if we plug in the best known 2-approximation algorithm of Jain et al. [2003] for the Robust Facility Location problem. If we are willing to spend more computational effort and solve the RFLP exactly, we obtain a 4-competitive incremental solution with our framework.

4.3 Lower bound

We describe an instance for which no incremental solution can be better than 3-competitive. More precisely, we give a family of instances with more than 200 facilities which yield a lower bound of at least 2.99. The ratio for this construction does not seem to exceed a value of 3.

Let there be m facilities. Each facility $i = 1, \ldots, m$ has zero-cost-connections to a set R_i of 2^{i-1} customers. The sets R_i are mutually disjoint. Additionally, every facility j is connected to every customer in the set R_i, for all $i \neq j$, with a very high assignment cost $M \gg 0$; note that the connection costs would still induce a metric. We refer to the facility together with its customers of zero service cost as a cluster.

Let $x_1, \ldots, x_m$ be the facility opening costs. Consider the following system of linear inequalities:

$$x_{i+1} + \sum_{j=1}^{i-1} x_j \geqslant \alpha \cdot x_i \qquad \text{for } i = 1, \ldots, m-1.$$

Here, α is the achieved minimal competitive ratio by these inequalities. As an example, for $m = 4$ this system is feasible for $\alpha = 2.246$, but infeasible for $\alpha = 2.247$. By setting the facility costs to a solution of this system, we get a lower bound of 2.246.

The intuitive explanation behind these inequalities is the following. Suppose one looks at a point in any sequence that gives a solution serving 2^{i-1} customers. To keep the competitive ratio of the sequence below α for serving just one customer, one needs to first open facility 1 and serve the single customer in R_1, as cluster 2 is α times as expensive. By proceeding with such an inductive reasoning, the above inequalities give an incentive to open the clusters one by one in the order of their size, as the competitive ratio for the sequence is immediately as bad as α as soon as one skips a cluster. On the other hand, the optimal solution to serve 2^{i-1} customers is to open facility i and serve customers in the set R_i. α should now be chosen carefully, so that the ratio between the solution provided by the sequence in opening all facilities from $1, \ldots, i$ to that of opening facility i alone is at least α for some i, i.e., $\alpha \leqslant \max_{i=1,\ldots,m} \frac{\sum_{j=1}^{i} x_j}{x_i}$. An example for $m = 4$, $\alpha = 2.246$ can be seen in Figure 4.3.1 on the following page. As we increase the value

of m, the value of α increases as well and it numerically converges to 3. For a value of $m = 200$, we get $\alpha \approx 2.99$. Using an optimal solution for these lower bound instances, our incremental algorithm would achieve a competitive factor of 4.

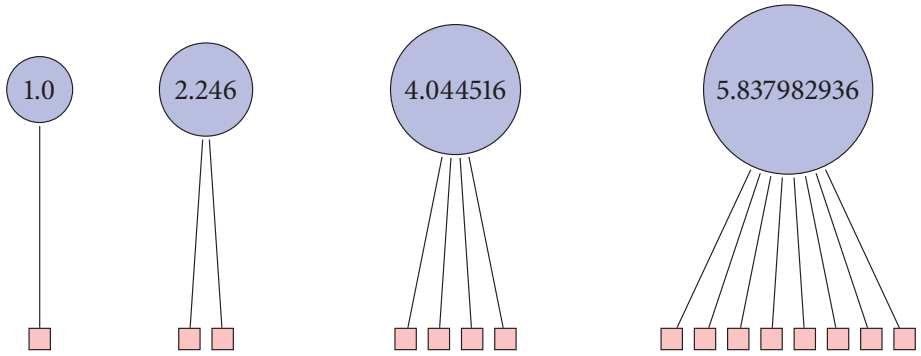


Figure 4.3.1: Lower bound example for $m = 4$, $\alpha = 2.246$. Blue circles depict the facilities, while the red squares are customers. All assignment costs are equal to zero.

4.4 Computational experiments

We tested the quality of the incremental solution produced by our algorithm against the optimal non-incremental solution. We present the maximum and average gap over all $\ell = 1 \dots n$ for every instance. We also report the running time of the algorithms. The experiments were carried out on a i7-4771 machine with 16GB of RAM. We used GUROBI 5.6 to obtain the optimal solution of ℓ-RFLP. The instances were the set of benchmark instances for the uncapacitated facility location problem from the UflLib library provided by Hoefer [2007]. All instances are complete bipartite graphs and in all instances, the number of facilities equals the number of customers.

The results from our computational study are presented in Table 4.1 on page 124. For each instance, we provide the size of the instance and the worst-case gap for some $\ell = 1, \dots, n$ and the average gap over all customers. Note that we are comparing our solution against the optimal solution of serving any ℓ customers and this is not an incremental solution. As pointed out in Section 4.3, we need to pay a price for obtaining an incremental solution. In almost all instances, the worst-case gap is about 50% (corresponding to a factor of 1.5 from the optimal) and the average gap is less than 15% for all instances. We also report the running time (in seconds) of our implementation. All instances took less than an hour.

4.5 Conclusions

In this chapter, we presented an incremental framework to provide a sequence of facilities to be opened and customers to be served along with their fixed assignments that provides a constant-factor competitive guarantee. We proved a lower bound of 2.99 through a family of examples. We gave an implementation of the algorithm and the competitive factor on a set

of benchmark instances was computed and presented. A number of questions are still open: Is the analysis tight for the provided algorithm or can the worst-case guarantee be improved? Can the lower bound be increased? Another interesting direction of future research is to study our incremental setting for other variations of the facility location problem that arise in other practical settings, such as the Buy-At-Bulk Network Design problem and the Connected Facility Location problems.

Instance	# Fac/Cust	Max gap (%)	Ave Gap (%)	Time[sec]
c10-mp1	200	59.50	15.88	286.25
c10-mp2	200	50.65	13.73	315.34
c10-mq1	300	40.06	13.41	2048.95
c10-mq2	300	40.06	11.38	2057.54
c15-mp1	200	59.50	15.88	284.62
c15-mp2	200	50.65	13.73	316.19
c15-mq1	300	40.06	13.41	2076.61
c15-mq2	300	40.06	11.38	2077.86
c20-mp1	200	59.50	15.88	289.52
c20-mp2	200	50.65	13.73	319.90
c20-mq1	300	40.06	13.41	2052.22
c20-mq2	300	40.06	11.38	2061.86
c5-mp1	200	59.50	15.88	287.77
c5-mp2	200	50.65	13.73	316.01
c5-mq1	300	40.06	13.41	2058.01
c5-mq2	300	40.06	11.38	2060.26
d10-mp1	200	59.50	15.88	285.44
d10-mp2	200	50.65	13.73	316.20
d10-mq1	300	40.06	13.41	2059.18
d10-mq2	300	40.06	11.38	2071.02
d15-mp1	200	59.50	15.88	285.26
d15-mp2	200	50.65	13.73	315.42
d15-mq1	300	40.06	13.41	2049.25
d15-mq2	300	40.06	11.38	2058.37
d20-mp1	200	59.50	15.88	287.47
d20-mp2	200	50.65	13.73	317.52
d20-mq1	300	40.06	13.41	2216.26
d20-mq2	300	40.06	11.38	2080.21
d5-mp1	200	59.50	15.88	286.38
d5-mp2	200	50.65	13.73	316.51
d5-mq1	300	40.06	13.41	2054.96
d5-mq2	300	40.06	11.31	2079.71

Table 4.1: Results of computational experiments

5 Incremental Connected Facility Location

5.1 The Incremental Connected UFL problem

In this chapter, we analyze MIP models for the Incremental Connected Facility Location problem. This part of the work is joint work with Ashwin Arulselvan, Andreas Bley, Stefan Gollowitzer and Ivana Ljubić and appeared in Lectures Notes in Computer Science, volume 6701 "Network Optimization" (Arulselvan et al. [2011]).

5.1.1 Introduction

The problem under consideration is how to design an optimal network topology in the context of a multi-period planning of local access networks. In this setting, a telecommunication company wants to increase the speed of broadband connections by combining fiber-optic technology with existing copper connections, i.e., by means of the *Fiber-to-the-Curb* (FTTC) technology. Street segments along which fiber optic cables can be installed, determine the core network. Potential optical and existing copper cables intersect at locations where potential multiplexor devices need to be installed. Between a multiplexor and a customer, the existing copper connection is used. The existing copper paths are preprocessed and in this process, an assignment network is constructed whose edges are assignment links between potential multiplexor locations and customers. To build an FTTC network, one has to decide on which locations to install multiplexor devices so that each customer is assigned to a multiplexor, and each multiplexor is connected to the Central Office (CO) by a fiber-optic path.

Due to the huge investments needed to build an FTTC network, the deployment is done in several stages. The company takes the strategic decision of fixing a minimal percentage of customer demands that should be served at each of the stages. Thereby, demand of a customer is defined as the number of end-subscribers (e.g., offices and/or households) behind the customer's address. The coverage of customer demands needs to be increased over time.

5.1.2 Problem setting

We define the *Incremental Connected Facility Location problem*, denoted as the INCCFL problem, as follows. The input consists of three disjoint sets of nodes: a set of facilities F, a set of customers R, and a set of Steiner nodes M. We denote $S = F \cup M$ and $V = S \cup R$. The potential connections among the nodes in S build the core network and are given as the undirected edge set E_S. The corresponding directed arc set is $A_S = \{(i,j),(j,i) \mid ij \in E_S\}$. Possible connections between the facilities F and the customers R are given by the edges $E_R \subseteq F \times R$, which define the directed arc set $A_R = \{(i,j) \in F \times R \mid ij \in E_R\}$. Note that it is sufficient to consider only arcs directed from facilities to customers here. We let $A = A_S \cup A_R$ and $E = E_S \cup E_R$. The considered planning

horizon is given as a discrete set of (not necessarily equally long) time periods $T = \{1, \dots, \mathcal{T}\}$, $\mathcal{T} > 1$. In addition, we are given fixed costs for edges $c\colon E \to \mathbb{R}_+$ and facilities $g\colon F \to \mathbb{R}_+$ for opening the edge or facility for the first time, and maintenance costs for edges $m\colon E \to \mathbb{R}_+$ and facilities $m_f\colon F \to \mathbb{R}_+$ that arise for each period an edge or a facility is actually used. The pre-period revenue for serving the customers is given by $p\colon R \to \mathbb{R}_+$. Finally, we are given customer demands $d\colon R \to \mathbb{Z}_+$ and a minimum coverage requirement D^t for each time period $t \in T$.

We wish to construct a schedule that describes at each point in time the partial network we have built up to that point. For each time step, it needs to describe which subset of facilities are in use, which set of customers are served by these facilities, how served customers are assigned to the open facilities and how the core network is constructed in order to connect the open facilities to the network. We require that in each time step, the total demand of the served customers satisfies the minimum coverage requirements and the chosen edges E_S in the core network connect the open facilities. Furthermore, a customer must be served in all periods after it has been served for the first time. The goal is to maximize the net present value of the network, which is a standard objective function that takes into account the discounted value of money that will be received in the future.

5.1.3 Related work

Multi-Period optimization problems Facility location problem over time is a well-studied problem. A recent survey is given in Owen and Daskin [1998]. In a recent work, Albareda-Sambola et al. [2009] consider a *Multi-period Incremental Facility Location* problem, where the coverage of customer demand needs to be increased over time. The authors combine subgradient optimization and a Lagrangian approach and generate feasible solutions with a Lagrangian based heuristic.

Multi-period Network Design problems There has been intense research on multi-period network design problems since publication of the seminal articles by Christofides and Brooker [1974], Doulliez and Rao [1975] and Zadeh [1974]. Optimization methods have been used for designing networks for telecommunication, transportation (Ukkusuri and Patil [2009]), distribution of gas or water (Suhl and Hilbert [1998]) and many others.

Most of the literature on applications in the telecommunications sector consider capacitated problems. Recent contributions are, e.g., Bienstock et al. [2006] and Lardeux and Nace [2007]. Much less literature is available on the *Connected Facility Location problem* (CFL).

Single-Period Connected Facility Location Early work on the CFL problem mainly includes approximation algorithms. The problem can be approximated within a constant ratio and the currently best-known approximation ratio is provided by Eisenbrand et al. [2010]. Ljubić [2007] describes a hybrid heuristic combining Variable Neighborhood Search with a reactive tabu search method. The author compares it with an exact Branch&Cut approach. In Tomazic and Ljubić [2008], a Greedy Randomized Adaptive Search Procedure (GRASP) for the unrooted CFL problem is presented. The authors also provide a transformation that enables solving CFL as the Steiner arborescence problem. Bardossy and Raghavan [2010] develop a dual-based local

search (DLS) heuristic for a generalization of the CFL problem. The presented DLS heuristic computes lower and upper bound using a dual-ascent and then improves the solution with a local search procedure. Gollowitzer and Ljubić [2011] study Mixed-Integer Programming (MIP) formulations for CFL both theoretically and computationally. The authors provide a complete hierarchy of ten MIP formulations with respect to the quality of their LP bounds.

Our contributions We consider the Incremental Connected Facility Location problem (INCCFLP), in which we are given a set of potential facilities, a set of interconnection nodes, a set of customers with demands, and a planning horizon. For each time period, we have to select a set of facilities to open, a set of customers to be served, the assignment of these customers to the open facilities, and a network that connects the open facilities. Once a customer is served, it must also be served in subsequent periods. Furthermore, in each time period the total demand of all customers served must be at least equal to a given minimum coverage requirement for that period. The objective is to maximize the net present value of the network, which is given by the discounted revenues of serving the customers and by the discounted investments and maintenance costs for the facilities and the network.

In Section 5.2 we present integer programming formulations for the Incremental CFL problem and discuss a class of valid inequalities that may be used to strengthen these formulations. Section 5.3 provides a description of the separation subroutines that we implemented in order to solve these models. In Section 5.4 we describe the benchmark data sets, details of our implementation of the Branch&Cut algorithm and the results of our computational experiments.

5.2 MIP models

In this subsection we present two alternative integer programming formulations for the Incremental CFL problem.

For modeling purposes, we assume that one of the facilities, denoted as root r, is open and used in all time periods. In a physical network, this node corresponds to the central office with an uplink to the backbone network serving the physical area modeled by this instance.

In order to ensure the connectivity among the open facilities, it is sufficient to ensure that all other open facilities in F are connected to the root r (compare also the approach in Gollowitzer and Ljubić [2011]). For notational simplicity, we let F denote the set of all facilities except r throughout the remainder of this chapter. Furthermore, we denote $\delta^{in}(W) := \{(i,j) \in A \mid i \notin W, j \in W\}$ for all $W \subseteq V$ and $F(j) := \{i \in F \mid (i,j) \in A_R\}$ for all $j \in R$.

In order to describe which customers and facilities are served and used at each time period, we introduce binary variables $y_j^t \in \{0,1\}$ for all $j \in R$ and $t \in T$, and $z_i^t \in \{0,1\}$ for all $i \in F$ and for all $t \in T$. These two variable sets are interpreted as

$$y_j^t = \begin{cases} 1 \text{ if customer } j \text{ is served in time period } t, \\ 0 \text{ otherwise,} \end{cases}$$

$$z_i^t = \begin{cases} 1 \text{ if facility } i \text{ is used in time period } t, \\ 0 \text{ otherwise.} \end{cases}$$

The assignment of the served customers to the open facilities and the network connecting the open facilities to the root node are modeled by the arc variables $x_{ij}^t \in \{0,1\}$ for all directed arcs $(i,j) \in A$ and for all time periods $t \in T$, which are interpreted as

$$x_{ij}^t = \begin{cases} 1 \text{ if arc } (i,j) \text{ is used in time period } t, \\ 0 \text{ otherwise.} \end{cases}$$

To describe the initial opening of facilities and edges, we also introduce the facility variables $\tilde{z}_i^t \in \{0,1\}$ for all $i \in F$ and all $t \in T$ and the aggregated edge variables $\tilde{x}_e^t \in \{0,1\}$ for all $e \in E$ and all $t \in T$, which are interpreted as

$$\tilde{z}_i^t = \begin{cases} 1 \text{ if facility } i \text{ is opened for the first time in time period } t, \\ 0 \text{ otherwise,} \end{cases}$$

$$\tilde{x}_e^t = \begin{cases} 1 \text{ if edge } e \text{ is opened for the first time in time period } t, \\ 0 \text{ otherwise.} \end{cases}$$

Observe that the variables $\tilde{x}_e^t$ are associated to edges instead to arcs of the core network for the following reason. In the general case, a facility $i \in F$ may be opened in period $t \in T$, and closed in period $t + k \in T$ ($k > 0$). Consequently, an arc that was oriented like (i,j) in period t, may be used in the opposite direction in period $t + k$. Since the edge opening costs need to be payed only once, we have to leave the direction of set-up variables $\tilde{x}$ unspecified.

With these variables and notations, the objective function of the incremental CFL problem can be formulated as follows:

$$f(x,y,z) = \sum_{t=1}^{T} (1+\alpha)^{-t} \left[\sum_{j \in R} p_j y_j^t - \sum_{e \in E} c_e \tilde{x}_e^t - \sum_{(i,j) \in A} m_{ij} x_{ij}^t \quad - \sum_{i \in F} g_i \tilde{z}_i^t - \sum_{i \in F} m_i z_i^t \right]$$

This construction using exponentially decreasing factors is called the *net present value*, which takes the time-dependent value of money into account.

For each period $t \in T$, the objective function comprises the collected profit for customers served in period t decreased by the investment (maintenance) costs that need to be paid for each edge and facility that are opened (used) in this period. The following mixed integer programming formulation models the Incremental CFL problem:

The model is presented in Figure 5.2.1 on the following page. We now explain its constraints one by one.

Constraints (5.2.1) model the fact that a customer is served only if there is a facility connected to it. Constraints (5.2.2) enforce that a facility is open if it is used to serve a customer. Inequalities (5.2.3) and (5.2.4) ensure that we open edges and facilities as soon as they are used. Constraint set (5.2.5) expresses the minimum demand coverage requirement for each time period. Inequalities (5.2.6) enforce the continuance of service for each customer (i.e., if customer j was served in period $t \in T$, it also needs to remain served in all consecutive periods).

$$
\begin{aligned}
&\text{(CUTF):} && \max\ f(x,y,z) && && \\
&\text{subject to} && \sum_{i\in F(j)} x_{ij}^t = y_j^t && \forall j\in R,\ t\in T && (5.2.1)\\
& && x_{ij}^t \leqslant z_i^t && \forall (i,j)\in A_R,\ t\in T && (5.2.2)\\
& && x_{ij}^t + x_{ji}^t \leqslant \sum_{k=1}^{t} \tilde{x}_e^k && \forall (i,j)=e\in E,\ t\in T && (5.2.3)\\
& && z_i^t \leqslant \sum_{k=1}^{t} \tilde{z}_i^k && \forall i\in F,\ t\in T && (5.2.4)\\
& && \sum_{j\in R} d_j y_j^t \geqslant D^t && \forall t\in T && (5.2.5)\\
& && y_j^t \geqslant y_j^{t-1} && \forall j\in R,\ t\in T && (5.2.6)\\
& && \sum_{(u,v)\in\delta^{in}(W)} x_{uv}^t \geqslant z_j^t && \forall W\subseteq S\setminus\{r\}, j\in W\cap F\neq\emptyset, t\in T && (5.2.7)\\
& && x_{kl}^t, y_j^t, z_i^t \in \{0,1\} && \forall (k,l)\in A,\ j\in R,\ i\in F,\ t\in T && (5.2.8)\\
& && \tilde{x}_e^t, \tilde{z}_i^t \in \{0,1\} && \forall e\in E,\ i\in F,\ t\in T && (5.2.9)
\end{aligned}
$$

Figure 5.2.1: ILP model for the Incremental CFL problem

Finally, the exponentially large constraint set (5.2.7) ensures that, in each time period, all open facilities are connected to the root node. The inequalities in constraint set (5.2.7) enforce that for every subset $W \subseteq S$ that includes a facility j and does not include the root node r, at least one of the arcs in the set of all incoming arcs in W must be *used* if facility j is open. These inequalities correspond to the directed cutset inequalities in the Steiner tree formulations used in Ljubić [2007] and Gollowitzer and Ljubić [2011].

Instead of enforcing at least one arc in each directed cut that separates a chosen facility from the root node, as done by constraints (5.2.7), we may model the connectivity constraints by enforcing at least one arc in every directed cut that separates a chosen customer from the root node. This leads to the following alternative formulation for the Incremental CFL problem:

$$
\begin{aligned}
&(CUT_R): \quad \max\ f(x,y,z)\\
&\qquad (x,y,z) \text{ satisfies (5.2.1) – (5.2.6)}\\
&\qquad \sum_{(u,v)\in\delta^{in}(W)} x_{uv}^t \geqslant y_j^t \qquad \forall W\subseteq V\setminus\{r\},\ j\in W\cap R,\ t\in T \qquad (5.2.10)
\end{aligned}
$$

5.2.1 Valid inequalities

In this subsection we provide two new families of valid inequalities that can strengthen the previous two models. The third group of constraints presented here are several degree-inequalities that were very useful throughout our computations.

Cover inequalities: The minimum coverage constraints (5.2.5) imply a set of *cover inequalities* that can be defined for each single period $t \in T$. We call a subset of facilities $I^t \subseteq F$ a *cover* if its complement, $\bar{I}^t = F \setminus I^t$, cannot serve enough customers to satisfy the minimal demand requirements for the time period t. We denote by $COV^t \subseteq 2^F$ the family of all covers for period t. An inclusion-wise minimal such facility set I^t is called a *minimal cover*. In other words, I^t is a minimal cover if $\bar{I}^t$ cannot satisfy the minimum demand requirement of period t even if all the facilities in $\bar{I}^t$ are open, but for any $i \in I^t$ the facility set $J = \bar{I}^t \cup \{i\}$ would allow serving enough customers to meet the minimum coverage constraint. In such a case, obviously at least one facility from I^t needs to be opened. Consequently, the following set of *cover inequalities* are valid for all solutions of (CUT_F) and (CUT_R):

$$\sum_{i \in I^t} z_i^t \geqslant 1 \quad \forall t \in T,\ I^t \in COV^t \tag{5.2.11}$$

It is easy to verify that the cover inequality for any non-minimal cover I^t is dominated by the cover inequality for any minimal cover $I^t_{min} \subseteq I^t$. Furthemore, any non-minimal cover I^t can be easily turned into a minimal cover by iteratively removing all those facilities, whose removal still results in a cover.

It is also not difficult to construct examples where the addition of cover inequalities (5.2.11) strengthens the LP relaxations of (CUT_F) and (CUT_R). These inequalities are similar to the cover inequalities studied for knapsack constraints.

We note that one could also consider cover inequalities at the customers, which should be especially effective if the coverage requirement for the current period is very high.

The separation of cover inequalities is a modified version of the knapsack problem. We describe our separation approach for these inequalities in Section 5.3.

Cut-Set-Cover inequalities: The set of cover inequalities (5.2.11) also implies the following exponentially large family of cut-set inequalities, that we will refer to as *cut-set-cover inequalities*:

$$\sum_{uv \in \delta^{in}(W)} x_{uv}^t \geqslant 1 \quad \forall t \in T,\ I^t \in COV^t,\ W \subseteq S \setminus \{r\}, I^t \subseteq W \tag{5.2.12}$$

These inequalities state that, in each period $t \in T$, we have to establish a path between the root and at least one of the facilities from the set I^t. Once the corresponding covers I^t become known, the separation of these new inequalities can be done in polynomial time by means of a maximum flow algorithm, see Section 5.3.

Again, it is not difficult to show that the addition of the cut-set-cover inequalities (5.2.12) strengthens the LP relaxations of (CUT_F) and (CUT_R).

In-Arc inequalities: The requirement that, in each time period, the root node is connected to any open facility, implies the following *in-arc inequalities*:

$$z_i^t \leqslant \sum_{(j,i) \in \delta^{in}(i)} x_{ji}^t \qquad \forall i \in F,\ t \in T \tag{5.2.13}$$

$$x_{ik}^t \leqslant \sum_{(j,i) \in \delta^{in}(i) : j \neq k} x_{ji}^t \qquad \forall (i,k) \in A_S,\ i \neq r,\ t \in T \tag{5.2.14}$$

Inequalities (5.2.13) imply that there is at least one arc entering any chosen facility from the time it is opened. Inequalities (5.2.14) ensure that there is at least one arc entering any facility or Steiner node if there is an arc leaving that node.

Note that these inequalities are implied by the cut inequalities (5.2.7) or (5.2.10), but not vice versa. However, there is only a polynomial number of inequalities of type (5.2.13) and (5.2.14), so they can easily be added at the beginning of the computations without any separation algorithm. This fact makes these inequalities very useful in practical computations (Gollowitzer and Ljubić [2011]; Koch and Martin [1998]).

Furthermore, we add the inequalities

$$\sum_{(j,i)\in\delta^{in}(i)} x_{ji}^t \leqslant 1 \qquad \forall i \neq r,\ t \in T \tag{5.2.15}$$

to the LP relaxations of (CUT_F) and (CUT_R), which ensure that the indegree of every node except the root node is at most 1. While these constraints may cut off feasible solutions, there always exists an optimal solution of the Incremental CFL problem that satisfies these inequalities, as there are no capacity constraints associated with the facilities and edges. Adding these inequalities to the formulations substantially reduced the solution times in our experiments.

5.3 Separation Algorithms

In this section, we explain separation algorithms for the cover inequalities and the three groups of cut-set inequalities described above.

Separation of Cut-Set inequalities

We now present the separation routine to generate cut inequalities of type (5.2.7) which is similar to the one used in Chapter 2, see Lemma 2.2.1 there. Let $\hat{x}^t$ and $\hat{z}^t$ be the values of the arc variables and of the facility variables of the current optimal LP solution. In order to find a violated inequality of type (5.2.7), we compute for each time period $t \in T$ and each facility node $j \in F$ a minimum r-j-cut in the digraph $G(S, A_S)$ with arc capacities $\hat{x}^t$, solving the corresponding maximum flow problem. Let $\Gamma(r, j)$ be the set of arcs in the minimum cut obtained from this maximum flow computation. If the corresponding maximum flow value is less than $\hat{z}_i^t$, the corresponding cut inequality

$$\sum_{(u,v)\in\Gamma(r,j)} x_{uv}^t \geqslant z_j^t \tag{5.3.1}$$

is violated and we add this inequality to the current formulation.

The separation of the customer based cutset inequalities (5.2.10) is carried out analogously. We now consider the entire digraph $G(V, A)$ with capacities $\hat{x}^t$ given by the LP solution's arc variable values and solve the maximum flow problem with the root node r as the source and the customer node j as the sink for each customer $j \in R$ and each time period. Again, let $\Gamma(r, j)$ be

the arcs of the corresponding minimum cut. If the maximum flow value is less than y_i^t, we add the violated cut

$$\sum_{(u,v)\in\Gamma(r,j)} x_{uv}^t \geqslant y_i^t \,. \tag{5.3.2}$$

Separation of Cover and Cut-Set-Cover inequalities

Let $t \in T$ and let $\hat{z}^t$ be the values of the facility variables in the current LP solution. In order to find a cover I^t for which the corresponding cover inequality (5.2.11) is violated, we introduce variables $\alpha_i \in \{0,1\}$ for all $i \in F$ indicating which facilities are contained in I^t and $\beta_j \in \{0,1\}$ for all $j \in R$ indicating which customers can be served by any of the facilities not in I^t. Clearly, a cover I^t that maximizes the violation of inequality (5.2.11) corresponds to an optimal solution of the following integer program:

$$\min \sum_{i\in F} \hat{z}_i^t \alpha_i \tag{5.3.3}$$

$$\text{subject to} \quad \sum_{j\in R} d_j \beta_j \leqslant D^t - \varepsilon \tag{5.3.4}$$

$$\beta_j \geqslant 1 - \alpha_i \qquad \forall (i,j) \in A_R \tag{5.3.5}$$

$$\alpha_i, \beta_j \in \{0,1\} \qquad \forall i \in F, j \in R \tag{5.3.6}$$

Inequalities (5.3.5) guarantee that all clients that have at least one neighboring facility not in I^t are served, while constraint (5.3.4) ensures that the total demand of all served clients is strictly less than the demand required to meet the coverage constraint. Together, these constraints ensure that, for any integer solution of (5.3.3) - (5.3.6), the set of facilities i with $\alpha_i = 1$ forms a cover. Note that the objective value of a solution of (5.3.3) - (5.3.6) is equal to the left hand side of the corresponding cover inequality for the current LP solution. Finding a violated cover inequality thus is equivalent to finding a time period $t \in T$ and a solution of (5.3.3) - (5.3.6) with objective value strictly less than 1. In our implementation, we solve this integer program for all $t \in T$.

To separate the cut-set-cover inequalities for a given cover I^t, we create an artificial sink node l and connect the nodes in I^t to l. We then compute a maximum r-l flow in the graph $G(S \cup \{l\}, A_S \cup I^t \times \{l\})$ with capacities $\hat{x}^t$ for the arcs in A_S and capacity 1 for the artificial arcs in $I^t \times \{l\}$. If the maximum flow value is less than 1, we add the violated cut-set-cover inequality

$$\sum_{(u,v)\in\Gamma(r,l)} x_{uv}^t \geqslant 1 \tag{5.3.7}$$

where $\Gamma(r,l)$ is the arc set of a corresponding minimum cut.

5.4 Experiments

Benchmark Instances In Gollowitzer and Ljubić [2011], a set of instances for Connected Facility Location was generated by combining a set of benchmark instances for the Uncapacitated

Facility location (UFL) problem from the UflLib Hoefer [2007] with instances of the Steiner tree problem (STP) from the OR-library Beasley [2013]. The CFL input graphs are generated in the following way: first f nodes of the STP instance are selected as potential facility locations (where f denotes the number of facilities in the corresponding UFL instance), and the node with index 1 is selected as the root. The number of facilities, the number of customers, opening costs and assignment costs are provided in UFL files. STP files provide edge-costs and additional Steiner nodes.

We consider a set of 32 instances obtained by combining four UFL instances `mp1,mp2` and `mq1,mq2` (of the size 200×200 and 300×300, respectively) with eight STP instances `{c,d}n`, for $n \in \{5, 10, 15, 20\}$. These instances define the core networks with between 500 and 1000 nodes and with up to 25,000 edges.

We extend these instances to include demands and time periods. We generate demands uniformly between 20 and 40 for each customer and we consider a time horizon $\mathcal{T} = 5$. In the test instances generated in Gollowitzer and Ljubić [2011], the facility set F and customers R induce a complete bipartite graph. We desire a more sparse setting for our demand satisfaction and the cover set inequalities. Therefore, we only considered the connections of the first 20 closest facilities for each customer. Such obtained instances contain up to 1300 nodes and 45,000 edges. Finally, the minimum coverage required for time period t is defined as

$$D^t = \frac{\sum_{j \in R} d_j}{1.25(T - t)} \text{ for } t \in \{0, 1, 2, 3, 4\} \text{ and } T = 5.$$

The experiments were performed on an Intel Core2 Quad 2.66 Ghz systems with 2GB RAM. Each run was carried out on a single processor.

Branch&Cut Implementation

To test the effectiveness of the presented formulations and inequalities, we implemented a Branch&Cut algorithm using CPLEX 12.2 and Python API, a commercial integer programming solver with a Branch&Cut framework.

The integer linear programs initially contain all variables and the constraints (5.2.1) – (5.2.6). The cut inequalities (5.2.7) and (5.2.10), the cover inequalities (5.2.11), and the cut-set-cover inequalities (5.2.12) are applied in a standard cutting plane approach, iteratively adding those inequalities that are violated by the current fractional solution.

We add all indegree constraints (5.2.15) to the initial LP formulation. We generate a cut pool with all the in-arc inequalities (5.2.13) and (5.2.14), which are added at the root node if they are violated. Then, the maximum flow separation routine is called that generates the inequalities (5.2.7). This separation consists of randomly selecting 50 terminals at every time period and generating the violated cuts. We restrict the number of calls to the separation routine at every node by 10, to enable branching and avoid multiple calls to the separation routines. In addition to the above, the separation routine is called at node depth of multiples of 10 and at every occasion an incumbent is rejected. The intuition behind this scheme is that it would provide us with a balance between the time spent in generating the cuts and branching, as branching helps us reduce the search space (due to the priority strategies described below). The enhanced cuts and customer cuts are combined in the same separation routine. Each test run

was limited to 2000 CPU seconds and the optimality gap at this point of time is reported in the results.

Branching: The assignment variables x_{ij}^t, when branched (set to 0 or 1), does not affect the search space as much as the facility variables z_i^t. So we give them the highest priority in the branching. This was also observed in Ljubić [2007], but unlike the Connected Facility Location problem, in our incremental version of the problem, we also have uncertainty in determining the set of customers to be served at each time period. So, we provide them with the next highest priority in branching.

Separation routine: We observed that the cuts generated by the maximum flow algorithm when the root is treated as source tend to generate cuts that are closer to the root node and there will be edges repeated in the various minimum cuts generated for various terminals. In order to avoid this, we treat the root as the sink and the facilities as the source. This was appropriately captured in the primal heuristic and the in-arc inequalities as well. We also perform nested cuts, wherein we resolve maximum flow for the same facility by setting the capacity of the edges in current minimum cutset to 1. The cover (5.2.11) and cut-set-cover inequalities (5.2.12) rely on solving an integer program at every call of the separation routine, which is run for every time period. The integer program terminates if the elapsed running time is over 100 seconds or if the objective value drops below 1. We use this exact separation to test the impact of these inequalities on the lower bound.

Primal Heuristics: We also implemented and tested a naive primal heuristic. After our initial runs we decided to turn off the CPLEX heuristics as this was leading to poor performance. The primal heuristic rounds up all the z variables that indicate the usage of a facility as well as the y variables, which indicate the service to a customer. We run a minimum cost flow algorithm with a linear cost estimator with the open facilities (rounded up values) as sinks and the root node as source to generate our Steiner tree.

Results

Our preliminary computational study has shown that the CUT_R formulation is not competitive against the CUT_F model, due to the size of the support graph and the large number of cut-set inequalities that need to be separated. This is also consistent with the results obtained by Gollowitzer and Ljubić [2011] for the single-period CFL.

Therefore, in our computational study, we compared the performance of the following two Branch&Cut settings:

- CUT_F formulation,
- CUT_F+ formulation extended by cover inequalities (5.2.11) and cut-set-cover inequalities (5.2.12).

For each of the two settings, we report on the following values given in Table 5.1 on the following page: the overall percentage gap obtained after the time limit of 2000 seconds calculated

as Gap $= (UB - LB)/LB$, where UB is the best obtained upper bound, and LB is the global lower bound; the number of all constraints separated throughout the execution of the algorithm, denoted by "Cuts"; the number of Branch&Bound nodes, denoted by "B&B".

		CUT_F				CUT_F + (5.2.11) + (5.2.12)			
Instance	best LB	best UB	Gap[%]	cuts	B&B	best UB	Gap[%]	cuts	B&B
c10-mp1	164,136	166,691	1.53	2103	170	166,670	1.52	695	104
c10-mp2	160,278	168,347	4.79	1781	99	168,258	4.74	297	20
c10-mq1	346,866	386,409	10.23	457	11	386,054	10.15	161	0
c10-mq2	348,929	387,501	9.95	559	14	387,088	9.86	136	0
c15-mp1	165,004	166,900	1.14	1539	107	166,985	1.19	212	15
c15-mp2	161,333	168,487	4.25	1508	56	168,525	4.27	299	16
c15-mq1	352,583	386,520	8.78	567	15	386,326	8.73	140	0
c15-mq2	348,640	387,614	10.05	422	10	387,452	10.02	138	0
c20-mp1	155,919	167,227	6.76	334	6	167,184	6.74	121	0
c20-mp2	154,157	168,658	8.60	360	3	168,656	8.60	137	0
c20-mq1	349,075	386,640	9.72	298	0	386,540	9.69	55	0
c20-mq2	348,628	387,792	10.10	311	0	387,681	10.07	60	0
c5-mp1	162,521	166,466	2.37	1927	190	166,320	2.28	590	75
c5-mp2	158,230	168,042	5.84	1630	45	167,892	5.76	347	19
c5-mq1	346,924	386,236	10.18	869	15	385,491	10.00	133	0
c5-mq2	348,453	387,330	10.04	817	15	386,744	9.90	143	0
d10-mp1	164,160	166,945	1.67	2008	45	166,706	1.53	126	10
d10-mp2	155,885	168,381	7.42	1902	21	168,167	7.30	291	15
d10-mq1	342,278	386,234	11.38	508	6	385,844	11.29	121	0
d10-mq2	348,389	387,584	10.11	642	9	387,062	9.99	110	0
d15-mp1	158,402	167,103	5.21	813	15	167,005	5.15	176	6
d15-mp2	158,835	168,398	5.68	1103	15	168,468	5.72	193	1
d15-mq1	346,494	386,579	10.37	407	0	386,258	10.29	95	0
d15-mq2	346,129	387,683	10.72	415	6	387,458	10.67	118	0
d20-mp1	155,821	167,168	6.79	256	0	167,208	6.81	139	0
d20-mp2	154,141	168,661	8.61	291	0	168,675	8.62	175	0
d20-mq1	348,738	386,621	9.80	180	0	386,580	9.79	62	0
d20-mq2	348,365	387,801	10.17	135	0	387,772	10.16	16	0
d5-mp1	163,074	166,680	2.16	2666	50	166,216	1.89	1387	225
d5-mp2	163,182	167,967	2.85	2495	104	167,634	2.66	1172	165
d5-mq1	346,120	386,141	10.36	1230	15	385,396	10.19	185	0
d5-mp1	344,089	387,304	11.16	1541	15	386,517	10.98	166	0

Table 5.1: Comparison of two Branch&Cut settings: plain CUT_F model vs. CUT_F extended by cover and cut-set-cover inequalities.

Comparing the number of inserted cuts by the two approaches, we observe that the inclusion of coverage-related cuts (i.e., (5.2.11) and (5.2.12)) reduces the overall number of cuts generated within a given time limit. This can easily be explained by the large separation times needed to solve the integer program (5.3.3)-(5.3.6). Despite the reduced number of separated inequalities, in 27 out of 32 instances we obtained reduced duality gaps when the coverage-related inequalities

were used. This indicates the strength of the coverage-related cuts, but also the trade-off between their strength and their separation time.

We also observe that due to the branching and separation strategies that we choose, there is no direct correlation between the usage of coverage-related constraints and the number of branch and bound nodes.

5.5 Conclusions

In this chapter we introduced a combinatorial optimization problem that models the design of fiber-to-the-curb networks over time. The problem is a multi-period version of the connected facility location problem that has been intensively studied in the literature in the last decade. Besides two mixed integer programming models, we also introduced two new families of valid inequalities derived from the incremental coverage constraints over time. We provided separation algorithms needed to detect the new coverage-related inequalities within a cutting plane framework. The problem was then solved by means of a Branch&Cut algorithm that makes use of the cut-set inequalities and the new coverage-related constraints. In the preliminary computational study we show that the new inequalities are useful for small and/or sparser instances, where the obtained duality gaps can be significantly reduced. For larger instances, it turns out that the there is a trade-off between the separation time of the coverage-related family of inequalities and the obtained improvement of the quality of lower bounds.

One of the problems addressed by our computational results is the computational inefficiency of the integer program needed to separate the coverage-related inequalities. To overcome this problem, one should aim to develop more efficient exact or heuristic approaches for the separation. Finally, it would be also interesting to compare decomposition based approaches (e.g. Lagrangian or Benders decomposition) with the proposed Branch&Cut framework.

6 Frequency Assignment in Optical Networks

6.1 Spectrum Assignment problem

In this chapter, we consider MIP models for the Spectrum Assignment problem. The research presented is joint work with Andreas Bley, Benjamin Müller and Mohsen Rezapour.

6.1.1 Introduction

One of the challenging optimization problems in optical telecommunication networks is the *Routing and Wavelength Assignment problem* (RWA) which is the following. We are given a graph $G = (V, E)$ and a set of endpoints that need to communicate with each other through paths in the graph G. With respect to a total number of available wavelengths, we must determine the paths through which the connections should be set up and determine the frequency slots that should be assigned to these paths to maximize the number of connections that can be established.

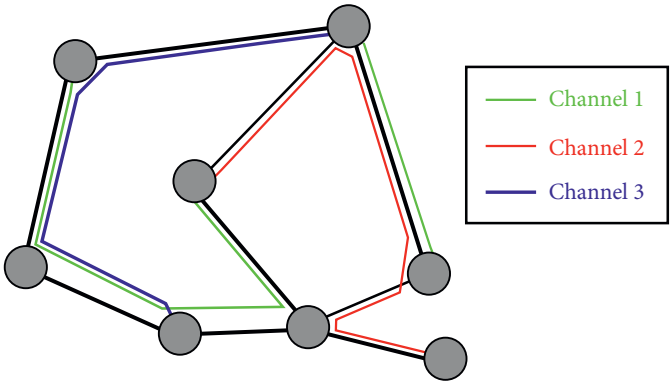


Figure 6.1.1: Lightpaths in a network. Different colors correspond to different paths, which establish connections between their endpoints using the same frequencies on all their edges.

While a simple heuristic could route the traffic on shortest paths between the endpoints, it may be advisable to also allow longer paths to avoid wavelength conflicts. There is a tradeoff between the number of allowed paths and thus the flexibility in the wavelength assignment part and the increasing complexity of the problem incurred by considering multiple paths. A standard multicommodity flow formulation for the RWA problem can be found in Ramaswami and Sivarajan [1995]. An exemplary solution of the RWA problem can be seen in Figure 6.1.1.

An important assumption in the RWA problem is that the frequency slots we assign correspond to intervals of some fixed size in the spectrum. For technical reasons, these fixed-size intervals include guard bands at both ends of the interval, which are used to ensure non-interlapping frequency ranges between different lightpaths. Networks employing this kind of allocation are called *Fixed grid networks*. In these networks, one must assign several frequency slots to paths whose demands exceed the provided capacity by just one slot. In that case, a path uses several guard bands, leading to a waste of usable resources; this problem is avoided in a more variable *Flexgrid architecture* (see for example Gerstel et al. [2012]).

In this chapter, we consider the *Routing and Spectrum Assignment (RSA)* problem, which arises in the planning of flexgrid Wavelength-Division Multiplexing (WDM) networks and is an extension of the Routing and Wavelength Assignment problem. In contrast to the RWA problem, the RSA problem allows using more flexible wavelength assignments: we allow the assignment of intervals of varying size, depending on the amount of traffic demand for the path. Instead of only assigning one fixed-size interval, in the RSA we allow assigning a contiguous interval in the frequency spectrum of variable size to a path.

We do not consider the full RSA problem, but focus on the subproblem of *Spectrum Assignment*, when the paths are already fixed. The solution methods for this subproblem could then be used in a decomposition approach for the full RSA problem.

Given a set $\mathcal{P}$ of paths in a network, with a specified spectral demand for each of them, our task therefore is to assign to each of them a spectral frequency interval of sufficient size to serve the demands. Also, paths sharing edges require disjoint frequency intervals.

6.1.2 Problem setting

We now give a formal definition of the Spectrum Assignment problem. We are given a simple graph $G = (V, E)$. The frequency spectrum is discretized into finitely many slots $[T] = \{1, ..., T\}$. Furthermore, we have a set of fixed paths $\mathcal{P} = \{P_1, ..., P_\ell\}$ with $P_i \subseteq E \; \forall i = 1, \ldots, \ell$. A spectral demand $d_i \in \mathbb{N}$ is associated to each path $P_i \in \mathcal{P}$. We can assume without generality that there are no paths with zero demand.

The task is to assign to each path P_i a frequency interval $[s_i, s_i + d_i - 1] \subseteq [T], s_i \in \mathbb{N}$ such that each pair of paths with shared edges have disjoint intervals, that is we have

$$[s_i, s_i + d_i - 1] \cap [s_j, s_j + d_j - 1] = \emptyset \quad \forall i, j, i \neq j \text{ such that } P_i \cap P_j \neq \emptyset.$$

We assume that the frequency slots are ordered in such a way that slot 1 corresponds to the lowest, slot T to the highest frequency slot. We consider the objective of minimizing the slot with the highest frequency that is used by any path. This value will be referred to as the *makespan* of a solution. In a sense, we treat the problem as a feasibility problem and try to decide for which values of T the problem has indeed a feasible solution. Our solution method can then be used in a *Parametric Pruning* approach to minimize T, for example by a binary search.

6.1.3 Computational complexity

The Spectrum Assignment problem is **NP**-hard. This can be seen in the following way. Given an instance of the problem, we can consider the corresponding *conflict graph*. In this graph,

the nodes represent the lightpaths in the original network. Two nodes in the conflict graph are connected by an edge if and only if the corresponding paths in the fiber network share an edge.

If we assume unit demands for all paths, i.e. $d_i = 1$ for all $i = 1, \ldots, \ell$, each path only gets exactly one color and therefore, the Spectrum Assignment problem on the original graph is then equivalent to the graph coloring problem on the conflict graph. As any simple graph can be realized as the conflict graph of some instance of the Spectrum Assignment problem (Boyacı et al. [2016]), this implies **NP**-hardness even for the unit demand case.

If we relax the requirement that the set of slots assigned to each path has to be a contiguous interval, the Spectrum Assignment problem reduces to solving the multicoloring problem on the conflict graph, a fact which is exploited later in a simple ILP lower bounding model.

6.1.4 Related work

Graph Coloring problem The graph coloring problem and its variants have been studied widely in the literature. In particular, several IP based approaches have been proposed for these problems so far. One of the first IP formulations for the coloring problem was proposed in Aardal et al. [1996] where the authors presented a compact formulation with variables corresponding to the assignment of different colors to the nodes. Since then this formulation has been strengthened by a series of papers (e.g. Coll et al. [2002]; Méndez-Díaz and Zabala [2006]; Lee and Margot [2007]; Méndez-Díaz and Zabala [2008]) by adding strong valid inequalities in order to eliminate the symmetry of the formulation and improve its integrality gap. In fact, the symmetry involved in these models is the main obstacle for IP solvers (Méndez-Díaz and Zabala [2006, 2008]).

A less symmetric formulation for the coloring problem was considered by Mehrotra and Trick [1996]. The authors introduced an exponential-sized IP formulation for the problem, using one variable associated with each maximal *stable set* of the graph. To deal with the exponential number of variables, they developed a Branch&Price algorithm based on the stable set formulation. In Mehrotra and Trick [2007] they show that their approach can be extended to the multicoloring problem. It is worth noting that a different formulation for the coloring problem that is also based on stable sets, called the *set packing* formulation, is considered by Hansen et al. [2009]. In this formulation, the condition that two stable sets cannot have a vertex in common is expressed as a constraint in the model. They showed that this formulation is equivalent to the stable set formulation in terms of the LP relaxation lower bound and that both formulations are equally efficient when used in a Branch&Price algorithm.

Other related problems For the bandwidth allocation problem in point-to-multipoint wireless systems similar polytopes arise as in our problem; facets of these polytopes have been analyzed in a series of papers (Marenco and Wagler [2006], Marenco and Wagler [2009a] and Marenco and Wagler [2009b]). Heuristics for the RSA problem have been proposed in Klinkowski and Careglio [2011] and Olszewski [2014]. For the spectrum assignment problem on a single path, a heuristic based on finding a good processing order for the classical *First-Fit* algorithm (Johnson [1974]) has been proposed in Peters [2015].

Dynamic Storage Allocation problem Furthermore, the Spectrum Assignment problem on a path is equivalent to the *Dynamic Storage Allocation problem* that has a long-standing line

of research; we only mention a few selected results. The problem can succinctly be described as the problem of "packing given axis-aligned rectangles into a horizontal strip of minimum height by sliding the rectangles vertically but not horizontally" (Buchsbaum et al. [2004]). Of course, the rectangles in the described strip may not overlap.

The sum over the heights of all rectangles at a fixed x-coordinate is called the *maximum load* of the given instance. Polynomial algorithms that find solutions which are not more than a multiplicative constant away from the maximum load were given by Kierstead [1988] (factor 80), Kierstead [1991] (factor 6), Gergov [1996] (factor 5) and Gergov [1999] (factor 3). The first constant-factor approximation algorithms for the problem with approximation guarantees of $\mathcal{O}(2+\varepsilon)$ for any $\varepsilon > 0$ were given by Buchsbaum et al. [2004]. Newer research mainly focuses on generalizations or modifications and on the online variant of the problem.

Our contributions

In this chapter, we devise a three-stage exact algorithm to solve the Spectrum Assignment problem. In the first stage presented in Section 6.2, we use a lower bound and a very fast combinatorial heuristic based on so-called *critical paths*. The *clique bound* is already tight for most instances we have and corresponding solutions can be found for most instances using this simple heuristic.

If the first stage was unsuccessful in solving the problem optimally, we now use an IP formulation of the multi-coloring relaxation of the problem. This stage is presented in Section 6.3. This model is still a simplification of the full problem and we aim to provide promising lower and upper bounds using the information provided by the relaxed model. We employ several heuristics to obtain good feasible solutions and also exploit the easy solvability of the relaxed model to improve the lower bounds.

If the second stage again fails to prove optimality, we then run our devised Branch&Price algorithm exploiting the information obtained in the first two stages. This stage is described in Subsection 6.4.4. We also compare the Branch&Price-approach with several compact formulations.

In Section 6.5, we provide the results of our computational experiments, showing that the described three-stage approach is computationally efficient and we often manage to avoid entering the third phase.

6.2 Phase I: Articulation Point Heuristic

Definition 6.2.1 (Conflict Graph)**.** Given a set of routing paths $\mathcal{P}$, we define $G^{\text{conf}}(\mathcal{P}) := (V, E)$ to be the *conflict graph* of the set of paths, where the nodes $v \in V$ represent the paths, that is, $V = \mathcal{P}$, and each edge $e = (i, j) \in E$ between two nodes of the graph represents a conflict between the two corresponding paths, i.e. $e = (i, j) \in E \Leftrightarrow P_i \cap P_j \neq \emptyset$. We usually drop the argument (P) from the notation if the context is clear.

We assume that the nodes of the conflict graph G^{conf} are ordered, that is, $V = \{v_1, \ldots, v_n\}$. By a *reordering*, we mean a bijective function $\sigma\colon [n] \to [n]$.

We first present a very important combinatorial lower bound.

6.2.1 Lower bounds

First, we define what we mean by a *clique*:

Definition 6.2.2. A subgraph $C = (W, F)$ of a graph $G = (V, E)$ is called a *clique* if

$$\forall w_1, w_2 \in W, w_1 \neq w_2 \Longrightarrow \{w_1, w_2\} \in F.$$

In other words, every two nodes in a clique are connected by an edge. In the conflict graph, this means that we cannot assign the same frequency slots to any of the elements of a clique. From this, a simple combinatorial lower bound, called the *clique bound*, follows:

Definition 6.2.3. Let C be the maximum node-weighted clique in G^{conf} with respect to the demands d. We define the clique bound as

$$T_{clique} := \sum_{i \in C} d_i.$$

Due the previous considerations, the clique bound is a valid lower bound for any feasible solution of the Spectrum Assignment problem.

Remark 6.2.4. T_{clique} is a lower bound, that is $T_{clique} \leqslant T_{opt}$, where T_{opt} denotes the frequency range of an optimal solution.

Proof. Let $O = (s_1, ..., s_\ell)$ be the frequency assignment corresponding to the optimal range T_{opt}. Since each $i \neq j \in C$ are in a conflict, we have

$$T_{opt} \geqslant \max_{i \in C}\{s_i + d_i\} \geqslant \max_{i \in C}\{s_i + d_i\} - \min_{i \in C}\{s_i\} \geqslant \sum_{i \in C} d_i = T_{clique}.$$

□

6.2.2 Heuristic algorithm

One basic idea for creating a feasible assignment of the paths to frequency slots is to use a *greedy* approach. We consider some ordering of the paths and try to assign them in this order to the lowest frequency interval that does not cause a conflict. We employ different initial job orders in the heuristic, including ordering jobs nondecreasingly or nonincreasingly by size or numbers of conflicts and randomly generated orders. A formal description of the greedy approach, based on the classical First-Fit algorithm for the Bin Packing problem, is given in Algorithm 5 on the following page.

This approach is also a standard approach for scheduling problems called *List-scheduling*. For scheduling problems with conflicts, it has been introduced and analyzed by Garey and Graham [1975]. It has also been similarly employed for other wavelength assignment problems, see for example Koster and Zymolka [2003].

We remind the reader that we use the notation Γ_i^* for all neighbors of the node i in a graph $G = (V, E)$. Also, we set $\Gamma_i := \Gamma_i^* \cup \{i\}$.

Algorithm 5 First-Fit Assignment

Input: conflict graph $G^{conf} = (V, E), V = \{v_1, \ldots, v_\ell\}$; demands $d_i \in \mathbb{N}, i = 1, \ldots, \ell$
Output: starting frequency slots $= (s_1, ..., s_\ell)$ for paths $P_i, i = 1, \ldots, \ell$;

```
1: for i := 1, ..., ℓ do
2:     s_i := 0
3:     for all j = 1, ..., i − 1 with j ∈ Γ_i^* do
4:         if [s_i, s_i + d_i] intersects ]s_j, s_j + d_j] then
5:             s_i := s_j + d_j
6:             continue
7:         end if
8:     end for
9: end for
10: return (s_1, ..., s_ℓ);
```

After determining an initial assignment using First-Fit, we compare the best found solution to the clique bound, as introduced in Subsection 6.2.1, and terminate execution if they already match.

Given an assignment created by the First-Fit algorithm, no job can be assigned to a lower frequency slot interval without creating a conflict. Due to this fact, we know that a *critical path* in the assignment has to exist.

Definition 6.2.5. Let $A = (s_1, ..., s_\ell)$ be a feasible assignment, $[0, T_A]$ be the frequency range used by A, and $P = (i_1, ..., i_m)$ be an ordered subset of nodes of the conflict graph G^{conf}, or in other words, an ordered subset of the routing paths $\mathcal{P}$. Let path i_k be in conflict with path i_{k+1} for all $k = 1, \ldots, m-1$. We call the path set P a *critical path set* if

- $s_{i_1} = 0$,
- $s_{i_1} + d_{i_1} = s_{i_2}, ..., s_{i_{k-1}} + d_{i_{k-1}} = s_{i_k}$,
- $s_{i_m} + d_{i_m} = T_A$.

From the definition, it is immediately clear that without changing the assignment of at least one path in the critical path set, all attempts to reduce the makespan of the solution must fail.

Given a solution, there might actually be several critical paths. Obviously, we need to change at least one assignment in every one of them to be able to decrease the solution value.

To do this efficiently, we compute the *critical network* for the assignment. This network has a node for each path that is in a critical path subset. Every node in the critical network has edges to all conflicting paths that end directly before it or start directly after it.

This critical network could be created as a graph with as many nodes as we have jobs by inserting an edge between two nodes whenever two jobs that are in conflict are scheduled directly after one another. This approach has the disadvantage of putting edges in the graph that are on no critical path. One way to avoid this problem is to run a Breadth-First Search (BFS) as a second step, beginning at an artificial *sink node* and ending at an artificial *source node*, while not

exploring from the artifical source node. All edges that are never in the seen in the breadth-first search can then be removed, leaving only the critical network.

The alternative approach we choose is to generate the network from back to front, in a sense creating the network by the breadth-first search described above.

As we have observed earlier, any modification of the solution that can improve the makespan has to interfere with all critical paths. We now describe how we can find such a modification. This is based on the following idea: In order to destroy all critical paths, we would like to modify some assignments such that none of the critical paths still exist. What we are looking for is set of nodes in the critical network whose removal would disconnect the artificial sink and source from one another. Denoting the artificial source and sink by s and t, such a set is called *s-t-node cut*. The most simple form of node cuts are *articulation nodes* of the network, if they exist. Therefore, by changing the assignment of the articulation nodes, we are able to interrupt all critical paths of the assignment.

A simple way to achieve this is to use the classical algorithm due to Hopcroft and Tarjan [1973] which finds all articulation nodes in $\mathcal{O}(|E|)$ using a special variant of depth-first search (DFS) and which we will describe next.

6.2.3 Hopcroft-Tarjan algorithm

We now describe the basic ideas underlying the algorithm, see also Hopcroft and Tarjan [1973]. For the description of the algorithm, we follow Cormen et al. [2000].

The setting is as follows. We have a connected, undirected graph $G = (V, E)$ and want to find its articulation points.

The algorithm is based on depth-first search. Let $r \in V$ be an arbitrarily chosen root of the graph G and let T be a tree resulting from a depth-first search starting in r.

Considering the root of the DFS tree, we know that r is an articulation point of G if and only if r has at least two children in T. This is due to the fact that we do a DFS, so another edge adjacent to r is used only if a whole component of $G - v$ has been completely explored.

This idea could already be used to find the articulation points in quadratic time by running a DFS on every node of the graph and checking each root. But there is another useful property that can be used to speed up this method: We can also find a necessary and sufficient criterion for a nonroot node of T to be an articulation point.

In a standard depth-first search, the distance inside the tree from each node to the root v is usually maintained. This is also called the *depth* of a node v. We can also maintain another useful information during the DFS, the so-called *lowpoint* of a node v. The node w is called a *descendant* of v in T if v lies on the unique path in T from r to w. The node $v \in V$ is called a *proper ancestor* of w if it is a descendant of v and $v \neq w$. Looking at all descendants of a node v in T, we define its lowpoint $\ell(v)$ as the minimum depth of all descendants of v. During a DFS, this information is quite easy to maintain—for any node, it can be computed at the moment when it is removed from the stack in the DFS, as we have considered all its descendants by then. The lowpoint is then the minimum of the depth of v, the lowpoints of all its children and the depth of all neighbors of v, excluding its parent.

If for a node v, we then have a child s of v with the property that $\ell(s) \geqslant \mathtt{depth}(v)$, we have found an articulation point. This is based on the following observation:

Lemma 6.2.6. *Let $v \in V, v \neq r$. Then v is an articulation point of G if and only if v has a child s in T such that there is no edge from s or any descendant of s to a proper ancestor of v.*

Using the articulation points of the critical network

Algorithm 6 Articulation Point Heuristic

Input: conflict graph $G^{conf} = (V, E), V = \{v_1, \ldots, v_\ell\}$; demands $d_i \in \mathbb{N}, i = 1, \ldots, \ell$, $K \in \mathbb{N}$
Output: starting frequency slots $= (s_1, ..., s_\ell)$ for paths $P_i, i = 1, \ldots, \ell$

```
(s_1, ..., s_ℓ) := First-Fit(G^conf, (d_1, ..., d_ℓ))
for i := 1, ..., K do
    G^crit := critical-network((s_1, ..., s_ℓ), (d_1, ..., d_ℓ), G^conf)
    (v_1, ..., v_k) := articulation-points(G^crit)
    Σ = (σ(1), ..., σ(k)) := random-permutation(v_1, ..., v_k)
    Π = (π_1, ..., π_ℓ) := Σ applied to (1, ..., ℓ)
    Reorder nodes of G^conf according to Π
    (s_1, ..., s_ℓ) :=First-Fit(G^conf, (d_1, ..., d_ℓ))
end for
return A = (s_1, ..., s_ℓ);
```

Having found all articulation points using the algorithm described above, we modify the initial order that was used to compute this scheduling by randomly permuting the articulation points within it. We then again use the greedy scheduling approach described above and find a new solution. This process can, in principle, be iterated indefinitely, or until no more articulation points exist (in that case, we could employ other methods to find node cuts as described above). The full heuristic algorithm is given in Algorithm 6 – there, the iteration is run K times.

The heuristic yields good solutions very fast, as we will show in the computational experiments in Section 6.5.

6.3 Phase II: Multicoloring formulation

We now propose an exponentially-sized relaxed formulation for the problem based on stable sets of the conflict graph. To solve it, we use a *Column generation* approach to try to avoid having to generate exponentially many variables. This approach was initially introduced by Dantzig and Wolfe [1960]. For a more recent description of Column generation techniques, compare Desaulniers et al. [2005].

Column generation

Column generation techniques are normally applied when we have a *master problem* that contains exponentially many variables, which can lead to memory problems and also very long running times for solving the subproblems.

The idea of *column generation* is to begin work with a small, promising subset of the variables. This variable-restricted problem is called the *restricted master problem (RMP)*.

The black-box plan for column generation goes approximately like this: We solve the RMP, which gives a current optimal objective value $\bar{z}$ and dual multipliers π, proving the optimality of the solution with the currently available variables. Because we want to solve the master problem, not the RMP, we then generate additional variables in such a way that they change the current optimal solution. This is accomplished by looking for variables with negative reduced costs which could be used to further improve the LP solution, if only they were part of the RMP. They can be found by solving another subproblem which we call the *pricing problem*. This consists of solving an LP whose objective is to minimize the reduced cost of not yet generated variables. We continue this process until there is no longer such a variable – then, we have solved the master problem optimally. Of course, this method works as described only for linear programs. To solve exponentially-sized ILPs using this method, it has to be embedded in a Branch&Bound framework; column generation is then used in every node of the search tree. This can lead to complications depending on the exact formulation of the problem, as the branching decisions by the Branch&Bound framework now interact with the pricing problem. For example, it could happen that in the pricing problem, one keeps generating some variable that can no longer enter the LP basic solution because some branching decision forbids using it.

If we simply use an ILP to solve the pricing subproblem, the branching decisions can be incorporated into the formulation of the pricing problem. More care needs to be taken if the subproblem is a combinatorial problem which can be solved more efficiently using a specialized algorithm. In that case, the branching decisions might destroy the structure of the subproblem, making the specialized algorithm useless. This is often a problem one would like to avoid by branching on variables that change the subproblem in such a way that its structure remains intact.

The multicoloring formulation

We now consider an exponentially-sized relaxed formulation for the multicoloring problem based on stable sets of the conflict graph that was proposed by Mehrotra and Trick [2007].

First, we observe that due to its construction, stable sets in G^{conf} correspond to sets of paths that are mutually not in conflict. Therefore, we can assign the same frequency slots to all paths inside such a stable set without creating a conflict. Let $\mathcal{S}$ be the set of all *maximal stable sets* of $G^{\text{conf}} = (P, E)$. We assume without loss of generality that we have an upper bound T for the optimal value. If we relax the requirement that frequency slots assigned to paths must be intervals, our problem reduces to the multicoloring problem.

We use nonnegative integer variables x_S in the ILP model for each stable set $S \in \mathcal{S}$. These indicate the number of slots in which this stable set is used.

$$\begin{aligned} \text{(IP-MC)} \qquad & \min \sum_{S \in \mathcal{S}} x_S \\ \text{subject to} \qquad & \sum_{S \in \mathcal{S} : i \in S} x_S \geqslant d_i \qquad \forall i \in P \\ & x_S \in \mathbb{N}_0 \qquad \forall S \in \mathcal{S} \end{aligned} \tag{6.3.1}$$

Constraints (6.3.1) ensure that we pick enough stable sets and hence, enough frequency slots to satisfy the demand of each path.

An optimal solution to (IP-MC) provides us with a lower bound for the Spectrum Assignment problem. We refer to this lower bound as T_{mcol}.

To illustrate the lower bounds, consider the example depicted in Figure 6.3.1. While an optimal frequency assignment needs at least $T_{opt} = 6$ frequency slots, the clique bound T_{clique} is equal to 4 and an optimal solution of the multicoloring model for the instance has a value of $T_{mcol} = 5$.

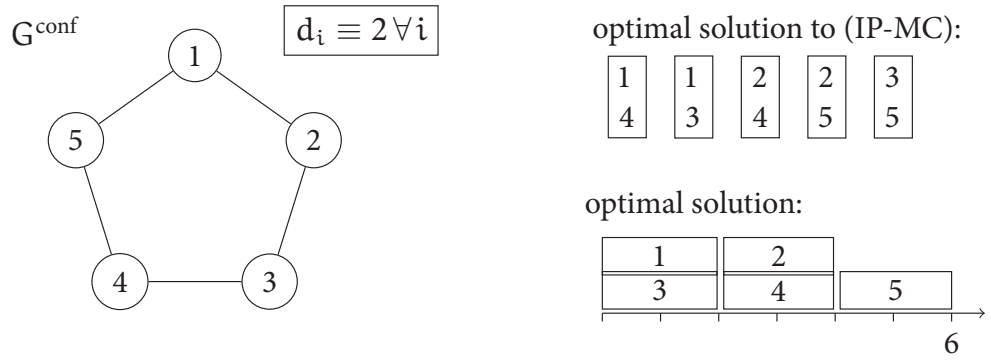


Figure 6.3.1: $T_{clique} < T_{mcol} < T_{opt}$

In the next section, we discuss how a solution to the multicoloring model can not only provide a good lower, but can be used to create good feasible solutions as well.

Upper Bounds

In order to solve the (IP-MC) model, we use the column generation model described as is, where we solve the pricing subproblem by a specialized Branch&Bound method available in SCIP (Achterberg [2009]).

Given an optimal solution to (IP-MC), let $\mathcal{A} = \{S_1, ..., S_k\}$ be the set of all stable sets picked by this optimal solution. In general, we may not be able to generate an optimal solution to our problem just by assigning frequency ranges to these stable sets, as there might be no way to assign them in such a way that the resulting frequency ranges form intervals. However, the stable sets picked can be used to provide some information about the problem structure to the heuristic described as Algorithm 5 on page 142. Any ordering of the stable sets can be used to generate a job reordering that may improve the solution found by that heuristic.

6.4 Phase III: Exact models

We now present several variants of modelling the problem exactly. We begin with several compact formulations and then present our Branch&Price-Approach that is, like the multicoloring model from the previous section, based on stable sets of the conflict graph.

We remind the reader that $\mathcal{P}$ denotes the set of paths, d_i are the path demands, $[T]$ is the set of available frequencies and E are the edges in the input graph.

6.4.1 Constraint Programming model

We begin with a constraint programming model. In this model, our objective is represented by an integer variable $0 \leqslant \text{makespan} \leqslant T$ that is to be minimized.

For each path $i \in P$, we introduce an integer variable z_i that determines the lowest frequency assigned to this path. To model the requirement that conflicting paths must have non-overlapping frequency intervals, we use a well-known constraint from the field of Constraint Programming known as the *cumulative constraint*. This constraint was introduced by Aggoun and Beldiceanu [1993] for resource-constrained scheduling problems. It models the condition that at no point in time, a set of jobs that is processed at the same time can exceed the resource bound. That is, if job j starts at time S_j, is processed for a time of p_j and uses r_j of some resource that has an upper bound of C, the condition

$$\sum_{j:S_j \leqslant t < S_j + p_j} r_j \leqslant C$$

has to be true for all times t.

The cumulative constraint appears here for every edge of the original graph. Any path that uses this edge has to use a frequency interval disjoint from all other paths using this edge. This can be expressed as a cumulative constraint with $C = 1$, $r_j = 1$ and a processing time of $p_j = d_j$.

Finally, we have the constraint that makespan $\geqslant \max_{i \in P} z_i + d_i$, linearized into one constraint for each path.

Let $P_e \subseteq P$ denote the set of paths containing the edge $e \in E$.

$$\begin{array}{llll} \text{(CP-SA)} & \min & \text{makespan} & \\ \text{subject to} & \text{cumulative} & (P_e, (z_p)_{p \in P_e}, (d_p)_{p \in P_e}, r = 1, C = 1) & \forall e \in E \quad (6.4.1) \\ & \text{makespan} \geqslant & z_p + d_p & \forall p \in P \quad (6.4.2) \\ & z_p \in & \mathbb{N}_0 & \forall p \in P \end{array}$$

6.4.2 Binary model

The *binary model* is a linearization of the constraint programming model—we simply express the constraints using binary variables and linear constraints on them.

For this, we have variables $x_{i,t}$ and $z_{i,t}$ for each $i \in P$ and $t \in [T]$, all of them binary.

The $z_{i,t}$ variable is an indicator variable determining the lowest frequency assigned to a path. Let $t(i)$ denote the index t of the lowest $z_{i,t}$ with a non-zero value, then the constraints will ensure that the necessary subsequent frequeny slots are used as well: $x_{i,j} = 1$ for all j with $j \in [t(i), t(i) + d_i - 1]$.

The $z_{i,t}$ variables for a fixed path $i \in P$ form a so-called *special ordered set of type* 1, meaning that only one of them can be non-zero. This is not forced by an explicit constraint in the model, but the objective function ensures this property for any optimal solution.

$$
\begin{aligned}
\text{(BM):} \qquad & \min \max_{i \in P} \Big(\sum_{t \in [T]} t \cdot z_{it} + d_i \Big) & & \\
\text{subject to} \qquad & \sum_{\tilde{t} \in \{0, T - d_i\}} z_{i,\tilde{t}} \geqslant 1 & & \forall i \in P \\
& \sum_{\tilde{t} \in [\max\{t - d_i + 1, 0\}, t]} z_{i,\tilde{t}} \leqslant x_{i,t} & & \forall i \in P, t \in [T] \\
& x_{i,t} + x_{j,t} \leqslant 1 & & \forall (i,j) \in E(G^{\text{conf}}), t \in [T] \\
& x_{i,t}, z_{i,t} \in \{0,1\} & &
\end{aligned}
$$

While the objective function is not linear, we can use a simple reformulation to linearize it. We introduce an integer variable y and write $y \geqslant \sum_{t \in [T]} t \cdot z_{it} + d_i$ for each path $i \in P$ as a constraint. The objective function is then $\min y$, which is now linear.

6.4.3 Conflict-Graph Orienting model

The Conflict-Graph Orienting model, introduced by Borndörfer et al. [1998], is based on the idea that any valid assignment of frequencies to paths determines an order for each pair $\{i, j\}$ of conflicting paths. This is because in the assignment, one of i and j has to be assigned to lower frequencies than the other. This way, any solution defines a partial order[1] for the conflict graph.

In the ILP model, we introduce integer variables l_i and r_i that define the borders of the frequency interval assigned to path i. For each conflict edge, we have a variable x_{ij} and a variable x_{ji} that determine an orientation of the conflict graph.

The objective function is expressed as the maximum over all r_i and can again be linearized like in the previous model.

$$
\begin{aligned}
\text{(CGO):} \qquad & \min \max_{i \in P} r_i & & & \\
\text{subject to} \qquad & l_i + d_i \leqslant r_i & & \forall i \in P & (6.4.3) \\
& l_j + T x_{ji} \geqslant r_i & & \forall ij \in E^{\text{conf}} & (6.4.4) \\
& l_i + T x_{ij} \geqslant r_j & & \forall ij \in E^{\text{conf}} & (6.4.5) \\
& x_{ij} + x_{ji} = 1 & & \forall ij \in E^{\text{conf}} & (6.4.6) \\
& x_{ij} \in \{0,1\} & &
\end{aligned}
$$

One can see immediately that the linking constraints (6.4.4) and (6.4.5) are of big-M type and because of this, the linear relaxation of this model will be quite weak.

[1] Partial means here that not all pairs of objects are comparable.

6.4.4 Branch&Price model

Our Branch&Price model is based on the idea of using stable sets as we already saw earlier in Phase II. In contrast to phase II, we will also enforce that the spectral ranges assigned to paths will be intervals.

The basic idea of the model will be the assignment of stable sets to frequency slots. The stable sets consist of nonconflicting paths, which therefore can all be assigned to the same frequency. Of course, if we use a stable set S containing a certain path $i \in P$ and the demand of path i is larger than 1, then several consecutive frequencies will have to use stable sets also containing path i, thereby restricting the possible choices of stable sets.

Similar ideas have previously been used for wireless frequency assignment problems, see for example Jaumard et al. [2001]. There, co-channel constraints are incorporated into variables that corresponds to sets that can be assigned to the same channel. These variables are then created in a column generation scheme. On the other hand, the "Adjacent Channel Interference Constraints" are explicitly expressed in the model, again restricting the possible choices for sets.

We now give a formal description of the model. For every $S \in \mathcal{S}$, $t \in [T]$, we introduce binary variables $x_{S,t} \in \{0,1\}$ that indicate if stable set S is used in slot t; that is, if the frequency slot t is assigned to paths corresponding to the nodes in S. For every $P_i \in \mathcal{P}$ and $t \in [T]$, we consider binary variables $z_{i,t} \in \{0,1\}$ that indicate if the contiguous set of slots assigned to path P_i starts at slot t. Finally, we use binary variables $u_t, t \in [T]$ that indicate whether slot t is used at all. Using the notation above, our ILP formulation is as follows.

$$\text{(IP-BP):} \qquad \min \sum_{t \in T} u_t$$

$$\text{subject to} \qquad \sum_{S \in \mathcal{S}: i \in S} x_{S,t} \geqslant \sum_{\tilde{t} \in [\max\{t-d_i+1,0\},t]} z_{i,\tilde{t}} \qquad \forall i \in P, t \in [T] \tag{6.4.7}$$

$$-\sum_{S \in \mathcal{S}} x_{S,t} \geqslant -u_t \qquad \forall t \in [T] \tag{6.4.8}$$

$$\sum_{\tilde{t} \in [0,T-d_i]} z_{i,\tilde{t}} \geqslant 1 \qquad \forall i \in P \tag{6.4.9}$$

$$u_{t-1} \geqslant u_t \qquad t \in [T] \tag{6.4.10}$$

$$x_{S,t} \in \{0,1\} \qquad \forall S \in \mathcal{S}, t \in [T]$$

$$z_{i,t} \in \{0,1\} \qquad \forall i \in P, t \in [T]$$

$$u_t \in \{0,1\} \qquad \forall t \in [T]$$

The objective is to minimize the number of slots used. As the slot ordering constraints (6.4.10) actually lead to a contiguous slot usage, this corresponds to minimizing the total length of the spectrum range used by the routing paths.

Constraints (6.4.7) enforce that a slot t must be assigned to some stable set containing path i if the set of slots assigned to path P_i has already started at frequency slot $\tilde{t} \in \{t - d_i + 1, \ldots, t\}$. This ensures that the frequency assignment for any path in some optimal solution to (IP-BP) is an interval. Constraints (6.4.8) impose that we do not use more than one stable set for each slot.

Finally, constraints (6.4.9) forbid to start a path so late that its frequency interval does not fit anymore within the range bound.

Note that it is possible that we get a solution to (IP-BP) in which the set of used frequencies for some paths is not an interval. In this case however, intermediate slots are unused. To alleviate this problem, we can simply shift the assignment to the left, thereby removing the unused slots, to get a contiguous interval.

Solving the B&P-model

The proposed formulation (IP-BP) contains an exponential number of variables. Therefore, the solution procedure we devise is based on the column generation technique. We consider as the *restricted master* problem the continuous relaxation of the (IP-BP) model including all the constraints and the z variables, but only the x variables corresponding to a subset $\mathcal{S}' \subseteq \mathcal{S}$ of the maximal stable sets of G^{conf}.

Initialization

We enrich the restricted master problem with solutions obtained in the first stage. Given an assignment A returned by Algorithm 2. We extract stable sets corresponding to this assignment as described in Algorithm 3; and then add them to $\mathcal{S}'$.

Algorithm 7 extract_stable_sets(A)

Require: assignment A
Ensure: family of stable sets $\mathcal{A}$
1: Let $B(t)$ be the set of path in S using slot t
2: $\mathcal{A} := \emptyset$, $t := 0$
3: **while** $B(t) \neq \emptyset$ **do**
4: add $B(t)$ to $\mathcal{A}$
5: increase t until $B(t)$ changes
6: **end while**
7: **return** $\mathcal{A}$

Column generation

We iteratively solve the restricted master problem and search for new columns with negative reduced cost. This can be computed using an optimal dual solution as follows. Let the dual variables corresponding to constraints (6.4.7) be $\pi_{i,t} \geqslant 0$, for all $i \in P, t \in [T]$. We also refer to the dual variables corresponding to constraints (6.4.8) as $\mu_t \geqslant 0$, for all $t \in [T]$. The corresponding pricing problem looks as follows.

$$\min_{t\in\{0,1,\ldots,T\}} \left\{ 1 + \mu_t - \max_{S\in\mathcal{S}} \left\{ \sum_{i\in S} \pi_{i,t} \right\} \right\} \tag{6.4.11}$$

We observe that, for a fixed slot t, this corresponds to solving a *maximum-weight stable set* problem on the conflict graph, while a weight $\pi_{i,t}$ is associated with each node i in P. So our pricing subroutine works as follows. Given a solution to the restricted master problem, we solve the corresponding maximum-weight stable set problem for every t independently. If the solution to the maximum-weight stable set problem has value less than $1 + \mu_t$ for all t, there is no additional variable with negative reduced cost and the solution to the restricted master problem is therefore optimal. Otherwise, the variable $x_{S,t}$ corresponding to the stable set and frequency attaining the optimal value in (6.4.11) will be added to the RMP. To speed up the column generation process, we add this variable for more than just this frequency. Let j be the path in S with the highest demand. We then add the variables corresponding to this stable set in the range $[t-d_j, t+d_j]$ immediately to our restricted master problem. The new restricted master problem is resolved and the process is iterated until the master problem has been completely solved.

Branching strategies

The optimal solution obtained at the end of the above pricing loop might not be integral. We hence apply the *Branch&Bound* technique (see Wolsey [1998], compare also Subsection 1.3.5) to handle integrality. Note that this technique when used together with column generation is called *Branch&Price* (see Lübbecke and Desrosiers [2005]). For this end, we propose and evaluate two branching rules.

We also remark that branching on the z variables is sufficient, as the x-variables can always be set to integral values without any loss. This is due to the fact that if in a certain solution the z-variables are fixed to integral values and we choose a fixed t, a certain subset of the constraints (6.4.7) are active enforcing certain nodes to be in the stable set S chosen for the slot t by the x-variables. We can choose any stable set that contains all the enforced nodes and set the corresponding x-variable to 1, fixing all other x-variables to zero, without losing any objective value, as this change has no influence on the u_t-variables.

Median-based spectrum range branching. We determine the z variable with the most fractional value, that is, if $\nu(z)$ denotes the fractional value of variable z, we choose one that maximizes $\min\{\nu(z), 1-\nu(z)\}$. Let P_i be the path it corresponds to. We define the partial sum $S_\ell = \sum_{t=0}^{\ell} z_{i,t}$. Let m be the first slot among the spectrum range $[T]$ for which $S_m \geqslant \frac{1}{2}$. We define t_{break} to be m, if $S_m \leqslant \frac{17}{20}$; $\max\{m-1, 0\}$, otherwise. The heuristic idea behind this is the following: If $S_m > \frac{17}{20}$, the value assigned to slot m is greater than $\frac{7}{20}$, so close to $\frac{1}{2}$ (as this was the first slot where the partial sum exceeds $\frac{1}{2}$). To get the branches as equal as possible, we should then assign m to the upper interval, as only a small part is left to get to 1. On the other hand, if $S_m \leqslant \frac{17}{20}$, we can assign m to the lower interval, as at least $\frac{3}{20}$ of fractional value can be assigned to the upper interval, hopefully even more. We create two branches, by imposing

$$\sum_{t=0}^{t_{\text{break}}} z_{i,t} = 0$$

in one child node and

$$\sum_{t=t_{break}+1}^{T} z_{i,t} = 0$$

in the other one.

Naive spectrum range halving. In this branching rule, we again identify the most fractional z variable as well as the path P_i it belongs to. We identify the first and the last slot among $[T]$, namely t_{begin} and t_{end}, whose $z_{i,t}$ variable is not yet upper bounded by zero (by constraints imposed in the parent nodes of the current branch-and-bound tree). We compute an intermediate point $t_{inter} := \lceil \frac{t_{begin}+t_{end}}{2} \rceil$. Then, two branches are created by imposing $\sum_{t=t_{begin}}^{t_{inter}-1} z_{i,t} = 0$ in one child node, and $\sum_{t=t_{inter}}^{t_{end}} z_{i,t} = 0$ in the other one.

Fractional fatness branching Let v be some branching candidate, that is, one of the z variables. We now introduce a score $s(v)$. Our branching rule then selects a branching candidate with maximum score.

For this purpose, let j_v denote the job and t_v denote the spectral slot belonging to the variable v. Let

$$f(v) := \min \left\{ 2 \cdot \sum_{t \in [t_v+1]} z^*_{i,t}, \quad 1 - \sum_{t \in [t_v+1]} z^*_{i,t} \right\}.$$

Furthermore, we set

$$w(v) = d(j_v) \cdot e(j_v),$$

where $d(j_v)$ denotes the spectral demand of the job j_v and $e_j(v)$ denotes the path length of j_v. We set $c(v) := \log(n) + 1$, where n denotes the number of $z_{j(v),t}$ variables currently set to non-zero.

We define the branching score as

$$s_v := f_v \cdot w_v \cdot c_v.$$

These are some heuristic ideas to avoid symmetries in the branching process and to favor paths in the branching that have not been restricted very much in the previous branching decisions. If a lot of paths are restricted early in the branch-and-bound tree, the hope is that strong cuts can be generated early using the model inequalities.

We remark that the branching decisions require no changes in the basic structure of the pricing problem, which remains a maximum-weight stable set problem.

Rounding heuristic

In order to improve the overall performance of our Branch&Price algorithm, it is helpful to generate good primal solutions during the solution procedure. Given an optimal, fractional solution (x^*, z^*) of some current node of the Branch&Bound-tree, we can use it to generate a job order to run the First-Fit heuristic. This is done by determining, for different threshold values $h \in [0, 1]$, the lowest integer t_h at which the sum $\sum_{t=0}^{t_h} z^*_{i,t}$ exceeds the value h. For every

chosen threshold value this yields a value $t_h(p)$ for every path, which we use to generate a job order by comparing these values. Then, the First-Fit Assignment Heuristic, Algorithm 5 on page 142, is called to determine a feasible assignment using this ordering.

Reusing previous information

To be able to finish each stage as quickly as possible, we try to reuse as much information as possible. From the first to the second stage, the clique bound and the best found solution are kept. The best solution is then transformed into a representation by stable sets and the corresponding columns are part of the starting variable set of the restricted master problem.

We also keep the best solution found in stage II and reuse the corresponding stable sets to initialize Phase III.

6.5 Computational experiments

Computational details

For our implementation, we used the Mixed-Integer Programming Solver SCIP (Achterberg [2009]) in version 3.1.1 with the LP solver SoPlex. The tests were run on an Intel i7-4771 CPU with 3.5 GHz and with 16GB of memory.

6.5.1 Test instances

We have 20 test instances. Of these, instances a-l are based on the data of the Eibone project (Bley et al. [2008]), while instances m-t are based on the SNDLib (Orlowski et al. [2010]). The routing is constructed manually via shortest path computations in the original network (distributing the load on two edge-disjoint paths if they exist), demands are based on the original demands downscaled to realistic values for our considered problem.

More information on properties of the test instances can be found in Table 6.1 on the following page. We denote the instance name, the number of nodes, the number of edges and the number of demand-pairs. In column *density*, we denote the percentage of edges the conflict graph has compared to a complete graph on the same number of nodes. It is followed by the minimum demand value, the maximum demand value and the average and median demand. In the last column, we mark the standard deviation of the demand values.

6.5.2 Computational experiments

Results

A time limit of 2 hours was used in all computations.

First, we explain the table for the Articulation Point Heuristic and the Multicoloring approach, compare Table 6.2 on page 155. Column *I* denotes the instance name. In columns UB^* and LB^*, the best results for these instances that we know are marked. We remark that all the lower bounds in the table match the clique bound explained earlier. So for all but one instance, we know the optimal solution values equals the clique bound.

inst	|V|	|E|	|P|	density	d_{min}	d_{max}	$\overline{d}$	d_{median}	$\sigma(d)$
a	17	17	16	0.16	3	12	7.31	7	2.59
b	17	16	60	0.46	3	3	3.00	3	0
c	17	16	25	0.27	3	12	7.32	8	1.67
d	17	16	33	0.49	3	3	3.00	3	0
e	17	16	26	0.22	3	3	3.00	3	0
f	17	16	47	0.35	3	12	5.64	4	2.19
g	17	16	59	0.42	3	12	5.10	5	1.30
h	17	16	18	0.27	3	12	3.50	3	2.06
i	17	16	31	0.30	3	12	5.10	5	2.10
j	17	17	99	0.45	3	12	4.23	4	0.97
k	17	17	97	0.41	3	12	4.10	4	0.92
l	17	16	63	0.44	3	12	3.62	3	1.50
m	12	15	132	0.47	3	12	3.53	3	1.42
n	15	22	210	0.44	3	12	4.11	4	1.32
o	37	57	1332	0.30	3	12	4.25	4	0.96
p	10	45	90	0.06	3	12	4.31	4	1.95
q	11	42	44	0.12	3	3	3.00	3	0.00
r	25	45	600	0.22	3	12	4.43	4	1.59
s	50	88	1324	0.17	3	3	3.00	3	0.00
t	40	89	1560	0.20	3	12	8.15	9	1.64

Table 6.1: Overview of basic instance properties. Instances a-l are based on the Eibone project, while instances m-t are based on the SNDLib.

In the following columns, we present the results for the *Articulation Point Heuristic*. By $\nu(S)$, we denote the best value the heuristic found. By *gap*[%], we compare the best value found against the clique bound. In the column gap_{UB}[%], we compare the best solution value found by the heuristic against the best overall upper bound. The number of iterations is denoted in the column #It. Finally, we mark the running time in seconds in the column t[s]. For the *Multicoloring + StableSet-FirstFit* approach, the same values are presented in the table.

We now explain the tables for the Constraint Programming and the Binary model, compare Table 6.3 on page 156. The main difference to the previous approaches is that these approaches generate their own lower bounds that increase throughout the computation. So in column *gap*[%], we write the final gap that was reported by the solver. In column gap_{LB^*}, we report the gap between the solution this approach generated and the best lower bound we know. Furthermore, in column #BB, we report the Branch&Bound-nodes processed by the solver.

In the last table of computational results, see Table 6.4 on page 158, we present the results for the Conflict-Graph Orienting model and the full column generation model using the last of the described branching rules. We do not present the exact results for the other branching rules, but remark that empirically, the branching rule presented here worked best among the three described. The selected branching rule has no influence on the largest few of the instances in these computations as the time limit is already exceeded before the branching process even begins.

			Articulation Point Heuristic					Multicoloring + StableSet-FirstFit				
I	UB*	LB*	$\nu(S)$	gap[%]	gap_{UB}[%]	#It	t[s]	$\nu(S)$	gap[%]	gap_{UB}[%]	#It	t[s]
a	41	41	41	0.00	0.00	1	<1	41	0.00	0.00	1	<1
b	78	78	78	0.00	0.00	1	<1	78	0.00	0.00	1	<1
c	62	62	62	0.00	0.00	6	<1	62	0.00	0.00	2	<1
d	45	45	45	0.00	0.00	9	<1	45	0.00	0.00	1	<1
e	24	24	24	0.00	0.00	1	<1	24	0.00	0.00	1	<1
f	108	108	108	0.00	0.00	1	<1	108	0.00	0.00	1	<1
g	137	137	137	0.00	0.00	1	<1	137	0.00	0.00	3	1
h	30	30	30	0.00	0.00	1	<1	30	0.00	0.00	1	<1
i	65	65	65	0.00	0.00	1	<1	65	0.00	0.00	1	<1
j	134	134	134	0.00	0.00	63	<1	140	4.29	4.29	14334	7200
k	110	110	110	0.00	0.00	16103	94	111	0.90	0.90	21133	7200
l	87	87	87	0.00	0.00	17	<1	87	0.00	0.00	16	8
m	153	153	153	0.00	0.00	3065	45	153	0.00	0.00	12486	7007
n	310	310	310	0.00	0.00	30	1	318	2.52	2.52	6927	7200
o	1044	1026	1049	2.19	0.48	841	7198	1108	7.40	5.78	9	7179
p	49	49	49	0.00	0.00	3	<1	49	0.00	0.00	1	<1
q	24	24	24	0.00	0.00	1	<1	24	0.00	0.00	1	<1
r	462	462	462	0.00	0.00	4	3	467	1.07	1.07	2168	7200
s	396	396	396	0.00	0.00	33	145	—	—	—	—	—
t	2333	2333	2333	0.00	0.00	11	106	2338	0.22	0.22	47	7030

Table 6.2: Computational results for Articulation Point and Multicoloring Heuristic

			Constraint program+FirstFit					Binary model+FirstFit				
Inst	UB*	LB*	$\nu(S)$	gap[%]	gap_{LB^*}	#BB	t[s]	$\nu(S)$	gap[%]	gap_{LB^*}	#BB	t[s]
a	41	41	41	0.00	0.00	1	<1	41	0.00	0.00	1	6
b	78	78	78	0.00	0.00	1	1	78	92.30	0.00	400	7082
c	62	62	66	27.27	6.06	228500	7195	62	0.00	0.00	9	87
d	45	45	51	66.66	11.76	156200	7192	45	82.22	0.00	32600	7200
e	24	24	24	0.00	0.00	1	<1	24	0.00	0.00	1	5
f	108	108	108	0.00	0.00	176	<1	113	78.76	4.42	1400	7198
g	137	137	137	0.00	0.00	335	4	151	88.08	9.27	1	7198
h	30	30	30	0.00	0.00	1	<1	30	0.00	0.00	20900	91
i	65	65	65	0.00	0.00	177	<1	65	0.00	0.00	2600	367
j	134	134	155	16.78	13.59	270100	7198	155	87.74	13.55	1	7200
k	110	110	110	0.00	0.00	21121	544	138	87.68	20.29	1	7199
l	87	87	87	0.00	0.00	501	8	93	79.57	6.45	200	7199
m	153	153	153	0.00	0.00	1642	39	170	87.06	10.00	1	7200
n	310	310	310	0.00	0.00	2465	63	330	92.73	6.06	1	7219
o	1044	1026	1155	12.29	11.17	214500	7197	—	—	—	—	374
p	49	49	56	14.29	12.50	160800	7197	49	6.12	0.00	19500	7200
q	24	24	24	0.00	0.00	1	1	24	0.00	0.00	1	10
r	462	462	465	11.83	0.65	255400	7199	465	—	0.65	0	7200
s	396	396	411	98.05	3.65	210800	7196	—	—	—	—	735
t	2333	2333	2462	49.59	5.24	219800	7198	—	—	—	—	628

Table 6.3: Computational results for the Constraint Programming and the Binary model approach

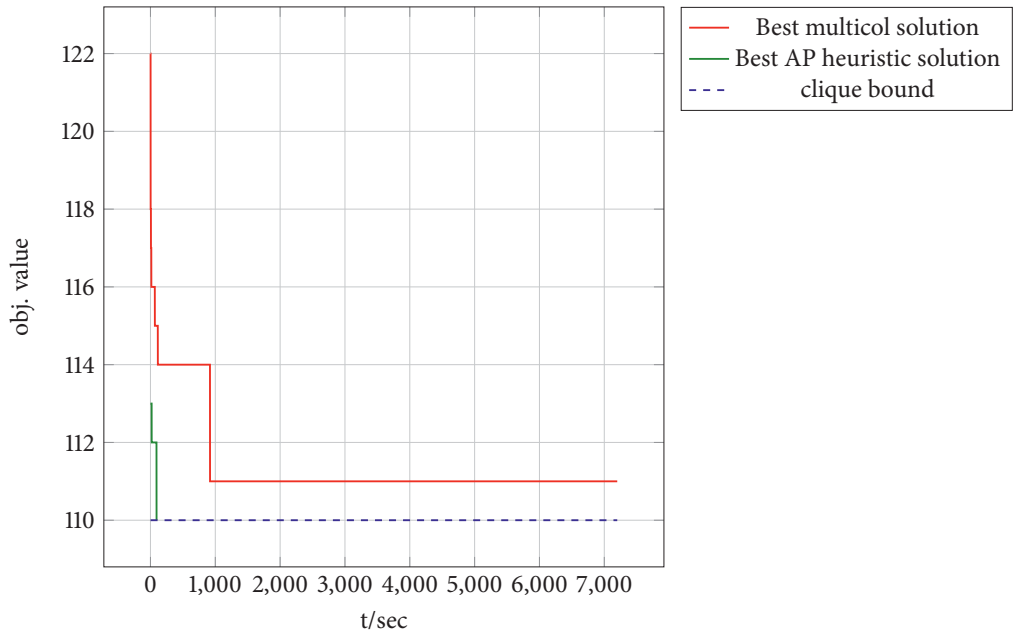


Figure 6.5.1: Multicoloring approach vs Articulation Point Heuristic on instance k

For the column generation model, we have two additional columns in the table, #PC and #PV, detailing the number of the times the pricer was called and the number of variables it generated in these calls. For the larger instances, the running time of the pricer is not a significant factor, as it takes much longer to solve the increasingly larger LP relaxations.

Results of the experiments

For the considered models, we conclude that the heuristic approaches work best in finding good solutions quickly. Also, for these instances it is simple to prove the optimality of heuristic solutions, because the clique bound equals the value of an optimal solution in all but one instance, and for the remaining instance, it might still be true, but we do not know, as a provably optimal solution for this instance seems to be out of reach with our methods.

Concerning the compact exact ILP models, we can see that while the small models get solved quickly, partly due to the additional FirstFit-Heuristic, for the larger instances, the gaps are very large. The gaps are quite a bit better in the Constraint Programming approach, but still are not very good. While the column generation approach managed to solve every instance optimally where it made any progress at all, it could not even solve the root node relaxation for the larger instances within the given time bound.

6.6 Conclusions

We introduced a combinatorial optimization problem for the spectrum assignment to optical lightpaths. We studied two heuristic approaches, one of them based on a relaxation, and four

			Conflict-Graph Orienting model					Exact-ZSM					
I	UB*	LB*	$v(S)$	gap[%]	gap_{LB^*}	#BB	t[s]	$v(S)$	gap[%]	#BB	#PC	#PV	t[s]
a	41	41	41	0.00	0.00	1	<1	41	0.00	1	2	540	1
b	78	78	78	87.18	0.00	250300	7195	78	0.00	1	5	793	5
c	62	62	62	6.45	0.00	5068000	7200	62	0.00	1	2	1049	3
d	45	45	45	71.11	0.00	5603000	7200	45	0.00	1	7	693	2
e	24	24	24	0.00	0.00	16900	26	24	0.00	1	3	134	<1
f	108	108	108	62.04	0.00	4630000	7200	108	0.00	1	3	3488	33
g	137	137	137	81.02	0.00	2320000	7200	137	0.00	1	3	3929	89
h	30	30	30	0.00	0.00	2300	3	30	0.00	1	2	128	<1
i	65	65	65	0.00	0.00	2253000	48	65	0.00	1	2	839	4
j	134	134	155	86.45	13.55	396500	7200	134	0.00	5	16	5069	5697
k	110	110	128	86.72	14.06	505400	7200	118	—	1	19	9936	7199
l	87	87	90	74.44	3.33	1193000	7200	87	0.00	1	7	3422	51
m	153	153	170	87.65	10.00	119800	7198	—	—	0	6	7547	7200
n	310	310	330	92.72	6.06	27300	7190	—	—	0	0	0	7200
o	1044	1026	1155	98.96	11.16	1	31	—	—	0	0	0	7200
p	49	49	49	0.00	0.00	138900	3601	49	0.00	1	3	2086	35
q	24	24	24	0.00	0.00	13300	21	24	0.00	1	4	139	1
r	462	462	465	96.56	0.65	1	7154	—	—	0	0	0	7200
s	396	396	411	99.27	3.65	3	411	—	—	0	0	0	7200
t	2333	2333	2462	99.51	5.24	12	28	—	—	0	0	0	7200

Table 6.4: Computational results for the Conflict-Graph Orienting model and the column generation model

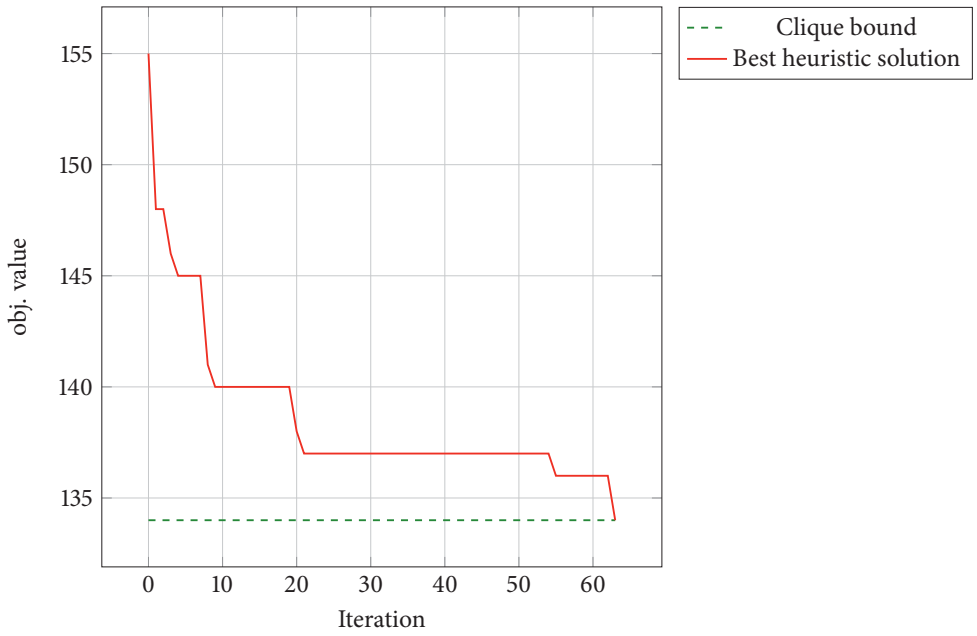


Figure 6.5.2: Articulation Point Heuristic performance on instance j. The total runtime was less than one second.

mathematical programming approaches, one of them based on Constraint Programming and one based on column generation techniques. In the computational study, we showed that the heuristic approaches are able to find good, often even optimal solutions quickly. For the two compact ILP models, we remarked that the lower bounds implied by the node relaxations are quite weak. The column generation model was shown to work quite well, but quickly runs out of time or memory given a time limit of 2 hours, often even without having solved the root node relaxation. One interesting follow-up question would be about the compact ILP models. The lower bounds are very weak, but there might be a way to improve them by separating additional cuts. The clique bound is easily computable, so could maybe also be used to provide additional strengthening inequalities. Another way to deal with the weakness of the lower bounds might involve extended formulations. As we saw in the computational study, the column generation model presented is strong, but also too large to be solved within a timeframe of hours and often even at all, because it uses too much memory. There might be a formulation of intermediate size that works better. Finally, it would be interesting to examine the Constraint Programming approach in more detail and maybe find some way to improve its lower bounds.

7 Online Scheduling

In this chapter, we introduce the concept of Competitive-Ratio Approximation Schemes and apply them to the problem of online scheduling to minimize the weighted average completion time. The work presented in this chapter is joint work with Elisabeth Lübbecke, Nicole Megow and Andreas Wiese. An extended journal version is currently pending review.

7.1 Online Scheduling problems

7.1.1 Introduction

The basic question in scheduling is how to allocate limited resources over time. The resources are usually called *machines* and the resource requests *jobs*. A basic example is the following. We have a set of jobs J that have to be processed on some machines. Each job has a processing time p_j and has to be assigned to one machine for this time. Usually, a machine can only process one job at a time and we try to minimize some property of the resulting schedule, for example the *makespan* which is defined as the last point in time at which one of the jobs is completed. Other common examples of objective functions include the flowtime, the weighted average completion time and variants thereof.

Notation of Scheduling problems For a given scheduling problem, the number of machines is denoted by m, while n denotes the number of jobs. Each job $j \in J$ has some running time p_j. In *weighted* variants, we also assign a weight w_j to each job. Jobs may not by available at the beginning of a schedule, but only be released at some *release date* r_j, which is a lower bound on their allowed starting time.

If we allow a schedule to interrupt a running problem and pick it up at a later time or on a different machine, we are in the setting of *Preemptive Scheduling*, otherwise *Non-preemptive Scheduling*, If we do not explicitly specify the setting, preemption will be not be allowed.

Online scheduling In *Offline Scheduling*, the entire input instance is known in advance. The goal usually is to find a polynomial-time algorithm that is able to create a good schedule, for example only worse than an optimal schedule by a constant factor.

In contrast, in *Online Scheduling*, jobs only become known when they *arrive*, that is, at their release date. The online algorithm then has to make decisions based only on the partial knowledge it possesses about previous jobs and the current situation.

7.1.2 Problem Setting

We consider here the online setting with three different machine environments: identical parallel machines (denoted by P), related machines (Q) where each machine i has associated a speed s_i and processing a job j on machine i takes p_j/s_i time and unrelated machines (R) where the processing time of a job j on each machine i is explicitly given as a value p_{ij}. The main problem considered in this chapter is to schedule the jobs on the given set of machines so as to minimize $\sum_{j\in J} w_j C_j$, where C_j denotes the completion time of job j.

We consider the problem with and without preemption. Using standard scheduling notation (Graham et al. [1979]), we denote the non-preemptive (resp. preemptive) problems that we consider in this chapter by $(Pm|\, r_j, (pmtn) \,|\, \sum w_j C_j)$, $(Qm|\, r_j, (pmtn) \,|\, \sum w_j C_j)$ and $(Rm|\, r_j, pmtn \,|\, \sum w_j C_j)$.

Relations to embedding problems of virtual networks The online scheduling problems considered in this chapter have some connections to online network embedding problems.

For example, consider a network embedding problem where all requests only require single nodes. In that case, the nodes in the substrate network can be interpreted as the machines of a scheduling problem. The network requests correspond to jobs that need to be scheduled online on a number of machines. These could be unrelated machines or even parallel, identical machines if the machines in the nodes of the substrate network are all equivalent. If we consider the weighted sum of completion times as the objective function, we might interpret it in the network embedding setting as rewards for not making a request wait to be embedded. The earlier we embed the request into the network the earlier it will be completed. If we consider a makespan objective function, we are asking for the shortest possible timeframe we need to embed all the requests.

If we want to allow reconfiguration of jobs to other machines while they are running, the corresponding setting in the online scheduling is to allow preemption of jobs. Reconfiguration can also be a desirable property of substrate networks, as it allows for a better utilization of resources.

Competitive analysis Given a minimization problem, a deterministic online algorithm A is called ρ-*competitive* if, for any problem instance $\mathcal{I}$, it achieves a solution of value

$$A(\mathcal{I}) \leqslant \rho \cdot \mathrm{OPT}(\mathcal{I}),$$

where $\mathrm{OPT}(\mathcal{I})$ denotes the value of an optimal offline solution for the same instance $\mathcal{I}$. The *competitive ratio* ρ_A of A is the infimum over all ρ such that A is ρ-competitive. The minimum competitive ratio ρ^* achievable by any online algorithm is called *optimal*. Note that there are no requirements on the computational complexity of competitive algorithms. Indeed, the competitive ratio measures the best possible performance under the lack of information given unbounded computational resources.

Competitive analysis (Sleator and Tarjan [1985]; Karlin et al. [1988]) is the most popular method for studying the performance of online algorithms. It provides an effective framework to analyze and classify algorithms based on their worst-case behavior compared to an optimal offline algorithm.

Scheduling for Weighted Average Completion Time A classical well-known problem is online scheduling to minimize the weighted average completion time. It has received a lot of attention in the past two decades. For different machine environments, a long sequence of papers (see below) emerged, introducing new techniques and algorithms, improving upper and lower bounds on the competitive ratio of particular algorithms as well as on the best possible competitive ratio that any online algorithm can achieve. Still, unsatisfactory gaps remain. As for most online problems, a provably optimal online algorithm with respect to competitive analysis among *all* online algorithms is only known for very special cases.

7.1.3 Related work

The offline variants of nearly all scheduling problems under consideration are **NP**-hard, but in most cases polynomial-time approximation schemes have been developed. The corresponding online settings have been a highly active field of research in the past fifteen years. A whole sequence of papers appeared introducing new algorithms, new relaxations and analytical techniques that decreased the gaps between lower and upper bounds on the optimal competitive ratio (Goemans et al. [2002]; Schulz and Skutella [2002a]; Hall et al. [1997]; Sitters [2010b,a]; Anderson and Potts [2004]; Chekuri et al. [2001]; Hoogeveen and Vestjens [1996]; Correa and Wagner [2009]; Goemans [1997]; Megow and Schulz [2004]; Megow [2007]; Chung et al. [2010]; Schulz and Skutella [2002b]; Liu and Lu [2009]; Lu et al. [2003]; Stougie and Vestjens [2002]; Phillips et al. [1998]; Chakrabarti et al. [1996]; Seiden [2000]; Epstein and van Stee [2003]). Despite the considerable effort, *optimal* competitive ratios are hardly known while generally unsatisfactory, even quite significant gaps remain. We give more details on the state of the art for the different problem classes under consideration in the subsections below. Before we do so, we discuss other previous work on computing techniques for competitive ratios.

To the best of our knowledge, there are only very few problems in online optimization for which an optimal competitive ratio can be determined, bounded, or approximated by computational means. Lund and Reingold [1994] present a framework for upper-bounding the optimal competitive ratio of randomized algorithms by a linear program. For certain cases, e.g., the 2-server problem in a space of three points, this yields a provably optimal competitive ratio. Ebenlendr et al. [2009] and Ebenlendr and Sgall [2011] study various online and semi-online variants of scheduling preemptive jobs on uniformly related machines to minimize the makespan. In contrast to our model, they assume the jobs to be given one by one (rather than over time). They prove that the optimal competitive ratio can be computed by a linear program for any given set of speeds. In terms of approximating the best possible performance guarantee, Augustine et al. [2008] is closest to ours. They show how to compute a nearly optimal power-down strategy for a processor with a finite number of power states.

Sum of weighted completion times. The offline variants of scheduling to minimize the total weighted completion time are **NP**-hard, in many cases even for the special case of a single machine (Labetoulle et al. [1984]; Lenstra et al. [1977]). Two restricted single-machine variants can be solved optimally in polynomial time. *Smith's Rule* solves the problem $(1||\sum w_j C_j)$ to optimality by scheduling jobs in non-increasing order of weight-to-processing-time ratios (Smith [1956]). Furthermore, scheduling by shortest remaining processing times yields an

optimal schedule for $(1|\,r_j, pmtn\,|\sum w_j C_j)$ (Schrage [1968]). However, for the other settings polynomial-time approximation schemes have been developed (Afrati et al. [1999]), even when the number of machines is part of the input.

We do not intend to give a detailed history of developments in online scheduling; instead, we refer the reader to overviews, e.g., in Megow [2007]; Correa and Wagner [2009]. Interestingly, despite the considerable effort, optimal competitive ratios are known only for $(1|\,r_j, pmtn\,|\sum C_j)$ (Schrage [1968]) and for non-preemptive single-machine scheduling (Anderson and Potts [2004]; Stougie and Vestjens [2002]; Hoogeveen and Vestjens [1996]; Chekuri et al. [2001]).

Our contributions

In this chapter, we close the aforementioned gaps, at least theoretically, by finding algorithmic methods to determine nearly optimal online scheduling algorithms. We provide what we call *competitive-ratio approximation schemes* that compute algorithms with a competitive ratio that is at most a factor $1+\varepsilon$ larger than the optimal ratio for any $\varepsilon > 0$. To that end, we introduce a new way of designing online algorithms. Apart from structuring and simplifying input instances, we find an abstract description of online scheduling algorithms which allows us to reduce the infinite-size set of all online algorithms to a relevant set of finite size, which is the key for allowing an enumeration scheme like the one we present.

Besides improving on previous algorithms, our method also provides an algorithm to determine the competitive ratio of the designed algorithm, and even the best possible competitive ratio, up to any desired accuracy. We are aware of only very few online problems for which a competitive ratio, or even the optimal competitive ratio, are known to be *computable* by some algorithm— at least for a not inherently finite problem. Typically, there are no means of enumerating all possible input instances and all possible online algorithms – even for only one given algorithm, usually one cannot compute its competitive ratio simply due to difficulties like the halting problem.

We believe that our concept of abstraction for online algorithms can be applied successfully to some other problems. We show this for other scheduling problems with jobs arriving online over time. Our techniques are presented focussing on the problems

1. $(Pm|\,r_j, (pmtn)\,|\sum w_j C_j)$ and
2. $(Qm|\,r_j, (pmtn)\,|\sum w_j C_j)$,

under the assumption of a constant range of machine speeds in the related machines case without preemption. For any $\varepsilon > 0$, we show that the competitive ratios of our new algorithms are by at most a factor $1+\varepsilon$ larger than the respective optimal competitive ratios.

To achieve our results, we present a novel abstraction in which online algorithms are formalized as *algorithm maps*. Such a map receives as input a set of unfinished jobs together with the schedule computed so far. Based on this information, it returns a schedule for the next time instant. This view captures exactly how online algorithms operate under limited information. The total number of algorithm maps is unbounded. However, we show that there is a finite subset which approximates the entire set. More precisely, for any algorithm map there is a map in our subset whose competitive ratio is at most by a factor $1+\varepsilon$ larger. To achieve this

reduction, we first apply several standard techniques, such as geometric rounding, time-stretch, and weight-shift, to transform and simplify the input problem without increasing the objective value too much; see, e.g., Afrati et al. [1999]. The key, however, is the insight that it suffices for an online algorithm to base its decisions on the currently unfinished jobs and a very *limited part* of the so far computed schedule—rather than the entire history. This allows for an enumeration of all relevant algorithm maps (see also Manasse et al. [1988] for an enumeration routine for online algorithms for a fixed task system with finitely many states). For randomized algorithms we even show that we can restrict to instances with only constantly many jobs. As all our structural insights also apply to offline algorithms for the same problems, they might turn out to be useful for other settings as well.

Although the enumeration scheme for identifying the nearly optimal online algorithm heavily exploits unbounded computational resources, the resulting algorithm itself has polynomial running time. As a consequence, there are efficient online algorithms for the considered problems with almost optimal competitive ratios. Hence, the granted additional, even unbounded, computational power of online algorithms does not yield any significant benefit here.

In Section 7.2 we introduce several general transformations and observations that simplify the structural complexity of online scheduling in the $(Pm|r_j, pmtn|\sum w_j C_j)$ setting. Based on this, we present our abstraction of online algorithms and develop a competitive-ratio approximation scheme in Section 7.3. Finally, we sketch in Section 7.4 how to extend these techniques to the non-preemptive setting and also the setting of related machines.

Follow-up work

Soon after the first publication of our results, there has been follow-up work that extends our methods. Kurpisz et al. [2013] presented competitive-ratio approximation schemes for minimizing the makespan on unrelated machines (in the non-preemptive case), for the job shop problem, and for scheduling on a single machine with delivery times. Chen et al. [2015] presented such a scheme for the problem of scheduling jobs one by one on unrelated machines, the objective function being to minimize the makespan or the L_p-norm of the machine loads. For the case of identical machines and the makespan objective they even bound the running time by a polynomial in the number of machines. Independently, for the same online model Megow and Wiese [2013] found a competitive-ratio approximation scheme for minimizing the makespan on identical and related machines. They do not provide a polynomial bound on the running time, but their construction is more compact than the one by Chen et al. [2015]. Finally, Mömke [2013] provided a competitive-ratio approximation scheme for the k-server problem in fixed finite metrics.

7.2 General simplifications and techniques

In this section, we discuss several transformations that simplify the input and reduce the structural complexity of online schedules for $(Pm|r_j, pmtn|\sum w_j C_j)$. Later, we outline how to adapt these for more general settings. Our construction combines several transformation techniques known for offline PTASs (see Afrati et al. [1999] and their references) and a new technique to subdivide an instance online into parts which can be handled separately.

We will use the terminology that *at* $1+\mathcal{O}(\varepsilon)$ *loss we can restrict* to instances or schedules with certain properties. This means that we lose at most a factor $1+\mathcal{O}(\varepsilon)$, as $\varepsilon \to 0$, by limiting our attention to those. We bound several relevant parameters by constants. If not stated differently, any mentioned constant depends only on ε and m.

To begin with, we use the standard simplification technique of *geometric rounding* introduced in Afrati et al. [1999]. For the sake of completeness, we prove the required properties in the following lemma.

Lemma 7.2.1. *At* $1+O(\varepsilon)$ *loss we can restrict to instances where all processing times, release dates, and weights are powers of* $1+\varepsilon$*, no job is released before time* $t=1$*, and* $r_j \geqslant \varepsilon \cdot p_j$ *for all jobs* j.

Proof. We prove that any given schedule can be adapted at $1+O(\varepsilon)$ loss such that the required properties can be assumed for the corresponding instance. Obviously, weights can be rounded up to the next power of $1+\varepsilon$ by increasing the cost of a given schedule by at most a factor $1+\varepsilon$.

Within a given preemptive schedule, the job volume of each job j is assigned to a set of time intervals on machines (such that their lengths sum up to p_j). Multiplying the boundary values of each interval by $(1+\varepsilon)$ results in time intervals assigned to each j with a total length of $(1+\varepsilon)p_j$. Hence, we get a feasible schedule even when rounding up all processing times to the next powers of $(1+\varepsilon)$. The total completion time does not increase by more than a factor of $(1+\varepsilon)$.

Consider now a schedule with rounded processing times. We again multiply each boundary value of the intervals by $(1+\varepsilon)$ and shift the processing volume of each job j to the latest possible time intervals within its assigned intervals. Then, job j completes at time $(1+\varepsilon)C_j$, and the earliest point in time at which j is processed increases from S_j to $S_j+\varepsilon p_j$. Hence, by losing at most a factor $(1+\varepsilon)$ we may assume that each job $j \in J$ has a release date $r_j \geqslant \varepsilon p_j$. If necessary, the parameters of all jobs can be scaled by some power of $(1+\varepsilon)$ such that the earliest release date is at least one, since jobs with $r_j = p_j = 0$ can be ignored.

With a similar reasoning, we can finally round at a loss of $(1+\varepsilon)$ each release date to the next power of $(1+\varepsilon)$. □

The geometric rounding procedure allows us to see intervals of the form $I_x := [R_x, R_{x+1}), x \in \mathbb{N}$ with $R_x := (1+\varepsilon)^x$ as atomic entities. Note that $|I_x| = \varepsilon \cdot R_x$. An online algorithm can define the corresponding schedule at the beginning of an interval since no further jobs are released until the next interval. Moreover, we make, at $(1+\varepsilon)$ loss, the simplifying assumption that each job j finishing within the interval I_x contributes $w_j \cdot R_{x+1}$ to the objective function, i.e., we pretend that j finishes exactly at time R_{x+1}.

7.2.1 Simplification within intervals

The goal of this section is to reduce the number of situations that can arise at the beginning of an interval. For each ε we identify constantly many relevant inputs per interval. And for a given input, we reduce the number of relevant algorithmic actions within each interval to a constant.

To do so, we use the techniques of *time-stretching* and partitioning jobs released at time R_x into *large and small* jobs, see Afrati et al. [1999]. In an online interpretation of time-stretching, we shift the work assigned to any interval I_x to the interval I_{x+1}. When doing this operation

once we speak of *one time-stretch*. This can be done at a loss of $1+\varepsilon$ and we obtain free space of size $\varepsilon \cdot |I_x|$ in each interval I_{x+1}. For each $x \geqslant 0$ we define

$$L_x = \left\{ j \in J \;\middle|\; r_j = R_x, p_j > \varepsilon^2 |I_x| \right\}$$

to be the set of *large* jobs released at R_x and

$$S_x = \left\{ j \in J \;\middle|\; r_j = R_x, p_j \leqslant \varepsilon^2 |I_x| \right\}$$

to be the set of *small* jobs released at R_x.

We first take care of the small jobs. Since they are small, there is a lot of flexibility in scheduling them. We show that we can, at a small loss, fix in advance the order in which the jobs in each S_x are processed. This enables us to group very small jobs to job *packs*. Treating each pack as a single job we then get a lower bound on the processing time of the released jobs.

We denote w_j/p_j as the *Smith ratio* of a job j. A non-increasing ordering by the Smith ratios is an ordering according to *Smith's rule* (Smith [1956]). We say that a job is *partially processed* at some point in time if it has been processed, but not yet completed. For a set of jobs J we define $p(J) := \sum_{j \in J} p_j$ and $w(J) := \sum_{j \in J} w_j$.

Lemma 7.2.2. *At $1+\varepsilon$ loss we can restrict to schedules such that for each interval I_x the small jobs scheduled within this interval are chosen by Smith's rule from the set $S_{\leqslant x} := \bigcup_{x' \leqslant x} S_{x'}$ and no small job is preempted or only partially processed at the end of an interval. Therefore, we can restrict to instances with $p(S_x) \leqslant m \cdot |I_x|$ for each interval I_x.*

Proof. Consider some schedule S and apply one time-stretch. The resulting schedule S' has a free space of $\varepsilon|I_x|$ units in each interval I_{x+1}, and its total cost is within a factor $(1+\varepsilon)$ of the cost of S. Remove now all small jobs from each interval I_{x+1}. Denote by $P_{x,i}$ the amount of removed processing time of small jobs from machine i in interval I_x of S' and let $P_x := \sum_i P_{x,i}$. We denote by A_x the set of removed small jobs from I_x in S'. Now, we apply for each $x \geqslant 0$ the following procedure consecutively. First, we remove all large jobs from the interval I_x. Denote those jobs by L_x. We now schedule the jobs in L_x on the first $|I_x|$ units of processing time machine-wise, that is, we fill the machines consecutively and with preemption until we run out of large jobs. By this procedure, we do not lose any cost and as $p(L_x) \leqslant m \cdot |I_x|$, we can fit the large jobs there. We refer to this type of schedule as a McNaughton-type schedule, see also McNaughton [1959]. We now have a number of $k \geqslant 0$ machines in I_x whose first $|I_x|$ units of processing time is completely filled and $m-k$ machines where this is not the case.

Now, fill the idle time of each machine $i = k+1, \ldots, m$ with unscheduled jobs from $S_{\leqslant x}$ chosen in the order of their Smith ratio without preemption until an amount of $P_{x,i}$ is achieved or no further small job is available. This procedure increases the processing time on the latter machines by at most $\varepsilon^2|I_x|$, the maximum size of small jobs, which is smaller than the created extra space of $\varepsilon|I_{x-1}|$ for sufficiently small ε. Denote by B_x the set of small jobs assigned to I_x in this step.

Inductively, we can prove that for each I_x the total weight of jobs from $S_{\leqslant x}$ that are completed before the end of I_x has not decreased compared to S'. For the first interval I_{x_1} with $P_{x_1} > 0$ it holds that $w(A_{x_1}) \leqslant w(B_{x_1})$ since the jobs of B_{x_1} are assigned according to Smith's rule and $p(B_{x_1}) \geqslant P_{x_1} = p(A_{x_1})$. This proves the base case since each job of B_{x_1} is completed before

the end of I_{x_1}. For the induction step we use that $w(A_{\leqslant x}) \leqslant w(B_{\leqslant x})$ with $A_{\leqslant x} := \bigcup_{x' \leqslant x} A_{x'}$ and $B_{\leqslant x}$ respectively for some x. Assume by contradiction $w(A_{\leqslant \tilde{x}}) > w(B_{\leqslant \tilde{x}})$ for the next $\tilde{x} > x$ with $P_{\tilde{x}} > 0$. This implies that

$$w(A_{\tilde{x}}) > w(B_{\leqslant x}) - w(A_{\leqslant x}) + w(B_{\tilde{x}}).$$

Intuitively, the total weight of jobs in $A_{\tilde{x}}$ is higher than the surplus weight gained in the earlier intervals together with the total weight of $B_{\tilde{x}}$. This contradicts the fact that we assigned for each $x' \leqslant \tilde{x}$ jobs with a total processing time of $p(B_{\leqslant x'}) \geqslant p(A_{\leqslant x'})$ via Smith's rule. Hence, we get $w(A_{\leqslant \tilde{x}}) \leqslant w(B_{\leqslant \tilde{x}})$. The induction hypothesis follows since each job in $B_{\tilde{x}}$ is completed before the end of $I_{\tilde{x}}$. Recall that we modified the objective function by rounding up the completion times of jobs to the end of the respective interval. Consequently, the observation on the completed weights before each interval end yields that the objective value is not increased by the described reassignment. Furthermore, the procedure ensures that no small job is preempted and that any small job finishes in the same interval where it started.

Since we can now assume that the jobs of each S_x are chosen to be scheduled within I_x in non-increasing order of their Smith ratios—completely and without preemption— we can conclude the last claim of the lemma. Note that the total processing time in interval I_x is $m|I_x|$. Pick the jobs of S_x in the respective order until the total processing time of picked jobs just exceeds $m|I_x|$. The remaining jobs of S_x cannot be scheduled within I_x and hence, by the above argument, we can safely move their release dates to R_{x+1}. □

Lemma 7.2.3. *At $1 + \mathcal{O}(\varepsilon)$ loss we can restrict to instances such that $p_j \geqslant \frac{\varepsilon^2}{4} \cdot |I_x|$ for each job $j \in S_x$. In these instances, the number of distinct processing times of each set S_x is bounded from above by $\log_{(1+\varepsilon)} 4$.*

Proof. We call a job $j \in S_x$ *tiny* if $p_j \leqslant \frac{\varepsilon^2}{4} \cdot |I_x|$. Let $T_x = \{j_1, j_2, ..., j_{|T_x|}\}$ denote all tiny jobs released at R_x. W.l.o.g. assume that they are ordered non-increasingly by their Smith ratios w_j/p_j. Let ℓ be the largest integer such that $\sum_{i=1}^{\ell} p_i \leqslant \frac{\varepsilon^2}{2} \cdot |I_x|$. We define the pack $P_x^1 := \{j_1, ..., j_\ell\}$. We define the processing time of pack P_x^1 to be $\sum_{i=1}^{\ell} p_i$ and its weight to be $\sum_{i=1}^{\ell} w_i$. We continue iteratively until we assigned all tiny jobs to packs. By definition of the processing time of tiny jobs, the processing time of all but possibly the last pack released at time R_x is in the interval $\left[\frac{\varepsilon^2}{4} \cdot |I_x|, \frac{\varepsilon^2}{2} \cdot |I_x|\right]$.

We apply one time-stretch. In any schedule that assigns small jobs according to Smith's rule to intervals there is for each I_x at most one partially processed job pack per machine from each of the previous release dates $R_{x'} \leqslant R_x$. Since $\sum_{x'<x} \varepsilon^2 |I_{x'}| \leqslant \varepsilon |I_x|$, we can schedule all of them in the newly created space. This also includes space to increase the processing time of the very last pack of each $S_{x'}$ to $\frac{\varepsilon^2}{4} \cdot |I_{x'}|$, if necessary. Therefore, we can enforce that at $1 + O(\varepsilon)$ loss all tiny jobs of the same pack are scheduled in the same interval on the same machine. Hence, we can treat each pack as a single job whose processing time and weight matches the respective values of the pack.

Finally, at $1 + O(\varepsilon)$ loss we can ensure that the processing times and weights of the new jobs (which still remain small) are powers of $1 + \varepsilon$. Consequently, the processing times of jobs in S_x are of the form $(1 + \varepsilon)^y$ within the following range:

$$\frac{e^3}{4} \cdot (1+\varepsilon)^x \leqslant (1+\varepsilon)^y \leqslant \varepsilon^2 |I_x| = \varepsilon^3 (1+\varepsilon)^x.$$

The number of integers y satisfying these inequalities is bounded from above by $\log_{(1+\varepsilon)} 4$. □

For the large jobs released at the beginning of an interval we obtain an upper bound on their length by their relation of release date and processing time by Lemma 7.2.1. This induces a constant upper bound on the number of occurring processing times.

Lemma 7.2.4. *The number of distinct processing times of jobs in each set* L_x *is bounded from above by* $4 \log_{(1+\varepsilon)} \varepsilon^{-1}$.

Proof. Let $j \in L_x$ be a large job released at R_x with processing time $p_j = (1+\varepsilon)^y > \varepsilon^2 |I_x|$ for some integer y. By Lemma 7.2.1, we know that $p_j \leqslant \frac{1}{\varepsilon} r_j = \frac{1}{\varepsilon}(1+\varepsilon)^x$ and hence,

$$\varepsilon^3 (1+\varepsilon)^x = \varepsilon^2 |I_x| < (1+\varepsilon)^y \leqslant \varepsilon^{-1} (1+\varepsilon)^x.$$

The number of integers y which satisfy the above inequalities is upper-bounded by the constant claimed in the lemma. □

We say that two large jobs are of the same *type* if they have the same processing time and the same release date. By an exchange argument, we can restrict ourselves without any loss to schedules in which at each point in time at most m large jobs of each type are partially scheduled. Since the amount of work that can be processed within each interval is bounded, the number of large jobs of the same type can also be bounded.

Lemma 7.2.5. *Without loss, we can restrict to instances with*

$$|L_x| \leqslant (m/\varepsilon^2 + m) 4 \log_{(1+\varepsilon)} \varepsilon^{-1}$$

for each set L_x.

Proof. Let $L_{x,p} \subseteq L_x$ denote the set of jobs in L_x with processing time p, i.e., they are of the same type. Since $p_j > \varepsilon^2 |I_x|$ for each job $j \in L_x$, at most $m/\varepsilon^2 + m$ jobs in $L_{x,p}$ can be started before I_{x+1}. By an exchange argument we can assume that they are among the $m/\varepsilon^2 + m$ jobs with the largest weight in $L_{x,p}$. Hence, the release date of all other jobs in $L_{x,p}$ can be moved to R_{x+1} without any cost. By Lemma 7.2.4, there are at most $4 \log_{(1+\varepsilon)} \frac{1}{\varepsilon}$ distinct processing times p of large jobs in L_x and, thus, the claim follows. □

As we simplified the objective function by pretending all jobs to complete at the end of an interval, the only information needed for computing the objective function value is the interval in which a job completes. The only part that has not been bounded yet is the amount that large jobs are processed within the single intervals.

Lemma 7.2.6. *There is a constant* $\mu \in \mathbb{N}$ *such that at* $1 + O(\varepsilon)$ *loss we can restrict to schedules such that at the end of each interval, each large job* j *is processed to an extent which is an integer multiple of* p_j/μ.

Proof. We apply one time-stretch and choose $\mu \in \mathbb{N}$ to be a constant such that $1/\mu$ is smaller than $\varepsilon^4/(8\log_{(1+\varepsilon)}\frac{1}{\varepsilon})$. Consider now an interval I_x and the jobs that are scheduled in I_x before the time-stretch. For each job $j \in L_{x'}, x' \leqslant x$ that was partially processed at time R_{x+1} we must extend the amount of time the job j is processed within the interval I_{x+1} by at most p_j/μ to achieve the stated property. Using Lemma 7.2.1 this value can be bounded by

$$\frac{p_j}{\mu} \leqslant \frac{R_{x'}}{\varepsilon} \cdot \frac{\varepsilon^4}{8\log_{(1+\varepsilon)}\varepsilon^{-1}} = \frac{\varepsilon^2|I_{x'}|}{2 \cdot 4\log_{(1+\varepsilon)}\varepsilon^{-1}}.$$

Recall that we can assume without any loss that at the end of each interval at most one large job per job type is partially processed on each machine. Since the number of job types of $L_{x'}$ is bounded by $4\log_{(1+\varepsilon)}\varepsilon^{-1}$ (compare Lemma 7.2.4), the space of $\varepsilon^2|I_{x'}|/2$ is sufficient to handle each job type of $L_{x'}$. Applying

$$\sum_{x'<x} \varepsilon^2|I_{x'}| \leqslant \varepsilon|I_x|,$$

we see that this amount of free space was created by the time-stretch. □

To summarize, we get the following simplifications within intervals concerning input and scheduling decisions. Recall that we can calculate the value of μ that depends only on ε. Furthermore, we define Δ to be a constant upper bound on the number of released jobs in each interval, so we set

$$\Delta := \left\lceil \frac{4m}{\varepsilon^2} + (\frac{m}{\varepsilon^2} + m)4\log_{(1+\varepsilon)}\varepsilon^{-1} \right\rceil.$$

Corollary 7.2.7. *At $1+\mathcal{O}(\varepsilon)$ loss we can assume that for each interval I_x,*

1. *for each job j released at time R_x there is some integer k such that j has a processing time of*

$$p_j = (1+\varepsilon)^k \in [\frac{\varepsilon^3}{4}R_x, \frac{1}{\varepsilon}R_x],$$

2. *there are at most $\log_{(1+\varepsilon)}(4/\varepsilon^4)$ distinct processing time values of jobs released at R_x,*
3. *at most Δ jobs are released at R_x,*
4. *each small job starting in I_x completes in I_x without preemption, and*
5. *at the end of I_x, each job j is processed to an extent of $\ell_{x,j} \cdot p_j/\mu$ for some $\ell_{x,j} \in \{0,\dots,\mu\}$.*

7.2.2 Irrelevant history

The schedule for an interval returned by an online algorithm may depend on the set of currently unfinished jobs and possibly the entire schedule computed so far. In the remainder of this section we show why we can assume that an online algorithm only takes a finite amount of history into account when taking its decisions, namely, the jobs with relatively large weight released in the last constantly many intervals.

Firstly, we show that we may assume that each job completes within constantly many intervals after its release.

Lemma 7.2.8. *There is a constant* s *such that at* $1+\mathcal{O}(\varepsilon)$ *loss we can restrict to schedules such that for each interval* I_x *there is a subinterval of* I_{x+s-1} *which is large enough to process all jobs released at* R_x *and during which only those jobs are executed. We call this subinterval the safety net of interval* I_x*. We can assume that each job released at* R_x *finishes before time* R_{x+s}*.*

Proof. By using Lemma 7.2.1, Lemma 7.2.2, and Lemma 7.2.5 we get

$$\begin{aligned} p(S_x) + p(L_x) &\leqslant m \cdot |I_x| + (m/\varepsilon^2 + m) \cdot \cdot \varepsilon^{-1} (1+\varepsilon)^x \cdot 4 \log_{(1+\varepsilon)} \varepsilon^{-1} \\ &\leqslant m \cdot (1+\varepsilon)^x \left(\varepsilon + 8\varepsilon^{-3} \log_{(1+\varepsilon)} \varepsilon^{-1} \right) \\ &= \varepsilon \cdot |I_{x+s-2}| \end{aligned}$$

for a suitable constant s, depending on ε and m. Stretching time once, we gain enough free space at the end of each interval I_{x+s-1} to establish the safety net for each job set $p(S_x) + p(L_x)$. □

Given the bound on the number of intervals between release and completion times of jobs, we partition the time horizon into periods such that no job is "alive" for more than two periods. For each integer $k \geqslant 0$, we define *period* Q_k to consist of the s consecutive intervals $I_{k\cdot s}, \ldots, I_{(k+1)\cdot s-1}$. We add an artificial period Q_{-1} for the interval $[0,1)$ in which no job is released. Hence, we can assume by Lemma 7.2.8 that each job released in period Q_k is completed by the end of period Q_{k+1}. For ease of notation, we will treat a period Q as the set of jobs released in that period. For a set of jobs J we denote by

$$rw(J) := \sum_{j \in J} r_j w_j$$

their *release weight*. Note that $rw(J)$ forms a lower bound on the quantity that these jobs must contribute to the objective value in *any* schedule. Due to Lemma 7.2.8, we also obtain an upper bound of $(1+\varepsilon)^s \cdot rw(J)$ for the latter quantity.

We will now determine at the end of each period how important its released jobs are compared to the previous periods. If the job weights released in a series of periods grow large enough from period to period, we will see that the overall objective value is dominated by the contribution of the constantly many last periods. If otherwise the job weights of a period are too low compared to the preceding ones and its jobs can be moved to their safety net with a small loss, we will see that the following periods can be treated independently. To this end, we define that $p > 0$ consecutive periods $Q_k, \ldots, Q_{k+p-1}$ are a *sequence of significant periods* if

$$rw(Q_{k+\ell}) > \frac{\varepsilon}{(1+\varepsilon)^s} \cdot \sum_{i=0}^{\ell-1} rw(Q_{k+i})$$

for each $\ell = 1, \ldots, p-1$. This implies exponential growth for the series of partial sums of release weights and we can prove the dominance of a few of the last periods.

Lemma 7.2.9. *There is a constant* K *such that for each sequence* $Q_k, Q_{k+1}, \ldots, Q_{k+p-1}$ *of significant periods, the following inequality is valid:*

$$\sum_{i=0}^{p-K-1} (1+\varepsilon)^s \, rw(Q_{k+i}) \leqslant \varepsilon \cdot \sum_{i=p-K}^{p-1} rw(Q_{k+i}).$$

Proof. Let $\delta := \varepsilon(1+\varepsilon)^{-s}$. Since we consider a sequence of significant periods, we get

$$\mathrm{rw}(Q_{k+\ell}) > \delta \cdot \sum_{i=0}^{\ell-1} \mathrm{rw}(Q_{k+i}) \qquad \forall \ell = 1, \ldots, p-1.$$

This implies that

$$(1+\delta)\,\mathrm{rw}(Q_{k+\ell}) > \delta \cdot \sum_{i=0}^{\ell} \mathrm{rw}(Q_{k+i}) \qquad \forall \ell = 1, \ldots, p-1.$$

Hence, we get

$$\frac{\sum_{i=0}^{\ell-1} \mathrm{rw}(Q_{k+i})}{\sum_{i=0}^{\ell} \mathrm{rw}(Q_{k+i})} = 1 - \frac{\mathrm{rw}(Q_{k+\ell})}{\sum_{i=0}^{\ell} \mathrm{rw}(Q_{k+i})} < 1 - \frac{\delta}{1+\delta} = \frac{1}{1+\delta} < 1 \qquad \forall \ell = 1, \ldots, p-1$$

which implies

$$\sum_{i=0}^{\ell-1} \mathrm{rw}(Q_{k+i}) < \frac{1}{1+\delta} \sum_{i=0}^{\ell} \mathrm{rw}(Q_{k+i}) \qquad \forall \ell = 1, \ldots, p-1. \tag{7.2.1}$$

In other words, if we remove $Q_{k+\ell}$ from $\bigcup_{i=0}^{\ell} Q_{k+i}$, the total release weight of the set decreases by a factor of at least $1/(1+\delta) < 1$. Recursively applying (7.2.1) we get for any K

$$\begin{aligned}\sum_{i=0}^{p-1-K} \mathrm{rw}(Q_{k+i}) &< \frac{1}{(1+\delta)^K} \sum_{i=0}^{p-1} \mathrm{rw}(Q_{k+i}) \\ &= \frac{1}{(1+\delta)^K} \left(\sum_{i=0}^{p-K-1} \mathrm{rw}(Q_{k+i}) + \sum_{i=p-K}^{p-1} \mathrm{rw}(Q_{k+i}) \right)\end{aligned}$$

and hence

$$\left(1 - \left(\frac{1}{(1+\delta)^K}\right)\right) \sum_{i=0}^{p-K-1} \mathrm{rw}(Q_{k+i}) < \frac{1}{(1+\delta)^K} \sum_{i=p-K}^{p-1} \mathrm{rw}(Q_{k+i}).$$

To ensure that

$$(1+\varepsilon)^s \sum_{i=0}^{p-K-1} \mathrm{rw}(Q_{k+i}) < \varepsilon \cdot \sum_{i=p-K}^{p-1} \mathrm{rw}(Q_{k+i}),$$

we choose K sufficiently large such that

$$\frac{(1+\delta)^{-K}}{1 - \frac{1}{(1+\delta)^K}} = \frac{1}{(1+\delta)^K - 1} < \delta = \frac{\varepsilon}{(1+\varepsilon)^s}.$$

Hence, K is some constant integer larger than $\log_{(1+\delta)}(1/\delta + 1)$ and depends only on ε. □

Using the safety net (Lemma 7.2.8), the above lemma implies that an ε-fraction of the weighted completion time of the last $K-1$ periods of a sequence of significant periods yields an upper bound on the weighted completion time of the previous periods of this sequence. Therefore, the objective value is essentially dominated by the contribution of the last $K-1$ periods. We will need this later to show that at $1+\mathcal{O}(\varepsilon)$ loss we can assume that an online algorithm bases its decisions only on a constant amount of information.

To that end, we consider a sequence of significant periods $Q_k,\dots,Q_{k+p-1}$ and an interval I_x of period Q_{k+p} where the newly released jobs are just revealed to an online algorithm. Recall that the online algorithm does not know the release weight of the current period Q_{k+p} unless I_x is the last interval of Q_{k+p}. Nevertheless, we know by Lemma 7.2.9 that the costs of $Q_k,\dots,Q_{k+p-1}$ are dominated by the last $K-1$ periods. Hence, the costs until interval I_x are dominated by the last important $\Gamma := Ks$ intervals including I_x. In addition to the jobs that have been released very early, also the jobs with very small weight in comparison to at least one other job can be almost neglected for the total costs. Therefore, we partition the jobs into relevant and irrelevant jobs using these two criteria.

Definition 7.2.10. Let J be a set of jobs. A job $j \in J$ with $r_j \leqslant R_x$ is called *recent at time* R_x if $R_{x-\Gamma} \leqslant r_j$. Otherwise, it is called *old at time* R_x.

Job j is called *dominated at time* R_x if it is dominated at time R_{x-1} or if there is a job $j' \in J$ being recent and not dominated at time R_x such that

$$w_j < \frac{\varepsilon}{\Delta\cdot\Gamma\cdot(1+\varepsilon)^{\Gamma+s}} w_{j'}.$$

Now, job j is *irrelevant at time* R_x if it is old or dominated at time R_x and otherwise *relevant at time* R_x. We denote the respective subsets of a job set J by $\mathrm{Rec}_x(J)$, $\mathrm{Old}_x(J)$, $\mathrm{Dom}_x(J)$, $\mathrm{Ir}_x(J)$, and $\mathrm{Rel}_x(J)$.

The subsequent lemma states that the irrelevant jobs can almost be ignored for the objective value of a schedule even when scheduled in their safety nets. This implies that we can restrict at $1+O(\varepsilon)$ loss to online algorithms which schedule the remaining part of a job in its safety net, once it has become irrelevant.

Lemma 7.2.11. *Let $Q_k,\dots,Q_{k+p-1}$ be a sequence of significant periods and let J be the jobs of $Q_k,\dots,Q_{k+p}$ released until the beginning of an interval $I_x \in Q_{k+p}$. Then*

$$(1+\varepsilon)^s \,\mathrm{rw}(\mathrm{Ir}_x(J)) \leqslant 6\varepsilon\cdot\mathrm{rw}(\mathrm{Rel}_x(J)).$$

Proof. By definition, an irrelevant job at time R_x is either old or dominated. Hence, $\mathrm{Ir}_x(J)$ is the disjoint union of $\mathrm{Old}_x(J)$ and $\mathrm{Dom}_x(J)\cap\mathrm{Rec}_x(J)$.

We start by giving a bound on the recent but dominated jobs and define

$$\mathrm{rw}_{\max} := \max\{\mathrm{rw}(j) \mid j \in \mathrm{Rel}_x(J)\cup\mathrm{Old}_x(J)\}.$$

Consider a job $j\in J$ that is dominated and recent at time R_x. By definition, there is a time $r_j \leqslant R_{x'} \leqslant R_x$ at which another job j' that is relevant at time $R_{x'}$ dominates j. Choose $R_{x'}$ to be as late as possible. As j' is recent at time $R_{x'}$, we get by $R_{x'}(1+\varepsilon)^{-\Gamma} \leqslant r_{j'}$ that $r_j \leqslant R_{x'} \leqslant$

$(1+\varepsilon)^{\Gamma} r_{j'}$. Because we have chosen $R_{x'}$ to be as late as possible, we can assume that j' is not dominated at time R_x. Hence, j' is either relevant or old at time R_x. Therefore, there is for each $j \in \mathrm{Dom}_x(J) \cap \mathrm{Rec}_x(J)$ a job j' in $\mathrm{Rel}_x(J) \cup \mathrm{Old}_x(J)$ such that

$$\begin{aligned} w_j r_j &\leqslant \frac{\varepsilon}{\Delta \cdot \Gamma \cdot (1+\varepsilon)^{\Gamma+s}} w_{j'} r_j \\ &\leqslant \frac{\varepsilon}{\Delta \cdot \Gamma \cdot (1+\varepsilon)^{\Gamma+s}} w_{j'} r_{j'} (1+\varepsilon)^{\Gamma} \\ &\leqslant \frac{\varepsilon}{\Delta \cdot \Gamma \cdot (1+\varepsilon)^{s}} \, rw_{\max} . \end{aligned}$$

Since at most Δ jobs are released at the beginning of each interval (Corollary 7.2.7) and $rw_{\max}$ is the release weight of a job in $\mathrm{Rel}_x(J) \cup \mathrm{Old}_x(J)$ we get:

$$\begin{aligned} (1+\varepsilon)^{s} \, rw(\mathrm{Dom}_x(J) \cap \mathrm{Rec}_x(J)) &= (1+\varepsilon)^{s} \sum_{j \in \mathrm{Dom}_x(J) \cap \mathrm{Rec}_x(J)} w_j r_j \\ &\leqslant (1+\varepsilon)^{s} \Delta \cdot \Gamma \frac{\varepsilon}{\Delta \cdot \Gamma \cdot (1+\varepsilon)^{s}} \, rw_{\max} \\ &= \varepsilon \cdot rw_{\max} \\ &\leqslant \varepsilon \, (rw(\mathrm{Rel}_x(J)) + rw(\mathrm{Old}_x(J))) . \end{aligned}$$

Moreover, Lemma 7.2.9 implies for the old jobs at time R_x that

$$\begin{aligned} (1+\varepsilon)^{s} \, rw(\mathrm{Old}_x(J)) &\leqslant \varepsilon \cdot rw(\mathrm{Rec}_x(J)) \\ &= \varepsilon \cdot (rw(\mathrm{Rel}_x(J)) + rw(\mathrm{Dom}_x(J) \cap \mathrm{Rec}_x(J)) . \end{aligned}$$

Combining all these arguments yields

$$\begin{aligned} (1+\varepsilon)^{s} \, rw(\mathrm{Ir}_x(J)) &= (1+\varepsilon)^{s} \, (rw(\mathrm{Old}_x(J)) + rw(\mathrm{Dom}_x(J) \cap \mathrm{Rec}_x(J)) \\ &\leqslant \varepsilon \cdot rw(\mathrm{Rel}_x(J)) + 2\varepsilon \cdot (rw(\mathrm{Rel}_x(J)) + rw(\mathrm{Old}_x(J))) \\ &\leqslant 3\varepsilon \cdot rw(\mathrm{Rel}_x(J)) + 2\varepsilon \cdot rw(\mathrm{Ir}_x(J)). \end{aligned}$$

With $\varepsilon < \frac{1}{3}$ we obtain the bound $(1+\varepsilon)^{s} \, rw(\mathrm{Ir}_x(J)) \leqslant 6\varepsilon \cdot rw(\mathrm{Rel}_x(J))$. □

With the preceding lemmas, we have identified for each point in time a subset of jobs that is relevant at this time, but only under the assumption that we consider a sequence of significant periods $Q_k, \ldots, Q_{k+p-1}$. We now consider the case where the immediately following period Q_{k+p} is not significant. We then know that

$$(1+\varepsilon)^{s} \, rw(Q_{k+p}) \leqslant \varepsilon \cdot \sum_{i=0}^{p-1} rw(Q_{k+i}).$$

Hence, completing all unfinished jobs of Q_{k+p} in their safety nets costs only an ε-fraction of the costs caused by the preceding significant periods. And since an online algorithm is allowed to preempt these jobs, it can now schedule the succeeding periods independently. However, in

the following we will state a more general result which will be useful for eventually proving that online algorithms can forget all irrelevant jobs.

We define the consecutive periods $Q_k, \dots, Q_{k+p}$, for some $p > 0$, to be a *part* if the sequence $Q_k, \dots, Q_{k+p-1}$ consists only of significant periods and

$$(1+\varepsilon)^s \operatorname{rw}(Q_{k+p}) \leqslant 8\varepsilon \cdot \sum_{i=0}^{p-1} \operatorname{rw}(Q_{k+i}).$$

In this case, we call Q_{k+p} *insignificant*, even though it is possible that $Q_k, \dots, Q_{k+p}$ is a sequence of significant periods.

We now consider a partitioning of a given instance $\mathcal{I}$ into parts $P_i, i = 0, ..., \ell$, and denote the insignificant period of each part P_i by $Q_{a_{(i+1)}}$. With $a_0 := -1$ each part P_i consists of the periods $Q_{a_i+1}, ..., Q_{a_{(i+1)}}, i = 0, ..., \ell$. Again, we identify with P_i all jobs released in this part.

With the possibility to preempt jobs, we now treat each part P_i of a partition as a separate instance that we feed into a given online algorithm. For the final output, we concatenate the computed schedules for the different parts. By the following lemma, it then suffices to bound $\mathsf{A}(P_i)/\mathrm{OPT}(P_i)$ for each part P_i.

Lemma 7.2.12. *At $1 + O(\varepsilon)$ loss we can restrict to instances which consist of only one part.*

Proof. Consider an online algorithm A, some instance $\mathcal{I}$ and a partition of $\mathcal{I}$ into parts $P_i, i = 0, ..., \ell - 1$. We define an adapted version A' that applies $\mathsf{A}(P_i)$ to each part P_i and moves all jobs of P_i that are unfinished at the end of P_i to their respective safety nets. Recall that due to Lemma 7.2.8, these unfinished jobs must be released within the last period $Q_{a_{(i+1)}}$ of part P_i and this period is insignificant. Hence, these jobs contribute at most

$$\sum_{i=1}^{\ell+1} (1+\varepsilon)^s \operatorname{rw}(Q_{a_i}) \leqslant \sum_{i=1}^{\ell+1} 8\varepsilon \cdot \sum_{p=a_{i-1}+1}^{a_i-1} \operatorname{rw}(Q_p) \leqslant \sum_{i=0}^{\ell} O(\varepsilon) \cdot \mathrm{OPT}(P_i)$$

to the objective value. Therefore, we get that

$$\mathsf{A}'(\mathcal{I}) \leqslant (1 + \mathcal{O}(\varepsilon)) \sum_{i=0}^{\ell} \mathsf{A}(P_i).$$

Applying $\sum_{i=0}^{\ell} \mathrm{OPT}(P_i) \leqslant \mathrm{OPT}(\mathcal{I})$ we can bound the competitive ratio of A' as follows:

$$\frac{\mathsf{A}'(\mathcal{I})}{\mathrm{OPT}(\mathcal{I})} \leqslant (1 + \mathcal{O}(\varepsilon)) \frac{\sum_{i=0}^{\ell} \mathsf{A}(P_i)}{\sum_{i=0}^{\ell} \mathrm{OPT}(P_i)} \leqslant (1 + \mathcal{O}(\varepsilon)) \max_{i=1,\dots,\ell} \frac{\mathsf{A}(P_i)}{\mathrm{OPT}(P_i)}.$$

□

Note that there might be different partitions of one instance into parts since it is possible that a part is at the same time a sequence of significant periods. In the following section, we will consider online algorithms that work in principle only on relevant jobs while forgetting irrelevant jobs after they have been moved to their safety net. Hence, we need a partition into parts that can be determined regardless of the irrelevant jobs. To that end, we define $x(a) := (a+1)s - 1$ for a period Q_a to be the index of the last interval of period Q_a.

Lemma 7.2.13. *For a given instance $\mathcal{I}$, let $a_0 := -1$ and let $a_1 < ... < a_{\ell+1}$ be all indices such that*

$$(1+\varepsilon)^s \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_{a_i}) \leqslant \left(\varepsilon + \frac{6\varepsilon^2}{(1+\varepsilon)^s}\right) \sum_{p=a_{i-1}+1}^{a_i-1} \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_p), \tag{7.2.2}$$

where each P_{i-1} consists of all periods

$$Q_{a_{(i-1)}+1}, \ldots, Q_{a_i}$$

for $i = 1, \ldots, \ell+1$. Then, each P_i is a part of $\mathcal{I}$ for $i = 0, \ldots, \ell$.

Proof. We first prove that the periods between Q_{a_i} and $Q_{a_{(i+1)}}$ build a sequence of significant periods. To that end, we can inductively apply Lemma 7.2.11 and get for each $a_{i-1} < a < a_i$:

$$\begin{aligned}
(1+\varepsilon)^s \operatorname{rw}(Q_a) &\geqslant (1+\varepsilon)^s \operatorname{rw}(\operatorname{Rel}_{x(a)}(P_{i-1}) \cap Q_a) \\
(7.2.2) &> \varepsilon \cdot \left(1 + \frac{6\varepsilon}{(1+\varepsilon)^s}\right) \sum_{p=a_{i-1}+1}^{a-1} \operatorname{rw}(\operatorname{Rel}_{x(a)}(P_{i-1}) \cap Q_p) \\
&= \varepsilon \cdot \left(1 + \frac{6\varepsilon}{(1+\varepsilon)^s}\right) \operatorname{rw}(\operatorname{Rel}_{x(a)}(P_{i-1}) \cap \bigcup_{p=a_{i-1}+1}^{a-1} Q_p \\
\text{(Lemma 7.2.11)} &\geqslant \varepsilon \cdot \sum_{p=a_{i-1}+1}^{a-1} \operatorname{rw}(Q_p).
\end{aligned}$$

That is why we can now apply Lemma 7.2.11 to each complete P_{i-1} and get that each Q_{a_i} is insignificant. Let $\mathcal{A}$ denote the interval $[a_{i-1}+1, a_i-1]$. We get:

$$\begin{aligned}
(1+\varepsilon)^s \operatorname{rw}(Q_{a_i}) &= (1+\varepsilon)^s \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_{a_i}) + (1+\varepsilon)^s \operatorname{rw}(\operatorname{Ir}_{x(a_i)}(P_{i-1}) \cap Q_{a_i}) \\
&\leqslant \left(\varepsilon + \frac{6\varepsilon^2}{(1+\varepsilon)^s}\right) \sum_{p \in \mathcal{A}} \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_p) + 6\varepsilon \cdot \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1})) \\
&\leqslant \varepsilon \cdot \left(1 + \frac{6\varepsilon}{(1+\varepsilon)^s} + 6\right) \cdot \sum_{p \in \mathcal{A}} \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_p) + \\
&\qquad 6\varepsilon \cdot \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_{a_i}) \\
&\leqslant \varepsilon \cdot \left(7 + \frac{6\varepsilon}{(1+\varepsilon)^s}\right) \cdot \sum_{p \in \mathcal{A}} \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_p) + \\
&\qquad \frac{6\varepsilon^2}{(1+\varepsilon)^s} \left(1 + \frac{6\varepsilon}{(1+\varepsilon)^s}\right) \cdot \sum_{p \in \mathcal{A}} \operatorname{rw}(\operatorname{Rel}_{x(a_i)}(P_{i-1}) \cap Q_p) \\
&\leqslant 8\varepsilon \cdot \sum_{p \in \mathcal{A}} \operatorname{rw}(Q_p).
\end{aligned}$$

□

We conclude this section by summarizing those consequences of our considerations that we need in the following section. To that end, we denote the maximum ratio between weights of relevant jobs at any point in time by $W = \frac{\Delta \cdot \Gamma \cdot (1+\varepsilon)^{\Gamma+s}}{\varepsilon}$. Recall that the values of $\mu, \Delta, s, K, \Gamma$, and hence W depend only on ε and m. Furthermore, for some given schedule we denote by $o_{x,j}$ the amount of a job j that is processed within an interval I_x.

Corollary 7.2.14. *At $1 + \mathcal{O}(\varepsilon)$-loss we can assume for each instance $\mathcal{I}$ with job set J and each interval I_x that*

1. $p_j \in \{(1+\varepsilon)^k \mid \frac{\varepsilon^3}{4}(1+\varepsilon)^{-\Gamma} R_x \leqslant (1+\varepsilon)^k \leqslant \frac{1}{\varepsilon} R_x\}$ *for each* $j \in \mathrm{Rel}_x(J)$,
2. $r_j \in \{(1+\varepsilon)^k \mid (1+\varepsilon)^{-\Gamma} R_x \leqslant (1+\varepsilon)^k \leqslant R_x\}$ *for each* $j \in \mathrm{Rel}_x(J)$,
3. $w_j \in \{(1+\varepsilon)^k \mid w_x \leqslant (1+\varepsilon)^k \leqslant W \cdot w_x\}$ *for some value* w_x *and each* $j \in \mathrm{Rel}_x(J)$,
4. $o_{x,j} \in \{\ell \cdot p_j/\mu \mid \ell \in \{0, \dots, \mu\}\}$ *for each* $j \in \mathrm{Rel}_x(J)$,
5. *the cardinality of* $\mathrm{Rel}_x(J)$ *is bounded by* $\Gamma \cdot \Delta$, *and*
6. $\sum_{j \in \mathrm{Ir}_x(J)} w_j C_j \leqslant O(\varepsilon) \cdot \mathit{OPT}(\mathrm{Rel}_x(J))$.

Note that the respective sets of relevant processing times, release dates, weights and processing time fractions have constant size.

7.3 Abstraction of online algorithms

In this section we show how to construct a competitive-ratio approximation scheme based on the simplifications of Section 7.2. To do so, we restrict ourselves to such simplified instances and schedules. The key idea is to characterize the behavior of an online algorithm by a map: For each interval, the map gets as input the schedule computed so far and all information about the currently unfinished jobs. Based on this information, the map outputs how to schedule the available jobs within this interval.

More precisely, we define the input by a *configuration* and the output by an *interval-schedule*.

Definition 7.3.1. An *interval-schedule* S for an interval I_x is defined by

- the index x of the interval,
- a set of jobs $J(S)$ available for processing in I_x together with the properties r_j, p_j, w_j of each job $j \in J(S)$ and its already finished part $o_j < p_j$ up to R_x,
- for each job $j \in J(S)$ the information whether j is relevant at time R_x, and
- for each job $j \in J(S)$ and each machine i a value q_{ij} specifying for how long j is processed by S on machine i during I_x.

An interval-schedule is called *feasible* if there is a feasible schedule in which all values for the jobs of $J(S)$ in the interval-schedule fit to the corresponding values of the schedule within the interval I_x. Denote the set of feasible interval-schedules as $\mathcal{S}$.

Definition 7.3.2. A *configuration* C for an interval I_x consists of

- the index x of the interval,
- a set of jobs $J(C)$ released up to time R_x together with the properties r_j, p_j, w_j, o_j of each job $j \in J(C)$,
- an interval-schedule for each interval $I_{x'}$ with $x' < x$.

A configuration is called *feasible* if there is a feasible schedule in which all values for the jobs of $J(C)$ in the configuration (including each interval-schedule) fit to the corresponding values of the schedule. The set of all feasible configurations, respecting the adaptations of Section 7.2, is denoted by $\mathcal{C}$. An *end-configuration* is a feasible configuration C for an interval I_x such that at time R_x, and not earlier, all jobs have been released and all relevant jobs have completely finished processing.

We say that an interval-schedule S is *feasible for a configuration* C if the set of jobs in $J(C)$ which are unfinished at time R_x matches the set $J(S)$ with respect to release dates, total and remaining processing time, weight and relevance of the jobs.

Instead of online algorithms we work from now on with *algorithm maps*, which are defined as functions $f\colon \mathcal{C} \to \mathcal{S}$. An algorithm map determines a schedule $f(\mathcal{I})$ for a given scheduling instance $\mathcal{I}$ by iteratively applying f to the corresponding configurations. W.l.o.g. we consider only algorithm maps f such that $f(C)$ is feasible for each configuration C and $f(\mathcal{I})$ is feasible for each instance I. Call these algorithm maps *feasible*. Like for online algorithms, we define the competitive ratio ρ_f of an algorithm map f by $\rho_f := \max_{\mathcal{I}} f(\mathcal{I})/\mathrm{OPT}(\mathcal{I})$. Due to the following observation, algorithm maps are a natural generalization of online algorithms.

Proposition 7.3.3. *For each online algorithm A there is an algorithm map f_A such that when A is in configuration $C \in \mathcal{C}$ at the beginning of an interval I_x, algorithm A schedules the jobs according to $f_A(C)$.*

Recall that we restrict our attention to algorithm maps describing online algorithms which obey the simplifications introduced in Section 7.2. The essence of such online algorithms are the decisions for the relevant jobs. To this end, we define equivalence classes for configurations and for interval-schedules. Intuitively, two interval-schedules (configurations) are equivalent if we can obtain one from the other by scalar multiplication with the same value, while ignoring the irrelevant jobs.

Definition 7.3.4. Let S, S' be two feasible interval-schedules for two intervals $I_x, I_{x'}$. Let $\sigma\colon \tilde{J} \to \tilde{J}'$ be a bijection from a subset $\tilde{J} \subseteq J(S)$ to a subset $\tilde{J}' \subseteq J(S')$ and let y an integer. The interval-schedules S, S' are called (σ, y)-*equivalent* if for all $j \in \tilde{J}$ and $i = 1, \ldots, m$, the following conditions are satisfied:

- $r_{\sigma(j)} = r_j(1+\varepsilon)^{x'-x}$,

- $p_{\sigma(j)} = p_j(1+\varepsilon)^{x'-x}$,
- $o_{\sigma(j)} = o_j(1+\varepsilon)^{x'-x}$,
- $q_{i\sigma(j)} = q_{ij}(1+\varepsilon)^{x'-x}$ and
- $w_{\sigma(j)} = w_j\,(1+\varepsilon)^{y}$.

Denote by $J_{Rel}(S) \subseteq J(S)$ and $J_{Rel}(S') \subseteq J(S')$ the jobs of $J(S)$ relevant at time R_x and of $J(S')$ relevant at time $R_{x'}$. The interval-schedules S, S' are *equivalent* (denoted by $S \sim S'$) if a bijection $\sigma\colon J_{Rel}(S) \to J_{Rel}(S')$ and an integer y exist such that they are (σ, y)-equivalent.

Definition 7.3.5. Let C, C' be two feasible configurations for two intervals $I_x, I_{x'}$. Denote by $J_{Rel}(C), J_{Rel}(C')$ the jobs which are relevant at times $R_x, R_{x'}$ in C, C', respectively. The configurations C, C' are *equivalent* (denoted by $C \sim C'$) if there is a bijection $\sigma\colon J_{Rel}(C) \to J_{Rel}(C')$ and an integer y such that for all $j \in J_{Rel}(C)$

- $r_{\sigma(j)} = r_j(1+\varepsilon)^{x'-x}$,
- $p_{\sigma(j)} = p_j(1+\varepsilon)^{x'-x}$,
- $o_{\sigma(j)} = o_j(1+\varepsilon)^{x'-x}$ and
- $w_{\sigma(j)} = w_j\,(1+\varepsilon)^{y}$

and if the interval-schedules of I_{x-k} and $I_{x'-k}$ are (σ, y)-equivalent for each $k \in \mathbb{N}$.

The restriction of equivalence to relevant jobs allows a reasonable measurement of the performance of end-configurations contained in the same equivalence class. On the one hand we get equal performance ratios for equivalent configurations. On the other hand we can approximate the actual competitive ratio of the complete solution since relevant jobs dominate the objective value. Consider therefore an end-configuration C. We denote the objective value of a subset $\tilde{J} \subseteq J(C)$ in the history of C by $val_C(\tilde{J})$. We further define $\rho(C) := val_C(J_{Rel}(C))/OPT(J_{Rel}(C))$ to be the achieved competitive ratio of C when restricted to the relevant jobs.

Lemma 7.3.6. *For each end-configuration $C \in \mathcal{C}$ we have that*

1. $(1+\mathcal{O}(\varepsilon))^{-1}\rho(C) \leqslant val_C(J(C))/OPT(J(C)) \leqslant (1+\mathcal{O}(\varepsilon)) \cdot \rho(C)$ *and*
2. $\rho(C) = \rho(C')$ *for any $C' \in \mathcal{C}$ with $C \sim C'$.*

Proof. The first property follows from Lemma 7.2.11 via

$$\frac{val_C(J(C))}{OPT(J(C))} \leqslant \frac{val_C(J(C))}{OPT(J_{Rel}(C))} \leqslant (1+\mathcal{O}(\varepsilon))\frac{val_C(J_{Rel}(C))}{OPT(J_{Rel}(C))} = (1+\mathcal{O}(\varepsilon))\rho(C).$$

Similarly, we get

$$\rho(C) = \frac{val_C(J_{Rel}(C))}{OPT(J_{Rel}(C))} \leqslant \frac{val_C(J(C))}{OPT(J_{Rel}(C))} \leqslant (1+\mathcal{O}(\varepsilon))\frac{val_C(J(C))}{OPT(J(C))}.$$

For the second property, denote the job weights and the resulting completion times of C (C') by w_j (w'_j) and C_j (C'_j). Since $C \sim C'$ there is an integer y and a bijection $\sigma\colon J_{Rel}(C) \to J_{Rel}(C')$ such that

$$\begin{aligned} val_{C'}(J_{Rel}(C')) &= \sum_{j \in J_{Rel}(C')} w'_j C'_j = \sum_{j \in J_{Rel}(C)} w'_{\sigma(j)} C'_{\sigma(j)} \\ &= \sum_{j \in J_{Rel}(C)} (1+\varepsilon)^{y} \, w_j \, (1+\varepsilon)^{x'-x} \, C_j \\ &= (1+\varepsilon)^{y+x'-x} \, val_C(J_{Rel}(C)). \end{aligned}$$

If we transform the solution $OPT(J_{Rel}(C))$ by scaling weights by a factor of $(1+\varepsilon)^y$ and scaling all time values by a factor of $(1+\varepsilon)^{x'-x}$ using the map σ, we get a solution for $J_{Rel}(C')$ with value

$$(1+\varepsilon)^{y+x'-x} \, OPT(J_{Rel}(C)).$$

Vice versa, applying the inverse transformation to the solution $OPT(J_{Rel}(C'))$ yields a solution for $J_{Rel}(C)$ with value

$$(1+\varepsilon)^{x-x'-y} \, OPT(J_{Rel}(C')).$$

By

$$\begin{aligned} OPT(J_{Rel}(C')) &\leqslant (1+\varepsilon)^{y+x'-x} \, OPT(J_{Rel}(C)) \\ &\leqslant (1+\varepsilon)^{y+x'-x} \, (1+\varepsilon)^{x-x'-y} \, OPT(J_{Rel}(C')) = OPT(J_{Rel}(C')) \end{aligned}$$

we can conclude equality. This finally yields

$$\rho(C') = \frac{val_{C'}(J_{Rel}(C'))}{OPT(J_{Rel}(C'))} = \frac{(1+\varepsilon)^{y+x'-x} \, val_C(J_{Rel}(C))}{(1+\varepsilon)^{y+x'-x} \, OPT(J_{Rel}(C))} = \rho(C).$$

□

The following lemma shows that we can restrict the set of algorithm maps under consideration to those which treat equivalent configurations equivalently. We call algorithm maps obeying this condition in addition to the restrictions of Section 7.2 *simplified algorithm maps*. A configuration C is called *realistic for an algorithm map* f if there is an instance $\mathcal{I}$ such that if f processes $\mathcal{I}$ then at time R_x it is in configuration C.

Lemma 7.3.7. *At* $1 + O(\varepsilon)$ *loss we can restrict to algorithm maps* f *such that* $f(C) \sim f(C')$ *for any two equivalent configurations* C, C'.

Proof. Let f be an algorithm map. We now construct a new algorithm map $\bar{f}$ which is simplified and has competitive ratio $\rho_{\bar{f}} \leqslant (1 + \mathcal{O}(\varepsilon))\rho_f$ almost as good as f. Therefore, we pick for each equivalence class $\mathcal{C}_e \in \mathcal{C}/_\sim$ of the set of configurations a representative C_e (i.e. $[C_e] = \mathcal{C}_e$) which is realistic for f. For each configuration $C \in [C_e]$ which is equivalent to C_e with bijection σ and integer y, we define $\bar{f}$ by setting $\bar{f}(C)$ to be the interval-schedule for C which is (σ, y)-equivalent to $f(C_e)$. One can show by induction that $\bar{f}$ is always in a configuration such that an equivalent

configuration is realistic for f. Hence, equivalence classes without realistic configurations for f are not relevant.

Consider now an instance $\bar{\mathcal{I}}$. We show that there is an instance $\mathcal{I}$ such that

$$\bar{f}(\bar{\mathcal{I}})/\mathrm{OPT}(\bar{\mathcal{I}}) \leqslant (1+\mathcal{O}(\varepsilon))f(\mathcal{I})/\mathrm{OPT}(\mathcal{I})$$

which implies the claimed competitive ratio for $\bar{f}$. Let $\bar{C}$ for interval $I_{\bar{x}}$ be the end-configuration obtained when $\bar{f}$ is applied iteratively on $\bar{\mathcal{I}}$. Let C_e be the representative of the equivalence class of $\bar{C}$, which was chosen above and which is realistic for f (C_e is also an end-configuration). Therefore, there is an instance $\mathcal{I}$ such that C_e is reached at time R_x when f is applied on $\mathcal{I}$. Hence, by Lemma 7.3.6 and $C_e \sim \bar{C}$ we get that $\mathcal{I}$ is the required instance. □

Lemma 7.3.8. *There are only constantly many simplified algorithm maps. Each simplified algorithm map can be described using finite information.*

Proof. Assuming the simplifications introduced in Section 7.2 we can apply Corollary 7.2.14. Therefore, the domain of the algorithm maps under consideration contains only constantly many equivalence classes of configurations. Also, the target space contains only constantly many equivalence classes of interval-schedules. For an algorithm map f which obeys the restrictions of Section 7.2, the interval-schedule $f(C)$ is fully specified when knowing only C and the equivalence class which contains $f(C)$ (since the irrelevant jobs are moved to their safety net anyway). Since $f(C) \sim f(C')$ for a simplified algorithm map f if $C \sim C'$, we conclude that there are only constantly many simplified algorithm maps. Finally, each equivalence class of configurations and interval-schedules can be characterized using only finite information, and hence the same holds for each simplified algorithm map. □

The next lemma shows that up to a factor $1+\varepsilon$, worst case instances of simplified algorithm maps span only constantly many intervals. Using this property, we will show in the subsequent lemmas that the competitive ratio of a simplified algorithm map can be determined algorithmically up to a $1+\varepsilon$ factor.

Lemma 7.3.9. *There is a constant* E *such that for any instance* I *and any simplified algorithm map* f *there is a realistic end-configuration* $\tilde{C}$ *for an interval* $I_{\tilde{x}}$ *with* $\tilde{x} \leqslant E$ *which is equivalent to the corresponding end-configuration when* f *is applied to* I.

Proof. Consider a simplified algorithm map f. For each interval I_x, denote by $\mathcal{C}^f_x$ the set of realistic equivalence classes for I_x, i.e., the equivalence classes which have a realistic representative for I_x. Since there are constantly many equivalence classes and thus constantly many *sets* of equivalence classes, there must be a constant E independent of f such that $\mathcal{C}^f_{\bar{x}} = \mathcal{C}^f_{\bar{x}'}$ for some $\bar{x} < \bar{x}' \leqslant E$. Since f is simplified it can be shown by induction that $\mathcal{C}^f_{\bar{x}+k} = \mathcal{C}^f_{\bar{x}'+k}$ for any $k \in \mathbb{N}$, i.e., f *cycles* with period length $\bar{x}' - \bar{x}$.

Consider now some instance I. Let C with interval I_x be the corresponding end-configuration when f is applied to I. If $x \leqslant E$, we are done. Otherwise, there must be some $k \leqslant \bar{x}' - \bar{x}$ such that $\mathcal{C}^f_{\bar{x}+k} = \mathcal{C}^f_x$, since f cycles with this period length. Hence, by definition of $\mathcal{C}^f_{\bar{x}+k}$ there must be a realistic end-configuration $\tilde{C}$ which is equivalent to C for the interval $I_{\tilde{x}}$ with $\tilde{x} := \bar{x}+k \leqslant E$. □

Lemma 7.3.10. *Let f be a simplified algorithm map. There is an algorithm which approximates ρ_f up to a factor $1+\varepsilon$, i.e., it computes a value ρ' with $\rho' \leqslant \rho_f \leqslant (1+O(\varepsilon))\rho'$.*

Proof. According to Lemma 7.3.6 and Lemma 7.3.9, it suffices to construct the sets $\mathcal{C}_0^f, ..., \mathcal{C}_E^f$ in order to approximate the competitive ratio of all end-configurations in these sets. Due to Corollary 7.2.14, we know all possible values for all parameters of jobs explicitly with lower and upper bounds. Due to Lemma 7.2.13 we can additionally ensure that each equivalence class of configurations corresponds to at most one part. So the enumeration can be done in a finite amount of time. We start with $\mathcal{C}_0^f$ and determine $f(C_e)$ for one representative C_e of each equivalence class $[C_e] \in \mathcal{C}_0^f$. Based on this we determine the set $\mathcal{C}_1^f$. We continue inductively to construct all sets $\mathcal{C}_x^f$ with $x \leqslant E$.

We define ρ_{max} to be the maximum ratio $\rho(C)$ for an end-configuration $C \in \cup_{0\leqslant x\leqslant E}\mathcal{C}_x^f$. Due to Lemma 7.3.9 and Lemma 7.3.6 the value ρ_{max} implies the required ρ' satisfying the properties claimed in this lemma. □

Our main algorithm works as follows. We first enumerate all simplified algorithm maps. For each simplified algorithm map f, we approximate ρ_f using Lemma 7.3.10. We output the map f with the minimum (approximated) competitive ratio. Note that the resulting online algorithm has polynomial running time: All simplifications of a given instance can be done efficiently and for a given configuration, the equivalence class of the schedule for the next interval can be found in a look-up table of constant size.

Theorem 7.3.11. *$(Pm|r_j, pmtn|\sum w_j C_j)$ admits a competitive-ratio approximation scheme for any $m \in \mathbb{N}$.*

7.4 Extensions to other settings

Certain arguments in Section 7.2 do not transfer directly to more complex scheduling settings. In this section, we argue how to overcome the increased complexity. If we are able to obtain similar statements as for Corollary 7.2.7 and Corollary 7.2.14 for the extended settings competitive-ratio approximation schemes are also possible.

7.4.1 Non-preemptive scheduling

In this section we review which statements or proofs of Section 7.2 make use of the possibility to preempt jobs and explain how to proceed in the non-preemptive case. One difference is that a schedule can be defined by a start time S_j for each job j with a resulting completion time of $C_j = S_j + p_j$ on its assigned machine i_j. Nevertheless, Lemma 7.2.1 can be proven similarly. We adapt the definition of time-stretch to this setting: when applying a time-stretch, we now shift each completion time $C_j \in I_x$ to the next interval while keeping the offset with respect to the beginning of the interval, so

$$C_j' = R_{x+1} + (C_j - R_x).$$

For two completion time values $C_1 < C_2$ with $C_1 \in I_{x_1}$ and $C_2 \in I_{x_2}$ we observe that the difference after a time-stretch is

$$(C_2' - C_1') = (C_2 - C_1) + \sum_{x_1 \leqslant x < x_2} \varepsilon |I_x|.$$

Hence, idle time is inserted right before each job that is partially processed at some interval bound. Therefore, the proof of Lemma 7.2.2 needs the small adaptation that we place the idle time for the small jobs on each machine respectively. Lemma 7.2.3, Lemma 7.2.4 and Lemma 7.2.5 of Subsection 7.2.1 are not affected and remain valid as they are stated. Since splitting of jobs into small atoms as in Lemma 7.2.6 is not applicable here, we give a variant of this technique using start times. For each job j we define $s(j)$ and $c(j)$ to be interval indices such that

$$S_j \in I_{s(j)}, C_j \in I_{c(j)}.$$

Recall that with our adapted objective function, each ordering of jobs with $s(j) = c(j)$ assigned to one machine yields the same objective value. Hence, we only have to take care of the exact start times of those jobs with $s(j) < c(j)$ since they determine the amount of processing time assigned to each interval.

Lemma 7.4.1. *There is a constant $\mu \in \mathbb{N}$ such that at $1+\varepsilon$ loss we can restrict to schedules where each large job j with $s(j) < c(j)$ has a start time of the form*

$$S_j = R_{s(j)} + \ell_j \cdot |I_{s(j)}|/\mu$$

with $\ell_j \in \{0, \ldots, \mu - 1\}$.

Proof. We apply one time-stretch on a considered schedule yielding a new start time S_j for each job $j \in J$. Choose $\mu \in \mathbb{N}$ to be a constant integer such that $1/\mu$ is smaller than ε^2.

Consider now each interval I_x. For each machine i there is at most one job j with $x = s(j) < c(j)$ running on i. Due to the time-stretch there is idle time of at least $(1+\varepsilon)|I_{x-1}|$ before the start of j. We now set $\ell_j = \lfloor (S_j - R_x)\mu/|I_x| \rfloor$ and decrease the start time of j to $S_j' = R_x + \ell_j |I_x|/\mu$. This yields

$$S_j' \geqslant R_x + ((S_j - R_x)\frac{\mu}{|I_x|} - 1)\frac{|I_x|}{\mu} = S_j - \frac{|I_x|}{\mu} \geqslant S_j - \varepsilon^2 |I_x| \geqslant S_j - \varepsilon |I_{x-1}|.$$

Hence, we get a feasible schedule of the required form with costs increased by at most a factor of $(1+\varepsilon)$. □

Therefore, a similar variant of Corollary 7.2.7 with a restatement of its item 5 holds in the non-preemptive setting.

The conclusion that also in the non-preemptive case only a constant amount of history is relevant demands a bit more work. First, the definition of the safety net (Lemma 7.2.8) needs to be adjusted since it might be that all machines are executing jobs during the entire interval I_{x+s-1}. However, with the adapted definition of time-stretching we know that there is reserved space on one machine in $[R_x, R_{x+s})$ to process all jobs released at time R_x. It remains to verify

that an online algorithm (that does not know the future) can determine the beginning of this reserved space before it actually happens. Let j be the job scheduled to cover R_{x+s} on the last empty machine at this time. Denote its start and completion times before the time-stretch by $S_j \in I_{s(j)}, C_j \in I_{c(j)}$. From the above observation on time-stretches we can additionally conclude that

$$C'_j - p_j - \sum_{s(j) \leqslant x < c(j)} \varepsilon |I_x| \geqslant R_{s(j)+1}.$$

Hence, the safety net can be scheduled within interval $I_{s(j)+1}$ by our online algorithm since it decides about the complete schedule of an interval at its beginning.

Having the existence of the safety net established, we can consider similar periods and the proof of Lemma 7.2.9 that only counts release weights of periods remains valid. For the following we need the small adaptation of $\Gamma := (K+1)\cdot s$. This adjusted value is then used in Definition 7.2.10 about recent, old, dominated, irrelevant and relevant jobs at time R_x.

The proof of Lemma 7.2.11 still holds. This is unfortunately not true for Lemma 7.2.12. Since some remaining jobs at the end of a part may have already started processing, we cannot simply move them to their safety net. Hence, parts cannot be treated independently. Therefore, we switch back to consider complete instances. If we can no longer assume that an instance is mainly a sequence of significant periods, we lose the premise of Lemma 7.2.9. We cannot simply use that only a constant amount of history is relevant.

To solve this problem we will consider only special instances, where the weights after each insignificant period are sufficiently high such that they dominate the complete past until this period. The following lemma states the exact condition for these instances and proves again for each point in time that the relevant history of an instance with constant length dominates the irrelevant past even in the presence of insignificant periods. In a second step, we will reason why it is sufficient to consider only those instances satisfying the given condition. To state the first lemma, we need the following definitions.

For a given instance $\mathcal{I}$, let $a_1 < \cdots < a_{\ell+1}$ be again all indices satisfying Condition (7.2.2) of Lemma 7.2.13 (and $a_0 := -1$). Recall that the periods between $Q_{a_{i-1}}$ and Q_{a_i} build a sequence of significant periods for each $i = 1, \ldots, \ell$. Let

$$\text{first}(i), i = 1, \ldots, \ell$$

denote the job that is released first after period Q_{a_i} at release time R_{x_i}. Without loss of generality, we can assume that $I_{x_i} \in Q_{a_i+1}$ since we can otherwise split the instance again after an empty period. For a given algorithm A we denote by

$$A(\mathcal{I}|i)$$

the contribution of the jobs in $P_i = \bigcup_{p=a_i+1}^{a_{(i+1)}} Q_p$ to the objective value of the solution $A(\mathcal{I})$. We denote by

$$\mathcal{I}(i) := \bigcup_{k \leqslant i} P_k$$

the instance up to period $Q_{a_{(i+1)}}$. Note that $I_{x_{(\ell+1)}}$ denotes the first interval after $Q_{a_{(\ell+1)}}$.

Since we increased Γ by the length of one period and $I_{x_i} \in Q_{a_i+1}$, we can use Lemma 7.2.9 to extend Lemma 7.2.11 such that

$$(1+\varepsilon)^s \cdot \mathrm{rw}(\mathrm{Ir}_x(P_{i-1})) \leqslant 6\varepsilon \cdot \mathrm{rw}(\mathrm{Rel}_x(P_{i-1}))$$

actually holds for each $x_{(i-1)} \leqslant x < x_i, i = 1, \ldots, \ell+1$.

Lemma 7.4.2. *Let* I *be an instance such that for each* $i = 1, \ldots, \ell$ *it holds that*

$$\mathrm{rw}(\mathrm{Rel}_{x_i-1}(\mathcal{J}(i-1))) \leqslant \frac{\varepsilon}{(1+\varepsilon)^s\,(1+7\varepsilon)} \cdot \mathrm{rw}(\mathrm{first}(i)) \quad \textit{and} \tag{7.4.1a}$$

$$\max\{w_j \mid j \in \mathrm{Rel}_{x_i-1}(\mathcal{J}(i-1))\} \leqslant w_{\mathrm{first}(i)}. \tag{7.4.1b}$$

Then, for each $i = 1, \ldots, \ell+1$ *and each* $x_{i-1} \leqslant x < x_i$, *we have*

$$(1+\varepsilon)^s\,\mathrm{rw}(\mathrm{Ir}_x(\mathcal{J}(i-1))) \leqslant 7\varepsilon \cdot \mathrm{rw}(\mathrm{Rel}_x(\mathcal{J}(i-1))). \tag{7.4.2}$$

Proof. We prove this by induction and start with the base case $i = 1$. Consider some $x_0 \leqslant x < x_1$. By definition, all periods before Q_{a_1} are significant. By Lemma 7.2.11 we get

$$(1+\varepsilon)^s\,\mathrm{rw}(\mathrm{Ir}_x(\mathcal{J}(0))) \leqslant 7\varepsilon \cdot \mathrm{rw}(\mathrm{Rel}_x(\mathcal{J}(0))).$$

For the inductive step consider some $i = 2, \ldots, \ell+1$ and assume that (7.4.2) holds for $i-1$ and each $x_{(i-2)} \leqslant x < x_{(i-1)}$. Observe that $\mathcal{J}(i-1) = \mathcal{J}(i-2) \cup P_{i-1}$. Due to condition (7.4.1b), we know that $\mathrm{Dom}_x(\mathcal{J}(i-1)) \cap P_{i-1} = \mathrm{Dom}_x(P_{i-1})$ for each $x_{i-1} \leqslant x < x_i$ which yields:

$$\begin{aligned}
\mathrm{rw}(\mathrm{Ir}_x(\mathcal{J}(i-1))) &= \mathrm{rw}(\mathrm{Ir}_x(\mathcal{J}(i-1)) \cap \mathcal{J}(i-2)) + \mathrm{rw}(\mathrm{Ir}_x(\mathcal{J}(i-1)) \cap P_{i-1}) \\
&\leqslant \mathrm{rw}(\mathcal{J}(i-2)) + \mathrm{rw}(\mathrm{Ir}_x(P_{i-1})) \\
(7.4.2) \quad &\leqslant (1+7\varepsilon)\,\mathrm{rw}(\mathrm{Rel}_{x_{(i-1)}-1}(\mathcal{J}(i-2))) + \mathrm{rw}(\mathrm{Ir}_x(P_{i-1})) \\
(7.4.1) \quad &\leqslant \frac{\varepsilon}{(1+\varepsilon)^s} \cdot \mathrm{rw}(\mathrm{first}(i-1)) + \mathrm{rw}(\mathrm{Ir}_x(P_{i-1}))) \\
(\text{Lemma 7.2.11}) \quad &\leqslant \frac{\varepsilon}{(1+\varepsilon)^s} \cdot \mathrm{rw}(\mathrm{first}(i-1)) + \frac{6\varepsilon}{(1+\varepsilon)^s} \cdot \mathrm{rw}(\mathrm{Rel}_x(P_{i-1}))) \\
&\leqslant \frac{7\varepsilon}{(1+\varepsilon)^s} \cdot \mathrm{rw}(\mathrm{Rel}_x(\mathcal{J}(i-1))),
\end{aligned}$$

which proves the hypothesis. □

Consider now some online algorithm A with competitive ratio ρ_A that performs with ratio $\overline{\rho_A} := \max\{A(\mathcal{J})/\mathrm{OPT}(\mathcal{J}) \mid \mathcal{J} \text{ satisfies (7.4.1)}\} \leqslant \rho_A$ on our desired instances where the first released job after each insignificant period dominates the complete instance up to this period. We now want to find a new online algorithm A' that modifies the weights of each given arbitrary instance $\mathcal{I}$ to an instance $\mathcal{I}'$ satisfying (7.4.1) and creates a solution $A(\mathcal{I}')$ that yields a schedule $A'(\mathcal{I})$ for the original instance with

$$\frac{A'(\mathcal{I})}{\mathrm{OPT}(\mathcal{I})} \leqslant (1+\mathcal{O}(\varepsilon)) \max_i \frac{A(\mathcal{I}'(i))}{\mathrm{OPT}(\mathcal{I}'(i))} \leqslant (1+\mathcal{O}(\varepsilon))\overline{\rho_A} \leqslant (1+\mathcal{O}(\varepsilon))\rho_A.$$

Therefore, an online algorithm that proves to be good on only these instances yields an online algorithm for general instances with almost the same competitive ratio. Hence, it is sufficient to consider only these instances for the enumeration of simplified algorithm maps.

Lemma 7.4.3. *At* $(1+\mathcal{O}(\varepsilon))$ *loss, we can restrict to instances satisfying* (7.4.1).

Proof. For a considered online algorithm A we define a new algorithm A' that creates for a given instance $\mathcal{I}$ the new instance $\mathcal{I}'$ and applies A on $\mathcal{I}'$ as follows:

At the beginning we set $\mathcal{I}'(0) = \mathcal{I}(0)$. After period Q'_{a_i} we add for each further $i = 1, \dots, \ell$ and for each job $j \in P_i$ of instance $\mathcal{I}$ a new job j' to $\mathcal{I}'$ with equal processing time and release date but new weight $w_{j'} := \nu_i \cdot w_j$ such that $w_{j'} \leqslant w_{\text{first}(i)'}$ for each $j' \in \text{Rel}_{x_i-1}(\mathcal{I}'(i-1))$ and

$$\text{rw}(\text{Rel}_{x_i-1}(\mathcal{I}'(i-1))) \leqslant \frac{\varepsilon}{(1+\varepsilon)^s\,(1+7\varepsilon)} \cdot \text{rw}(\text{first}(i)').$$

Hence, $\mathcal{I}'$ satisfies (7.4.1). The schedules for $A'(\mathcal{I}(i))$ are then defined by applying $A(\mathcal{I}'(i))$.

We then observe that

$$A'(\mathcal{I}|i) \cdot \nu_i = A(I'|i) \leqslant A(I'(i))$$

and by Lemma 7.4.2,

$$\begin{aligned}
\text{OPT}(\mathcal{I}'(i)) &\leqslant \text{OPT}(\mathcal{I}|i) \cdot \nu_i + (1+\varepsilon)^s \,\text{rw}(\mathcal{I}'(i-1)) \\
(7.4.2) &\leqslant \text{OPT}(\mathcal{I}|i) \cdot \nu_i + (1+\varepsilon)^s\,(1+7\varepsilon)\,\text{rw}(\text{Rel}_{x_i-1}(\mathcal{I}'(i-1))) \\
(7.4.1) &\leqslant \text{OPT}(\mathcal{I}|i) \cdot \nu_i + \varepsilon \cdot \text{rw}(\text{first}(i)') \\
&\leqslant (1+\varepsilon)\nu_i \cdot \text{OPT}(\mathcal{I}|i).
\end{aligned}$$

The combination finally yields

$$\frac{A'(\mathcal{I})}{\text{OPT}(\mathcal{I})} \leqslant \max_i \frac{A'(\mathcal{I}|i)}{\text{OPT}(\mathcal{I}|i)} \leqslant \max_i \frac{(1+\varepsilon)\nu_i \cdot A(\mathcal{I}'(i))}{\text{OPT}(\mathcal{I}'(i)) \cdot \nu_i} \leqslant (1+\varepsilon) \max_i \frac{A(\mathcal{I}'(i))}{\text{OPT}(\mathcal{I}'(i))}$$

Hence, we get $\rho_{A'} \leqslant (1+\mathcal{O}(\varepsilon))\overline{\rho_A} \leqslant (1+\mathcal{O}(\varepsilon))\rho_A$. □

To conclude, even without treating parts independently, we can restrict to instances where only a constant amount of history is relevant. Therefore, we can state an adapted version of Corollary 7.2.14 for the non-preemptive setting:

Corollary 7.4.4. *At* $1+\mathcal{O}(\varepsilon)$ *loss we can assume for each instance* $\mathcal{I}$ *with job set* J *and each interval* I_x *that*

1. $p_j \in \{(1+\varepsilon)^k \mid \frac{\varepsilon^3}{4}(1+\varepsilon)^{-\Gamma} R_x \leqslant (1+\varepsilon)^k \leqslant \frac{1}{\varepsilon} R_x\}$ *for each* $j \in \text{Rec}_x(J)$,
2. $r_j \in \{(1+\varepsilon)^k \mid (1+\varepsilon)^{-\Gamma} R_x \leqslant (1+\varepsilon)^k \leqslant R_x\}$ *for each* $j \in \text{Rec}_x(J)$,
3. $w_j \in \{(1+\varepsilon)^k \mid w_x \leqslant (1+\varepsilon)^k \leqslant W \cdot w_x\}$ *for some value* w_x *and each* $j \in \text{Rel}_x(J)$,
4. $s(j), c(j) \in \{x-\Gamma, \dots, x+s\}$ *for each* $j \in \text{Rec}_x(J)$ *and* $S_j \in \{R_{s(j)} + \ell_j \cdot |I_{s(j)}|/\mu \mid \ell_j \in \{0, \dots, \mu-1\}\}$ *if* $s(j) < c(j)$,

5. *the cardinality of* $\mathrm{Rec}_x(J) \supseteq \mathrm{Rel}_x(J)$ *is bounded by* $\Gamma \cdot \Delta$*, and*

6. $\sum_{j \in \mathrm{Ir}_x(J)} w_j C_j \leqslant O(\varepsilon) \cdot OPT(\mathrm{Rel}_x(J))$.

We now continue with adaptations for Section 7.3. Since schedules are defined in the non-preemptive case via assigned machines and start times we discard the values of o_j and q_{ij} from the definitions of interval-schedules and configurations (compare Definition 7.3.1 and Definition 7.3.2). Instead, each interval-schedule S for interval I_x knows in addition to all properties of $J(S)$ for each already running job j with $s(j) < x \leqslant c(j)$ the values S_j and i_j and assigns these two values to each job j scheduled to start within I_x (i.e. $s(j) = x$). Also each configuration C for interval I_x contains additionally the values S_j and i_j for each job $j \in P(C) = \{j \in J(C) \mid s(j) < x \leqslant c(j)\}$ that is partially processed at time R_x.

In Definition 7.3.4 of two (σ, y)-equivalent interval-schedules S, S', we replace the o_j- and q_{ij}-conditions by:

$$i_{\sigma(j)} = i_j$$

and

$$x - x' = s(j) - s(\sigma(j)) = c(j) - c(\sigma(j))$$

for each $j \in \tilde{J}$ and

$$S_{\sigma(j)} = S_j(1+\varepsilon)^{x'-x}$$

for each $j \in \tilde{J}$ with $s(j) < c(j)$.

In the non-preemptive setting it is possible that jobs running at the beginning of an interval are also dominated and hence irrelevant at that time. Nevertheless, configurations with different dominated jobs partially processed at time R_x must be treated differently. To deal with this circumstance we have to extend the definition of equivalent configurations appropriately:

Definition 7.4.5. Two feasible configurations C, C' for the intervals $I_x, I_{x'}$ are called equivalent

- if they are equivalent in the sense of Definition 7.3.5 without the o_j-conditions
- and if there is a bijection $\psi\colon P(C) \to P(C')$ such that for each $j \in P(C)$,
  - $r_{\psi(j)} = r_j(1+\varepsilon)^{x'-x}$,
  - $p_{\psi(j)} = p_j(1+\varepsilon)^{x'-x}$,
  - $i_{\sigma(j)} = i_j$ and
  - $S_{\sigma(j)} = S_j(1+\varepsilon)^{x'-x}$.

This extension allows a similar proof of Lemma 7.3.7 since for any feasible algorithm-map f, a (σ, y)-equivalent interval schedule of $f(C_e)$ for any configuration $C \in [C_e]$ equivalent to the representative C_e has enough idle time where the running dominated jobs of $P(C)$ can be feasibly continued. Also, Lemma 7.3.6 is still valid.

Note that each job of $P(C)$ is recent at time R_x since no job remains unfinished for more than one period. Corollary 7.4.4 explicitly takes care of which statements are valid not only for the set of relevant jobs, but also for the set of recent jobs at any point in time. Therefore, we can

use Corollary 7.4.4 in the proof of Lemma 7.3.8 instead of Corollary 7.2.14 to conclude again that there are only constantly many simplified algorithm maps. With this, the implications for Lemma 7.3.9 and Lemma 7.3.10 apply similarly and we can construct a competitive-ratio approximation scheme as in Section 7.3.

Theorem 7.4.6. $(Pm|r_j|\sum w_j C_j)$ *admits a competitive-ratio approximation scheme for any* $m \in \mathbb{N}$.

7.4.2 Scheduling on related machines

We now consider the setting of scheduling related machines, where each machine i has a certain speed s_i associated to it. Processing job j on machine i takes p_j/s_i time units.

We review the statements and proofs of Section 7.2 and discuss where adjustments need to be made. Without loss of generality we can assume that the slowest machine has unit speed. Let s_{max} denote the maximum speed in an instance and denote by $s_\sum$ the sum of the processing speeds of all machines. We use the following adjusted version of Lemma 7.2.1:

Lemma 7.4.7. *At* $1+O(\varepsilon)$ *loss we can restrict to instances where all processing times, release dates, and weights are powers of* $1+\varepsilon$*, no job is released before time* $t = 1$*, and* $r_j \geqslant \varepsilon \cdot p_j/s_{max}$ *for all jobs* j.

Proof. Everything but the last statement follows exactly as in the proof of Lemma 7.2.1. For the last claim, let $A_1 = [\ell_1, u_1), \ldots, A_k = [\ell_k, u_k)$ be the time intervals during which job j is being processed in an optimal schedule that satisfies the other properties. Let t_ε denote the time where exactly an $\frac{\varepsilon}{1+\varepsilon}$-fraction of j has been processed. We multiply every time event by a factor of $(1+\varepsilon)$. Then we have enough processing time reserved that we can shift the starting time of j to $r'_j := (1+\varepsilon)t_\varepsilon$. If j started at time $\ell_1(1+\varepsilon)$, then at time $(1+\varepsilon)t_\varepsilon$ we would already have processed an ε-fraction of j. Therefore, $r'_j \geqslant \frac{\varepsilon p_j}{s_{max}}$ is true even if j was processed on the fastest machine up to time r'_j. □

We define

$$L_x = \left\{ j \in J \;\middle|\; r_j = R_x, p_j > \frac{\varepsilon^2 |I_x|}{s_{max}} \right\}$$

to be the set of *large* jobs released at R_x and

$$S_x = \left\{ j \in J \;\middle|\; r_j = R_x, p_j \leqslant \frac{\varepsilon^2 |I_x|}{s_{max}} \right\}$$

to be the set of *small* jobs released at R_x.

Lemma 7.4.8. *At* $1+\varepsilon$ *loss we can restrict to schedules such that for each interval* I_x *the small jobs scheduled within this interval are chosen by Smith's rule from the set* $\bigcup_{x' \leqslant x} S_{x'}$ *and no small job is preempted or only partially processed at the end of an interval. Therefore, we can restrict to instances with* $p(S_x) \leqslant m \cdot |I_x| \cdot s_\sum$ *for each interval* I_x.

The proof of Lemma 7.4.8 works the same way as the proof of Lemma 7.2.2, except that after shifting the large jobs to the beginning of the interval, we fill the machines with small jobs in some nonincreasing order of their speeds. Also, the bound on processing times now includes a factor of $s_{\sum}$.

For Lemma 7.2.3, we have a slight modification:

Lemma 7.4.9. *At $1+\mathcal{O}(\varepsilon)$ loss we can restrict to instances such that $p_j \geqslant \frac{\varepsilon^2}{4s_{max}} \cdot |I_x|$ for each job $j \in S_x$. In these instances, the number of distinct processing times of each set S_x is bounded from above by $\log_{(1+\varepsilon)} 4$.*

Proof. We call a job $j \in S_x$ *tiny* if

$$p_j \leqslant \frac{\varepsilon^2}{4s_{max}} \cdot |I_x|.$$

Let ℓ be the largest integer such that

$$\sum_{i=1}^{\ell} p_i \leqslant \frac{\varepsilon^2}{2s_{max}} \cdot |I_x|.$$

We construct the packs in the same way as in Lemma 7.2.3. Then, the processing time of all but possibly the last pack released at time R_x is in the interval

$$\left[\frac{\varepsilon^2}{4s_{max}} \cdot |I_x|, \frac{\varepsilon^2}{2s_{max}} \cdot |I_x|\right].$$

We apply one time-stretch and the rest of the argumentation holds. Each processing time of a job in S_x then equals $(1+\varepsilon)^y$ for some integer y such that

$$\frac{\varepsilon^3}{4s_{max}} \cdot (1+\varepsilon)^x \leqslant (1+\varepsilon)^y \leqslant \frac{\varepsilon^2 |I_x|}{s_{max}} = \frac{\varepsilon^3}{s_{max}} \cdot (1+\varepsilon)^x.$$

The number of integers y satisfying these inequalities is bounded from above by the claimed constant. □

Possible processing times of large jobs fall in the range $\left[\frac{\varepsilon^2}{s_{max}} \cdot |I_x|, \frac{r_j s_{max}}{\varepsilon}\right]$ and therefore, we get an upper bound on the number of distinct processing times of large jobs of $8 \log_{(1+\varepsilon)} \varepsilon \cdot s_{max}$.

Lemma 7.4.10. *Without loss, we can restrict to instances with*

$$|L_x| \leqslant (m/\varepsilon^2 + m) 8 s_{max}^2 \log_{(1+\varepsilon)} \frac{1}{\varepsilon}$$

for each set L_x.

Lemma 7.4.11. *There is a constant $\mu \in \mathbb{N}$ such that at $1+O(\varepsilon)$ loss we can restrict to schedules such that at the end of each interval, each large job j is processed to an extent which is an integer multiple of p_j/μ.*

Proof. We choose $\mu \in \mathbb{N}$ such that $1/\mu$ is smaller than $\varepsilon^4/(s_{max}^2 \cdot 8 \log_{(1+\varepsilon)} \frac{1}{\varepsilon})$. The remainder of the proof is identical to the argumentation in the proof of Lemma 7.2.6. □

The following corollary summarizes the simplifications that we need for the setting of related machines.

Corollary 7.4.12. *At $1+\mathcal{O}(\varepsilon)$ loss we can assume that for each interval I_x,*

1. *for each job j released at time R_x exists some integer k such that j has a processing time*

$$p_j = (1+\varepsilon)^k \in \left[\frac{\varepsilon^3 \cdot R_x}{4s_{\max}}, \frac{R_x \cdot s_{\max}}{\varepsilon}\right],$$

2. *there are at most*

$$8s_{\max}^2 \log_{(1+\varepsilon)} \frac{1}{\varepsilon} + \log_{(1+\varepsilon)} 4$$

distinct processing time values of jobs released at R_x,

3. *at most Δ jobs are released at R_x,*

4. *each small job started in I_x is completed in I_x without preemption, and*

5. *at the end of I_x, each job j is processed to an extent of $\ell_{x,j} \cdot p_j/\mu$ for some $\ell_{x,j} \in \{0,\dots,\mu\}$.*

We establish the safety net for the jobs of each release date R_x only on the *fastest* machine and thereby ensure the condition of Lemma 7.2.8 in the related machine setting. For the non-preemptive setting, we incorporate the adjustments introduced in Subsection 7.4.1. At $1+\varepsilon$ loss we can round the speeds of the machines to powers of $1+\varepsilon$. Thus, for a constant number of machines m, there are only constantly many possible vectors s characterizing the speeds of the machines if the machine speeds lie in a constant range. Assuming the latter, we apply our enumeration scheme to each of these speed vectors of the machines and obtain a competitive-ratio approximation scheme for this case.

Theorem 7.4.13. *For any $m \in \mathbb{N}$, we obtain competitive-ratio approximation schemes for*

$$(Qm|\, r_j, pmtn \,|\, \sum w_j C_j)$$

and

$$(Qm|\, r_j \,|\, \sum w_j C_j),$$

assuming that the speeds of any two machines differ by at most a constant factor.

In the preemptive setting we can strengthen the result and give a competitive-ratio approximation scheme for the case that machine speeds are part of the input, that is, we obtain a nearly optimal competitive ratio for *any* speed vector. The key is to bound the variety of different speeds. To that end, we show that at $1+\varepsilon$ loss a very fast machine can simulate $m-1$ very slow machines.

Lemma 7.4.14. *For $(Qm|\, r_j, pmtn \,|\, \sum w_j C_j)$, we can at $1+O(\varepsilon)$ loss restrict to instances in which $s_{\max}$ is bounded by m/ε.*

Proof. Given a schedule on related machines with speed values $s_1, ..., s_{max}$, we stretch time twice. Thus, we gain in each interval I_x free space of size εI_x on the fastest machine. For each machine whose speed is at most $\frac{\varepsilon}{m} s_{max}$, we take its schedule of the interval I_x and simulate it on the fastest machine. Thus, those slow machines are not needed and can be ignored. The remaining machines have speeds in $[\frac{\varepsilon}{m} s_{max}, s_{max}]$. Assuming the slowest machines have unit speed then yields the desired bound. □

As speeds are geometrically rounded, we also have for each value m only finitely many speed vectors. Our enumeration scheme finds a nearly optimal online algorithm with a particular routine for each speed vector.

Theorem 7.4.15. *For any $m \in \mathbb{N}$, we obtain a competitive-ratio approximation scheme for*

$$(Qm|r_j, pmtn|\sum w_j C_j).$$

7.5 Conclusions

We introduced the concept of competitive-ratio approximation schemes that compute online algorithms with a competitive ratio arbitrarily close to the best possible competitive ratio. We provided such schemes for various problem variants of scheduling jobs online to minimize the weighted sum of completion times.

The techniques derived in this chapter provide a new and interesting view on the behavior of online algorithms. We believe that they contribute to the understanding of such algorithms, possibly open a new viewpoint for understanding them, and open a new line of research which yields even further insights. In particular, recent results show that our methods can also be applied to other scheduling objective settings (Kurpisz et al. [2013]), other online models (Chen et al. [2015]; Megow and Wiese [2013]) and also to different online problems such as the k-server problem (Mömke [2013]).

Bibliography

[Aardal et al. 1996] Aardal, K. I. ; Hipolito, A. ; Hoesel, C.P.M. v. ; Jansen, B. ; Roos, C. ; Terlaky, T.: A branch-and-cut algorithm for the frequency assignment problem. In: *Research Memorandum 96/011, Maastricht University* (1996). – http://pub.maastrichtuniversity.nl/196b6d6f-4d2b-4ee7-a235-23f14f5e27e3

[Achterberg 2009] Achterberg, Tobias: SCIP: Solving constraint integer programs. In: *Mathematical Programming Computation* 1 (2009), Issue 1, pp 1–41. – http://mpc.zib.de/index.php/MPC/article/view/4

[Afrati et al. 1999] Afrati, Foto N. ; Bampis, Evripidis ; Chekuri, Chandra ; Karger, David R. ; Kenyon, Claire ; Khanna, Sanjeev ; Milis, Ioannis ; Queyranne, Maurice ; Skutella, Martin ; Stein, Clifford ; Sviridenko, Maxim: Approximation Schemes for Minimizing Average Weighted Completion Time with Release Dates. In: *Proceedings of the 40th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1999, pp 32–43. – https://dx.doi.org/10.1109/SFFCS.1999.814574

[Aggoun and Beldiceanu 1993] Aggoun, Abderrahmane ; Beldiceanu, Nicolas: Extending chip in order to solve complex scheduling and placement problems. In: *Mathematical and Computer Modelling* 17 (1993), Issue 7, pp 57–73. – https://dx.doi.org/10.1016/0895-7177(93)90068-A

[Albareda-Sambola et al. 2009] Albareda-Sambola, M. ; Fernández, E. ; Hinojosa, Y. ; Puerto, J.: The multi-period incremental service facility location problem. In: *Computers & Operations Research* 36 (2009), May, pp 1356–1375. – https://dx.doi.org/10.1016/j.cor.2008.02.010

[Álvarez-Miranda et al. 2013] Álvarez-Miranda, E. ; Ljubić, I. ; Mutzel, P.: The Maximum Weight Connected Subgraph Problem. In: *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*. Springer, 2013, pp 245–270. – https://dx.doi.org/10.1007/978-3-642-38189-8_11

[Anderson and Potts 2004] Anderson, Edward J. ; Potts, Chris N.: Online Scheduling of a Single Machine to Minimize Total Weighted Completion Time. In: *Mathematics of Operations Research* 29 (2004), pp 686–697. – https://dx.doi.org/10.1287/moor.1040.0092

[Arora and Barak 2009] Arora, Sanjeev ; Barak, Boaz: *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009

[Arulselvan et al. 2011] Arulselvan, Ashwin ; Bley, Andreas ; Gollowitzer, Stefan ; Ljubić, Ivana ; Maurer, Olaf: MIP modeling of incremental connected facility location. In: *Network Optimization* (2011), pp 490–502. – https://dx.doi.org/10.1007/978-3-642-21527-8_54

[Arulselvan et al. 2015] Arulselvan, Ashwin ; Maurer, Olaf ; Skutella, Martin: An incremental algorithm for the uncapacitated facility location problem. In: *Networks* 65 (2015), Issue 4, pp 306–311. – https://dx.doi.org/10.1002/net.21595

[Augustine et al. 2008] Augustine, John ; Irani, Sandy ; Swamy, Chaitanya: Optimal Power-Down Strategies. In: *SIAM Journal on Computing* 37 (2008), Issue 5, pp 1499–1516. – https://dx.doi.org/10.1137/05063787X

[Balakrishnan et al. 1994] Balakrishnan, A. ; Magnanti, T.L. ; Mirchandani, P.: Modeling and heuristic worst-case performance analysis of the two-level network design problem. In: *Management Science* 40 (1994), Issue 7, pp 846–867. – https://dx.doi.org/10.1287/mnsc.40.7.846

[Balinski 1961] Balinski, Michel L.: On the graph structure of convex polyhedra in n-space. In: *Pacific Journal of Mathematics* 11 (1961), Issue 2, pp 431–434. – https://dx.doi.org/10.2140/pjm.1961.11.431

[Bar-Yehuda and Moran 1984] Bar-Yehuda, R. ; Moran, S.: On approximation problems related to the independent set and vertex cover problems. In: *Discrete Applied Mathematics* 9 (1984), pp 1–10. – https://dx.doi.org/10.1016/0166-218X(84)90086-6

[Bardossy and Raghavan 2010] Bardossy, M. G. ; Raghavan, S.: Dual-Based Local Search for the Connected Facility Location and Related Problems. In: *INFORMS Journal on Computing* 22 (2010), Issue 4, pp 584–602. – https://dx.doi.org/10.1287/ijoc.1090.0375

[Beasley 2013] Beasley, J. E.: *OR-Library*. 2013. – http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html

[Bertsimas and Tsitsiklis 1997] Bertsimas, D. ; Tsitsiklis, J.N.: *Introduction to linear optimization*. Athena Scientific, 1997. – ISBN 978-1-886529-19-9

[Bertsimas and Sim 2003] Bertsimas, Dimitris ; Sim, Melvyn: Robust discrete optimization and network flows. In: *Mathematical programming* 98 (2003), Issue 1, pp 49–71. – https://dx.doi.org/10.1007/s10107-003-0396-4

[Bertsimas and Weismantel 2005] Bertsimas, Dimitris ; Weismantel, Robert: *Optimization over Integers*. Dynamic Ideas, 2005. – ISBN 0-9759146-2-6

[Bienstock et al. 2006] Bienstock, D. ; Raskina, O. ; Saniee, I. ; Wang, Q.: Combined Network Design and Multiperiod Pricing: Modeling, Solution Techniques, and Computation. In: *Operations Research* 54 (2006), Issue 2, pp 261–276. – https://dx.doi.org/10.1287/opre.1050.0259

[Bland 1977] Bland, Robert G.: New Finite Pivoting Rules for the Simplex Method. In: *Mathematics of Operations Research* 2 (1977), Issue 2, pp 103–107. – https://dx.doi.org/10.1287/moor.2.2.103

[Bley et al. 2013] Bley, Andreas ; Ljubić, Ivana ; Maurer, Olaf: Lagrangian decompositions for the two-level FTTx network design problem. In: *EURO Journal on Computational Optimization* 1 (2013), Issue 3-4, pp 221–252. – https://dx.doi.org/10.1007/s13675-013-0014-z

[Bley et al. 2008] Bley, Andreas ; Menne, Ullrich ; Klaehne, Roman ; Raack, Christian ; Wessaely, Roland: Multi-layer network design – A model-based optimization approach. In: *Proceedings of the PGTS 2008, Berlin, Germany*, 2008, pp 107 – 116

[Bonnans et al. 2003] Bonnans, J.F. ; Gilbert, J.C. ; Lemaréchal, C. ; Sagastizaábal, C.A.: *Numerical Optimization*. Springer, 2003

[Borndörfer et al. 1998] Borndörfer, Ralf ; Eisenblätter, Andreas ; Grötschel, Martin ; Martin, Alexander: *The Orientation Model for Frequency Assignment Problems*. 1998. – https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/562

[Boyacı et al. 2016] Boyaci, Arman ; Ekim, Tınaz ; Shalom, Mordechai ; Zaks, Shmuel: Graphs of Edge-intersecting and Non-splitting Paths. In: *Theor. Comput. Sci.* 629 (2016), May, Issue C, pp 40–50. – https://dx.doi.org/10.1016/j.tcs.2015.10.004

[Buchsbaum et al. 2004] Buchsbaum, Adam L. ; Karloff, Howard ; Kenyon, Claire ; Reingold, Nick ; Thorup, Mikkel: OPT Versus LOAD in Dynamic Storage Allocation. In: *SIAM Journal on Computing* 33 (2004), Issue 3, pp 632–646. – https://dx.doi.org/10.1137/S0097539703423941

[Byrka and Aardal 2010] Byrka, Jarosław ; Aardal, Karen: An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In: *SIAM Journal on Computing* 39 (2010), Issue 6, pp 2212–2231. – https://dx.doi.org/10.1137/070708901

[Byrka et al. 2013] Byrka, Jarosław ; Grandoni, Fabrizio ; Rothvoss, Thomas ; Sanità, Laura: Steiner Tree Approximation via Iterative Randomized Rounding. In: *Journal of the ACM* 60 (2013), February, Issue 1, pp 6:1–6:33. – https://dx.doi.org/10.1145/2432622.2432628

[Carvajal et al. 2013] Carvajal, R. ; Constantino, M. ; Goycoolea, M. ; Vielma, J. P. ; Weintraub, A.: Imposing Connectivity Constraints in Forest Planning Models. In: *Operations Research* 61 (2013), pp 824–836. – https://dx.doi.org/10.1287/opre.2013.1183

[Chakrabarti et al. 1996] Chakrabarti, Soumen ; Phillips, Cynthia A. ; Schulz, Andreas S. ; Shmoys, David B. ; Stein, Clifford ; Wein, Joel: Improved algorithms for minsum criteria. In: Meyer, Friedhelm (ed.) ; Monien, Burkhard (ed.): *Automata, Languages and Programming: 23rd International Colloquium, ICALP '96 Paderborn, Germany, July 8–12, 1996 Proceedings* vol. 1099, 1996, pp 646–657. – https://dx.doi.org/10.1007/3-540-61440-0_166

[Chardy et al. 2012] Chardy, M. ; Costa, M.-C. ; Faye, A. ; Trampont, M.: Optimizing splitter and fiber location in a multilevel optical FTTH network. In: *European Journal of Operational Research* 222 (2012), Issue 3, pp 430–440. – https://dx.doi.org/10.1016/j.ejor.2012.05.024

[Charikar et al. 2001] Charikar, Moses ; Khuller, Samir ; Mount, David M. ; Narasimhan, Giri: Algorithms for facility location problems with outliers. In: *Proceedings of the twelfth annual ACM-SIAM Symposium on Discrete Algorithms*, 2001 (SODA '01), pp 642–651. – https://dl.acm.org/citation.cfm?id=365555

[Chekuri et al. 2001] Chekuri, Chandra ; Motwani, Rajeev ; Natarajan, B. ; Stein, Clifford: Approximation Techniques for Average Completion Time Scheduling. In: *SIAM Journal on Computing* 31 (2001), pp 146–166. – https://dx.doi.org/10.1137/S0097539797327180

[Chen et al. 2015] Chen, Lin ; Ye, Deshi ; Zhang, Guochuan: Approximating the Optimal Algorithm for Online Scheduling Problems via Dynamic Programming. In: *Asia-Pacific Journal of Operational Research* 32 (2015), Issue 01. – https://dx.doi.org/10.1142/S0217595915400114

[Cherkassky and Goldberg 1997] Cherkassky, B.V. ; Goldberg, A.V.: On Implementing the Push-Relabel Method for the Maximum Flow Problem. In: *Algorithmica* 19 (1997), pp 390–410. – https://dx.doi.org/10.1007/PL00009180

[Chopra 1989] Chopra, Sunil: On the Spanning Tree Polyhedron. In: *Operations Research Letters* 8 (1989), Issue 1, pp 25–29. – https://dx.doi.org/10.1016/0167-6377(89)90029-1

[Chopra and Rao 1994] Chopra, Sunil ; Rao, M.R.: The Steiner tree problem I: Formulations, compositions and extension of facets. In: *Mathematical Programming* 64 (1994), Issue 1-3, pp 209–229. – https://dx.doi.org/10.1007/BF01582573

[Christofides and Brooker 1974] Christofides, Nicos ; Brooker, P.: Optimal expansion of an existing network. In: *Mathematical Programming* 6 (1974), pp 197–211. – https://dx.doi.org/10.1007/BF01580236

[Chrobak et al. 2008] Chrobak, Marek ; Kenyon, Claire ; Noga, John ; Young, Neal: Incremental Medians via Online Bidding. In: *Algorithmica* 50 (2008), Issue 4, pp 455–478. – https://dx.doi.org/10.1007/s00453-007-9005-x

[Chudak and Shmoys 2003] Chudak, Fabian A. ; Shmoys, David B.: Improved Approximation Algorithms for the Uncapacitated Facility Location Problem. In: *SIAM Journal on Computing* 33 (2003), Issue 1, pp 1–25. – https://dx.doi.org/10.1137/S0097539703405754

[Chung et al. 2010] Chung, Christine ; Nonner, Tim ; Souza, Alexander: SRPT is 1.86-Competitive for Completion Time Scheduling. In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SIAM, 2010, pp 1373–1388. – https://dx.doi.org/10.1137/1.9781611973075.111

[Cisco 2015] Cisco: *Cisco Visual Networking Index: Forecast and Methodology,*2014 — 2019. 2015. – Cisco White Paper, available online at www.cisco.com

[Coll et al. 2002] Coll, Pablo ; Marenco, Javier ; Díaz, Isabel M. ; Zabala, Paula: Facets of the graph coloring polytope. In: *Annals of Operations Research* 116 (2002), Issue 1-4, pp 79–90. – https://dx.doi.org/10.1023/A:1021315911306

[Cormen et al. 2000] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L. ; Stein, C.: *Introduction to Algorithms, 2nd edition*. MIT Press, McGraw-Hill Book Company, 2000

[Cornuéjols et al. 1990] Cornuéjols, Gérard P. ; Nemhauser, George L. ; Wolsey, Laurence A.: The uncapacitated facility location problem. In: Mirchandani, P. (ed.) ; Francis, R. (ed.): *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990, pp 119–171. – dx.doi.org/10.1002/net.3230240212

[Correa and Wagner 2009] Correa, José R. ; Wagner, Michael R.: LP-Based online scheduling: from single to parallel machines. In: *Mathematical Programming* 119 (2009), pp 109–136. – https://dx.doi.org/10.1007/s10107-007-0204-7

[Dantzig and Wolfe 1960] Dantzig, George B. ; Wolfe, Philip: Decomposition principle for linear programs. In: *Operations research* 8 (1960), Issue 1, pp 101–111. – https://dx.doi.org/10.1287/opre.8.1.101

[Demaine et al. 2013] Demaine, Erik D. ; Hajiaghayi, Mohammadtaghi ; Klein, Philip N.: Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs. In: *ACM Transactions on Algorithms* 10 (2013), Issue 3, pp 13:1–13:20. – https://dx.doi.org/10.1145/2601070

[Desaulniers et al. 2005] Desaulniers, Guy ; Desrosiers, Jacques ; Solomon, Marius M.: *Column Generation*. Springer US, 2005. – https://dx.doi.org/10.1007/b135457. – ISBN 978-0-387-25485-2

[Diestel 2010] Diestel, Reinhard: *Graph Theory*. 4th. Springer, 2010. – ISBN 978-3-642-14278-9

[Doulliez and Rao 1975] Doulliez, P. J. ; Rao, M. R.: Optimal Network Capacity Planning: A Shortest-Path Scheme. In: *Operations Research* 23 (1975), Issue 4, pp 810–818. – https://dx.doi.org/10.1287/opre.23.4.810

[Ebenlendr et al. 2009] Ebenlendr, Tomáš ; Jawor, Wojciech ; Sgall, Jiří: Preemptive Online Scheduling: Optimal Algorithms for All Speeds. In: *Algorithmica* 53 (2009), pp 504–522. – https://dx.doi.org/10.1007/s00453-008-9235-6

[Ebenlendr and Sgall 2011] Ebenlendr, Tomáš ; Sgall, Jiří: Semi-Online Preemptive Scheduling: One Algorithm for All Variants. In: *Theory of Computing Systems* 48 (2011), Issue 3, pp 577–613. – https://dx.doi.org/10.1007/s00224-010-9287-2

[Eisenbrand et al. 2010] Eisenbrand, Friedrich ; Grandoni, Fabrizio ; Rothvoss, Thomas ; Schäfer, Guido: Connected facility location via random facility sampling and core detouring.

In: *Journal of Computer and System Sciences* 76 (2010), Issue 8, pp 709–726. – https://dx.doi.org/10.1016/j.jcss.2010.02.001

[Epstein and van Stee 2003] Epstein, Leah ; Stee, Rob van: Lower Bounds for On-line Single-Machine Scheduling. In: *Theoretical Computer Science* 299 (2003), pp 439–450. – https://dx.doi.org/10.1016/S0304-3975(02)00488-7

[Feige 1998] Feige, Uriel: A Threshold of ln n for Approximating Set Cover. In: *Journal of the ACM* 45 (1998), Issue 4, pp 634–652. – https://dx.doi.org/10.1145/285055.285059

[Fernandes and Gouveia 1998] Fernandes, Lucinda M. ; Gouveia, Luis: Minimal spanning trees with a constraint on the number of leaves. In: *European Journal of Operational Research* 104 (1998), Issue 1, pp 250–261. – https://dx.doi.org/10.1016/S0377-2217(96)00327-X

[Fischetti et al. 2015] Fischetti, M. ; Leitner, M. ; Ljubić, I. ; Luipersbeck, M. ; Monaci, M. ; Resch, M. ; Salvagnin, D. ; Sinnl, M.: Thinning out Steiner trees: A node based model for uniform edge costs. (2015). – Submitted

[Fotakis 2006] Fotakis, Dimitris: Incremental algorithms for Facility Location and k-Median. In: *Theoretical Computer Science* 361 (2006), Issue 2-3, pp 275–313. – https://dx.doi.org/10.1016/j.tcs.2006.05.015

[Fotakis 2008] Fotakis, Dimitris: On the Competitive Ratio for Online Facility Location. In: *Algorithmica* 50 (2008), Issue 1, pp 1–57. – https://dx.doi.org/10.1007/s00453-007-9049-y

[FTTx-Plan 2012] *FTTx-Plan: Kostenoptimierte Planung von FTTx-Netzen*. 2012. – http://www.fttx-plan.de/

[Fujie 2004] Fujie, Tetsuya: The maximum-leaf spanning tree problem: Formulations and facets. In: *Networks* 43 (2004), Issue 4, pp 212–223. – https://dx.doi.org/10.1002/net.20001

[Garey and Graham 1975] Garey, M. R. ; Graham, R. L.: Bounds for Multiprocessor Scheduling with Resource Constraints. In: *SIAM Journal on Computing* 4 (1975), Issue 2, pp 187–200. – https://dx.doi.org/10.1137/0204015

[Garey and Johnson 1979] Garey, Michael R. ; Johnson, David S.: *Computers and Intractability - A Guide to the Theory of NP-Completeness*. New York : W.H. Freeman and Company, 1979. – ISBN 0-7167-1045-5

[Garg et al. 2000] Garg, Naveen ; Konjevod, Goran ; Ravi, R.: A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. In: *Journal of Algorithms* 37 (2000), Issue 1, pp 66–84. – https://dx.doi.org/10.1006/jagm.2000.1096

[Gendron et al. 2014] Gendron, Bernard ; Lucena, Abilio ; Cunha, Alexandre S. da ; Simonetti, Luidi: Benders Decomposition, Branch-and-Cut, and Hybrid Algorithms for the

Minimum Connected Dominating Set Problem. In: *INFORMS Journal on Computing* 26 (2014), Issue 4, pp 645–657. – https://dx.doi.org/10.1287/ijoc.2013.0589

[Gergov 1996] Gergov, Jordan: *Algorithms — ESA '96: Fourth Annual European Symposium Barcelona, Spain, September 25–27, 1996 Proceedings*. Chap. Approximation algorithms for dynamic storage allocation, pp 52–61. Berlin, Heidelberg : Springer Berlin Heidelberg, 1996. – https://dx.doi.org/10.1007/3-540-61680-2_46. – ISBN 978-3-540-70667-0

[Gergov 1999] Gergov, Jordan: Algorithms for Compile-time Memory Optimization. In: *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics, 1999 (SODA '99), pp 907–908. – http://dl.acm.org/citation.cfm?id=314500.315082. – ISBN 0-89871-434-6

[Gerstel et al. 2012] Gerstel, O. ; Jinno, M. ; Lord, A. ; Yoo, S.J.B.: Elastic optical networking: a new dawn for the optical layer? In: *Communications Magazine, IEEE* 50 (2012), February, Issue 2, pp s12–s20. – https://dx.doi.org/10.1109/MCOM.2012.6146481

[Goemans 1997] Goemans, Michel X.: Improved Approximation Algorithms for Scheduling with Release Dates. In: *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Society for Industrial and Applied Mathematics, 1997, pp 591–598. – https://dl.acm.org/citation.cfm?id=314161.314394

[Goemans et al. 2002] Goemans, Michel X. ; Queyranne, Maurice ; Schulz, Andreas S. ; Skutella, Martin ; Wang, Yaoguang: Single machine scheduling with release dates. In: *SIAM Journal on Discrete Mathematics* 15 (2002), pp 165–192. – https://dx.doi.org/10.1137/S089548019936223X

[Goldberg 2012] Goldberg, A.: *Andrew Goldberg's Network Optimization Library*. 2012. – http://www.avglab.com/andrew/soft.html

[Gollowitzer et al. 2013] Gollowitzer, Stefan ; Gouveia, Luis ; Ljubić, Ivana: Enhanced formulations and branch-and-cut for the two level network design problem with transition facilities. In: *European Journal of Operational Research* 225 (2013), Issue 2, pp 211–222. – https://dx.doi.org/10.1016/j.ejor.2012.09.040

[Gollowitzer and Ljubić 2011] Gollowitzer, Stefan ; Ljubić, Ivana: MIP models for connected facility location: A theoretical and computational Study. In: *Computers & Operations Research* 38 (2011), Issue 2, pp 435–449. – https://dx.doi.org/10.1016/j.cor.2010.07.002

[Gouveia et al. 2015] Gouveia, Luís ; Lopes, Maria J. ; Sousa, Amaro de: Single PON network design with unconstrained splitting stages. In: *European Journal of Operational Research* 240 (2015), Issue 2, pp 361 – 371. – https://dx.doi.org/10.1016/j.ejor.2014.07.006

[Graham et al. 1979] Graham, Ronald L. ; Lawler, Eugene L. ; Lenstra, Jan K. ; Rinnooy Kan, Alexander H. G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In: *Annals of Discrete Mathematics* 5 (1979), pp 287–326. – https://dx.doi.org/10.1016/S0167-5060(08)70356-X

[Grötschel et al. 1993] Grötschel, Martin ; Lovász, László ; Schrijver, Alexander: *Algorithms and Combinatorics*. vol. 2: *Geometric Algorithms and Combinatorial Optimization*. Second corrected edition. Springer, 1993. – ISBN 3-540-56740-2, 0-387-56740-2 (U.S.)

[Grötschel et al. 2013] Grötschel, Martin ; Raack, Christian ; Werner, Axel: Towards optimizing the deployment of optical access networks. In: *EURO Journal on Computational Optimization* 2 (2013), Issue 1, pp 17–53. – https://dx.doi.org/10.1007/s13675-013-0016-x

[Gualandi et al. 2010a] Gualandi, Stefano ; Malucelli, Federico ; Sozzi, Domenico L.: On the Design of the Fiber To The Home Networks. In: Faigle, U. (ed.) ; Schrader, R. (ed.) ; Herrmann, D. (ed.): *9th CTW Workshop, Cologne, Germany, 2010. Extended Abstracts*, 2010, pp 65–68. – http://www.zaik.uni-koeln.de/AFS/conferences/CTW2010/CTW%202010%20Band/PDFs/16-(P73-76)-25-ftth-ctw.pdf

[Gualandi et al. 2010b] Gualandi, Stefano ; Malucelli, Feredico ; Sozzi, Domenico L.: On the Design of the Next Generation Access Networks. In: Lodi, Andrea (ed.) ; Milano, Michela (ed.) ; Toth, Paolo (ed.): *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* vol. 6140. Springer, 2010, pp 162–175. – https://dx.doi.org/10.1007/978-3-642-13520-0_20

[Guha and Khuller 1998] Guha, Sudipto ; Khuller, Samir: Approximation algorithms for connected dominating sets. In: *Algorithmica* 20 (1998), Issue 4, pp 374–387. – https://dx.doi.org/10.1007/PL00009201

[Guha and Khuller 1999a] Guha, Sudipto ; Khuller, Samir: Greedy Strikes Back: Improved Facility Location Algorithms. In: *Journal of Algorithms* 31 (1999), Issue 1, pp 228–248. – https://dx.doi.org/10.1006/jagm.1998.0993

[Guha and Khuller 1999b] Guha, Sudipto ; Khuller, Samir: Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets. In: *Information and computation* 150 (1999), Issue 1, pp 57–74. – https://dx.doi.org/doi:10.1006/inco.1998.2754

[Hall et al. 1997] Hall, Leslie A. ; Schulz, Andreas S. ; Shmoys, David B. ; Wein, Joel: Scheduling To Minimize Average Completion Time: Off-line and On-line Approximation Algorithms. In: *Mathematics of Operations Research* 22 (1997), Issue 3, pp 513–544. – https://dx.doi.org/10.1287/moor.22.3.513

[Halperin and Krauthgamer 2003] Halperin, Eran ; Krauthgamer, Robert: Polylogarithmic inapproximability. In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp 585–594. – https://dx.doi.org/10.1145/780542.780628

[Hansen et al. 2009] Hansen, Pierre ; Labbé, Martine ; Schindl, David: Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. In: *Discrete Optimization* 6 (2009), Issue 2, pp 135–147. – https://dx.doi.org/10.1016/j.disopt.2008.10.004

[Hedetniemi et al. 1986] Hedetniemi, S. T. ; Laskar, Renu ; Pfaff, John: A linear algorithm for finding a minimum dominating set in a cactus. In: *Discrete Applied Mathematics* 13 (1986), pp 287–292. – https://dx.doi.org/10.1016/0166-218X(86)90089-2

[Helmberg 2009] Helmberg, C.: Network Models with Convex Cost Structure like Bundle Methods. In: Barnhart, C. (ed.) ; Clausen, U. (ed.) ; Lauther, U. (ed.) ; Möhring, R.H. (ed.): *Models and Algorithms for Optimization in Logistics*. Dagstuhl, Germany : Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009 (Dagstuhl Seminar Proceedings 09261). – http://drops.dagstuhl.de/opus/volltexte/2009/2190. – ISSN 1862-4405

[Helmberg 2012] Helmberg, C.: *The ConicBundle Library for Convex Optimization*. 2012. – http://www-user.tu-chemnitz.de/~helmberg/ConicBundle/

[Helmberg and Kiwiel 2002] Helmberg, C. ; Kiwiel, K.C.: A Spectral Bundle Method with Bounds. In: *Mathematical Programming* 93 (2002), Issue 2, pp 173–194

[Hiriart-Urruty and Lemaréchal 1993] Hiriart-Urruty, J.B. ; Lemaréchal, C.: Convex Analysis and Minimization Algorithms. In: *Volume 306 of Grundlehren der mathematischen Wissenschaften*. Springer, 1993

[Hoefer 2007] Hoefer, Martin: *UflLib. UFLP-benchmarks, optimization code and benchmark generators*. 2007. – http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/

[Hoogeveen and Vestjens 1996] Hoogeveen, Han ; Vestjens, Arjen P. A.: Optimal On-Line Algorithms for Single-Machine Scheduling. In: Cunningham, William H. (ed.) ; McCormick, S. T. (ed.) ; Queyranne, Maurice (ed.): *Integer Programming and Combinatorial Optimization: 5th International IPCO Conference Vancouver, British Columbia, Canada, June 3–5, 1996 Proceedings* vol. 1084, 1996, pp 404–414. – https://dx.doi.org/10.1007/3-540-61310-2_30

[Hopcroft and Tarjan 1973] Hopcroft, John ; Tarjan, Robert: Algorithm 447: Efficient Algorithms for Graph Manipulation. In: *Communication of the ACM* 16 (1973), June, Issue 6, pp 372–378. – https://dx.doi.org/10.1145/362248.362272

[Jain et al. 2003] Jain, Kamal ; Mahdian, Mohammad ; Markakis, Evangelos ; Saberi, Amin ; Vazirani, Vijay V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. In: *Journal of the ACM* 50 (2003), Issue 6, pp 795–824. – https://dx.doi.org/10.1145/950620.950621

[Jaumard et al. 2001] Jaumard, Brigitte ; Marcotte, Odile ; Meyer, Christophe ; Vovor, Tsevi: Comparison of column generation models for channel assignment in cellular networks. In: *Discrete Applied Mathematics* 112 (2001), Issue 1–3, pp 217–240. – https://dx.doi.org/10.1016/S0166-218X(00)00317-6

[Johnson 1974] Johnson, David S.: Fast algorithms for bin packing. In: *Journal of Computer and System Sciences* 8 (1974), Issue 3, pp 272–314. – https://dx.doi.org/10.1016/S0022-0000(74)80026-7

[Kannan and Monma 1978] Kannan, Ravindran ; Monma, Clyde L.: *Optimization and Operations Research: Proceedings of a Workshop Held at the University of Bonn, October 2–8, 1977*. Chap. On the Computational Complexity of Integer Programming Problems, pp 161–172. Berlin, Heidelberg : Springer Berlin Heidelberg, 1978. – https://dx.doi.org/10.1007/978-3-642-95322-4_17. – ISBN 978-3-642-95322-4

[Karlin et al. 1988] Karlin, Anna R. ; Manasse, Mark S. ; Rudolph, Larry ; Sleator, Daniel D.: Competitive snoopy caching. In: *Algorithmica* 3 (1988), pp 79–119. – https://dx.doi.org/10.1007/BF01762111

[Kazovsky 2011] Kazovsky, Leonid: *Broadband optical access networks : emerging technologies and optical-wireless convergence*. Hoboken, N.J : Wiley-Interscience, 2011. – ISBN 978-0-470-18235-2

[Khandekar et al. 2012] Khandekar, Rohit ; Kortsarz, Guy ; Nutov, Zeev: Approximating fault-tolerant group-Steiner problems. In: *Theoretical Computer Science* 416 (2012), pp 55–64. – https://dx.doi.org/10.1016/j.tcs.2011.08.021

[Kierstead 1988] Kierstead, H. A.: The Linearity of First-Fit Coloring of Interval Graphs. In: *SIAM Journal on Discrete Mathematics* 1 (1988), Issue 4, pp 526–530. – https://dx.doi.org/10.1137/0401048

[Kierstead 1991] Kierstead, H.A.: A polynomial time approximation algorithm for dynamic storage allocation. In: *Discrete Mathematics* 88 (1991), Issue 2, pp 231 – 237. – https://dx.doi.org/10.1016/0012-365X(91)90011-P

[Kim et al. 2011] Kim, Y. ; Lee, Y. ; Han, J.: A splitter location-allocation problem in designing fiber optic access networks. In: *European Journal of Operational Research* 210 (2011), Issue 2, pp 425–435. – https://dx.doi.org/10.1016/j.ejor.2010.10.003

[Klein and Ravi 1995] Klein, P. ; Ravi, R.: A nearly best-possible approximation algorithm for node-weighted Steiner trees. In: *Journal of Algorithms* 19 (1995), pp 104–115. – https://dx.doi.org/10.1006/jagm.1995.1029

[Klinkowski and Careglio 2011] Klinkowski, Miroslaw ; Careglio, Davide: A routing and spectrum assignment problem in optical OFDM networks. In: *First European Teletraffic Seminar*, 2011. – http://hdl.handle.net/2117/13505

[Koch and Martin 1998] Koch, T. ; Martin, A.: Solving Steiner tree problems in graphs to optimality. In: *Networks* 32 (1998), pp 207Ð232. – https://dx.doi.org/10.1002/(SICI)1097-0037(199810)32:3<207::AID-NET5>3.0.CO;2-O

[Koch et al. 2000] Koch, T. ; Martin, A. ; Voss, S.: SteinLib: An Updated Library on Steiner Tree Problems in Graphs / Konrad-Zuse-Zentrum für Informationstechnik Berlin. Takustr. 7, Berlin, 2000 (ZIB-Report 00-37). – Technical Report. – http://elib.zib.de/steinlib

[Korte et al. 1991] Korte, B. H. ; Lovász, L. ; Schrader, R.: *Greedoids*. Springer-Verlag, 1991 (Algorithms and Combinatorics)

[Koster and Zymolka 2003] Koster, Arie M. ; Zymolka, Adrian: Minimum Converter Wavelength Assignment in All-Optical Networks / Konrad-Zuse-Zentrum für Informationstechnik Berlin. Takustraße 7, 14195 Berlin, December 2003 (ZIB-Report 03-45). – Technical Report. http://nbn-resolving.de/urn:nbn:de:0297-zib-7673

[Kurpisz et al. 2013] Kurpisz, Adam ; Mastrolilli, Monaldo ; Stamoulis, Georgios: *Approximation and Online Algorithms: 10th International Workshop, WAOA 2012, Ljubljana, Slovenia, September 13-14, 2012, Revised Selected Papers*. Chap. Competitive-Ratio Approximation Schemes for Makespan Scheduling Problems, pp 159–172. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. – https://dx.doi.org/10.1007/978-3-642-38016-7_14

[Labetoulle et al. 1984] Labetoulle, Jacques ; Lawler, Eugene L. ; Lenstra, Jan K. ; Rinnooy Kan, Alexander H. G.: Preemptive scheduling of uniform machines subject to release dates. In: *Progress in Combinatorial Optimization*. Academic Press Canada, 1984, pp 245–261. – http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-1832

[Lardeux and Nace 2007] Lardeux, B. ; Nace, D.: Multi-Period Network Design of Optical Transmission Networks. In: *12th IEEE Symposium on Computers and Communications (ISCC 2007)*, 2007, pp 935–940. – https://dx.doi.org/10.1109/ISCC.2007.4381618

[Lee and Margot 2007] Lee, Jon ; Margot, François: On a binary-encoded ILP coloring formulation. In: *INFORMS Journal on Computing* 19 (2007), Issue 3, pp 406–415. – https://dx.doi.org/10.1287/ijoc.1060.0178

[Leitner and Raidl 2011] Leitner, M. ; Raidl, G.R.: Branch-and-Cut-and-Price for Capacitated Connected Facility Location. In: *Journal of Mathematical Modelling and Algorithms* 10 (2011), pp 245–267. – https://dx.doi.org/10.1007/s10852-011-9153-5

[Lenstra et al. 1977] Lenstra, Jan K. ; Rinnooy Kan, Alexander H. G. ; Brucker, Peter: Complexity of machine scheduling problems. In: *Annals of Discrete Mathematics* 1 (1977), pp 243–362. – https://dx.doi.org/10.1016/S0167-5060(08)70743-X

[Li 2011] Li, Shi: A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem. In: Aceto, Luca (ed.) ; Henzinger, Monika (ed.) ; Sgall, Jiří (ed.): *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II* vol. 6756, 2011, pp 77–88. – https://dx.doi.org/10.1007/978-3-642-22012-8_5

[Lin et al. 2010] Lin, Guolong ; Nagarajan, Chandrashekhar ; Rajaraman, Rajmohan ; Williamson, David P.: A General Approach for Incremental Approximation and Hierarchical Clustering. In: *SIAM Journal on Computing* (2010), pp 3633–3669. – https://dx.doi.org/10.1137/070698257

[Liu and Lu 2009] Liu, P. ; Lu, X.: On-line scheduling of parallel machines to minimize total completion times. In: *Computers and Operations Research* 36 (2009), pp 2647–2652. – https://dx.doi.org/10.1016/j.cor.2008.11.008

[Ljubić 2007] Ljubić, Ivana: *Lecture Notes in Computer Science*. vol. 4771: *A hybrid VNS for connected facility location*. pp 157–169. In: Bartz-Beielstein, Thomas (ed.) ; Aguilera, Maria José B. (ed.) ; Blum, Christian (ed.) ; Naujoks, Boris (ed.) ; Roli, Andrea (ed.) ; Rudolph, Günter (ed.) ; Sampels, Michael (ed.): *Hybrid Metaheuristics* vol. 4771, Springer, 2007. – https://dx.doi.org/10.1007/978-3-540-75514-2_12

[Ljubić et al. 2012] Ljubić, Ivana ; Putz, Peter ; Salazar-González, Juan-José: Exact Approaches to the Single-Source Network Loading Problem. In: *Networks* 59 (2012), Issue 1, pp 89–106. – https://dx.doi.org/10.1002/net.20481

[Ljubić et al. 2006] Ljubić, Ivana ; Weiskircher, René ; Pferschy, Ulrich ; Klau, Gunnar W. ; Mutzel, Petra ; Fischetti, Matteo: An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem. In: *Mathematical Programming* 105 (2006), pp 427–449. – https://dx.doi.org/10.1007/s10107-005-0660-x

[Lu et al. 2003] Lu, Xiwen ; Sitters, René A. ; Stougie, Leen: A class of on-line scheduling algorithms to minimize total completion time. In: *Operations Research Letters* 31 (2003), pp 232–236. – https://dx.doi.org/10.1016/S0167-6377(03)00016-6

[Lübbecke and Desrosiers 2005] Lübbecke, Marco E. ; Desrosiers, Jacques: Selected topics in column generation. In: *Operations Research* 53 (2005), Issue 6, pp 1007–1023. – https://dx.doi.org/10.1287/opre.1050.0234

[Lund and Reingold 1994] Lund, Carsten ; Reingold, Nick: Linear programs for randomized on-line algorithms. In: *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics, 1994 (SODA '94), pp 382–391. – http://dl.acm.org/citation.cfm?id=314464.314574. – ISBN 0-89871-329-3

[Lund and Yannakakis 1994] Lund, Carsten ; Yannakakis, Mihalis: On the Hardness of Approximating Minimization Problems. In: *Journal of the ACM* 41 (1994), September, Issue 5, pp 960–981. – https://dx.doi.org/10.1145/185675.306789

[Manasse et al. 1988] Manasse, M. ; McGeoch, L. ; Sleator, D.: Competitive algorithms for on-line problems. In: *Proceedings of the 20th annual ACM Symposium on Theory of Computing (STOC)*, ACM, 1988, pp 322–333. – https://dx.doi.org/10.1145/62212.62243

[Marenco and Wagler 2006] Marenco, Javier ; Wagler, Annegret: On the combinatorial structure of chromatic scheduling polytopes. In: *Discrete Applied Mathematics* 154 (2006), Issue 13, pp 1865–1876. – https://dx.doi.org/10.1016/j.dam.2006.03.032

[Marenco and Wagler 2009a] Marenco, Javier ; Wagler, Annegret: Cycle-based facets of chromatic scheduling polytopes. In: *Discrete Optimization* 6 (2009), Issue 1, pp 51–63. – https://dx.doi.org/10.1016/j.disopt.2008.08.004

[Marenco and Wagler 2009b] Marenco, Javier ; Wagler, Annegret: Facet-inducing inequalities for chromatic scheduling polytopes based on covering cliques. In: *Discrete Optimization* 6 (2009), Issue 1, pp 64–78. – https://dx.doi.org/10.1016/j.disopt.2008.09.002

[Martens et al. 2010] Martens, M. ; Orlowski, S. ; Werner, A. ; Wessäly, R. ; Bentz, W.: FTTx-PLAN: Optimierter Aufbau von FTTx-Netzen. In: *Breitbandversorgung in Deutschland* vol. 220, VDE-Verlag, March 2010

[Martens et al. 2009] Martens, M. ; Patzak, E. ; Richter, A. ; Wessäly, R.: Werkzeuge zur Planung und Optimierung von FTTx-Netzen. In: *16. ITG-Fachtagung Kommunikationskabelnetze, Köln, Germany* vol. 218, VDE-Verlag, 2009, pp 37–41

[McNaughton 1959] McNaughton, Robert: Scheduling with deadlines and loss functions. In: *Management Science* 6 (1959), Issue 1, pp 1–12. – https://dx.doi.org/10.1287/mnsc.6.1.1

[Megow 2007] Megow, Nicole: *Coping with incomplete information in scheduling—stochastic and online models*, Technische Universität Berlin, Germany, PhD Thesis, 2007

[Megow and Schulz 2004] Megow, Nicole ; Schulz, Andreas S.: On-line scheduling to minimize average completion time revisited. In: *Operations Research Letters* 32 (2004), pp 485–490. – https://dx.doi.org/10.1016/j.orl.2003.11.008

[Megow and Wiese 2013] Megow, Nicole ; Wiese, Andreas: Competitive-Ratio Approximation Schemes for Minimizing the Makespan in the Online-List Model. In: *CoRR* abs/1303.1912 (2013). – http://arxiv.org/abs/1303.1912

[Mehlhorn 1988] Mehlhorn, K.: A faster approximation algorithm for the Steiner problem in graphs. In: *Information Processing Letters* 27 (1988), Issue 3, pp 125–128. – https://dx.doi.org/10.1016/0020-0190(88)90066-X

[Mehrotra and Trick 1996] Mehrotra, Anuj ; Trick, Michael A.: A column generation approach for graph coloring. In: *INFORMS Journal on Computing* 8 (1996), Issue 4, pp 344–354. – https://dx.doi.org/10.1287/ijoc.8.4.344

[Mehrotra and Trick 2007] Mehrotra, Anuj ; Trick, Michael A.: A branch-and-price approach for graph multi-coloring. In: *Extending the horizons: Advances in computing, optimization, and decision technologies* (2007), pp 15–29. – https://dx.doi.org/10.1007/978-0-387-48793-9_2

[Méndez-Díaz and Zabala 2006] Méndez-Díaz, Isabel ; Zabala, Paula: A branch-and-cut algorithm for graph coloring. In: *Discrete Applied Mathematics* 154 (2006), Issue 5, pp 826–847. – https://dx.doi.org/10.1016/j.dam.2005.05.022

[Méndez-Díaz and Zabala 2008] Méndez-Díaz, Isabel ; Zabala, Paula: A cutting plane algorithm for graph coloring. In: *Discrete Applied Mathematics* 156 (2008), Issue 2, pp 159–179. – https://dx.doi.org/10.1016/j.dam.2006.07.010

[Meyerson 2001] Meyerson, Adam: Online Facility Location. In: *42nd IEEE Symposium on Foundations of Computer Science (FOCS), Proceedings*, 2001, pp 426–431. – https://dx.doi.org/10.1109/SFCS.2001.959917

[Mömke 2013] Mömke, Tobias: A Competitive Ratio Approximation Scheme for the k-Server Problem in Fixed Finite Metrics. In: *CoRR* abs/1303.2963 (2013). – http://arxiv.org/abs/1303.2963

[Mukherjee 2006] Mukherjee, Biswanath: *Optical WDM networks*. New York : Springer, 2006. – ISBN 978-0-387-29188-8

[Olszewski 2014] Olszewski, Ireneusz: *Routing and Spectrum Assignment in Spectrum Flexible Transparent Optical Networks*. pp 407–417. In: S. Choras, Ryszard (ed.): *Image Processing and Communications Challenges 5*. Heidelberg : Springer International Publishing, 2014. – https://dx.doi.org/10.1007/978-3-319-01622-1_46

[Orlowski et al. 2011] Orlowski, S. ; Werner, A. ; Wessäly, R. ; Eckel, K. ; Seibel, J. ; Patzak, E. ; Louchet, H. ; Bentz, W.: Schätze heben bei der Planung von FTTx-Netzen: optimierte Nutzung von existierenden Leerrohren – eine Praxisstudie. In: *Breitbandversorgung in Deutschland* vol. 227, VDE-Verlag, March 2011

[Orlowski et al. 2010] Orlowski, S. ; Wessäly, R. ; Pióro, M. ; Tomaszewski, A.: SNDlib 1.0—Survivable Network Design Library. In: *Networks* 55 (2010), Issue 3, pp 276–286. – https://dx.doi.org/10.1002/net.20371

[Owen and Daskin 1998] Owen, S.H. ; Daskin, M.S.: Strategic facility location: A review. In: *European Journal of Operational Research* 111 (1998), Issue 3, pp 423–447. – https://dx.doi.org/10.1016/S0377-2217(98)00186-6

[Papadimitriou 1982] Papadimitriou, Christos: *Combinatorial optimization : algorithms and complexity*. Englewood Cliffs, N.J : Prentice Hall, 1982. – ISBN 978-0486402581

[Papadimitriou 1981] Papadimitriou, Christos H.: On the complexity of integer programming. In: *Journal of the ACM* 28 (1981), Issue 4, pp 765–768. – https://dx.doi.org/10.1145/322276.322287

[Papadimitriou 1993] Papadimitriou, Christos H.: *Computational Complexity*. Addison Wesley, 1993

[Peters 2015] Peters, Simon: *The Minimum Interval Coloring Problem*. Germany, Universität Kassel, Bachelor's Thesis (Studienarbeit), December 2015

[Phillips et al. 1998] Phillips, Cynthia ; Stein, Clifford ; Wein, Joel: Minimizing average completion time in the presence of release dates. In: *Mathematical Programming* 82 (1998), pp 199–223. – https://dx.doi.org/10.1007/BF01585872

[Plaxton 2003] Plaxton, Charles G.: Approximation Algorithms for Hierarchical Location Problems. In: *Proceedings of the 35th Annual ACM Symposium On Theory of Computing*, ACM, 2003, pp 40–49. – https://dx.doi.org/10.1145/780542.780549

[Putz 2012] Putz, Peter: *Fiber To The Home, Cost Optimal Design of Last-Mile Broadband Telecommunication Networks*, University of Vienna, PhD Thesis, 2012

[Ramaswami and Sivarajan 1995] Ramaswami, Rajiv ; Sivarajan, Kumar N.: Routing and wavelength assignment in all-optical networks. In: *IEEE/ACM Transactions on Networking (TON)* 3 (1995), Issue 5, pp 489–500. – https://dx.doi.org/10.1109/90.469957

[Ruan et al. 2004] Ruan, L. ; Du, H. ; Jia, X. ; Wu, W. ; Li, Y. ; Ko, K.: A greedy approximation for minimum connected dominating sets. In: *Theoretical Computer Science* 329 (2004), pp 325–330. – https://dx.doi.org/10.1016/j.tcs.2004.08.013

[Salman 2000] Salman, F.S.: *Selected Problems in Network Design: Exact and Approximate Solution Methods*, Carnegie Mellon University, Pittsburgh, PhD Thesis, 2000

[Schrage 1968] Schrage, Linus: Letter to the Editor — A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. In: *Operations Research* 16 (1968), pp 687–690. – https://dx.doi.org/10.1287/opre.16.3.687

[Schulz and Skutella 2002a] Schulz, Andreas S. ; Skutella, Martin: The Power of α-Points in Preemptive Single Machine Scheduling. In: *Journal of Scheduling* 5 (2002), pp 121–133. – https://dx.doi.org/10.1002/jos.93

[Schulz and Skutella 2002b] Schulz, Andreas S. ; Skutella, Martin: Scheduling Unrelated Machines by Randomized Rounding. In: *SIAM Journal on Discrete Mathematics* 15 (2002), pp 450–469. – https://dx.doi.org/10.1137/S0895480199357078

[Seiden 2000] Seiden, Steven S.: A guessing game and randomized online algorithms. In: *Proceedings of the 32nd ACM Symposium on the Theory of Computing (STOC)*, ACM, 2000, pp 592–601. – https://dx.doi.org/10.1145/335305.335385

[Shmoys et al. 1997] Shmoys, David ; Tardos Éva ; Aardal, Karen: Approximation algorithms for facility location problems. In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*, ACM, 1997, pp 265–274. – https://dx.doi.org/10.1145/258533.258600

[Sitters 2010a] Sitters, René: Competitive analysis of preemptive single-machine scheduling. In: *Operations Research Letters* 38 (2010), pp 585–588. – https://dx.doi.org/10.1016/j.orl.2010.08.012

[Sitters 2010b] Sitters, René: Efficient Algorithms for Average Completion Time Scheduling. In: Eisenbrand, Friedrich (ed.) ; Shepherd, F. B. (ed.): *Integer Programming and Combinatorial Optimization: 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings* vol. 6080, Springer, 2010, pp 411–423

[Sleator and Tarjan 1985] Sleator, Daniel D. ; Tarjan, Robert E.: Amortized Efficiency of List Update and Paging Rules. In: *Communications of the ACM* 28 (1985), Issue 2, pp 202–208. – https://dx.doi.org/10.1145/2786.2793

[Smith 1956] Smith, Wayne E.: Various optimizers for single-stage production. In: *Naval Research Logistics Quarterly* 3 (1956), Issue 1-2, pp 59–66. – https://dx.doi.org/10.1002/nav.3800030106

[Stougie and Vestjens 2002] Stougie, Leen ; Vestjens, Arjen P. A.: Randomized algorithms for on-line scheduling problems: how low can't you go? In: *Operations Research Letters* 30 (2002), Issue 2, pp 89–96. – https://dx.doi.org/10.1016/S0167-6377(01)00115-8

[Suhl and Hilbert 1998] Suhl, U.H. ; Hilbert, H.: A branch-and-cut algorithm for solving generalized multiperiod Steiner problems in graphs. In: *Networks* 31 (1998), Issue 4, pp 273–282

[Swamy and Kumar 2004] Swamy, Chaitanya ; Kumar, Amit: Primal–Dual Algorithms for Connected Facility Location Problems. In: *Algorithmica* 40 (2004), Issue 4, pp 245–269. – https://dx.doi.org/10.1007/s00453-004-1112-3

[Takara et al. 2012] Takara, Hidehiko ; Sano, Akihide ; Kobayashi, Takayuki ; Kubota, Hirokazu ; Kawakami, Hiroto ; Matsuura, Akihiko ; Miyamoto, Yutaka ; Abe, Yoshiteru ; Ono, Hirotaka ; Shikama, Kota ; Goto, Yukihiro ; Tsujikawa, Kyozo ; Sasaki, Yusuke ; Ishida, Itaru ; Takenaga, Katsuhiro ; Matsuo, Shoichiro ; Saitoh, Kunimasa ; Koshiba, Masanori ; Morioka, Toshio: 1.01-Pb/s (12 SDM/222 WDM/456 Gb/s) Crosstalk-managed Transmission with 91.4-b/s/Hz Aggregate Spectral Efficiency. In: *European Conference and Exhibition on Optical Communication*, Optical Society of America, 2012, pp Th.3.C.1. – https://www.osapublishing.org/abstract.cfm?URI=ECEOC-2012-Th.3.C.1

[Tomazic and Ljubić 2008] Tomazic, Alessandro ; Ljubić, Ivana: A GRASP Algorithm for the Connected Facility Location Problem. In: *Internal Symposium on Applications and the Internet (SAINT), Proceedings*, IEEE, 2008, pp 257–260. – https://dx.doi.org/10.1109/SAINT.2008.64

[Ukkusuri and Patil 2009] Ukkusuri, Satish V. ; Patil, Gopal: Multi-period transportation network design under demand uncertainty. In: *Transportation Research Part B: Methodological* 43 (2009), Issue 6, pp 625–642. – https://dx.doi.org/10.1016/j.trb.2009.01.004

[Vazirani 2003] Vazirani, Vijay V.: *Approximation Algorithms*. Springer, 2003

[Wang et al. 2015] Wang, Y. ; Buchanan, A. ; Butenko, S.: *On imposing connectivity constraints in integer programs*. 2015. – Submitted

[Wolsey 1998] Wolsey, Laurence A.: *Integer programming*. vol. 42. Wiley-Interscience, 1998. – ISBN 978-0-471-28366-9

[Zadeh 1974] Zadeh, N.: On building minimum cost communication networks over time. In: *Networks* 4 (1974), Issue 1, pp 19–34. – https://dx.doi.org/10.1002/net.3230040104

[Żotkiewicz et al. 2015] Żotkiewicz, Mateusz ; Mycek, Mariusz ; Tomaszewski, Artur: Profitable areas in large-scale FTTH network optimization. In: *Telecommunication Systems* 61 (2015), Issue 3, pp 591–608. – https://dx.doi.org/10.1007/s11235-015-0016-7