DISSERTATION

Solving Two Network Design Problems by Mixed Integer Programming and Hybrid Optimization Methods

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

ao. Univ.-Prof. Dipl.-Ing. Dr. Günther R. Raidl
Institut für Computergraphik und Algorithmen
Technische Universität Wien

und

ao. Univ.-Prof. Dipl.-Ing. Dr. Ulrich Pferschy
Institut für Statistik und Operations Research
Karl-Franzens-Universität Graz

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Dipl.-Ing. Markus Leitner
Matrikelnummer 0025315
Palffygasse 27/11, 1170 Wien

Wien, am 28. Mai 2010

Markus Leitner

Kurzfassung

Diese Dissertation behandelt zwei $\mathcal{NP}$-schwere kombinatorische Optimierungsprobleme aus dem Bereich Netzwerkdesign. Derartige Netzwerkdesignprobleme treten in vielen praktischen Anwendungen, wie etwa dem Design von Telekommunikationsnetzen, auf. Die beiden in dieser Arbeit behandelten Probleme erlauben die Modellierung von Szenarien welche beispielsweise bei der Planung von Glasfasernetzwerken auftreten. Aufgrund gestiegener Kundenanforderungen hinsichtlich verfügbarer Bandbreite sind Telekommunikationsfirmen gezwungen ihre Netze zu erweitern bzw. existierende Kupferverbindungen sukzessive duch Glasfaser zu ersetzen. Im Allgemeinen sind Kunden jedoch nicht bereit wesentlich höhere Preise für schnellere Breitbandanschlüsse zu bezahlen. Aus diesem Grund sind gute Algorithmen zur kosteneffizienten Planung derartiger Netzwerke von entscheidender Bedeutung.

Das $b_{\max}$-*Survivable Network Design Problem* ($b_{\max}$-SNDP) betrachtet die Aufgabe der effizienten Erweiterung von Fiber-to-the-home Netzwerken. Hierbei sind neben sogenannten Standardkunden mit einfachen Anbindungsanforderungen, welche auch als Typ-1 oder C_1 Kunden bezeichnet werden, zusätzlich Typ-2 (oder C_2) Kunden gegeben. Für diese muss die Verbindung an das Netzwerk redundant ausgeführt werden, sodass deren Konnektivität im Fall eines einfachen Fehlers garantiert werden kann. Nachdem diese Art der redundanten Anbindung jedoch häufig zu teuer ist, erlaubt das $b_{\max}$-SNDP eine Relaxierung dieser Anforderung. In diesem Fall darf das letzte Stück des Anschlusses eines C_2 Kunden nichtredundant ausgeführt sein. Diese nichtredundante *branch-line* darf jedoch eine vordefinierte Länge $b_{\max}$ nicht

überschreiten. In dieser Arbeit werden zwei, hinsichtlich der Zielfunktion unterschiedliche, Varianten des Problems behandelt. Während beim sogenannten "Operative Planning Task" (OPT) alle gegebenen Kunden möglichst kostengünstig an ein bestehendes Netzwerk angeschlossen werden sollen, wird beim "Strategic Simulation Task" (SST) eine möglichst profitable Lösung gesucht, in welcher eventuell nur eine Teilmenge aller Kunden versorgt wird.

Zur Lösung des Problems werden zwei auf gemischt-ganzzahliger linearer Programmierung beruhende Modelle vorgeschlagen. Diese können mittels Branch-and-Price gelöst werden und liefern beweisbar optimale Lösungen für kleine und mittelgroße Probleminstanzen. Eine Spezialität hierbei ist die Verwendung von alternativen dualen Lösungen im sogenannten Pricing Problem, wodurch die Lösung der linearen Relaxierung beider Modelle mittels Spaltengenerierung enorm beschleunigt wird. Weiters werden ein hybrider Optimierungsansatz, welcher auf Lagranger Relaxierung beruht, sowie metaheuristische Methoden zur näherungsweisen Lösung von sehr großen Instanzen von $b_{\max}$-SNDP vorgeschlagen. Die erzielten Testergebnisse belegen die Effektivität der vorgestellten Lösungsansätze.

Speziell in ländlichen Gebieten sind Fiber-to-the-home Netzwerke häufig nicht profitabel. In solchen Fällen wird oft eine Fiber-to-the-curb Strategie verfolgt. Hier wird das neue Netz nicht bis zum Kunden, sondern nur bis zu Übergabepunkten errichtet. Sind diese Übergabepunkte (facilities) nahe genug an den jeweils zugewiesenen Kunden, kann dennoch eine beträchtliche Steigerung der verfügbaren Bandbreite erzielt werden. Derartige Szenarien können als Varianten des Connected Facility Location Problems (ConFL), bei dem eine Menge an Facilities ausgewählt und miteinander verbunden werden sollen, modelliert werden. Zusätzlich müssen die Kunden noch diesen Facilities zugeordnet werden. Der zweite Teil dieser Arbeit beschäftigt sich mit dem *Capacitated Connected Facility Location Problem* (CConFL), welches ConFL um wichtige Nebenbedingungen erweitert. Dazu zählen etwa Kapazitätsbeschränkungen für Facilities aufgrund individueller Bandbreitenanforderungen von Kunden. Das Ziel der Optimierung ist es eine möglichst profitable Lösung zu berechnen bei der nicht zwangsweise alle potentiellen Kunden versorgt werden.

Es werden vier auf gemischt-ganzzahliger linearer Programmierung beruhende Verfahren vorgestellt mit welchen beweisbar optimale Lösungen für CConFL berechnet werden können. Diese werden anhand ihrer zugrundeliegenden Polyeder aus theoretischer Sicht verglichen. Weiters wird ein auf Lagranger Relaxierung basierender Ansatz vorgestellt, welcher im Anschluss mit lokaler Suche sowie "very large scale neighborhood search" hybridisiert wird. Die Ergebnisse der durchgeführten computationalen Studie zeigen klare Vorteile für zwei der vorgestellten Lösungsansätze.

Abstract

This thesis considers two $\mathcal{NP}$-hard combinatorial optimization problems (COPs) from the domain of network design. Network design problems (NDPs) arise in a multitude of real world applications such as the design of telecommunication networks. The NDPs addressed in this thesis are suitable to model certain real-world scenarios occurring in the extension of communication networks on the last mile. Nowadays, telecommunication companies need to upgrade and extend existing networks due to the rising bandwidth requirements of customers. Customers are, however, not usually willing to pay significantly more than for existing lower bandwidth connections. Thus good algorithms for finding cost-efficient network layouts are crucial.

The $b_{\max}$-*Survivable Network Design Problem* ($b_{\max}$-SNDP), which allows for modeling fiber-to-the-home scenarios, aims to efficiently extend an existing network to supply new customers. Here, two sets of customers are given. Standard customers which are denoted as type-1 or C_1 customers need to be connected by simple routes, while type-2 (or C_2) customers need a more reliable connection. For the latter, connectivity needs to be ensured even when a single link or routing node fails, i.e. pairs of node-disjoint paths are required. Furthermore, these redundancy requirements are occasionally relaxed by allowing a connection via a final non-redundant branch line that does not exceed a certain length $b_{\max}$. In this thesis, two different variants with respect to the objective of $b_{\max}$-SNDP are considered. In the operative planning task (OPT) a cheapest network feasibly connecting all given customers needs to be identified, while in the strategic simulation task (SST) return-on-investments are additionally considered. Here, the objective is to identify a most profitable solution supplying only a subset of all customers.

Two mixed integer programming models, which can be solved by branch-and-price, are discussed and compared to existing approaches theoretically as well as by a computational study. They are suitable for solving small and medium sized instances of $b_{\max}$-SNDP to proven optimality. One main contribution within this section is the usage of alternative dual-optimal solutions in the pricing subproblem, which significantly accelerates the solution of the linear relaxation of both models.

Furthermore, a new hybrid optimization approach based on Lagrangian relaxation as well as metaheuristic methods for approximately solving large instances are described. Computational results demonstrate the efficiency of the proposed solution approaches.

Especially in rural districts covering larger areas by fiber optic networks often does not pay off economically. Thus, a compromise between the bandwidth offered to individual customers and the resulting network construction costs has to be made. In such situations the fiber-optic infrastructure is typically extended to so-called mediation points that bridge the high-bandwidth network with an older lower-bandwidth network, i.e. fiber-to-the-curb. From an optimization point of view such scenarios can be modeled as variants of the Connected Facility Location Problem (ConFL) where new facilities, which correspond to the above mentioned mediation points, need to be installed and connected to each other. Furthermore, customer nodes need to be assigned to them.

The *Capacitated Connected Facility Location Problem* (CConFL), which is addressed in the second part of this thesis, extends ConFL by considering additional real world constraints such as those imposed by the individual bandwidth demands of customers and given maximum assignable demands for each potential facility location (mediation point). Furthermore, CConFL aims to determine a most profitable network instead of simply minimizing the resulting costs while mandatorily supplying all customers.

Four new mixed integer programming models for solving instances of CConFL to proven optimality are presented and their solution is discussed in detail. Subsequently, a theoretical comparison of the polyhedra corresponding to these four models is given. Furthermore, a Lagrangian relaxation method which is hybridized with local search and very large scale neighborhood search is described. The results obtained from a computational study indicate clear advantages for two of the proposed solution methods.

Acknowledgments

First of all I want to thank Prof. Günther Raidl who gave me the opportunity to do my PhD in his group and introduced me into the field of combinatorial optimization. Thank you for your outstanding support and for providing ideas and guidance whenever I was facing difficulties during my research. I am also very grateful to Prof. Ulrich Pferschy, who agreed to be my second supervisor. His valuable comments and suggestions improved all parts of this thesis.

I also owe gratitude to all my current and former colleagues from the Algorithms and Data Structures Group of the Vienna University of Technology. Thank you for exchanging scientific ideas and providing helpful suggestions, but also for great private discussions. It is a pleasure working with you.

Many thanks to my former colleagues from the Carinthia University of Applied Sciences for giving me the possibility to work in an applied research project, but still providing enough time to concentrate on basic models and algorithms.

Special thanks to all members of my family and all my friends for all their support during the last years.

Last but not least I want to thank my wife Romana for her continuous love and support. Thank you for your patience and encouragement. This thesis would not have been possible without you. I love you.

Contents

Chapter 1

Introduction

This thesis is concerned with network design problems (NDPs), which form a large and important subclass of combinatorial optimization problems (COPs). Network design problems arise in a multitude of real world applications such as the design of communication networks. Well known NDPs are for instance the minimum spanning tree (MST) problem where all nodes of a graph need to be connected to each other in a cost efficient way, and the Steiner tree problem (STP) on a graph where only a subset of all given nodes (terminals) need to be included in the resulting network, while the others are not mandatory. The STP has been used to model real world problems such as the cost efficient design of telecommunication networks or the planning and extension of heating networks, see e.g. [124].

Other NDPs such as the survivable network design problem (SNDP) or the bounded diameter minimum spanning tree problem (BDMST) are also concerned with ensuring a certain quality of service level. The SNDP is an extension of the STP where each terminal node has individual redundancy requirements ensuring connectivity in case of failures. The BDMST respects maximum transmission delays by imposing a bound on the maximum number of nodes (i.e. routers) between any two nodes. Next to respecting each problem's individual side constraints, the solutions should also ensure that the costs for constructing the resulting networks are as small as possible. A broad overview on NDPs occurring in the area of telecommunication is e.g. given in [160].

Furthermore, several problems from other domains such as transportation and logistics can be modeled as NDPs. One example is the traveling salesman problem (TSP), arguably one of the most prominent and well analyzed COPs. Here, given a

graph the objective is to find a cost-minimal round trip visiting each node exactly once.

Often NDPs can be modeled on a graph $G = (V, E)$ with a cost function $c : E \to \mathbb{R}$ on the edges. A solution is then given by some subgraph $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E$, respecting the problem's side constraints. The cost function c assigns an objective value $c(G') = \sum_{e \in E'} c_e$, to each feasible solution G' and the objective usually is to identify an optimal solution, which is feasible and has minimal (or maximal) objective value, compare [95]. Furthermore, a solution's objective value might also depend on its node set V', e.g. if the estimated profit due to connecting some node is also considered.

While the MST problem can be solved efficiently by the the classic algorithms of Prim [149] or Kruskal [113], most practically relevant network design problems are $\mathcal{NP}$-hard [69]. Thus, no efficient algorithm – that is an algorithm whose runtime is polynomially bounded on the input size – for solving instances of such a problem to optimality is currently known and it is likely that no efficient algorithms for $\mathcal{NP}$-hard problems do exist.

Nevertheless, due to their practical relevance it is very desirable to solve NDPs as good as possible. For instance, better algorithmic methods for cost-efficiently designing fiber-optic networks on the last mile might reduce the estimated costs to construct such a network. These savings might decide whether or not building a network does make sense from an economic perspective. Thus, developing better solution methods for COPs generally does not only improve the business opportunities of companies, but influences the life of everyone profiting from new infrastructure that potentially would not have been created otherwise.

During the last decades great progress in solving COPs has been made. Methods like dynamic programming [20], constraint programming [161], and especially the methods from (integer) linear programming such as branch-and-cut or branch-and-cut-and-price [16, 143] have shown to be able to solve moderately sized instances of difficult COPs to proven optimality.

Furthermore, heuristics and metaheuristics such as tabu search [74], simulated annealing [109], evolutionary algorithms [10], variable neighborhood search [82], or ant colony optimization [38] can be used to compute good but not necessarily optimal solutions for instances which cannot be solved by above mentioned exact methods in reasonable time.

More recently, so-called hybrid optimization methods aim to combine different optimization methods to profit from their different advantages while avoiding their individual drawbacks. A special class of hybrid approaches are matheuristics, which

combine metaheuristic approaches with mathematical programming based exact methods [151, 157].

1.1 Combinatorial Optimization Problems

As implicitly introduced above, a combinatorial optimization problem (COP) is a problem where some optimum solution is searched from a finite set of feasible solutions according to some function assigning a numerical objective value to each feasible solution. The following definitions are due to Aarts and Lenstra [1].

Definition 1 *A combinatorial optimization problem is specified by a set of problem instances and is either a minimization or a maximization problem.*

Definition 2 *An instance of a combinatorial optimization problem is a pair (S, c), where S is the set of feasible solutions and the cost or profit function c is a mapping $c : S \to \mathbb{R}$. The problem is to find a globally optimal solution $x^* \in S$. In case of a minimization problem this is a solution such that*

$$c(x^*) \leq c(x)$$

holds for all feasible solutions $x \in S$, while for a maximization problem

$$c(x^*) \geq c(x)$$

must hold.

1.2 Considered Problems

This thesis considers two NDPs suitable to model different variants of the extension of real world communication networks.

The $b_{\max}$-*Survivable Network Design Problem* ($b_{\max}$-SNDP), which can be used to model fiber-to-the-home scenarios, aims to efficiently extend an existing network to supply new customers. Here, next to standard customers, which are denoted as type-1 or C_1 customers and need to be connected by simple routes, a second set of type-2 (or C_2) customers is given, which need a more reliable connection. For these type-2 customers, connectivity needs to be ensured even when a single link or routing node fails, i.e. pairs of node-disjoint paths are required. Furthermore, these

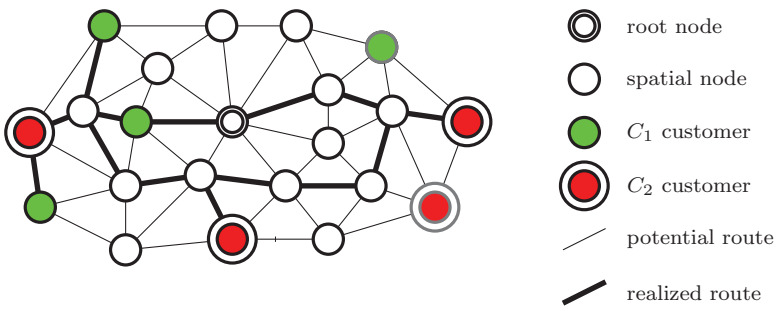


Figure 1.1: An exemplary solution to $b_{\max}$-SNDP.

redundancy requirements are occasionally relaxed by allowing a connection via a final non-redundant branch line that does not exceed a certain length.

In this thesis two different variants with respect to the objective of $b_{\max}$-SNDP are considered. On the one hand, in the operative planning task (OPT) a cheapest network feasibly connecting all given customers needs to be identified. On the other hand, in the strategic simulation task (SST) we are interested in identifying a most profitable solution supplying a subset of all customers only.

Figure 1.1 depicts an exemplary solution for the SST variant of $b_{\max}$-SNDP, connecting several customers of each type to the root node which models some existing network.

$b_{\max}$-SNDP is $\mathcal{NP}$-hard, since it corresponds to the $\mathcal{NP}$-hard Steiner tree problem (STP) on a graph [103] if only type-1 customers are considered.

Especially in rural areas, covering larger areas by fiber optic networks often does not pay off economically. To make a compromise between the bandwidth offered to individual customers and the resulting network construction costs providers frequently implement a fiber-to-the-curb strategy. Here, the fiber-optic network is extended to so-called mediation points that bridge the new network with an already existing lower-bandwidth network. While the original network is still used between a customer and its correspondingly assigned mediation point, the newly installed high-bandwidth routes are used in the remaining network. In this way, the bandwidth available for each customer can be significantly increased while the costs for constructing the network are typically much smaller compared to a fiber-to-the-home scenario.

Depending on additional side constraints that need to be considered, such scenarios can be modeled as different variants of the connected facility location problem

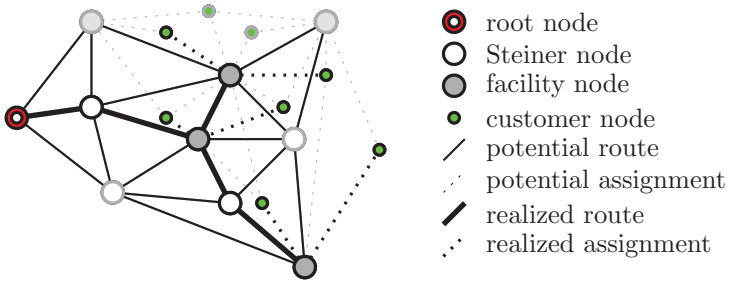


Figure 1.2: A solution to CConFL.

(ConFL) [125]. In ConFL, a set of facilities (mediation points) supplying the given customers needs to be installed and connected by a Steiner tree. In this thesis, the *Capacitated Connected Facility Location Problem* (CConFL) which extends ConFL by additionally considering capacity constraints on facilities and estimated profits due to customers prizes is considered. As for the SST variant of $b_{\max}$-SNDP, CConFL aims to identify a most profitable solution instead of mandatorily supplying all customers.

Figure 1.2 depicts an exemplary solution to CConFL.

CConFL is $\mathcal{NP}$-hard, since it is a combination of the STP [103] and the single source capacitated facility location problem (SSCFLP) [42] which are both $\mathcal{NP}$-hard.

1.3 Overview of the Thesis

The remainder of this thesis is organized as follows. First an overview of solution methods for COPs and their relevant theoretical background is presented in Chapter 2. Next to an introduction to exact methods based on (integer) linear programming, a short overview on (meta-) heuristics and hybrid optimization approaches is given.

Chapter 3 discusses solution approaches for the $b_{\max}$-SNDP. First two exact models which can be solved by branch-and-price are discussed and theoretically compared to existing approaches. Afterwards, a hybrid optimization approach for $b_{\max}$-SNDP based on Lagrangian relaxation as well as metaheuristic methods for approximately solving large instances are presented. Finally, test instances are described and computational results are given before conclusions are drawn.

Earlier versions of various parts of Chapter 3 have been published in

> Markus Leitner, Günther R. Raidl, and Ulrich Pferschy. Branch-and-Price for a Survivable Network Design Problem. Technical Report TR 186-1-10-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2010.
>
> Markus Leitner and Günther R. Raidl. Strong Lower Bounds for a Survivable Network Design Problem. In Proceedings of International Symposium on Combinatorial Optimization (ISCO 2010), Hammamet, Tunisia, March 2010.
>
> Markus Leitner, Günther R. Raidl, and Ulrich Pferschy. Accelerating Column Generation for a Survivable Network Design Problem. In M. G. Scutellá et al., editors, Proceedings of the International Network Optimization Conference 2009, Pisa, Italy, April 2009.
>
> Markus Leitner and Günther R. Raidl. Lagrangian Decomposition, Metaheuristics, and Hybrid Approaches for the Design of the Last Mile in Fiber Optic Networks. In M. J. Blesa et al., editors, Hybrid Metaheuristics 2008, volume 5296 of LNCS, pages 158-174, Malaga, Spain, October 2008. Springer-Verlag Berlin Heidelberg.

Furthermore, a talk on preliminary results has been given at the Austrian workshop on metaheuristics 5.

> Markus Leitner. A Lagrangian Relaxation Approach for the Design of the Last Mile in Real-World Fiber Optic Networks. Joint Workshop: Austrian Workshop on Metaheuristics 5 (AWM 5 '07) & Experimental Economics (EXLab), Graz, Austria, November 27, 2007.

Chapter 4 is devoted to CConFL. Here, four models for solving instances of CConFL to proven optimality are presented and theoretically compared. Furthermore, a Lagrangian relaxation method which is subsequently hybridized with local search and very large scale neighborhood search is proposed. After describing the used test instances, computational results are presented, before finally some conclusions are drawn.

Earlier versions of parts of Chapter 4 have been published in

> Markus Leitner and Günther R. Raidl. Branch-and-Cut-and-Price for Capacitated Connected Facility Location. Technical Report TR 186-1-10-01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2010.

> Markus Leitner and Günther R. Raidl. Combining Lagrangian Decomposition with Very Large Scale Neighborhood Search for Capacitated Connected Facility Location, Post-Conference Book of the Eight Metaheuristics International Conference MIC 2009, accepted 2010.
>
> Markus Leitner and Günther R. Raidl. A Lagrangian Decomposition Based Heuristic for Capacitated Connected Facility Location. In S. Voß and M. Caserta, editors, Proceedings of the 8th Metaheuristic International Conference (MIC 2009), Hamburg, Germany, July 2009.

Furthermore, a hybrid variable neighborhood search approach for a slightly different version of CConFL has been published in

> Markus Leitner and Günther R. Raidl. Variable Neighborhood Search for a Prize Collecting Capacity Constrained Connected Facility Location Problem. In Proceedings of the 2008 International Symposium on Applications and the Internet, SAINT 2008, pages 233-236, Turku, Finland, 2008. IEEE Computer Society.

Finally, in Chapter 5 overall conclusions are drawn.

Chapter 2

Methodologies

This chapter reviews concepts and approaches for solving combinatorial optimization problems. Following the usual classification into exact and heuristic methods, first a brief overview on exact methods is presented. Given an instance of some combinatorial optimization problem, these approaches aim to derive a proven optimal solution, that is an optimal solution of a considered instance together with a proof that no better feasible solution does exist.

However for $\mathcal{NP}$-hard problems, applying exact methods might involve an exponential number of steps, unless $\mathcal{P}$=$\mathcal{NP}$. Thus, solving instances of realistic size to proven optimality often turns out to be too time consuming in practice. Here, heuristic methods which will be reviewed in the second part of this chapter turn out to be useful. They can be used to generate good, but not necessarily optimal solutions with reasonable computational effort. Finally, the third part of this chapter is devoted to hybrid approaches which try to combine the advantages of different optimization methods while avoiding the drawbacks of each individual method.

2.1 Exact Methods

Whenever the considered instances and the available computational resources allow for, one should try to solve instances of a considered combinatorial optimization problem to proven optimality by applying exact methods.

Many COPs can be modeled as (mixed) integer linear programs (MIPs) and the solution methods proposed for solving MIPs have proven to be usually a good choice

when solving moderately sized instances of COPs. In the following a brief summary of important concepts from the field of (mixed integer) linear programming, focusing on those methods that will be used in the remaining chapters, is given. Afterwards, Lagrangian relaxation, a method which usually does not compute optimal solutions, but generates tight dual bounds of an optimal solution's value is discussed.

This section is based on the books of Bertsimas and Tsitsiklis [23], Bertsimas and Weismantel [24], Dantzig [45], Nemhauser and Wolsey [143], and Schrijver [162].

2.1.1 Linear Programming

As will be described in Section 2.1.2, for solving integer linear programs (IPs), it is usually necessary to repeatedly solve linear programs (LPs). Thus, this section is devoted to linear programming, which means to optimize over a linear objective function subject to a set of side constraints modeled as linear inequalities.

Formally, an LP in minimization form is defined by model (2.1)–(2.3), where A is an $m \times n$ matrix with rational entries, c is a rational vector of dimension n, and b a rational vector of size m.

$$z^{\mathrm{LP}} = \min \quad c^{\mathrm{T}}x \tag{2.1}$$
$$\text{s.t.} \quad Ax \geq b \tag{2.2}$$
$$x \in \mathbb{R}^n_+ \tag{2.3}$$

Note that it is possible to describe any LP by an equivalent model where all side constraints are written as equalities instead of inequalities by adding so-called slack and surplus variables. Furthermore, any minimization problem can be transformed into an equivalent maximization problem and vice versa. However, since all problems considered in this thesis are minimization problems, this section considers the minimization variant only.

Each LP can be alternatively written in a more compact way in the form

$$z^{\mathrm{LP}} = \min\{c^{\mathrm{T}}x : Ax \geq b, x \in \mathbb{R}^n_+\}. \tag{2.4}$$

Duality

In the following, the concept of duality in linear programming and some of its implications are introduced. For any *primal* LP (2.1)–(2.3), we can state its *dual* LP (2.5)–(2.7).

$$w^{\mathrm{LP}} = \max \quad u^{\mathrm{T}} b \tag{2.5}$$
$$\text{s.t.} \quad u^{\mathrm{T}} A \leq c^{\mathrm{T}} \tag{2.6}$$
$$u \in \mathbb{R}^m_+ \tag{2.7}$$

Let (P) denote the primal LP (2.4). Then its dual (D) can be stated in a compact way as:

$$\text{(D)} \quad w^{\mathrm{LP}} = \max\{u^{\mathrm{T}} b : u^{\mathrm{T}} A \leq c^{\mathrm{T}}, u \in \mathbb{R}^m_+\}. \tag{2.8}$$

The following proposition unveils that it is in fact not important which of the LPs we denote as the primal and which as the dual.

Proposition 1 *The dual of the dual problem is the primal problem.*

A vector $x^* \in \mathbb{R}^n_+$ is called *primal feasible*, if it satisfies all side constraints of the primal problem, i.e. if $Ax^* \geq b$ does hold. Analogously, a vector $u^* \in \mathbb{R}^m_+$ is called *dual feasible* if $(u^*)^{\mathrm{T}} A \leq c^{\mathrm{T}}$. Using the concepts of primal and dual feasibility, we can state the weak duality theorem, see e.g. [143].

Theorem 1 (Weak Duality) *Let (P) denote a primal LP and (D) its corresponding dual problem. Then, $c^{\mathrm{T}} x^* \geq z^{\mathrm{LP}} \geq w^{\mathrm{LP}} \geq (u^*)^{\mathrm{T}} b$ holds if x^* is primal feasible and u^* is dual feasible.*

In particular the weak duality theorem implies that if a primal problem (P) is unbounded – i.e. $z^{\mathrm{LP}} = -\infty$ in case of a minimization problem – then its dual (D) is infeasible.

The strong duality theorem shows that if for any primal dual pair of linear programs, either the primal or the dual has a finite optimal solution, then the optimal solution to the other is finite too and has the same objective value.

Theorem 2 (Strong Duality) *If either z^{LP} or w^{LP} is finite, then both (P) and (D) have finite optimal solution values and $z^{\mathrm{LP}} = w^{\mathrm{LP}}$.*

Corollary 1 *For any primal dual pair of LPs (P) and (D) there are exactly four possibilities*

- *both (P) and (D) have finite and equal optimal solution values, i.e. $z^{\mathrm{LP}} = w^{\mathrm{LP}}$*
- *(P) is unbounded – i.e. $z^{\mathrm{LP}} = -\infty$ – and (D) is infeasible*
- *(D) is unbounded – i.e. $w^{\mathrm{LP}} = \infty$ – and (P) is infeasible*
- *both (P) and (D) are infeasible*

Another important relation between primal and dual solutions is given by the *complementary slackness* conditions.

Proposition 2 *If x^* is an optimal solution of (P) and u^* is an optimal solution of (D), then*

$$x_j^* \left((u^*)^{\mathrm{T}} A - c^{\mathrm{T}}\right)_j = 0 \quad \textit{for all } j, \text{ and}$$
$$u_i^* (b - Ax)_i = 0 \quad \textit{for all } i$$

One important theorem that can be proved using LP-duality and complementary slackness is the *max-flow min-cut* theorem [58]. It states, that given a directed graph $D = (V, A)$ with capacities on the arcs, the maximum flow between two nodes $r, s \in V$ is equivalent to the minimum capacity of an r-s-cut, compare [162].

Geometric Interpretation of Linear Programs

This section discusses important concepts and definitions with respect to the geometric interpretation of linear programs. These form the basis of the simplex algorithm for solving LPs and for further properties that will be relevant in the following sections.

The presentation of this part follows Nemhauser and Wolsey [143].

Definition 3 *A* polyhedron *$\mathcal{P} \subseteq \mathbb{R}^n$ is a set of points that satisfy a finite number of linear inequalities, i.e. $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \geq b\}$ where A is an $m \times n$ matrix and b is vector in $\mathbb{R}^m$.*

Since the side constraints of any LP (2.4) can be described in the form $Ax \geq b$, the set of feasible solutions to (2.4) obviously is a polyhedron. Each polyhedron can be either infinitely large (unbounded) or bounded, in which case it is called a polytope.

Definition 4 *A polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$ is bounded if there exists a scalar $\omega \in \mathbb{R}_+$ such that $\mathcal{P} \subseteq \{x \in \mathbb{R}^n : -\omega \leq x_j \leq \omega \text{ for } j \in 1, \ldots, n\}$. A bounded polyhedron is called* polytope.

Another property, which will turn out to be important is that a polyhedron is a convex set.

Definition 5 *$T \subseteq \mathbb{R}^n$ is a* convex set *if $x, y \in T$ implies that $\lambda x + (1 - \lambda)y \in T$, for all $0 \leq \lambda \leq 1$.*

Proposition 3 *A polyhedron is a convex set.*

For the following, we assume that A does not contain redundant equations, i.e. $rank(A) = m \leq n$ and to be given an LP with equality constraints only, i.e.

$$\min\{c^{\mathrm{T}}x : Ax = b, x \in \mathbb{R}^n_+\}. \tag{2.9}$$

As already mentioned any LP can be transformed into an equivalent LP corresponding to (2.9) by adding slack and surplus variables. Hence, above assumption can be taken without loss of generality.

Let a_j, $1 \leq j \leq n$, be the j-th column of A. Then A contains a nonsingular $m \times m$ submatrix $A_B = (a_{B_1}, \ldots, a_{B_m}) = (B_1, \ldots, B_m)$. By reordering the columns of A, we can write A as $A = (A_B, A_N)$ such that $A_B x_B + A_N x_N = b$ with $x = (x_B, x_N)$. Then a solution to (2.9) is given by $x_B = A_B^{-1} b$ and $x_N = 0$.

Definition 6 *Let A_B be nonsingular $m \times m$ submatrix of A which is called* basis. *Then $x = (x_B, x_N)$, $x_B = A_B^{-1} b$, $x_N = 0$, is a* basic solution *of the system $Ax = b$, where x_B is the vector of* basic variables *and x_N the vector of* nonbasic variables. *If $A_B^{-1} b \geq 0$, (x_B, x_N) is called a* basic primal feasible solution *and A_B is called a* primal feasible basis.

For the presentation of the simplex algorithm the definitions of adjacent basic solutions and degeneracy are further relevant [143].

Definition 7 *Two bases A_B, $A_{B'}$ are* adjacent *if they differ in only one column. If A_B and $A_{B'}$ are adjacent, the two basic solutions they define are said to be adjacent.*

Definition 8 *A primal basic feasible solution $x = (x_B, x_N)$, $x_N = 0$, is* degenerate *if $(x_B)_i = 0$, for some i.*

Before being able to show that the set of basic feasible solution of an LP corresponds to the set of vertices of its corresponding polyhedron, a few more definitions including the important concept of valid inequalities are necessary.

Definition 9 *A polyhedron $\mathcal{P}$ is of dimension k if the number of affinely independent points in $\mathcal{P}$ is $k+1$, which is denoted as $dim(\mathcal{P}) = k$.*

Definition 10 *The inequality $a^{\mathrm{T}}x \geq b_j$ is called a* valid inequality *for a set $\mathcal{P}$ if it is satisfied by all points in $x \in \mathcal{P}$.*

Definition 11 *If $a^{\mathrm{T}}x \geq b_j$ is a valid inequality for $\mathcal{P}$ and $F = \{x \in \mathcal{P} \mid a^{\mathrm{T}}x = b_j\}$, F is called a* face *of $\mathcal{P}$.*

Definition 12 *A face F of $\mathcal{P}$ is a* facet *of $\mathcal{P}$ if $dim(F) = dim(\mathcal{P}) - 1$.*

Definition 13 *Let $\mathcal{P}$ be a polyhedron. A vector $x \in \mathcal{P}$ is an* extreme point *of $\mathcal{P}$ if we cannot find two vectors $y, z \in \mathcal{P}$, $x \neq y$, $x \neq z$, and a scalar $0 \leq \lambda \leq 1$, such that $x = \lambda y + (1-\lambda)z$.*

Note that one could alternatively characterize an extreme point of $\mathcal{P}$ as a zero-dimensional face.

Corollary 2 *Each polyhedron has only a finite number of extreme points.*

Definition 14 *Let $\mathcal{P}$ be a polyhedron. A vector $x \in \mathcal{P}$ is a* vertex *of $\mathcal{P}$ if there exists some vector c such that $c^{\mathrm{T}}x \leq c^{\mathrm{T}}y$ holds for all $y \in \mathcal{P}$, $y \neq x$.*

Theorem 3 *Let $\mathcal{P}$ be a nonempty polyhedron and let $x^* \in \mathcal{P}$. Then the following are equivalent:*

- *x^* is a vertex*
- *x^* is an extreme point*
- *x^* is a basic feasible solution*

From Corollary 2 and Theorem 3 we conclude that the number of basic feasible solutions is finite for any LP and due to Theorem 4 at least one of them is an optimal solution.

Theorem 4 *Consider the linear programming problem of minimizing $c^{\mathrm{T}}x$ over a polyhedron $\mathcal{P}$. Suppose that $\mathcal{P}$ has at least one extreme point and that there exists an optimal solution. Then, there exists an optimal solution which is an extreme point of $\mathcal{P}$.*

Theorem 5 *A nonempty and bounded polyhedron is the convex hull of its extreme points.*

Comparing Linear Programming Formulations

To theoretically evaluate and compare different LP formulations for a problem, usually their corresponding polyhedra are compared. However, since different formulations often involve different design variables one needs to project each of the polyhedra onto some common subspace, typically defined by the variables used in all formulations that should be compared.

Definition 15 *Let $\mathcal{P} = \{(x, y) : Dx + By \geq d\}$ be a polyhedron. The* projection *of $\mathcal{P}$ on the set of x-variables is defined as*

$$proj_x(\mathcal{P}) = \{x \mid \text{there exists some } y \text{ with } (x, y) \in \mathcal{P}\}$$

Using Definition 15 we can define the concept of domination between polyhedra.

Definition 16 *Given two LP formulations P and P' with associated polyhedra $\mathcal{P}$ and $\mathcal{P}'$, respectively. Let furthermore x be a set of variables included in both P and P'. Then P* dominates *P' if $proj_x(\mathcal{P}) \subseteq proj_x(\mathcal{P}')$ and* strictly dominates *P' if $proj_x(\mathcal{P}) \subsetneq proj_x(\mathcal{P}')$.*

It is also common to say P is stronger (or tighter) than P' if P strictly dominates P'.

However, the concepts of dominance and strict dominance can also be established for LP formulations P, P' that do not involve a common subset of variables. In this case P dominates P' if there exists a transformation that maps any feasible solution of P into a feasible solution of P'. If on the contrary, no such transformation from P' to P exists, P strictly dominates P'.

Solving Linear Programs

Linear programs can be solved in polynomial time using the ellipsoid method [107] or interior point methods [102]. Although it might involve an exponential number of steps [23], the simplex algorithm proposed by Dantzig in 1947 [44] is still widely used due to its good practical performance.

The main idea of the simplex algorithm is to start from an initial basic feasible solution and to iteratively move from one basic feasible solution to an adjacent one in the so-called *pivoting* step. Given a basic feasible solution $x = (x_B, x_N)$, in the pivoting step, exactly one basic variable $x_i \in x_B$ leaves the basis and one nonbasic variable $x_j \in x_N$ enters the basis, compare Definition 7. For deciding which of the variables should leave and enter the basis, the *reduced costs* $\bar{c}_j$ of each variable $x_j \in x$ are considered.

Definition 17 *Let x be a basic solution, B its associated basis matrix, and c_B the vector of costs of the basic variables. For each j, we define the* reduced costs $\bar{c}_j$ *of the variable x_j as*

$$\bar{c}_j = c_j - c_B^{\mathrm{T}} B^{-1} A_j.$$

While the reduced costs of all basis variables are obviously equal to zero, Theorem 6 defines conditions for a basic feasible solution to be optimal.

Theorem 6 *Let $\bar{c}$ be the vector of reduced costs corresponding to a basic feasible solution x and its associated basis matrix B.*

- *If $\bar{c}_j \geq 0$, $\forall j$, then x is optimal.*
- *If x is optimal and non-degenerate, then $\bar{c}_j \geq 0$, $\forall j$.*

If x is non-degenerate, by exchanging a basic variable by a nonbasic variable with negative reduced costs, we obtain a basic feasible solution x' whose cost is less than those of x. Since the number of basic feasible solutions is finite, the simplex algorithm will terminate after a finite number of pivoting steps in the non-degenerated case.

Note that, Theorem 6 allows for negative reduced costs for some variable in an optimal but degenerated solution. However, similar optimality criterions considering degeneracy do exist. Nevertheless, in presence of degeneracy – i.e. if at least one basis variable is equal to zero – a pivoting step might not modify the solution and thus cycling might occur. To always ensure the termination of the simplex method, one has to prevent cycling by considering so-called *pivoting rules* such as the lexicographic pivoting rule or the smallest subscript rule, also known as Bland's rule.

Geometrically speaking, the simplex algorithm starts by a vertex of the polyhedron corresponding to the given LP and iteratively moves to a neighboring vertex with better objective value. Since, we are optimizing over a convex set (see Theorem 5) the simplex algorithm terminates in a vertex – i.e. in a basic feasible solution – corresponding to a global optimal solution.

Thus, if the simplex algorithm starts by an initial feasible solution, it will terminate with an optimal solution after a finite number of steps. By solving an auxiliary linear program involving additional artificial variables in its first phase, the so-called two-phase simplex method guarantees to find an initial basic feasible solution if it exists. The two-phase simplex then proceeds with the standard simplex method as presented above in its second phase.

A more detailed description of the simplex method can for instance be found in [23].

2.1.2 Integer Linear Programming

In many COPs arising in real world applications, decision variables need to have integral values rather than continuous ones as assumed in the previous section. These usually can be modeled as concrete instances of the following *integer linear program* (IP)

$$\text{(IP)} \quad z^{\text{IP}} = \min\{c^{\text{T}}x : x \in X\} \tag{2.10}$$

where $X = \mathcal{P} \cap \mathbb{Z}^n_+$ and $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \geq b\}$. As for the LP (2.4), A is an $m \times n$ matrix, c a vector of dimension n, and b a vector of dimension m, each of which having rational data. Alternatively, we can write (2.10) as (2.11)–(2.13).

$$z^{\mathrm{IP}} = \min \quad c^{\mathrm{T}}x \tag{2.11}$$
$$\text{s.t.} \quad Ax \geq b \tag{2.12}$$
$$x \in \mathbb{Z}^n_+ \tag{2.13}$$

Further important formulations for modeling real world problems include *mixed integer programs* (MIPs) where only some design variables are restricted to be integral and so-called *0-1 integer problems* which are also called *binary integer problems* (BIPs) where all variables are restricted to be binary, i.e. $x_i \in \{0, 1\}$, $1 \leq i \leq n$.

However, for simplicity we concentrate on pure integer linear programs (IPs) in the following.

Note that the LP (2.4) corresponds to the IP (2.10) when removing the integrality conditions. Thus, (2.4) is called the *linear programming relaxation* (LP relaxation) of (2.10). Obviously, any feasible solution of (2.10) is feasible for (2.4) too. Furthermore for any IP, the optimal solution value of its LP relaxation is a lower bound of its optimal solution value, i.e. $z^{\mathrm{LP}} \leq z^{\mathrm{IP}}$.

The following theorem shows a further important relation between LPs and IPs.

Theorem 7 *Let $\mathcal{P} = \{x \in \mathbb{R}^n_+ : Ax \geq b\}$, where A is a rational $m \times n$ matrix and b is a rational vector of dimension m and $X = \mathcal{P} \cap \mathbb{Z}^n_+$. Then conv$(X)$ is a rational polyhedron.*

Theorem 7 in particular implies that we can solve the IP (2.10), by solving the LP

$$\min\{c^{\mathrm{T}}x \mid x \in \mathrm{conv}(X)\}$$

However, finding a (compact) description of conv(X) is difficult for $\mathcal{NP}$-hard combinatorial optimization problems. Thus other solution methods for solving IPs are needed. Note that in general solving IPs is $\mathcal{NP}$-hard.

2.1.3 LP based Branch-and-Bound

Branch-and-bound is based on intelligent, restricted enumeration of an IPs feasible solutions [144]. Given some IP $\mathcal{F}$, branch-and-bound is based on successively partitioning the search space into easier subproblems (*branching*) and the computation of lower bounds $\underline{z}_i$ for each considered subproblem $\mathcal{F}_i$, i.e. *bounding*. Furthermore, a global upper bound U is maintained and updated due to found feasible solutions. Hence, after solving a subproblem $\mathcal{F}_i$ the following cases are possible:

- $\underline{z}_i = U$: The optimal solution to $\mathcal{F}_i$ has been computed.
- $\underline{z}_i > U$: The lower bound is higher than the global upper bound. Thus, $\mathcal{F}_i$ does not contain a new best solution and can be pruned.
- $\underline{z}_i < U$: $\mathcal{F}_i$ might contain a new best solution and thus needs to be further partitioned.

As summarized in Algorithm 2.1, in LP based branch-and-bound the LP relaxation of a current subproblem is solved to generate lower bounds for a subproblem.

Algorithm 2.1: Generic LP based branch-and-bound algorithm

$U = \infty$ *// global upper bound*
$L = \{\mathcal{F}\}$ *// list of unprocessed subproblems*
while $L \neq \emptyset$ **do**
  choose a subproblem $\mathcal{F}_i \in L$
  $L = L \setminus \{\mathcal{F}_i\}$
  solve the LP relaxation of $\mathcal{F}_i$ to obtain its solution x_i^{LP} and $\underline{z}_i$
  if $\mathcal{F}_i$ is infeasible **then**
    prune
  if $\underline{z}_i > U$ **then**
    prune
  if x_i^{LP} is integer **then**
    $U = \underline{z}_i$
    new incumbent solution $x^* = x_i^{\mathrm{LP}}$
    prune
  else
    create subproblems $\mathcal{F}_i^{(1)}, \ldots \mathcal{F}_i^{(k)}$ of $\mathcal{F}_i$
    $L = L \cup \{\mathcal{F}_i^{(j)} \mid 1 \leq j \leq k\}$
x^* is the optimal solution of $\mathcal{F}$

Branching

An important design aspect of branch-and-bound approaches is how to break a problem $\mathcal{F}_i$ into subproblems. While in general any meaningful number of subproblems can be generated, most approaches used in practice generate two subproblems only by rounding a single fractional variable. If x_j is a variable, for which its optimal solution x_j^* to the LP relaxation of $\mathcal{F}_i$ is not integral, two subproblems can be generated by adding the two constraints

$$x_j \leq \lfloor x_j^* \rfloor \quad \text{and} \quad x_j \geq \lceil x_j^* \rceil, \text{ respectively.}$$

Thus, $\mathcal{F}_i^{(1)} = \{x \mid x \in \mathcal{F}_i \wedge x_j \leq \lfloor x_j^* \rfloor\}$ and $\mathcal{F}_i^{(2)} = \{x \mid x \in \mathcal{F}_i \wedge x_j \geq \lceil x_j^* \rceil\}$. Which among all fractional variables to chose best is a difficult question. Common strategies are most infeasible, pseudocost, and strong branching [3].

Subproblem Selection

Finally, one needs to define which among the currently open subproblems to consider next. Common rules that a priory define an order include *depth first search* and *breadth first search* [24] while *best upper bound* and *best estimate* additionally consider each subproblems bounds, see e.g. [143] for a more detailed discussion.

2.1.4 Cutting Plane Methods and Branch-and-Cut

Theorem 7 shows that any IP can be described as a corresponding LP. However, as already mentioned finding such a description is usually not possible in practice for $\mathcal{NP}$-hard problems. In the following, cutting plane methods and their embedding in branch-and-bound methods are discussed.

Cutting Plane Methods

Cutting plane methods are based on the concept of *valid inequalities* as introduced by Definition 10. As shown by Algorithm 2.2 one way to solve the IP (2.10) is to iteratively solve its linear relaxation and subsequently identify and add valid inequalities for IP violated by the current LP solution in the *separation problem* as long as the optimal LP solution x^* is not integral.

Definition 18 *Given an IP (2.10) and* $x^* \in \mathbb{R}^n_+$, $x^* \notin conv(X)$. *Then the* separation problem *is to find a valid inequality* $a^{\mathrm{T}}x \geq b_j$ *that is violated by* x^*.

Algorithm 2.2: Generic cutting plane algorithm

```
solve the LP relaxation (LP) of the IP (2.10)
Let x* be an optimal solution to (LP)
while x* is not feasible for the IP (2.10) do
    find a valid inequality for (2.10) which is violated by x*
    add this inequality to (LP)
    resolve (LP)
```

In general the number of valid inequalities that need to be added can be exponentially large and one might need to terminate Algorithm 2.2 before finding an integer solution. In this case a – often extremely tight – lower bound x^* instead of an optimal solution is generated by Algorithm 2.2.

Branch-and-Cut

LP based branch-and-bound approaches often perform relatively bad since too many branching nodes need to be considered. In branch-and-cut methods, cutting planes are generated at each node of the branch-and-bound tree to tighten the lower bounds.

2.1.5 Column Generation and Branch-and-Price

As opposed to cutting plane methods which start with a small set of constraints and subsequently identify and add valid inequalities, in column generation one dynamically generates variables on demand only.

Column Generation

If a linear program involves a too large number of variables, its linear relaxation cannot be solved directly. Formulations based on a (exponentially) large set of variables occur for instance after reformulating an existing model by applying Dantzig-Wolfe decomposition [46] to tighten its LP relaxation.

For solving an LP involving too many variables, delayed column generation as summarized by Algorithm 2.3 is typically used. Instead of initially considering a subset

of all constraints, in column generation one starts with a small subset of variables and iteratively adds new variables to the model determined by solving the pricing subproblem.

Column generation has been first used by Gilmore and Gomory [71, 72] for the cutting stock problem and has been applied to a large number of problems since then, see e.g. [16, 49, 130] for recent and comprehensive surveys.

Consider the linear program (2.14)–(2.16) to which we denote as the (linear) master problem (MP).

$$\begin{aligned} \text{(MP)} \quad \min \quad & \sum_{j \in J} c_j x_j && && (2.14) \\ \text{s.t.} \quad & \sum_{j \in J} A_j x_j \geq b && && (2.15) \\ & x_j \geq 0 && \forall j \in J && (2.16) \end{aligned}$$

If J is too large, we cannot solve (MP) directly. Thus we define the so-called restricted master problem (RMP) (2.17)–(2.19) where we consider only a small subset of variables x_j, $j \in \tilde{J} \subsetneq J$, otherwise (RMP) corresponds to (MP).

$$\begin{aligned} \text{(RMP)} \quad \min \quad & \sum_{j \in \tilde{J}} c_j x_j && && (2.17) \\ \text{s.t.} \quad & \sum_{j \in \tilde{J}} A_j x_j \geq b && && (2.18) \\ & x_j \geq 0 && \forall j \in \tilde{J} && (2.19) \end{aligned}$$

According to Theorem 6, one needs to consider additional variables x_j, $j \in J \setminus \tilde{J}$, as long as at least one such variable has negative reduced costs. Given the vector $u \geq 0$ of dual variable values, the reduced costs for a variable x_j, $j \in J$, are

$$\overline{c}_j = c_j - u^{\mathrm{T}} a_j. \tag{2.20}$$

The pricing problem is to find at least one variable x_j, $j \in J \setminus \tilde{J}$, yielding negative reduced costs, or to prove that no such variable exists.

Next to the classical Dantzig rule, in which the variable with the most negative reduced cost is added in each iteration, various other schemes like full, partial, or multiple pricing have been considered [37].

Algorithm 2.3: General column generation algorithm.

choose a subset of variables x_j, $j \in \tilde{J} \subseteq J$, defining the RMP
solve RMP
while a variable x_j, $j \in J \setminus \tilde{J}$ with $\overline{c}_j < 0$ exists **do**
    determine a variable x_j, $j \in J \setminus \tilde{J}$, with negative reduced costs $\overline{c}_j$
    add x_j to RMP
    resolve RMP

Despite the elegant idea, solving large scale linear programs by column generation often involves computational difficulties. Vanderbeck [172] describes five major efficiency problems that often occur in simplex based column generation. These include the generation of irrelevant columns in the beginning (*heading-in effect*), primal degeneracy leading to the *plateau effect*, and slow convergence (*tailing-off effect*).

Different approaches – to which typically is referred to as stabilization techniques – have been proposed to overcome these problems, see e.g. [130] for a review. Problem independent approaches include bounding the dual variable values [4], the use of the boxstep method [134, 135], and the more flexible concept of stabilized column generation [22, 53]. Ben Amor et. al [22] suggested the use of so-called dual optimal inequalities to accelerate and speed-up the solution process and applied this concept to the cutting stock problem. This problem specific concept has also been applied to the three-stage two-dimensional bin packing problem by Puchinger et al. [152].

Branch-and-Price

Similar to branch-and-cut, which is the combination of branch-and-bound and the generation of cutting planes at each node of the branch-and-bound tree, branch-and-price combines column generation with branch-and-bound. Here, column generation is used for solving each subproblem's LP relaxation.

Special care must be taken with respect to branching decisions. In general branching should be performed on the original design variables – i.e. before reformulating the problem – since branching on the – potentially exponentially – large set of variables usually leads to strong asymmetries in the search space resulting in a large amount of branching nodes that need to be considered. Additional constraints imposed by branching decisions might, however, complicate the pricing subproblem. Branch-and-price approaches in which the pricing subproblems structure does not change due to branching decisions are usually referred to as *robust*.

2.1.6 Branch-and-Cut-and-Price

As suggested by its name, branch-and-cut-and-price refers to approaches where the generation of cutting planes is performed together with column generation at each node of a branch-and-bound search, see e.g. [15, 16]. The combination of row and column generation was probably first used by Nemhauser and Park [142] for the edge coloring problem. In their approach, however, the structure of the pricing subproblem does change due to additionally generated cutting planes. More recently, several successful so-called *robust branch-and-cut-and-price* algorithms have been proposed where the pricing subproblems structure does not change due to cut generation, see e.g. [48, 171].

2.1.7 Lagrangian Relaxation

Instead of computing an optimal solution to a given instance of a COP, Lagrangian relaxation (LR) is a technique that can be used to derive lower bounds of the optimal objective value. As will be discussed in the following, these lower bounds might be tighter than a models LP relaxation.

Lagrangian relaxation whose name has been introduced by Geoffrion [70] has been first applied to the traveling salesman problem by Held and Karp [90, 91]. Due to the efficiency of some of the early approaches, the method has received considerable attention since then, see e.g. [65] for a survey. Typically, Lagrangian relaxation based approaches are not only used to derive dual bounds but additionally incorporate Lagrangian heuristics to derive primal feasible solutions during the course of solving the so-called Lagrangian dual problem. Frequently, Lagrangian relaxation approaches are hybridized with metaheuristic methods [86, 146] to further improve the obtained primal solutions or incorporated into a branch-and-bound framework to compute proven optimal solutions, see e.g. [94].

Consider the IP given by model (2.21)–(2.24) whose m side constraints consist of m_1 "relatively easy" constraints $Bx \geq d$ and $m_2 = m - m_1$ "nasty" constraints $Dx \geq f$ which significantly complicate the solution of the model.

$$
\begin{aligned}
z^{\mathrm{IP}} = \min \quad & c^{\mathrm{T}}x && (2.21)\\
\text{s.t.} \quad & Bx \geq d && (2.22)\\
& Dx \geq f && (2.23)\\
& x \in \mathbb{Z}^n_+ && (2.24)
\end{aligned}
$$

The Lagrangian relaxation (LR(λ)) of model (2.21)–(2.24) with respect to constraints (2.23) is defined by model (2.25)–(2.27). Here, model (2.21)–(2.24) is relaxed by dropping the complicating constraints (2.23) and a corresponding term $\lambda(f - Dx)$, which penalizes violations of these constraints is added to the objective function. Vector $\lambda^{\mathrm{T}} \in \mathbb{R}_{+}^{\mathrm{m_2}}$ consists of the so-called Lagrangian multipliers $\lambda_i \geq 0$, $1 \leq i \leq m_2$.

$$\begin{aligned} (\mathrm{LR}(\lambda)) \quad z^{\mathrm{LR}}(\lambda) = \min \quad & c^{\mathrm{T}}x + \lambda^{\mathrm{T}}(f - Dx) && (2.25)\\ \text{s.t.} \quad & Bx \geq d && (2.26)\\ & x \in \mathbb{Z}_{+}^{n} && (2.27) \end{aligned}$$

Model (LR(λ)) is a relaxation of model (2.21)–(2.24) since any feasible solution x^* to (2.21)–(2.24) obviously is feasible for model (2.25)–(2.27), too. Furthermore, $z^{\mathrm{LR}}(\lambda)$ is a lower bound of z^{IP} if $\lambda > 0$.

The *Lagrangian dual problem* (LDP) describes the resulting optimization problem which is to find the best possible lower bound, i.e.

$$(\mathrm{LDP}) \quad z^{\mathrm{LDP}} = \max_{\lambda \geq 0} z^{\mathrm{LR}}(\lambda). \tag{2.28}$$

It can be shown that the lower bounds obtained by solving (LDP) are tighter than those of the simpler linear relaxation of model (2.21)–(2.24) if and only if model (LR(λ)) does not possess the *integrality property.*

Definition 19 *An IP has the* integrality property *if its optimal solution is equal to the optimal solution of its linear relaxation.*

Thus, when applying Lagrangian relaxation one should generally try to relax a set of constraints such that the resulting formulation does not possess the integrality property. However, it is also important that, given some Lagrangian multipliers λ, model (LR(λ)) can be solved more efficiently than the original model.

Solving the Lagrangian Dual Problem

For solving the Lagrangian dual problem (LDP) one needs to determine optimal Lagrangian multipliers λ^* such that $z^{\mathrm{LR}}(\lambda^*) \geq z^{\mathrm{LR}}(\lambda)$, $\forall \lambda \geq 0$. Since $z^{\mathrm{LR}}(\lambda)$ is piecewise linear and convex, subgradient based methods are well suited for approximately solving it [18]. Among different variants, the volume algorithm [13] has proven to outperform other methods on a number of occasions, see e.g. [11, 86]. However, it sometimes might converge too quickly in which case it has been outperformed by other variants, see e.g. [29, 85].

Lagrangian Heuristic

When solving (LDP) by some subgradient based algorithm, one derives values for all variables x in each iteration. However, these variable values usually do not correspond to a primal feasible solution, since some of the problem's constraints have been relaxed. As mentioned earlier, Lagrangian relaxation approaches often incorporate so-called *Lagrangian heuristics* which usually try to derive primal feasible solutions based on the actual variable values. Thus, both lower and upper bounds are generated and when embedded in a branch-and-bound procedure, Lagrangian relaxation can even be used for computing proven optimal solutions.

Lagrangian Decomposition

Lagrangian decomposition (LD) is a special form of Lagrangian relaxation where a problem is decomposed into several subproblems by duplicating some variables and adding corresponding coupling constraints – which are subsequently relaxed again – to the model, compare [86].

To decompose model (2.21)–(2.24) additional variables $y \in \mathbb{Z}^n_+$ and corresponding coupling constraints (2.31) are included, yielding model (2.29)–(2.34).

$$
\begin{aligned}
z^{\mathrm{IP}} = \min \quad & c^{\mathrm{T}} x && (2.29)\\
\text{s.t.} \quad & Bx \geq d && (2.30)\\
& x = y && (2.31)\\
& Dy \geq f && (2.32)\\
& x \in \mathbb{Z}^n_+ && (2.33)\\
& y \in \mathbb{Z}^n_+ && (2.34)
\end{aligned}
$$

Relaxing constraints (2.31) in the usual Lagrangian way and associating nonnegative Lagrangian multipliers λ to them, yields the relaxed model (LD(λ)).

$$
\begin{aligned}
(\mathrm{LD}(\lambda)) \quad z^{\mathrm{LD}}(\lambda) = \min \quad & c^{\mathrm{T}}x + \lambda^{\mathrm{T}}(y - x) && (2.35)\\
\text{s.t.} \quad & Bx \geq d && (2.36)\\
& Dy \geq f && (2.37)\\
& x \in \mathbb{Z}_+^n && (2.38)\\
& y \in \mathbb{Z}_+^n && (2.39)
\end{aligned}
$$

Obviously, (LD(λ)) decomposes into two subproblems (2.40)–(2.42) and (2.43)–(2.45), respectively, which can be solved independently.

$$
\begin{aligned}
\min \quad & c^{\mathrm{T}}x - \lambda^{\mathrm{T}}x && (2.40)\\
\text{s.t.} \quad & Bx \geq d && (2.41)\\
& x \in \mathbb{Z}_+^n && (2.42)
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & \lambda^{\mathrm{T}}y && (2.43)\\
\text{s.t.} \quad & Dy \geq f && (2.44)\\
& y \in \mathbb{Z}_+^n && (2.45)
\end{aligned}
$$

Lagrangian decomposition is sometimes also used to denote Lagrangian relaxation approaches where the constraints coupling the various subproblems are already present in the original model, i.e. no additional artificial variables need to be introduced.

2.2 Heuristic Methods

Realistic instances of $\mathcal{NP}$-hard COPs often cannot be solved to proven optimality due to the available computational resources. In such situations, heuristic approaches which provide good but not necessarily optimal solutions often turn out to be the only possible choice. The remainder of this section is organized as follows. First the concept of constructive heuristics which aim to generate initial feasible solutions from scratch is introduced, before considering approximation algorithms additionally providing bounds on the maximum gap to an optimal solution. After introducing local search, finally metaheuristic methods are discussed.

2.2.1 Constructive Heuristics

Constructive heuristics build a solution to some COP from scratch. Typically, they start from an empty solution and iteratively add certain solution components afterwards, until a feasible solution has been computed. Often, the decision which component to add next is based on some weighting function and simply the component with the smallest weight is taken. This yields the family of *greedy heuristics*. While typically being relatively fast, the solutions obtained due to such heuristics might not meet the requirements [27].

2.2.2 Approximation Algorithms

A particular class of heuristics are *approximation algorithms* that allow to a priori state a bound of a solutions quality in the worst case. The following is due to Kellerer et al. [104].

Let $c^*(I)$ denote the optimal solution value of some instance I and $c^{\mathrm{A}}(I)$ denote the solution value computed by some algorithm A for some minimization problem.

Definition 20 *An algorithm A is an approximation algorithm with* absolute performance guarantee *k, $k > 0$, if*

$$c^{\mathrm{A}}(I) - c^*(I) \leq k$$

for all problem instances I.

Definition 21 *An algorithm A is an approximation algorithm with* relative performance guarantee *k if*

$$\frac{c^{\mathrm{A}}(I)}{c^*(I)} \leq k$$

for all problem instances I.

If its relative performance guarantee can be adjusted by means of a parameter ϵ, A is usually called an ϵ-approximation scheme.

Definition 22 *An algorithm A is an* ϵ-approximation scheme *if for every input ϵ, $0 < \epsilon < 1$,*

$$\frac{c^{\mathrm{A}}(I)}{c^*(I)} \leq 1 + \epsilon$$

holds for all problem instances I.

If the running time of an ϵ-approximation scheme is polynomial in the instance size, A is called a *polynomial time approximation scheme* (PTAS), while it is called *fully polynomial time approximation scheme* (FPTAS) if its runtime is additionally polynomial in $\frac{1}{\epsilon}$.

More detailed reviews of approximation algorithms can be found in the books of Vazirani [173] and Kellerer et al. [104].

2.2.3 Local Search

Starting from some initial feasible solution $x \in S$, local search [26, 144] searches for better solutions in some neighborhood $N(x) \subseteq S$ and replaces x by a better solution $x' \in N(x)$ if one could be found. As detailed in Algorithm 2.4 this procedure is repeated until some termination criterion is met, e.g. no better solution could be found in $N(x)$ or due to some maximum runtime bound.

Definition 23 *A* neighborhood structure *is a function $N : S \rightarrow 2^S$ which assigns a set of neighbors $N(x) \in S$ to each feasible solution $x \in S$.*

Definition 24 *Let $x \in S$ be a feasible solution to an instance of some combinatorial optimization problem and $c(x)$ its objective value. Then x is called a* local optimum *with respect to some neighborhood structure N if and only if $c(x) \leq c(x')$, $\forall x' \in N(x)$.*

Obviously a global optimum is a local optimum with respect to all neighborhood structures.

Algorithm 2.4: Basic Local Search

```
x ← initial solution
repeat
    select x' ∈ N(x)
    if c(x') ≤ c(x) then
        x ← x'
until termination condition is met
```

Concerning the method used to compute a neighbor solution $x' \in N(x)$ one usually distinguishes *random improvement* where x' is chosen randomly, *next improvement* where $N(x)$ is searched in a fixed order and the first solution better than x is taken, and *best improvement* where the best solution $x' \in N(x)$ is taken.

2.2.4 Metaheuristics

The term metaheuristic was introduced by Glover [73] and refers to a class of algorithms which aim to efficiently and effectively explore a problem's search space [27]. According to Voß [176], a metaheuristic can be defined as follows:

> "A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method for example."

Two important classes of metaheuristic methods are local search based approaches which extend the simple local search approach to escape from local optima and nature inspired methods originating from analogies from nature.

In the following, variable neighborhood descent (VND), variable neighborhood search (VNS), and greedy randomized adaptive search procedures (GRASP) are discussed in more detail, as they will be used in the following chapters. Afterwards, a short overview on further local search based methods as well as on nature inspired metaheuristics is given.

Variable Neighborhood Descent

Variable neighborhood descent (VND) [82, 83, 84] is based on the observation that a local optimum with respect to some neighborhood structure is not necessarily a local optimum for another neighborhood structure, while a global optimum is a local optimum for all neighborhood structures.

As detailed in Algorithm 2.5 the simple but effective idea of VND is to systematically search a set of different neighborhood structures N_l, $1 \leq l \leq l_{\max}$, yielding a solution that is a local optimum for all considered neighborhood structures.

Next to selecting a set of neighborhoods $N_l(x)$, $1 \leq l \leq l_{\max}$, one needs to define the order in which they are applied. Since choosing a good sequence is not always obvious, approaches based on dynamically deciding during runtime which neighborhood structure to consider next have been proposed. Relaxation guided VND [153] considers relaxations of each neighborhood to estimate the profit obtained by searching it, while self-adaptive VND [97] dynamically reorders the neighborhood structures based their success rates and execution times.

Algorithm 2.5: Variable neighborhood descent(Solution x).

```
l = 1
while l ≤ l_max do
    find the best neighbor x' ∈ N_l(x)
    if c(x') < c(x) then
        x = x'
        l = 1
    else
        l = l + 1
return x
```

Variable Neighborhood Search

The various variants of variable neighborhood search (VNS) [82, 83, 84] are based on the same observations as VND. Basic VNS, uses a set of k, $1 \leq k \leq k_{\max}$, shaking neighborhood structures, which are usually larger than the VND's neighborhood structures, to escape from a local optimum generated by some local search method which is applied after each shaking move. Another variant of VNS is general VNS, which uses VND as a local improvement subordinate [84], see Algorithm 2.6. In this case the set of shaking neighborhood structures must be different from those used in the VND subordinate.

Algorithm 2.6: General variable neighborhood search (Solution x).

```
repeat
    k = 1
    while k ≤ k_max do
        select x' ∈ N_k(x) randomly  // shaking
        x' = VND(x') // local search by VND
        if c(x') < c(x) then
            // new so far best local optimum
            x = x'
            k = 1
        else
            k = k + 1
until termination condition is met
return x
```

Greedy Randomized Adaptive Search Procedure

A greedy randomized adaptive search procedure (GRASP) is a multi-start method based on generating a feasible solution in a first phase and improving this solution by local search in its second phase [64]. It has been first used by Feo and Resende [60], see also [61] for a survey on GRASP by the same authors. As its name suggests, the construction phase of GRASP is based on randomizing some greedy heuristic. Here, instead of always adding the most promising solution component, a restricted candidate list (RCL) of promising solution components is generated, among the one to add is chosen at random.

Algorithm 2.7 gives a high-level description of GRASP. For applying GRASP to some problem, next to defining a problem specific randomized greedy heuristic and some local improvement method, one has to decide how to organize the RCL. Common strategies include simply fixing its absolute size or dynamically adapting its size due the difference between the currently best and worst potential solution component to add.

Algorithm 2.7: Greedy randomized adaptive search procedure

```
initialize best solution x
repeat
    x' = ∅
    while x' is not a complete solution do
        build RCL
        select component x_i from RCL randomly
        x' = x' ∪ {x_i}
    improve x'
    if c(x') < c(x) then
        x = x'
until termination condition is met
return x
```

Further Local Search based Metaheuristics

Beside the metaheuristics mentioned so far, several other local search based methods utilizing different strategies to escape from local optima have been proposed. These methods include simulated annealing [109], tabu search [74, 75], and iterated local search [129]. In simulated annealing [109] solutions worse than a current one are probabilistically accepted considering an acceptance criterion which is based on the

simulation of a cooling process in metallurgy. Tabu search [74, 75] generally accepts the best solution in the neighborhood of a current solution but restricts this neighborhood by forbidding certain moves leading to just visited solutions. Iterated local search [129] randomly perturbs reached local optimal solutions in order to continue the search and escape the current bounds of attraction.

Nature inspired Metaheuristics

Another important strategy which is frequently used to develop metaheuristics is to mimic successful behavior from nature. As opposed to the methods discussed above, the resulting approaches often operate on a whole set of solutions, called population, rather than on single solutions only. An important class of nature inspired methods are evolutionary algorithms [10] which imitate the theory of evolution as described by Darwin [47] and Mendel [139]. Here, new candidate solutions are created by selecting parental solutions and inheriting their properties on a random basis. In analogy to mutation in nature, these offsprings are further randomly modified to a small degree. Concrete variants of this concept include evolution strategies [159], evolutionary programming [66], and genetic algorithms [92].

A further important variant of evolutionary algorithms which additionally incorporates local search to further improve promising candidate solutions is called memetic algorithms [141].

Another important stream of nature inspired metaheuristics are methods that try to simulate cooperative behavior of individuals instead of their competition as described above. Prominent methods in this area include the diverse variants of ant colony optimization [38, 51] and the simulation of swarm intelligence such as particle swarm optimization [105].

More complete and thorough overviews on above mentioned and further metaheuristics can be found in the books of Dreo et al. [52] or Glover and Kochenberger [76].

2.3 Hybrid Methods

Each of the methods presented in the previous sections have their individual advantages and drawbacks. While (meta-) heuristics are normally able to generate feasible solutions relatively fast, usually no information on the gap to an optimal solution is available. On the contrary, exact methods are in principle able to compute proven

optimal solutions. However, due to their dramatical runtime and resource usage increase if the problem sizes get larger, they might terminate yielding a huge optimality gap or even no feasible solution at all after their maximum allowed runtime. Thus, many successful hybrid methods trying to avoid the drawbacks of individual methods have been proposed recently, see e.g. [25, 133, 158].

Hybrid methods can be classified into *collaborative combinations* where algorithms exchange information but are not part of each other and *integrative combinations*, where one method is used as a subordinate of another. In the collaborative approach one can further distinguish between sequential, intertwined, and parallel approaches depending on the execution order of the different methods [151, 156, 157].

Another important distinction can be made between approaches combining different metaheuristic methods on the one hand and approaches combining exact methods and metaheuristics on the other hand. In particular, combinations of metaheuristics and mathematical programming techniques turned out to often be very fruitful [133]; such hybrids are also frequently called matheuristics.

Combinations of different metaheuristics have been successfully used in many applications. Prominent examples are memetic algorithms [141] or the integration of local search based methods in population based metaheuristics such as EAs and ACO approaches. See e.g. [25] for an overview on and several successful applications of hybrid metaheuristics.

Using exact methods as subordinates of metaheuristic methods often allows for finding extremely good solutions. In very large scale neighborhood search (VLSN) [5], subproblems of appropriate size are frequently solved by exact methods. Successful applications of this concept include *dynasearch* [39, 40] where dynamic programming is used to explore large neighborhoods or the embedding of IP techniques within VNS approaches [96, 148].

Other successful approaches are based on using metaheuristics within IP based methods. Those include the metaheuristic generation of cutting planes [80] as well as the use of metaheuristics for solving the pricing subproblem in column generation methods, see e.g. [145, 150, 152].

Finally, another frequently used combination is the use of metaheuristics to obtain tighter upper bounds within Lagrangian relaxation based methods, see e.g. [116, 146].

An extensive survey on combinations of integer programming techniques with metaheuristics can also be found in [154].

Chapter 3

The $b_{\max}$-Survivable Network Design Problem

3.1 Introduction

The $b_{\max}$-Survivable Network Design Problem ($b_{\max}$-SNDP) is a real-world communication network design problem which arises for instance in the expansion of fiber optic networks. Recently, fiber-to-the-home has become economically feasible for individual households in urban areas. However, covering larger districts with such networks requires enormous financial resources from an operators point of view. Since customers are usually not willing to pay significantly more than for existing lower bandwidth connections, good algorithms for finding cost-efficient network layouts are crucial.

$b_{\max}$-SNDP considers the problem of augmenting an existing network infrastructure by additional links and switches in order to connect additional customer nodes. Here, we distinguish between standard (type-1) customer nodes for which a single link connection suffices and type-2 customer nodes representing business customers which require a more reliable connection, ensuring connectivity even when a single link or routing node fails. Since offering full redundancy to each type-2 customer often is too expensive and does not pay off from an economic point of view, we consider a problem variant where the redundancy condition for type-2 customers is relaxed in the sense that a connection is allowed via a final non-redundant branch line that does not exceed a certain length $b_{\max}$. Thus, we restrict the length of the

non-redundant part of a connection taking a compromise between reliability and construction costs.

In this chapter, we formally introduce $b_{\max}$-SNDP in Section 3.2 and review previous and related work in Section 3.3. Afterwards, we present two new mixed integer programming approaches for solving $b_{\max}$-SNDP to proven optimality based on an exponential number of so-called connection variables which can be solved by branch-and-price. As one main contribution within this section, we show how to significantly speed up the solution of the linear relaxation of these models by using alternative dual-optimal solutions in the pricing subproblem. Theoretical comparisons of the corresponding polyhedra of those two as well to previously existing formulations are given in Section 3.7.

Section 3.8 details a Lagrangian decomposition (LD) approach, while Section 3.9 introduces neighborhood structures for $b_{\max}$-SNDP. In Section 3.10, we present a heuristic for generating feasible solutions to $b_{\max}$-SNDP as well as two metaheuristic approaches based on variable neighborhood search (VNS) and greedy randomized adaptive search (GRASP), respectively. The details of hybrid approaches combining Lagrangian decomposition with variable neighborhood descent (VND) are given in Section 3.11. Test instances for benchmarking our various approaches and computational results are discussed in Sections 3.12 and 3.13, before we finally draw conclusions and outline potential future work in Section 3.14.

The approaches presented in this chapter have been previously published in [116, 122, 120, 123].

3.2 Problem Definition

Formally, we are given a connected undirected graph $G^{\mathrm{o}} = (V^{\mathrm{o}}, E^{\mathrm{o}})$ representing the spatial topology of the surrounding area of potential customers. Each edge $e = (u, v) \in E^{\mathrm{o}}$ corresponding to a possible cable route between its end points $u, v \in V^{\mathrm{o}}$ is given with its length $l_e \geq 0$ and costs $c_e^{\mathrm{o}} \geq 0$ for installing the corresponding fiber optic link. The node set $V^{\mathrm{o}} = S \cup C \cup V_{\mathrm{I}}$ is the disjoint union of customer nodes C, spatial nodes S (switches, possible Steiner nodes) and nodes of the already existing network infrastructure V_{I}. The set of customers $C = C_1 \cup C_2$ is partitioned into type-1 customer nodes C_1 without specific redundancy requirements and type-2 customer nodes C_2 that need to be redundantly connected by means of two node disjoint paths to the existing infrastructure. Each customer node $k \in C$ further has associated a prize $p_k \geq 0$ modeling the expected return on investment when supplying customer

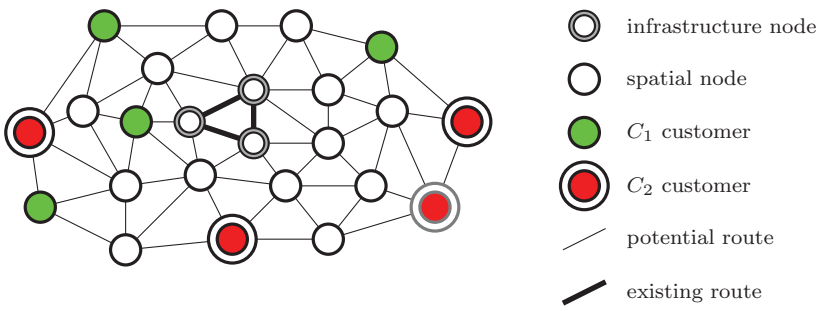


Figure 3.1: An instance of $b_{\max}$-SNDP.

k. Finally, the already existing network infrastructure is represented by the subgraph $(V_\mathrm{I}, E_\mathrm{I})$, $V_\mathrm{I} \subsetneq V^\mathrm{o}$, $E_\mathrm{I} \subsetneq E^\mathrm{o}$, see Figure 3.1.

In a first preprocessing step, we create a reduced graph $G = (V, E)$ by shrinking the whole existing network infrastructure into a single root node $r \in V$. From all edges $(u, v) \in E^\mathrm{o}$ connecting a node $u \in V^\mathrm{o} \setminus V_\mathrm{I}$ to the existing infrastructure – i.e. $v \in V_\mathrm{I}$ – only the cheapest edge (r, u) from the root node to u is included in E. Formally, $G = (V, E)$ is defined by its node set $V = \{r\} \cup S \cup C$, and its edge set $E = \{(u, v) \mid u, v \in V \wedge (u, v) \in E^\mathrm{o}\} \cup \{(r, v) \mid \exists (u, v) \in E^\mathrm{o} \wedge u \in V_\mathrm{I} \wedge v \in V^\mathrm{o} \setminus V_\mathrm{I}\}$, see Figure 3.2. Customers with associated prizes and edge lengths are adopted from the original graph $G^\mathrm{o} = (V^\mathrm{o}, E^\mathrm{o})$. Since we include one edge (r, v) for each original edge connecting v with some node of the existing infrastructure $u \in V_\mathrm{I}$, edge costs c_e, are defined as follows:

$$c_e = \begin{cases} c^\mathrm{o}_e, & \text{if } u, v \notin V_\mathrm{I} \\ \min\{c^\mathrm{o}_f \mid f = (w, v) \in E^\mathrm{o} : w \in V_\mathrm{I}\} & \text{otherwise} \end{cases}, \forall e = (u, v) \in E.$$

Let $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E$, represent a solution network to an instance of $b_{\max}$-SNDP. The following conditions specify how customer nodes are to be connected:

- *Simple connection:*
  A type-1 customer node k from C_1 is feasibly connected iff there exists a path from node r to k.
- *Redundant connection:*
  A customer node k from C_2 is feasibly connected iff there exist two node (and edge) disjoint paths from node r to k, see Figure 3.3.

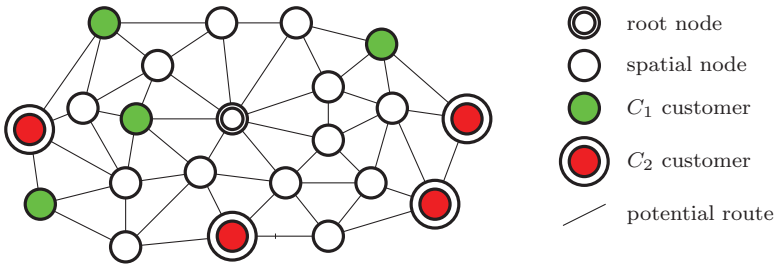


Figure 3.2: The instance of $b_{\max}$-SNDP from Figure 3.1 after shrinking the existing infrastructure.

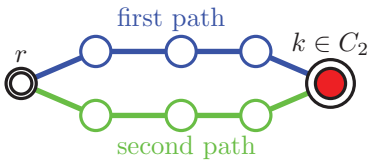


Figure 3.3: A feasible connection to $k \in C_2$ with $b_{\max}(k) = 0$.

- $b_{\max}$-*redundant connection:*
  Occasionally, the biconnectivity condition for the nodes in set C_2 is relaxed in the sense that such a node $k \in C_2$ may be connected to any biconnected (Steiner or customer) node $j \in V$ (the *branch node* of k) via a single path of maximum total length $b_{\max}(k) > 0$. This (optional) single path is called *branch line* and $b_{\max}(k)$ the *maximum branch line length* for customer k, see Figure 3.4. We denote the set of potential branch nodes for a customer $k \in C_2$, by $\mathcal{B}(k) \subseteq V$.
  Since each type-2 customer is a potential branch node of itself whereas k is the only potential branch node if $b_{\max}(k) = 0$, $k \in \mathcal{B}(k)$ holds for all type-2 customers $k \in C_2$ independent of a concrete problem instance and a given maximum branch line length.

Note that, we assume $r \notin \mathcal{B}(k)$, $\forall k \in C_2$, since above mentioned shrinking of the existing infrastructure into the root node r might influence the optimal solution value otherwise.

Regarding the objective, we distinguish between two alternative goals:

- In the *Operative Planning Task* (OPT) we focus on finding a minimum-cost

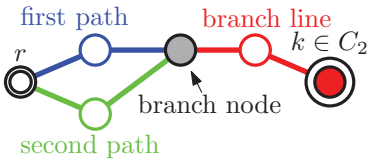


Figure 3.4: A feasible connection to $k \in C_2$ with $b_{\max}(k) > 0$.

subgraph G' feasibly connecting all customers C, with the total costs being

$$o_{\text{OPT}}(G') = \sum_{e \in E'} c_e. \tag{3.1}$$

This case can be considered a generalization of the classical Steiner tree problem on a graph (STP) where a special form of redundancy is required for the nodes in C_2.

- In the *Strategic Simulation Task* (SST) customers' prizes are also considered, and the objective is to determine a subset $C' \subseteq C$ of customers which are connected so that the costs for building the network minus the earned prizes are minimized. In order to always have positive total costs, which eases some parts of our algorithms and notations, we perform a simple transformation by adding the constant $\sum_{k \in C} p_k$ to the objective function, yielding

$$o_{\text{SST}}(G') = \sum_{e \in E'} c_e - \sum_{k \in C'} p_k + \sum_{k \in C} p_k = \sum_{e \in E'} c_e + \sum_{k \in C \setminus C'} p_k. \tag{3.2}$$

This problem variant is a generalization of the prize-collecting Steiner tree problem (PCSTP).

Figure 3.5 depicts an exemplary solution to the OPT variant of $b_{\max}$-SNDP without considering $b_{\max}$-redundancy – i.e. $b_{\max}(k) = 0$, $\forall k \in C_2$ – while Figure 3.6 shows an exemplary solution to the SST variant including $b_{\max}$-redundancy.

As already the classical Steiner tree problem on a graph is $\mathcal{NP}$-hard [103], this obviously also holds for both of our problem variants. In the following presentation of our solution approaches, we primarily consider the more complex SST case if not explicitly stated and assume $p_k = \infty$, $\forall k \in C$, to include the OPT case.

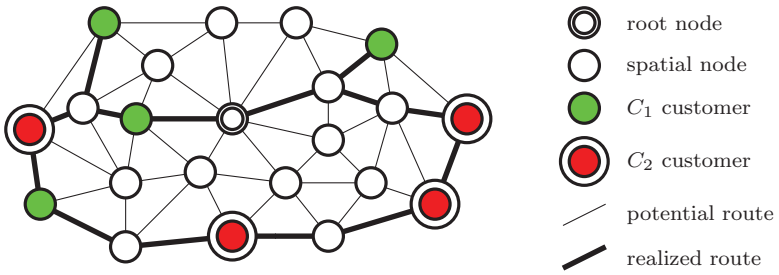


Figure 3.5: An exemplary solution to the OPT variant of $b_{\max}$-SNDP with $b_{\max}(k) = 0, \forall k \in C_2$.

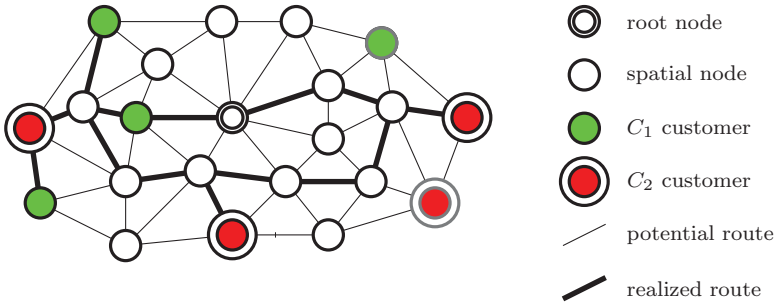


Figure 3.6: An exemplary solution to the SST variant of $b_{\max}$-SNDP with $b_{\max}(k) \neq 0, \forall k \in C_2$.

3.3 Related Work

$b_{\max}$-SNDP has been introduced by Bachhiesl et al. [9]. Ljubić [124] introduced its name[1] and pointed out the relation to $\{0, 1, 2\}$-SNDP [106] which corresponds to $b_{\max}$-SNDP if $b_{\max}(k) = 0$, $\forall k \in C_2$.

Wagner et al. [178] presented mixed integer programming (MIP) approaches for $b_{\max}$-SNDP based on multi-commodity flows. With the general purpose ILP-solver CPLEX 10.0, instances with up to 190 total nodes, 377 edges but only 6 customer nodes could be solved to proven optimality, and instances up to 2804 nodes, 3082 edges and 12 customer nodes could be solved with a final LP gap of about 7%. Unfortunately, this approach turned out to be unsuitable for larger instances and/or in particular instances with larger number of customer nodes, as already solving the linear programming (LP) relaxation of the MIP requires too much time due to the huge number of variables involved. In [177], the same authors approached $b_{\max}$-SNDP with a different formulation based on connectivity constraints. While this formulation involves only a reasonable number of variables, the number of inequalities is exponentially large. By using a branch-and-cut algorithm, this model could be solved relatively well, and they were able to find proven optimal solutions for instances with up to 190 nodes, 377 edges, and 13 customer nodes. For larger, practical instances this approach unfortunately still is not applicable at all or finds quite poor solutions with huge LP-gaps only.

Modeling redundant connections by pairs of reversely oriented paths, Chimani et al. [36, 35] further came up with strong formulations for $\{0, 1, 2\}$-SNDP based on multi-commodity flows and directed connection cuts, theoretically dominating those of Wagner et al. [178, 177] for the case of $b_{\max}(k) = 0$, $\forall k \in C_2$. Their formulations were able to solve larger instances and to consider a greater number of customer nodes than the approaches of Wagner et al. However, their directed model cannot be easily adapted to consider $b_{\max}$-redundancy too.

The classical Steiner tree problem (STP) on graphs has been considered by many authors, see e.g. [180] for a survey. Among the various authors that considered integer programming models for the STP, Koch and Martin [110] described an effective branch-and-cut method based on directed connectivity cuts. More recently, Bahiense et al. [11] presented a Lagrangian relaxation based approach which often yields near-optimal solutions. Well known problem specific heuristic methods have e.g. been described by Takahashi and Matsuyama [167] and Duin and Voß [54].

[1] Ljubić used the name $k_{\max}$-SNDP.

The prize collecting Steiner tree problem (PCSTP) was introduced by Segev [163] who considered the node weighted STP, which is a special version of the PCSTP. The term "prize collecting" has first been used by Balas [12] for the prize collecting traveling salesman problem. Ljubić et al. [128] presented an exact method for the PCSTP based on directed connection cuts. Other successful mathematical programming based approaches include a relax-and-cut by Cunha et al. [43] and a cutting plane method by Lucena et al. [131]. Canuto et al. [30] described an effective multi-start local search approach based on perturbation of the nodes prizes, where path-relinking and variable neighborhood search are used to further improve the obtained solutions. Tests to reduce the number of nodes and edges that need to be considered in an instance of the PCSTP have been described by Uchoa [170], whereas Chapovska et al. [32] discuss complexity of and solution methods for several variants of the PCSTP.

Other related problems are the various variants of the survivable network design problem (SNDP) [68]. Among these, especially the "low connectivity" variants such as above mentioned $\{0, 1, 2\}$-SNDP are relevant for $b_{\max}$-SNDP, see e.g. [106, 164] for relevant surveys.

3.4 Individual Optimal Connections

Most of the algorithms and approaches presented in this chapter require at some point to solve the subproblem of computing the cheapest feasible connection from the root node r to a single customer node $k \in C$. In this section, the corresponding algorithms to efficiently solve these subproblems are detailed for each customer type.

3.4.1 Optimal Connections to Type-1 Customers

Given, nonnegative edge costs $c'_e \geq 0$, $\forall e \in E$, the problem of computing the cheapest connection between the root node r and some customer node $k \in C_1$ can obviously be solved by a simple cheapest path calculation. Among the various existing algorithms for computing cheapest paths – see e.g. [50, 21, 87, 88] – we use a binary heap implementation of Dijkstra's' algorithm [50], resulting in a worst case time complexity of $O(|E| + |V| \log |V|)$.

3.4.2 Optimal Connections to Type-2 Customers

Without yet considering $b_{\max}$-redundancy, computing the cheapest connection to a type-2 customer $k \in C_2$ means to find a cheapest node disjoint pair of paths between the root node r and k. Suurballe and Tarjan [165] showed how to compute a cheapest arc-disjoint pair of paths between two nodes s and t on a directed graph (V, A) with arc costs $c'_{i,j} \geq 0$, $\forall (i,j) \in A$, efficiently in time $O(|A| + |V| \log |V|)$; see also [100].

Their algorithm consists of the following main steps:

1. Determine a shortest path tree from node s. Let $d(i)$ represent the costs of a cheapest path from node s to node i. We replace the costs of arc (i,j) with $c'_{i,j} - d(j) + d(i)$.
2. Determine a shortest path P_1 from s to t. Reverse all arcs on P_1 and leave all costs as computed in step one (= residual graph).
3. Solve the cheapest path problem between nodes s and t on this new graph with the new edge costs. Let P_2 represent this shortest path.
4. If any of the reversed P_1 arcs belong to P_2, eliminate these arcs from P_1 and P_2 to form arc sets P'_1 and P'_2. The set $P'_1 \cup P'_2$ corresponds to a cheapest pair of arc-disjoint paths between s and t.

Since $G = (V, E)$ is undirected, we define a corresponding directed graph (V, A), by replacing each edge $e = (u, v) \in E$ by two oppositely directed arcs $(u, v), (v, u) \in A$. Given nonnegative costs $c'_e \geq 0$, $\forall e \in E$, arc costs $c'_{u,v}$, $\forall (u, v) \in A$, are given by $c'_{u,v} = c'_{v,u} = c'_e$, $\forall e = (u, v) \in E$.

The algorithm of Suurballe and Tarjan [165] can be used to compute a node-disjoint pair of paths by applying it to the *split graph* of the original graph. The split graph is obtained by replacing each node $v \in V$ by a pair of nodes v' and v''. For each such pair, we add an arc (v', v'') with zero costs, while we include arcs (u'', v'), (v'', u') – with arc costs $c'_{u'',v'} = c'_{u,v}$, $c'_{v'',u'} = c'_{v,u}$ – for each pair of oppositely directed arcs $(u, v), (v, u) \in A$.

It is obvious that a shortest pair of edge-disjoint paths from s'' to t' is also node-disjoint, since each node v' has only one outgoing arc and each node v'' has only one ingoing arc.

In case of $b_{\max}$-redundancy, the above algorithm must further be extended. A naive approach considers each node $v \in \mathcal{B}(k)$ in the $b_{\max}$-neighborhood of node $k \in C_2$ and determines a cheapest pair of paths to this node. Furthermore, a cheapest length constrained shortest path from node k to each potential branch node must be computed.

The overall cheapest combination is the final result. Since, computing a (length) constrained cheapest path is $\mathcal{NP}$-hard [69] relaxing the biconnectivity constraints by means of $b_{\max}$-redundancy turns out to significantly increase the subproblem's complexity not only from a computational but also from a theoretic point of view. However, several pseudo-polynomial algorithms for solving constrained shortest path problems have been proposed, see e.g. [19, 56]. In our work, we use the approach described by Gouveia et al. [79] which solves this problem for a customer $k \in C_2$ in $O(b_{\max}(k)|\mathcal{E}(k)|)$, where $\mathcal{E}(k) = \{e = (u, v) \in E \mid u, v \in \mathcal{B}(k)\}$. Since $b_{\max}(k)$ and thus $|\mathcal{E}(k)|$ is typically rather small, we are able to solve this $\mathcal{NP}$-hard problem by above mentioned dynamic programming based approach without increasing the computational effort too much.

Suurballe and Tarjan [165] further presented an algorithm for computing cheapest arc disjoint pair of paths from a single source node to all other nodes with the same worst case complexity as for the single destination variant. Here, the necessary shortest path calculations on the various residual graphs – which are closely related to each other – are combined into a single calculation.

However, since the allowed branch line lengths' will be relatively small in all relevant scenarios, the number of potential branching nodes $|\mathcal{B}(k)|$ can be regarded as a constant for all type-2 customers $k \in C_2$. Thus, above mentioned naive algorithm is used in the following to compute optimal connections to type-2 customers $k \in C_2$ with $b_{\max}(k) > 0$.

3.5 The Undirected Connection Formulation for $b_{\max}$-SNDP

To model $b_{\max}$-SNDP as a mixed integer program (MIP) we consider the set of all possible feasible connections $\mathcal{F}_k$ for each customer $k \in C$. For type-1 customers $k \in C_1$, $\mathcal{F}_k$ corresponds to the set of all paths from the root node r to k, i.e.

$$\mathcal{F}_k = \{p \subseteq E \mid p \text{ forms a path from } r \text{ to } k\},$$

while for type-2 customers $k \in C_2$, $\mathcal{F}_k$ can be expressed as follows:

$$\mathcal{F}_k = \{p \subseteq E \mid p \text{ forms two node disjoint paths from } r \text{ to some node } j \in \mathcal{B}(k) \text{ and one path from } j \text{ to } k \text{ whose length does not exceed } b_{\max}(k)\}.$$

We formulate the SST variant of $b_{\max}$-SNDP by the following *integer master problem* (Col) using variables $0 \leq f_p^k \leq 1$, $\forall k \in C$, $\forall p \in \mathcal{F}_k$, to indicate whether a corresponding connection $p \in \mathcal{F}_k$ is realized ($f_p^k = 1$) or not ($f_p^k = 0$), decision variables $x_e \in \{0, 1\}$, $\forall e \in E$, to specify whether an edge e is part of the solution ($x_e = 1$) or not ($x_e = 0$), and variables $0 \leq y_k \leq 1$, $\forall k \in C$, to denote whether a feasible route to customer k is installed ($y_k = 1$) or not ($y_k = 0$). Variables y_k are fixed to one in the OPT variant.

$$\text{(Col)} \quad z = \min \sum_{e \in E} c_e x_e + \sum_{k \in C} p_k (1 - y_k) \tag{3.3}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{F}_k} f_p^k - y_k \geq 0 \qquad \forall k \in C \tag{3.4}$$

$$x_e - \sum_{p \in \mathcal{F}_k | e \in p} f_p^k \geq 0 \qquad \forall k \in C,\ \forall e \in E \tag{3.5}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{3.6}$$

$$0 \leq y_k \leq 1 \qquad \forall k \in C \tag{3.7}$$

$$f_p^k \geq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{F}_k \tag{3.8}$$

Constraints (3.4) ensure that a customer's prize can only be earned if it is feasibly connected to r, while constraints (3.5) link connection variables to edge variables. We define only lower and upper bounds for variables y_k, and f_p^k in inequalities (3.7) and (3.8). However, if all edge variables x_e, $\forall e \in E$, are integral, each set of potentially existing fractional connections to some customer $k \in C$ can be replaced by a integral connection without including additional edges and thus without modifying the solutions objective value. Since customers prizes reduce the objective value, variables y_k, $\forall k \in C$, we further conclude that they will automatically become integer.

The linear relaxation of (Col) – the *linear master problem* $(\text{Col})^{\text{LP}}$ – is given by substituting the integrality constraints (3.6) by

$$x_e \geq 0 \qquad \forall e \in E \tag{3.9}$$

Let $\mu_k \geq 0$, $\forall k \in C$, be the dual variables associated to the convexity constraints (3.4) and $\pi_{k,e} \geq 0$, $\forall k \in C$, $\forall e \in E$, be the dual variables associated to the coupling constraints (3.5).

Furthermore, let $F = \{f_p^k \mid k \in C, p \in \mathcal{F}_k\}$ be the set of all f_p^k variables representing columns in $(\text{Col})^{\text{LP}}$. Since F consists of an exponential number of variables we cannot solve (Col) directly, but use column generation [16, 49] for solving its linear relaxation $(\text{Col})^{\text{LP}}$. Here, we define the *restricted master problem* $(\text{Col})^{\text{RMP}}$ using only a small subset of connection variables $\tilde{F} \subsetneq F$. Otherwise $(\text{Col})^{\text{RMP}}$ corresponds to $(\text{Col})^{\text{LP}}$.

When solving $(\text{Col})^{\text{RMP}}$ we obtain optimal dual variable values μ_k^* and $\pi_{k,e}^*$, defining reduced prices $\bar{c}_{k,p}$ for variables $f_p^k \in F \setminus \tilde{F}$:

$$\bar{c}_{k,p} = -\mu_k^* + \sum_{e \in p} \pi_{k,e}^*$$

The pricing problem is now to find $(k^*, p^*) = \text{argmin}_{k \in C, p \in F_k} \{\bar{c}_{k,p}\}$. If $\bar{c}_{k^*,p^*} \geq 0$ we have obtained an optimal solution to $(\text{Col})^{\text{LP}}$. Otherwise, we add at least one column with negative reduced costs and resolve $(\text{Col})^{\text{RMP}}$.

More generally speaking, in the pricing subproblem we have to find a feasible connection for some $k \in C$ yielding negative reduced costs $\bar{c}_{k,p} = -\mu_k^* + \sum_{e \in p} \pi_{k,e}^*$ or prove that no such connection exists. For this purpose we need to determine a cheapest feasible connection on graph $G = (V, E)$ with modified edge costs $\pi_{k,e} \geq 0$, $\forall e \in E$, for each customer node $k \in C$. When the costs of such a connection are less then μ_k, we have found an appropriate connection, i.e. the corresponding variable f_p^k can be added to $(\text{Col})^{\text{RMP}}$. Since, we have strictly nonnegative edge costs, we can use the algorithms explained in Section 3.4 for solving the pricing subproblem.

3.5.1 Analyzing the Restricted Dual Problem

It is well known that (simplex based) column generation approaches often suffer from inefficiency resulting in a large number of required pricing iterations as well as long computation times. Vanderbeck [172] describes five major efficiency issues of simplex based column generation.

Several stabilization techniques to reduce their effects have been proposed, see e.g. [53] or [130] for reviews on those methods. From the issues described by Vanderbeck preliminary tests showed that *primal degeneracy* as well as the *heading-in effect* are mainly relevant in our case, compare also Section 2.1.5. The occurrence of primal degeneracy is based on the fact that typically only very few connection and edge variables will have nonzero values in a solution of $(\text{Col})^{\text{RMP}}$.

Instead of using a problem-independent stabilization approach we analyze the dual of $(\text{Col})^{\text{RMP}}$ to take advantage of problem specific characteristics. Let $\lambda_k \leq 0$ denote the dual variables associated to inequalities (3.7). As mentioned before $\tilde{F} \subsetneq F$ denotes the set of variables representing connections to customers in $(\text{Col})^{\text{RMP}}$. Then the dual of the restricted master problem $(\text{Col})^{\text{RMP}}$ – i.e. the *restricted dual problem* – for the SST variant is given by model (3.10)–(3.16).

$$\max \quad \sum_{k \in C} \lambda_k + p_k \tag{3.10}$$

$$\sum_{k \in C} \pi_{k,e} \leq c_e \qquad \forall e \in E \tag{3.11}$$

$$\mu_k - \sum_{e \in p} \pi_{k,e} \leq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{F}_k | \exists f_p^k \in \tilde{F} \tag{3.12}$$

$$-\mu_k + \lambda_k \leq -p_k \qquad \forall k \in C \tag{3.13}$$

$$\pi_{k,e} \geq 0 \qquad \forall k \in C,\ \forall e \in E \tag{3.14}$$

$$\mu_k \geq 0 \qquad \forall k \in C \tag{3.15}$$

$$\lambda_k \leq 0 \qquad \forall k \in C \tag{3.16}$$

Let $E'' \subseteq E$ denote the subset of edges which are not part of any so far included connection, i.e. $E'' = \left\{ e \in E \mid \nexists f_p^k \in \tilde{F} : e \in p \right\}$. For edges $e \in E''$, only inequalities (3.11) are relevant. Thus all values $\pi_{k,e} \geq 0$, $\forall k \in C$, $\forall e \in E''$, are dual optimal as long as $\sum_{k \in C} \pi_{k,e} \leq c_e$ holds.

Since almost the complete edge set E will not be in any included connection in the beginning of our column generation procedure, dual variable values $\pi_{k,e}$ used as edge costs in the pricing subproblem will not be meaningful. Furthermore, in order to be able to solve $(\text{Col})^{\text{LP}}$ efficiently, we aim to keep the number of included connection variables as well as the set $E \setminus E''$ as small as possible.

Generally speaking, the structure of model (3.10)–(3.16) imposes the generation of many irrelevant columns having identical reduced prices. This observation explains the occurrence of the heading in effect. This effect is even intensified by the fact that CPLEX [98] – which we use for solving the linear relaxation of our model – generates minimal dual-optimal values for all dual variables, i.e. most of them will be zero.

For the OPT variant, the dual of $(\text{Col})^{\text{RMP}}$ is given by model (3.17)–(3.21).

$$\max \quad \sum_{k \in C} \mu_k \tag{3.17}$$

$$\sum_{k \in C} \pi_{k,e} \leq c_e \qquad \forall e \in E \tag{3.18}$$

$$\mu_k - \sum_{e \in p} \pi_{k,e} \leq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{F}_k | \exists f_p^k \in \tilde{F} \tag{3.19}$$

$$\pi_{k,e} \geq 0 \qquad \forall k \in C,\ \forall e \in E \tag{3.20}$$

$$\mu_k \geq 0 \qquad \forall k \in C \tag{3.21}$$

Although model (3.17)–(3.21) slightly differs from formulation (3.10)–(3.16) for the SST case, the main observations about primal degeneracy and non-meaningful dual variable values – especially for edges not part of any so far included connection – remain valid.

3.5.2 Alternative Dual-Optimal Solutions

In the following, we detail our stabilization procedure for generating meaningful dual variable values in the pricing problem. Here, we exploit different dual-optimal solutions to improve the convergence properties of our column generation algorithm. This approach can be interpreted as a stabilization technique that "centers" an actual LP solution.

Let $D^* = (\lambda^*, \mu^*, \pi^*)$ be an optimal solution to the restricted dual problem (3.10)–(3.16). As shown in the previous section, for edges $e \in E''$ all values $\pi_{k,e} \geq 0, \forall k \in C$, are dual optimal as long as $\sum_{k \in C} \pi_{k,e} \leq c_e$. Furthermore, for edges $e \in E \setminus E''$, we can increase the sum of dual variable values $\sum_{k \in C} \pi_{k,e}$ by $\delta_e = c_e - \sum_{k \in C} \pi_{k,e}$.

As mentioned earlier CPLEX [98] generates minimal values for dual variables (i.e. $\pi_{k,e} = 0,\ \forall k \in C,\ \forall e \in E''$; usually $\delta_e > 0$ for some edges $e \in E \setminus E''$). For creating meaningful dual variable values and thus keeping the set of edges and connection variables that will be finally included relatively small, we aim to increase variable values $\pi_{k,e}$, $\forall k \in C$, $\forall e \in E$, while maintaining dual optimality.

The probably simplest and most obvious strategy is to use the alternative dual optimal solution $D' = (\lambda^*, \mu^*, \pi')$ with $\pi'_{k,e} = \frac{c_e}{|C|}, \forall k \in C,\ \forall e \in E''$ and $\pi'_{k,e} = \pi^*_{k,e} + \frac{\delta_e}{|C|}, \forall k \in C,\ \forall e \in E \setminus E''$. However, as will be illustrated by our computational results we can do even better by initially using different dual-optimal solutions $D^{(k,d)} =$

$\left(\lambda^*, \mu^*, \pi^{(k,d)}\right)$, for all $k \in C$ – controlled by parameter d ($1 \leq d \leq |C|$) – which finally converge to D' for $d = |C|$. When considering client $k \in C$ in the pricing problem, we use dual values $\pi_{k,e}^{(k,d)} = \frac{c_e}{d}$, $\forall e \in E''$, and $\pi_{k,e}^{(k,d)} = \pi_{k,e}^* + \frac{\delta_e}{d}$, $\forall e \in E \setminus E''$. Note that assuming $\pi_{k',e}^{(k,d)} = 0$, $\forall k' \neq k \in C$, $\forall e \in E''$, and $\pi_{k',e}^{(k,d)} = \pi_{k,e}^*$, $\forall k' \neq k \in C$, $\forall e \in E \setminus E''$ we again only use dual optimal solutions when solving the pricing problem. As shown in Algorithm 3.1 parameter d is initially set to one and gradually incremented up to $|C|$ in case no column with negative reduced cost could be priced in and reset to one in case columns including new edges have been added to $(\text{Col})^{\text{RMP}}$. Since we essentially use D' if $d = |C|$ we can terminate the column generation process if no column with negative reduced costs could be found for $d = |C|$.

We further apply a simpler variant of $D^{(k,d)}$ where d is initially set to one and set to $|C|$ in case no connection yielding negative reduced costs could be identified. In this strategy – we refer to the resulting dual optimal solutions by $D^{(k,d')}$ – d will not be decreased any more.

While the above mentioned strategies are feasible for both the SST as well as the OPT variant of $b_{\max}$-SNDP, we further consider a fourth approach for the SST variant taking each customer's prize into consideration. For each edge $e \in E$, we add a value corresponding to its prize relative to the sum of all prizes, i.e. for all customers $k \in C$ we set $\pi_{k,e}^{(\text{p})} = c_e \frac{p_k}{\sum_{l \in C} p_l}$ if $e \in E''$ and $\pi_{k,e}^{(\text{p})} = \pi_{k,e}^* + \delta_e \frac{p_k}{\sum_{l \in C} p_l}$ otherwise. The resulting alternative dual optimal solution is denoted by $D^{(\text{p})} = \left(\lambda^*, \mu^*, \pi^{(\text{p})}\right)$.

3.6 The Directed Connection Formulation for $b_{\max}$-SNDP

Since directed formulations are in many cases theoretically stronger than undirected ones, and frequently also outperform those from a computational point of view it is natural to ask whether it is possible to replace the undirected formulation (Col) presented in the previous section by a directed one.

Chimani et al. [36] showed that any feasible solution to $\{0, 1, 2\}$-SNDP can be transformed into a directed graph with a simple path from r to each connected type-1 customer and two oppositely directed, internally node disjoint paths between r and any connected type-2 customer $k \in C_2$. Interpreting a feasible connection to some customer $k \in C_2$ with $b_{\max}(k) > 0$ as two independent connections – a non-redundant from r to k and a fully redundant connection to its branching node $v \in \mathcal{B}(k)$ – the orientability of any solution to $b_{\max}$-SNDP follows from the result of Chimani et al.

Algorithm 3.1: Column generation for (Col).

$d = 1$
create and add set of initial columns $\tilde{F}$
$E'' = \{e \in E \mid \nexists f_p^k \in \tilde{F} : e \in p\}$
$m = true$
while m **do**
  $m = false$
  solve $(\text{Col})^{\text{RMP}}$
  $\delta_e = \left(c_e - \sum_{k \in C} \pi_{k,e}\right)/d, \ \forall e \in E \setminus E''$
  forall the $k \in C$ **do**
    $$c'_e = \begin{cases} c_e/d & \text{if } e \in E'', \\ \pi_{k,e} + \delta_e & \text{else.} \end{cases} \quad \forall e \in E$$
    $p =$ shortest connection to k using edge costs c'
    $E_p = \{e \in E \mid e \in p\}$
    if $\sum_{e \in E_p} c'_e < \mu_k$ **then**
      add corresponding variable f_p^k to $(\text{Col})^{\text{RMP}}$
      if $E_p \not\subseteq E''$ **then**
        $d = 1$
      $E'' = E'' \setminus (E_p \cap E'')$
      $m = true$
  if $m == false \wedge d < |C|$ **then**
    $m = true$
    $d++$

In this section, we introduce model (dCol), resembling a directed variant of model (Col), which exploits the orientability of solutions to $b_{\max}$-SNDP. Let $A = \{(u,v),(v,u) \mid e = (u,v) \in E\}$ consist of two oppositely directed arcs for each original edge $e \in E$. To model $b_{\max}$-SNDP we utilize binary variables $a_{u,v} \in \{0,1\}$, $\forall (u,v) \in A$, indicating whether or not arc $(u,v) \in A$ is part of the (oriented) solution ($a_{u,v} = 1$) or not ($a_{u,v} = 0$). As for model (Col), variables $0 \leq y_k \leq 1$, $\forall k \in C$, specify whether a customer is feasibly connected according to its redundancy requirements or not. We further use variables $0 \leq h_p^k \leq 1$, $\forall k \in C$, $\forall p \in \mathcal{H}_k$, where $\mathcal{H}_k$ is the set of all feasible directed connections for customer $k \in C$, indicating whether the corresponding connection is realized ($h_p^k = 1$) or not ($h_p^k = 0$).

Analogously to (Col), for type-1 customers $k \in C_1$, $\mathcal{H}_k$ corresponds to the set of all directed paths from the root node r to k, i.e.

$$\mathcal{H}_k = \{p \subseteq A \mid p \text{ forms a directed path from } r \text{ to } k\},$$

while for type-2 customers $k \in C_2$, $\mathcal{H}_k$ can be expressed as follows:

$$\begin{aligned}\mathcal{H}_k = \{p \subseteq A \mid\ & p \text{ forms two oppositely directed, internally node disjoint paths} \\ & \text{between } r \text{ and some node } j \in \mathcal{B}(k) \text{ and a directed path from} \\ & j \text{ to } k \text{ whose length does not exceed } b_{\max}(k)\}.\end{aligned}$$

Using directed arc costs $c_{u,v} = c_e$, $\forall (u,v) \in A$, $e = (u,v) \in E$, we can model $b_{\max}$-SNDP as model (dCol) described by (3.22)–(3.28).

$$\text{(dCol)} \quad z = \min \sum_{(u,v)\in A} c_{u,v} a_{u,v} + \sum_{k \in C} p_k (1 - y_k) \tag{3.22}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{H}_k} h_p^k - y_k \geq 0 \qquad \forall k \in C \tag{3.23}$$

$$a_{u,v} - \sum_{p \in \mathcal{H}_k \mid (u,v) \in p} h_p^k \geq 0 \qquad \forall k \in C,\ \forall (u,v) \in A \tag{3.24}$$

$$a_{u,v} + a_{v,u} \leq 1 \qquad \forall e = (u,v) \in E \tag{3.25}$$

$$a_{u,v} \in \{0,1\} \qquad \forall (u,v) \in A \tag{3.26}$$

$$0 \leq y_k \leq 1 \qquad \forall k \in C \tag{3.27}$$

$$h_p^k \geq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{H}_k \tag{3.28}$$

Constraints (3.23) ensure that a customer's prize can only be earned if it is feasibly connected to r, while constraints (3.24) link connection variables to arc variables. Inequalities (3.25) guarantee that at most one out of each pair of oppositely directed arcs is used in a solution. Note that for variables y_k and h_p^k only lower and upper bounds are defined in (3.27) and (3.28). They will automatically become integer by the same arguments as for model (Col).

As for model (Col), there are exponentially many variables $H = \{h_p^k \mid k \in C,\ p \in \mathcal{H}_k\}$ corresponding to feasible directed connections. Thus, we cannot directly solve the linear relaxation $(\text{dCol})^{\text{LP}}$ of model (3.22)–(3.28) which is given by substituting inequalities (3.26) by

$$a_{u,v} \geq 0 \qquad \forall (u,v) \in A. \tag{3.29}$$

We apply column generation [16, 49] for solving $(\text{dCol})^{\text{LP}}$ analogously to the undirected connection formulation presented in the last section. Again, we start with a small subset of connection variables $\tilde{H} \subsetneq H$ considered in the restricted master problem $(\text{dCol})^{\text{RMP}}$, and dynamically add further variables $h \in H \setminus \tilde{H}$ by iteratively solving the pricing problem.

Let $\nu_k \geq 0$, $\forall k \in C$, be the dual variables associated to constraints (3.23) and $\omega_{k,u,v} \geq 0$, $\forall k \in C$, $\forall (u,v) \in A$, denote the dual variables associated to constraints (3.24). Then, when solving $(\text{dCol})^{\text{RMP}}$ reduced prices $\overline{c}_{k,p}$ for connection variables $h_p^k \in H \setminus \tilde{H}$ can be computed by

$$\overline{c}_{k,p} = -\nu_k + \sum_{(u,v)\in p} \omega_{k,u,v}.$$

In the pricing problem, we need to find $(k^*, p^*) = \text{argmin}_{k\in C, p\in\mathcal{H}_k}\{\overline{c}_{k,p}\}$. As long as at least one variable with negative reduced costs does exist, we add it to $\tilde{H}$ and resolve $(\text{dCol})^{\text{RMP}}$.

In other words, in the pricing problem we need to determine a cheapest directed connection to each customer $k \in C$ in $D = (V, A)$ with arc costs $\omega_{k,u,v} \geq 0$, $\forall (u,v) \in A$. If the total costs of such a connection are smaller than ν_k, the corresponding connection variable has negative reduced costs and can be included in $(\text{dCol})^{\text{RMP}}$. Since arc costs are non-negative we can efficiently solve the pricing problem for type-1 customers, by simple cheapest path calculations. For customers $k \in C_2$ with

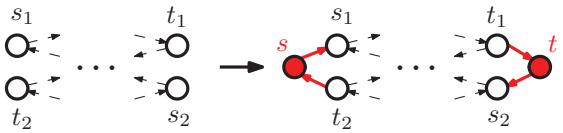


Figure 3.7: Transformation of 2DP on (s_1, t_1), (s_2, t_2) into ODP on (s, t).

$b_{\max}(k) = 0$ we need to compute the cheapest pair of oppositely directed, internally node disjoint paths (ODP) between the root node r and k.

As shown in Figure 3.7 any instance of the directed disjoint pair of paths problem (2DP) for two source-destination pairs (s_1, t_1), (s_2, t_2), which is known to be $\mathcal{NP}$-hard [67], can be transformed into an instance of ODP for s, t by adding nodes s, t and arcs $\{(s, s_1), (t_2, s), (t_1, t), (t, s_2)\}$. We conclude that ODP as well as the pricing problem for the more general case of customers $k \in C_2$ with $b_{\max}(k) > 0$ are $\mathcal{NP}$-hard.

While, several algorithms for solving the directed disjoint pair of paths problem have been proposed for special cases such as planar graphs or dual arc costs [78], its general case has gained surprisingly few consideration so far.

3.6.1 Solving the Pricing Problem by Mixed Integer Programming

We solve the pricing problem for each customer $k \in C_2$ using the MIP (3.30)–(3.43), where $\mathcal{A}(k) = \{(u, v) \in A \mid u, v \in \mathcal{B}(k)\}$ denotes the set of potential edges in the customer's branch line. Each feasible connection is represented by a directed cycle containing r and at least one potential branching node $w \in \mathcal{B}(k)$ and a path from r to k using arcs not on this cycle for the branch line only. The directed cycle containing r and the finally selected branch node is described by variables $q_{u,v} \in \{0, 1\}$, $\forall (u, v) \in A$. Variables $s_{u,v} \in \{0, 1\}$, $\forall (u, v) \in A$, indicate whether an arc is part of the non-redundant path from the root to k, while variables $0 \leq b_{u,v} \leq 1$, $\forall (u, v) \in \mathcal{A}(k)$, denote whether an arc is part of the connections branch line, i.e. those arcs that are on the non-redundant path described by variables $s_{u,v}$ but not on the cycle described by variables $q_{u,v}$.

$$\min \quad \sum_{(u,v)\in A} \omega_{k,u,v} q_{u,v} + \sum_{(u,v)\in \mathcal{A}(k)} \omega_{k,u,v} b_{u,v} \tag{3.30}$$

$$\text{s.t.} \quad \sum_{(u,v)\in A} q_{u,v} - \sum_{(v,w)\in A} q_{v,w} = 0 \qquad \forall v \in V \tag{3.31}$$

$$\sum_{(r,v)\in A} q_{r,v} = 1 \tag{3.32}$$

$$q_{u,v} + q_{v,u} \leq 1 \qquad \forall (u,v) \in E \tag{3.33}$$

$$\sum_{(u,v)\in A} q_{u,v} \leq 1 \qquad \forall v \in V \setminus \{r\} \tag{3.34}$$

$$\sum_{v\in\mathcal{B}(k)} \sum_{(u,v)\in A} q_{u,v} \geq 1 \tag{3.35}$$

$$\sum_{(u,v)\in A} s_{u,v} - \sum_{(v,w)\in A} s_{v,w} = \begin{cases} -1 & \text{if } v = r \\ 1 & \text{if } v = k \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V \tag{3.36}$$

$$s_{u,v} + s_{v,u} \leq 1 \qquad \forall (u,v) \in E \tag{3.37}$$

$$s_{u,v} \leq q_{u,v} \qquad \forall (u,v) \in A \setminus \mathcal{A}(k) \tag{3.38}$$

$$b_{u,v} \geq s_{u,v} - q_{u,v} \qquad \forall (u,v) \in \mathcal{A}(k) \tag{3.39}$$

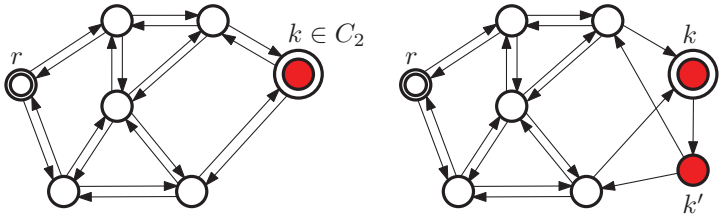
$$\sum_{(u,v)\in\mathcal{A}(k)} l_{u,v} b_{u,v} \leq b_{\max}(k) \tag{3.40}$$

$$q_{u,v} \in \{0,1\} \qquad \forall (u,v) \in A \tag{3.41}$$

$$s_{u,v} \in \{0,1\} \qquad \forall (u,v) \in A \tag{3.42}$$

$$0 \leq b_{u,v} \leq 1 \qquad \forall (u,v) \in \mathcal{A}(k) \tag{3.43}$$

The flow conservation constraints (3.31) ensure that the arcs (u,v) on which $q_{u,v} = 1$ form a directed cycle. Constraints (3.33) avoid the simultaneous usage of two oppositely directed arcs and constraints (3.34) prevent the repetition of nodes on the cycle. These constraints, in conjunction with constraints (3.32) and (3.35) which force the cycle to contain r and at least one potential branch node, ensure that the final cycle corresponds to two oppositely directed, internally node disjoint paths between r and some branch node. Due to the flow conservation constraints (3.36) together with constraints (3.37), variables $s_{u,v}$, $\forall (u,v) \in A$, describe a directed path from r to k. Furthermore, constraints (3.38) force this path to use arcs part of the above mentioned cycle outside the $b_{\max}$-neighborhood of k. Finally, constraints (3.39) ensure that variables $b_{u,v}$, $\forall (u,v) \in \mathcal{A}(k)$, indicate the arcs forming the branch line, whereas constraints (3.40) restrict the branch line's length.



a) Original graph: $G = (V, A)$ b) Transformed graph: $G' = (V', A')$

Figure 3.8: Transformation to ESPPRC for $k \in C_2$, with $b_{\max}(k) = 0$.

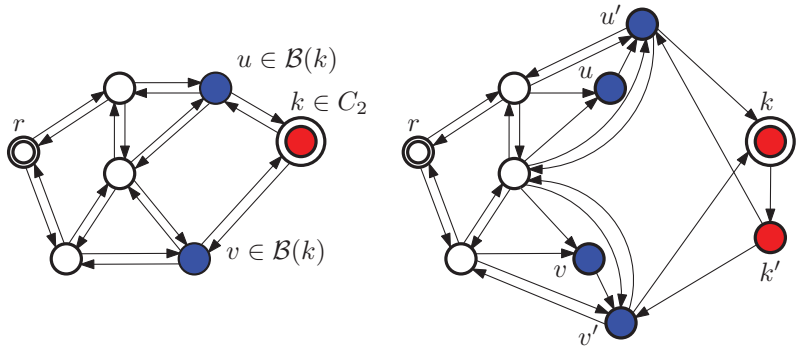
3.6.2 Modeling the Pricing Problem as an Elementary Shortest Path Problem with Resource Constraints

Without yet considering $b_{\max}$-redundancy, the pricing problem for a customer $k \in C_2$ can be interpreted as finding a cheapest cycle containing r and k. Finding negative cost cycles is a problem which frequently occurs as pricing problem in branch-and-price approaches from the context of vehicle routing and crew scheduling problems. Here, algorithms for solving the (elementary) shortest path problem with resource constraints (ESPPRC) are usually used for solving the pricing subproblem, see e.g. [99]. Thus, ESPPRC which is $\mathcal{NP}$-hard has recently gained great attention and several methods for solving it have been proposed [31, 59].

We transform the pricing subproblem for a customer $k \in C_2$ into an instance of the ESPPRC on graph $G'_k = (V'_k, A'_k)$ with the root node r being the source and destination node. The transformed graph – see Figure 3.8 for an example – is defined by its node set $V'_k = V \cup \{k'\}$ and its arc set $A'_k = \{(u, v) \in A \mid u \neq k\} \cup \{(k', v) \mid \exists (k, v) \in A\} \cup \{(k, k')\}$. Here, we replace node k by two nodes k and k' connected by an arc (k, k'). Each arc $(k, v) \in A$ emanating from k, is replaced by an arc $(k', v) \in A'_k$ going out from k'. We call k' the *split node* of k while we refer to arc (k, k') as *split arc* of k. Since k has only one outgoing arc, each non-trivial path in G'_k containing k which does not end at node k must also contain the split arc (k, k') between k and its split node k'.

Arc costs $c'_{u,v}$, are defined as

$$c'_{u,v} = \begin{cases} -\nu_k & \text{if } u = k \text{ and } v = k' \\ \omega_{k,k,v} & \text{if } u = k' \\ \omega_{k,u,v} & \text{otherwise} \end{cases} \qquad \forall (u, v) \in A'_k.$$



a) Original graph: $G = (V, A)$ b) Transformed graph: $G' = (V', A')$

Figure 3.9: Transformation to ESPPRC for $k \in C_2$, with $b_{\max}(k) > 0$.

As $(k, k') \in A'_k$ is the only arc with negative costs $c'_{k,k'} = -\nu_k$ in $G'_k = (V'_k, A'_k)$ and each pair of oppositely directed internally node-disjoint paths between r and k must have costs smaller than ν_k to price out favorably, we conclude that there is a one-to-one correspondence between the set of elementary shortest paths from r to itself with negative costs in $G'_k = (V'_k, A'_k)$ and the set of oppositely directed internally node-disjoint paths between r and k yielding negative reduced costs. As discussed by Boland et al. [28], node disjointness can be ensured by additionally adding one resource for each node $v \in V'_k$ with a maximum resource consumption of one for each individual node resource.

In the following, we slightly adapt the above mentioned transformation, in order to generalize it to the case of type-2 customer nodes $k \in C_2$ with $b_{\max}(k) > 0$. Here, we split each potential branch node $v \in \mathcal{B}(k)$ into nodes v, v' and add an arc (v, v') between each pair of nodes corresponding to one original node. In case, a path in G'_k corresponding to a feasible connection between r and k uses an arc between some potential branch node v and its split node v', v will be the branch node of the resulting connection. Since each potential branch node $v \in \mathcal{B}(k)$ except k can be used either as a connection's branch node or as a standard node of a connection to k, G'_k contains arcs (u, v) and (u, v') for each arc $(u, v) \in A$, $v \in \mathcal{B}(k)$, where $v \neq k$. Arcs $(v, w) \in A$ going out from $v \in \mathcal{B}(k)$ are replaced by arcs $(v', w) \in A'_k$. Formally the transformed graph $G'_k = (V'_k, A'_k)$ is defined by its node set $V'_k = V \cup \{v' \mid \exists v \in \mathcal{B}(k)\}$ and its arc set $A'_k = \{(u, v) \in A \mid u \neq \mathcal{B}(k)\} \cup \{(u', v) \mid \exists (u, v) \in A \wedge u \in \mathcal{B}(k)\} \cup \{(u, u') \mid u \in \mathcal{B}(k)\} \cup \{(u, v') \mid \exists (u, v) \in A \wedge v \in \mathcal{B}(k) \wedge v \neq k\}$, see Figure 3.9.

Let $\hat{c}_u \geq 0$, $\forall u \in \mathcal{B}(k)$, denote the costs of the precomputed branch line between u

and k when using node u as branch node of the connection between r and k with respect to arc costs $\omega_{k,u,w}$, $\forall (u,v) \in \mathcal{A}(k)$. Then arc costs $c'_{u,v}$ are defined as

$$c'_{u,v} = \begin{cases} -\nu_k + \hat{c}_u & \text{if } u \in \mathcal{B}(k) \\ \omega_{k,u,w} & \text{if } v \text{ is the split node of } w \\ \omega_{k,w,v} & \text{if } u \text{ is the split node of } w \\ \omega_{k,u,v} & \text{otherwise} \end{cases} \qquad \forall (u,v) \in A'_k.$$

Since only split arcs $(u,u') \in A'_k$, $\forall u \in \mathcal{B}(k)$, might eventually have negative costs $c'_{u,u'} = -\nu_k + \hat{c}_u$, and due to the above introduced transformation, there is a one-to-one correspondence between the set of feasible connections $p \in \mathcal{H}_k$ that price out favorably and the set of elementary shortest path from r to itself with negative costs in $G'_k = (V'_k, A'_k)$ using exactly one split arc.

Thus, by associating a resource of value one to each split arc, we can model the pricing problem for customer $k \in C$ as an elementary shortest path problem with resource constraints (ESPPRC) with a maximum resource consumption of one. Furthermore, above mentioned node resources for ensuring node disjointness need to be additionally considered.

3.6.3 Analyzing the Restricted Dual Problem

In accordance with Section 3.5.1, we analyze the dual problem of $(\text{dCol})^{\text{RMP}}$ to check whether we need to expect the same issues as for model (Col) when solving the linear relaxation of (dCol). If this is the case, we are interested if we can pursue a similar stabilization approach as proposed for the undirected model in Section 3.5.2.

Let $\gamma_e \leq 0$, $\forall e \in E$, denote the dual variable values associated to constraints (3.25) and $\rho_k \leq 0$, $\forall k \in C$, denote the dual variable values associated to constraints (3.27). Then the restricted dual problem – i.e. the dual of the restricted master problem $(\text{dCol})^{\text{RMP}}$ – for the SST variant of $b_{\max}$-SNDP is given by formulation (3.44)–(3.51).

$$\max \quad \sum_{k \in C} \rho_k + p_k + \sum_{e \in E} \gamma_e \tag{3.44}$$

$$\sum_{k \in C} \omega_{k,u,v} + \gamma_e \leq c_{u,v} \qquad \forall (u,v) \in A,\ e = (u,v) \in E \tag{3.45}$$

$$\nu_k - \sum_{(u,v) \in p} \omega_{k,u,v} \leq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{H}_k | \exists h_p^k \in \tilde{H} \tag{3.46}$$

$$-\nu_k + \rho_k \leq -p_k \qquad \forall k \in C \tag{3.47}$$

$$\omega_{k,u,v} \geq 0 \qquad \forall k \in C,\ \forall (u,v) \in A \tag{3.48}$$

$$\nu_k \geq 0 \qquad \forall k \in C \tag{3.49}$$

$$\gamma_e \leq 0 \qquad \forall e \in E \tag{3.50}$$

$$\rho_k \leq 0 \qquad \forall k \in C \tag{3.51}$$

Let $A'' = \{(u,v) \in A \mid \nexists h_p^k \in \tilde{H} : (u,v) \in p\}$ denote the set of arcs not included in any connection of $(\text{dCol})^{\text{RMP}}$. As only inequalities (3.45) are relevant for arcs $(u,v) \in A''$ and $\gamma_e \leq 0$, $\forall e \in E$, any variable values $\omega_{k,u,v} \geq 0$, $\forall k \in C$, $\forall (u,v) \in A''$, $e = (u,v) \in E$, are optimal with respect to model (3.44)–(3.51) as long as $\sum_{k \in C} \omega_{k,u,v} \leq c_{u,v} - \gamma_e$. In particular, it is easy to see that if $(u,v), (v,u) \in A''$, an optimal solution with $\gamma_e = 0$, $e = (u,v) \in E$, and $\omega_{k,u,v} = \omega_{k,v,u} = 0$, $\forall k \in C$, does exist.

Thus, next to the issue of degeneracy based upon the fact that only few arc and connection variables will be nonzero in any solution to $(\text{dCol})^{\text{RMP}}$, we observe that edge costs used in the pricing subproblems are not meaningful.

For the OPT variant the restrict dual problem is given by (3.52)–(3.57).

$$\max \quad \sum_{k \in C} \nu_k + \sum_{e \in E} \gamma_e \tag{3.52}$$

$$\sum_{k \in C} \omega_{k,u,v} + \gamma_e \leq c_{u,v} \qquad \forall (u,v) \in A,\ e = (u,v) \in E \tag{3.53}$$

$$\nu_k - \sum_{(u,v) \in p} \omega_{k,u,v} \leq 0 \qquad \forall k \in C,\ \forall p \in \mathcal{H}_k | \exists h_p^k \in \tilde{H} \tag{3.54}$$

$$\omega_{k,u,v} \geq 0 \qquad \forall k \in C,\ \forall (u,v) \in A \tag{3.55}$$

$$\nu_k \geq 0 \qquad \forall k \in C \tag{3.56}$$

$$\gamma_e \leq 0 \qquad \forall e \in E \tag{3.57}$$

We conclude, that all relevant observations described above for the SST variant remain valid for the slightly different formulation (3.52)–(3.57).

3.6.4 Alternative Dual-Optimal Solutions

Let $(\gamma^*, \rho^*, \nu^*, \omega^*)$ be an optimal solution to the restricted dual problem (3.44)–(3.51). As motivated for the undirected model in Section 3.5.2 we focus on increasing dual variable values ω^* used as arc costs in the pricing problem. Thus, the costs for individual connections will rise and we expect that less connections are finally included in $(\text{dCol})^{\text{RMP}}$. As inequalities (3.45) – respectively inequalities (3.53) – are the only constraints imposing upper bounds for dual variables ω, we only need to consider these constraints when increasing their values.

Let $\delta_{u,v} = c_{u,v} - \gamma_e - \sum_{k \in C} \omega_{k,u,v}$, $\forall (u,v) \in A$, $e = (u,v) \in E$, denote the total amount by which we can increase the sum of dual variable values ω on arc (u,v). It is easy to see that $\delta_{u,v} = c_{u,v}$, $\forall (u,v) \in A : (u,v), (v,u) \in A''$, where $A'' \subseteq A$ denotes the subset of arcs which are not part of any so far included connection[2]. Generally, $\delta_{u,v}$ will also be greater than zero at least for some arcs $(u,v) \in A \setminus A''$.

As in Section 3.5.2 we pursue four strategies for generating alternative dual optimal solutions. For $D' = (\gamma^*, \rho^*, \nu^*, \omega')$, we set $\omega'_{k,u,v} = \omega^*_{k,u,v} + \frac{\delta_{u,v}}{|C|}$, $\forall k \in C$, $\forall (u,v) \in A$. As $\sum_{k \in C} \omega'_{k,u,v} - \omega^*_{k,u,v} = \sum_{k \in C} \frac{\delta_{u,v}}{|C|} = \delta_{u,v}$, $\forall (u,v) \in A$, D' is feasible for the restricted dual problem in the OPT as well as in the SST case. Furthermore, since the objective value does not change due to our adaptation, D' is dual optimal.

We further apply the parameterized approach, where dual optimal solutions $D^{(k,d)} = (\gamma^*, \rho^*, \nu^*, \omega^{(k,d)})$ with $\omega^{(k,d)}_{k,u,v} = \omega^*_{k,u,v} + \frac{\delta_{u,v}}{d}$, $\forall k \in C$, $\forall (u,v) \in A$, are used. Here, we initialize d to be equal to one and gradually increment d up to $|C|$ if no column could be priced in and reset d to one in case a column including new arcs has been added. We further apply its simpler variant where d is immediately set to $|C|$ if no connection variable prices out favorably and d will not be decremented any more. We refer to the corresponding dual optimal solutions by $D^{(k,d')}$. All above mentioned strategies are valid for both, the SST as well as the OPT variant of our problem.

Finally in our last strategy which is feasible for the SST variant only we use dual optimal solutions $D^{(p)} = (\gamma^*, \rho^*, \nu^*, \omega^{(p)})$ with $\omega^{(p)}_{k,u,v} = \omega^*_{k,u,v} + \delta_{u,v} \frac{p_k}{\sum_{l \in C} p_l}$, $\forall k \in C$, $\forall (u,v) \in A$.

[2] Since CPLEX [98] will compute dual variable values equal to zero in this case.

3.7 Polyhedral Comparison

In this section, we theoretically compare the undirected and directed connection formulation to each other as well as to previous formulations introduced by Wagner et al. [178, 177] based on multi-commodity flows [178] and connectivity cuts [177], respectively. Hereby, we denote by $\mathcal{P}_{\text{col}}$ the polyhedron corresponding to the set of feasible solutions to the linear relaxation of model (Col). Similarly, $\mathcal{P}_{\text{dcol}}$ denotes the polyhedron induced by the LP relaxation of model (dCol), $\mathcal{P}_{\text{mcf}}$ those of the multi-commodity flow formulation (3.58)–(3.76) from [178], and $\mathcal{P}_{\text{cut}}$ the polyhedron corresponding to the cut formulation (3.77)–(3.93) from [177].

In their MCF formulation, Wagner et al. [178] used arc set $A_r = \{(r,j) \in E \mid j \in S\}$ denoting all arcs connecting r with Steiner nodes, the set of edges $E_S(k) = \{(i,j) \mid i,j \in V \setminus \{r,k\}\}$ connecting two Steiner nodes with respect to customer $k \in C$, as well as the corresponding arc set $A_S(k) = \{(i,j),\ (j,i) \mid (i,j) \in V \setminus \{r,k\}\}$. Furthermore, $A(k) = \{(i,k) \mid (i,j) \in E\}$ denotes the set of arcs to customer $k \in C$, and $A'(k) = A_r \cup A_S(k) \cup A(k)$ the set of all arcs relevant for a customer k. Finally, $B(k)$ denotes the set of arcs $(i,j) \in A'(k)$, with $i,j \in \mathcal{B}(k)$, i.e. those arcs that are potentially used in a branch line of a connection to customer k.

In formulation (3.58)–(3.76) introduced by Wagner et al. [178] variables $x_{i,j} \in \{0,1\}$, $\forall (i,j) \in E$, indicate whether edge (i,j) is used ($x_{i,j} = 1$) in a solution or not ($x_{i,j} = 0$). Flow variables $0 \leq m^k_{i,j} \leq 1$, $\forall k \in C$, $\forall (i,j) \in A'(k)$, and $0 \leq n^k_{i,j} \leq 1$, $\forall k \in C_2$, $(i,j) \in A'(k)$, model the connection to a customer node k. Here, the second set of flow variables is used to achieve redundancy for type-2 customers. Variables $0 \leq q^k_{i,j} \leq 1$, $\forall k \in C_2$, $\forall (i,j) \in B(k)$, indicate the edges used in the branch line to node k. Finally, variables $y_k \in \{0,1\}$, $\forall k \in C$, indicate in the SST variant whether customer node k is connected ($y_k = 1$) or not ($y_k = 0$). In the OPT variant, these variables are fixed to one. Using these sets and variables, $b_{\max}$-SNDP can be described by the following MIP:

$$\min \quad \sum_{(i,j)\in E} c_{i,j} x_{i,j} + \sum_{k \in C} p_k (1 - y_k) \tag{3.58}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A'(k)} m^k_{i,j} - \sum_{(j,i)\in A'(k)} m^k_{j,i} = \begin{cases} -y_k & \text{if } j = r \\ y_k & \text{if } j = k \ \ \forall k \in C,\ \forall j \in V \\ 0 & \text{otherwise} \end{cases} \tag{3.59}$$

$$\sum_{(i,j)\in A'(k)} n^k_{i,j} - \sum_{(j,i)\in A'(k)} n^k_{j,i} = \begin{cases} -y_k & \text{if } j = r \\ y_k & \text{if } j = k \quad \forall k \in C,\ \forall j \in V \\ 0 & \text{otherwise} \end{cases} \tag{3.60}$$

$$m^k_{i,j} \le x_{i,j} \qquad \forall k \in C,\ \forall (i,j) \in A_r \cup A(k) \tag{3.61}$$

$$m^k_{i,j} + m^k_{j,i} \le x_{i,j} \qquad \forall k \in C,\ \forall (i,j) \in E_S(k) \tag{3.62}$$

$$n^k_{i,j} \le x_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in A_r \cup A(k) \tag{3.63}$$

$$n^k_{i,j} + n^k_{j,i} \le x_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in E_S(k) \tag{3.64}$$

$$m^k_{i,j} + n^k_{j,i} \le x_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in A_S(k) \tag{3.65}$$

$$m^k_{i,j} + n^k_{i,j} \le x_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in A'(k) \setminus B(k) \tag{3.66}$$

$$m^k_{i,j} + n^k_{i,j} - q^k_{i,j} \le x_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \tag{3.67}$$

$$q^k_{i,j} \le m^k_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \tag{3.68}$$

$$q^k_{i,j} \le n^k_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \tag{3.69}$$

$$\sum_{(i,j)\in B(k)} (m^k_{i,j} + n^k_{i,j} - q^k_{i,j}) + \\ + \sum_{(i,j)\in A'(k)\setminus B(k)} (m^k_{i,j} + n^k_{i,j}) \le 1 \qquad \forall k \in C_2,\ \forall i \in V \setminus \{r,k\}, \tag{3.70}$$

$$\sum_{(i,j)\in B(k)} l_{i,j} q^k_{i,j} \le b_{\max}(k) \qquad \forall k \in C_2 \tag{3.71}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in E \tag{3.72}$$

$$y_k \in \{0,1\} \qquad \forall k \in C \tag{3.73}$$

$$0 \le m^k_{i,j} \le 1 \qquad \forall k \in C,\ \forall (i,j) \in A'(k) \tag{3.74}$$

$$0 \le n^k_{i,j} \le 1 \qquad \forall k \in C_2,\ \forall (i,j) \in A'(k) \tag{3.75}$$

$$0 \le q^k_{i,j} \le 1 \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \tag{3.76}$$

Lemma 8 *The multi-commodity flow formulation (3.58)–(3.76) from [178] does not dominate* (Col)*, i.e.* $proj_{x,y}(\mathcal{P}_{\text{mcf}}) \nsubseteq proj_{x,y}(\mathcal{P}_{\text{col}})$.

Proof Consider the instance of $b_{\max}$-SNDP given in Figure 3.10. Obviously, the optimal solution to $(\text{Col})^{\text{LP}}$ does not connect customer $j \in C_2$ since it does not pay off, i.e. all variables will be set to zero and thus the objective value is equal to five. $\mathcal{P}_{\text{mcf}}$, however, does contain the solution depicted in Figure 3.11, where both types of flows – i.e. m and n – to $j \in C_2$ each of which of value 0.5 are routed over the same arcs. Thus, by setting $y_j = 0.5$ and the resulting edge variables $x_{r,h}$ and $x_{h,j}$ to one, the costs for connecting customer j in such a way are lower than the resulting

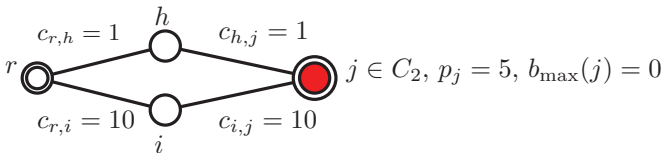


Figure 3.10: An exemplary instance of $b_{\max}$-SNDP with a single customer node.

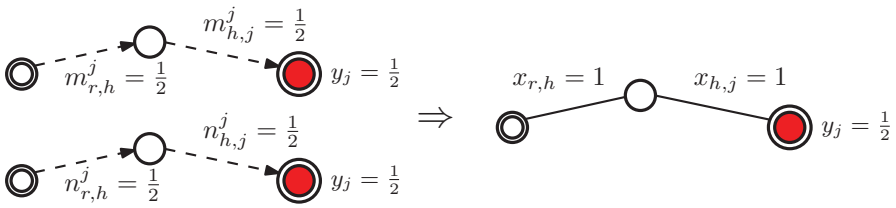


Figure 3.11: A feasible solution of $\mathcal{P}_{\text{mcf}}$ for the instance given in Figure 3.10.

profit. Finally, the objective value of the solution depicted in Figure 3.11 is equal to 4.5.

Lemma 9 *Let k be an arbitrary customer $k \in C$ connected in some – potentially fractional – solution $G' \in \mathcal{P}_{\text{col}}$ and y_k denote its variable value in G'. Furthermore, let $x_e \geq \sum_{p \in \mathcal{F}_k | e \in p} f^k_p$, $\forall e \in E$, denote the variable values of all edge variables induced by the (fractional) connections to k due to constraints (3.5).*
Then, variable values x_e, $\forall e \in E$, allow for describing a feasible connection to customer k of value y_k in $\mathcal{P}_{\text{mcf}}$.

Proof Let $f^k_p \in F_k$ be an arbitrary connection variable corresponding to connection $p \subseteq \mathcal{F}_k$. Since f_k contains only feasible connections to customer k, we can derive a feasible flow (m^k, n^k) of value f^k_p by orienting each edge $e \in p$ towards k. Furthermore, as the length constraints with respect to the branch line are met by definition of f_k, we conclude that each undirected connection $p \in f_k$ can be represented as a set of flow variables corresponding to a feasible connection of value f^k_p in model (3.58)–(3.76).
As $\sum_{p \in \mathcal{F}_k | e \in p} f^k_p \leq x_e$, $\forall e \in E$, holds due to constraints (3.5), we can simply use above mentioned equivalence between connections and feasible flows for each connection individually, yielding feasible flow variable values (m^k, n^k) that allow for setting the customer variable y_k to $\sum_{p \in \mathcal{F}_k} f^k_p \geq y_k$ in model (3.58)–(3.76).

Theorem 10 *The undirected connection formulation* (Col) *strictly dominates the multi-commodity flow formulation (3.58)–(3.76) from [178], i.e.* $proj_{x,y}(\mathcal{P}_{\text{col}}) \subsetneq proj_{x,y}(\mathcal{P}_{\text{mcf}})$.

Proof Due to Lemma 8, it is enough to show that any feasible solution to $(\text{Col})^{\text{LP}}$, can be projected into a feasible solution to formulation (3.58)–(3.76) with identical objective value, i.e. with identical values for x_e, $\forall e \in E$, and y_k, $\forall k \in C$. Since no constraint of formulation (3.58)–(3.76) considers multiple customers simultaneously, we can take into account each customer individually. Thus, Theorem 10 follows due to Lemma 9.

In their second formulation based on connectivity cuts, Wagner et al. [177] used variables $x_{i,j} \in \{0,1\}$, $\forall (i,j) \in E$, indicating edges being part of a solution. Variables $a_{i,j} \in \{0,1\}$, $\forall (i,j) \in A_D$, are defined on the arc set A_D, containing one arc going out of r and two oppositely directed arcs for the remaining edges. Variables $y_k \in \{0,1\}$, $\forall k \in C$, which are fixed to one in the OPT variant, specify whether a customer k is connected or not. Binary variables $z_i \in \{0,1\}$, $\forall i \in V$, indicate whether a node i has two node-disjoint paths to r and variables $b_j^k \in \{0,1\}$, $\forall k \in C_2$, $\forall j \in \mathcal{B}(k)$, denote whether j is the branching node of customer k. Finally, variables $q_{i,j}^k \in \{0,1\}$, $\forall k \in C_2$, $\forall (i,j) \in B(k)$, describe the branch line of the connection to customer k.

$$\min \quad \sum_{(i,j)\in E} c_{i,j}x_{i,j} + \sum_{k\in C} p_k(1-y_k) \tag{3.77}$$

$$\text{s.t.} \quad a(\delta^-(S)) \geq y_k \qquad \forall k \in C,\ \forall S \subseteq V \setminus \{r\} \mid k \in S \tag{3.78}$$

$$a(\delta^-(S)) \geq 2z_i \qquad \forall i \in V \setminus \{r\},\ \forall S \subseteq V \setminus \{r\} \mid i \in S \tag{3.79}$$

$$a(\delta^-_{V\setminus\{v\}}(S)) \geq z_i \qquad \forall i \in V \setminus \{r\},\ \forall v \in V \setminus \{r,i\},\ \forall S \subseteq V \setminus \{r,v\} \mid i \in S \tag{3.80}$$

$$\sum_{j\in\mathcal{B}(k)} b_j^k = y_k \qquad k \in C_2 \tag{3.81}$$

$$b_j^k \leq z_j \qquad \forall k \in C_2, \forall j \in \mathcal{B}(k) \tag{3.82}$$

$$q_{i,j}^k \leq a_{i,j} \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \tag{3.83}$$

$$q^k(\delta^-_{\mathcal{B}(k)}(S)) \geq b_j^k \qquad \forall k \in C_2,\ \forall j \in \mathcal{B}(k) \setminus \{k\},\ \forall S \subseteq \mathcal{B}(k) \setminus \{j\} \mid k \in S \tag{3.84}$$

$$\sum_{(i,j)\in B(k)} l_{i,j}q_{i,j}^k \leq b_{\max}(k) \qquad \forall k \in C_2 \tag{3.85}$$

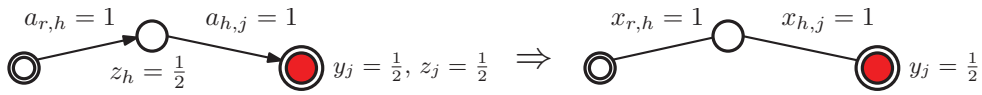


Figure 3.12: A feasible solution of $\mathcal{P}_{\text{cut}}$ for the instance given in Figure 3.10.

$$a_{i,j} \leq x_{i,j} \qquad \forall (i,j) \in A_D \qquad (3.86)$$
$$a_{j,i} \leq x_{i,j} \qquad \forall (i,j) \in A_D \qquad (3.87)$$
$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in E \qquad (3.88)$$
$$a_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A_D \qquad (3.89)$$
$$y_k \in \{0,1\} \qquad \forall k \in C \qquad (3.90)$$
$$z_i \in \{0,1\} \qquad \forall i \in V \qquad (3.91)$$
$$b_j^k \in \{0,1\} \qquad \forall k \in C_2,\ \forall j \in \mathcal{B}(k) \qquad (3.92)$$
$$q_{i,j}^k \in \{0,1\} \qquad \forall k \in C_2,\ \forall (i,j) \in B(k) \qquad (3.93)$$

Lemma 11 *The cut formulation (3.77)–(3.93) from [177] does not dominate* (Col)*, i.e.* $proj_{x,y}(\mathcal{P}_{\text{cut}}) \nsubseteq proj_{x,y}(\mathcal{P}_{\text{col}})$.

Proof Consider the instance given in Figure 3.10 with an optimal solution value to the LP relaxation of (Col) equal to five, where all variable values are set to zero. As the multi-commodity flow formulation, the cut model (3.77)–(3.93) does allow for "half-connecting" customer $j \in C_2$ via a single path where the corresponding arc and edge variables are set to one, see Figure 3.12.

Lemma 12 *Let k be an arbitrary customer $k \in C$ connected in some – potentially fractional – solution $G' \in \mathcal{P}_{\text{col}}$ and y_k denote its variable value in G'. Furthermore, let $x_e \geq \sum_{p \in \mathcal{F}_k | e \in p} f_p^k$, $\forall e \in E$, denote the variable values of all edge variables induced by the (fractional) connections to k due to constraints (3.5).*
Then, variables values x_e, $\forall e \in E$, allow for describing a feasible connection of value y_k in $\mathcal{P}_{\text{cut}}$.

Proof As already observed by Chimani et al. [36] model (3.77)–(3.93) uses directed variables $a_{i,j}$, $\forall (i,j) \in A_D$, but is equivalent to an undirected model since constraints (3.86) and (3.87) allow for simultaneously using oppositely directed arcs corresponding to a single edge simultaneously without increasing the cost function, i.e. $a_{i,j} = a_{j,i} = x_{i,j}$, $\forall (i,j) \in E$, $i \neq r$, $j \neq r$.

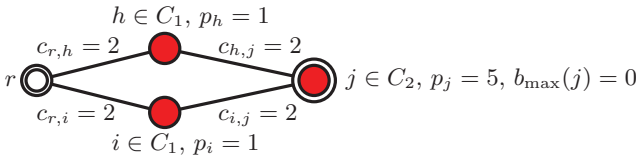


Figure 3.13: Another exemplary instance of $b_{\max}$-SNDP.

Let $p \in \mathcal{F}_k$ be an arbitrary connection to customer k. Due to constraints (3.5), p induces variable values $x_e^{(\mathrm{p})} \geq f_p^k$. By definition of $\mathcal{F}_k$, p is a feasible connection to k and thus setting $a_{i,j} = a_{j,i} = x_e^{(\mathrm{p})}$, $\forall e \in p$, allows for supplying customer k with a value of f_p^k in model (3.77)–(3.93). Due to constraints (3.4), $y_k \leq \sum_{p \in F_k} f_p^k$ holds and thus Lemma 12 follows.

Theorem 13 *The undirected connection formulation* (Col) *strictly dominates the cut formulation (3.77)–(3.93) from [177], i.e.* $proj_{x,y}(\mathcal{P}_{\mathrm{col}}) \subsetneq proj_{x,y}(\mathcal{P}_{\mathrm{cut}})$.

Proof Since model (3.77)–(3.93) considers each customer individually, Theorem 13 follows due to Lemmas 11 and 12.

Theorem 14 *The directed connection formulation* (dCol) *strictly dominates the undirected variant* (Col)*, i.e.* $proj_{x,y}(\mathcal{P}_{\mathrm{dcol}}) \subsetneq proj_{x,y}(\mathcal{P}_{\mathrm{col}})$.

Proof It is easy to see that $proj_{x,y}(\mathcal{P}_{\mathrm{dcol}}) \subseteq proj_{x,y}(\mathcal{P}_{\mathrm{col}})$ holds, if $proj_{x,y}(\mathcal{P}_{\mathrm{dcol}})$ denotes the obvious projection of $\mathcal{P}_{\mathrm{dcol}}$ into the space of $\mathcal{P}_{\mathrm{col}}$, i.e. $x_e = a_{i,j} + a_{j,i}$, $\forall e = (i,j) \in E$.
Consider the instance given in Figure 3.13 and the optimal solution G'_{col} of $(\mathrm{Col})^{\mathrm{LP}}$ to this instance as shown in Figure 3.14. Here, each type-1 customer is connected via two connections. The corresponding edges are also used for connecting the type-2 customer $j \in C_2$. Thus, it is possible to set $y_h = y_i = 1$ and $y_j = 0.5$, while all edge variables are set to 0.5. The objective value of the shown solution is $o(G'_{\mathrm{col}}) = 6.5$. On the other hand, the optimal oriented solution G'_{dcol} of $(\mathrm{dCol})^{\mathrm{LP}}$ does not connect any customers, i.e. $y_k = 0$, $\forall k \in \{h,i,j\}$, and $a_{u,v} = 0$, $\forall (u,v) \in A$. We conclude that (dCol) strictly dominates its undirected variant, i.e. $proj_{x,y}(\mathcal{P}_{\mathrm{dcol}}) \subsetneq proj_{x,y}(\mathcal{P}_{\mathrm{col}})$.

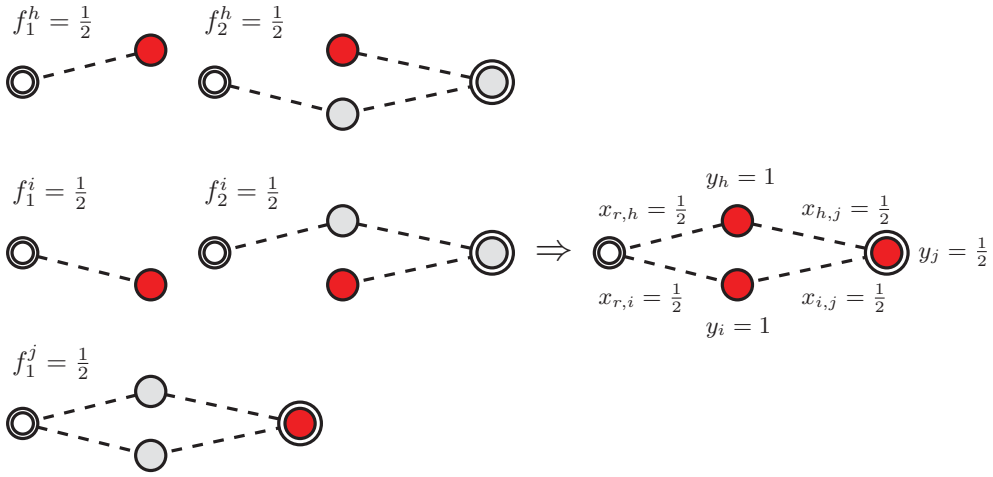


Figure 3.14: A feasible solution of $\mathcal{P}_{\text{col}}$ for the instance given in Figure 3.13.

3.8 Lagrangian Decomposition

In this section, we present a Lagrangian decomposition approach for $b_{\max}$-SNDP based on the abstract ILP model given by equations (3.94)–(3.98). Here, we utilize decision variables $x_e \in \{0,1\}$, $\forall e \in E$, indicating whether or not edge e is part of the solution, i.e. $x_e = 1 \leftrightarrow e \in E'$. For customer nodes $k \in C$ variables $y_k \in \{0,1\}$ denote whether or not feasible connections according to the customers' types and $b_{\max}(k)$ exist. Our model is based on the MCF formulation from [178], but all the different types of flow variables for each customer $k \in C$ on directed arcs are replaced by simple variables $f_e^k \in \{0,1\}$, $\forall e \in E$, indicating whether or not edge e is part of the single path (type-1) or pair of disjoint paths plus the eventual branch line (type-2) for connecting customer k; f^k denotes the vector of all these variables for a customer k.

Let F_k, $\forall k \in C$, be the set of all incidence vectors on E corresponding to feasible connections for customer k. We can now formulate the SST-variant of our problem in the following abstract way:

$$\min \quad \sum_{e \in E} c_e x_e + \sum_{k \in C} p_k (1 - y_k) \tag{3.94}$$

$$\text{s.t.} \quad f_e^k \leq x_e \qquad \forall k \in C,\ \forall e \in E \tag{3.95}$$

$$f^k \in F_k \text{ if } y_k = 1 \qquad \forall k \in C \tag{3.96}$$

$$f_e^k \in \{0, 1\} \qquad \forall k \in C,\ \forall e \in E \tag{3.97}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{3.98}$$

Inequalities (3.95) are called coupling constraints and enforce an edge to appear in the solution when it is used for connecting at least one customer. Conditions (3.96) ensure feasible connections for all selected customers ($y_k = 1$). The OPT-variant of the model is obtained by simply ignoring the second term in the objective function and the conditions on y_k in (3.96).

Note that in this form, the model is not yet a concrete ILP, as conditions (3.96) are not expressed by means of linear inequalities. Ideally, we would substitute them by a set of linear inequalities describing the convex hull $\text{conv}(F_k)$ of all incidence vectors of feasible connections for each customer k depending on variables y_k. Unfortunately, finding a (compact) set of such inequalities is not trivial. While this can be achieved for simple (type-1) connections via a network flow formulation, this task is difficult for the biconnected case involving branch lines (type-2).

The MCF model from [178] – see also Section 3.7 – represents a concrete instantiation of this abstract model. As can be easily shown, however, it does not contain a complete description of $\text{conv}(F_k)$ but just a formulation that is valid for integer solutions. We conclude that the MCF-formulation from [178] therefore is not as strong as an "ideal" instantiation of the abstract model.

We relax the coupling constraints (3.95) of our abstract model in a classical Lagrangian fashion, i.e. by substituting them with corresponding penalty terms in the objective function. This yields model (LR(λ)):

$$\min \quad \sum_{e\in E} c_e x_e + \sum_{k\in C} p_k(1-y_k) + \sum_{k\in C}\sum_{e\in E} \lambda_{k,e}\cdot(f_e^k - x_e) = \tag{3.99}$$

$$= \sum_{k\in C} p_k + \sum_{e\in E}\left(c_e - \sum_{k\in C}\lambda_{k,e}\right)x_e + \sum_{k\in C}\left(\sum_{e\in E}\lambda_{k,e}f_e^k - p_k y_k\right) \tag{3.100}$$

$$\text{s.t.} \quad f^k \in F_k \text{ if } y_k = 1 \qquad \forall k \in C \tag{3.101}$$

$$f_e^k \in \{0,1\} \qquad \forall k \in C,\ \forall e \in E \tag{3.102}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{3.103}$$

Parameters $\lambda_{k,e} \geq 0$, $\forall k \in C$, $\forall e \in E$, are the Lagrangian multipliers, and for any feasible instantiation of them an optimal solution of (LR(λ)) yields a lower bound on the optimal solution value of our original abstract model [18].

For a specific selection of λ, this relaxation can be efficiently solved, since it decomposes into independent subproblems corresponding to models (3.104)–(3.106) and (3.107)–(3.108), respectively.

$$\min \quad \sum_{k\in C} p_k + \sum_{k\in C}\left(\sum_{e\in E}\lambda_{k,e}f_e^k - p_k y_k\right) \tag{3.104}$$

$$\text{s.t.} \quad f^k \in F_k \text{ if } y_k = 1 \qquad \forall k \in C \tag{3.105}$$

$$f_e^k \in \{0,1\} \qquad \forall k \in C,\ \forall e \in E \tag{3.106}$$

Model (3.104)–(3.106) resembles $|C|$ independent problems of determining individual cheapest connections for each $k \in C$ on a graph whose edge costs are $\lambda_{k,e}$. A node k is finally connected ($y_k = 1$) and the variables f_e^k corresponding to the identified connection are set to one if and only if the connection pays off, i.e. if $\sum_{e\in E}\lambda_{k,e}f_e^k \leq p_k$. Otherwise, the connection is discarded by setting $y_k = 0$ and $f_e^k = 0$, $\forall e \in E$.

$$\min \quad \sum_{e\in E}\left(c_e - \sum_{k\in C}\lambda_{k,e}\right)x_e \tag{3.107}$$

$$\text{s.t.} \quad x_e \in \{0,1\} \qquad \forall e \in E \tag{3.108}$$

Optimal values for variables x_e, $e \in E$, are independently determined by simple inspection, i.e. $x_e = 1$ iff $c_e < \sum_{k\in C}\lambda_{k,e}$, $\forall e \in E$.

The Lagrangian dual problem is the challenge of finding an optimal vector of Lagrange multipliers λ^* so that the lower bound obtained by (LR(λ^*)) becomes as large as possible. As this maximization problem is convex and piecewise linear, subgradient algorithms are well suited for this purpose [18]. While different variants of such methods exist, the volume algorithm [13] has proven to be more effective than several alternatives on various occasions [11, 86], and we therefore apply it here. Also, our preliminary comparisons indicate the superiority of this algorithm over the standard subgradient strategy as described in [18], see also Section 2.1.7.

3.8.1 Theoretical Comparison to the MCF Formulation

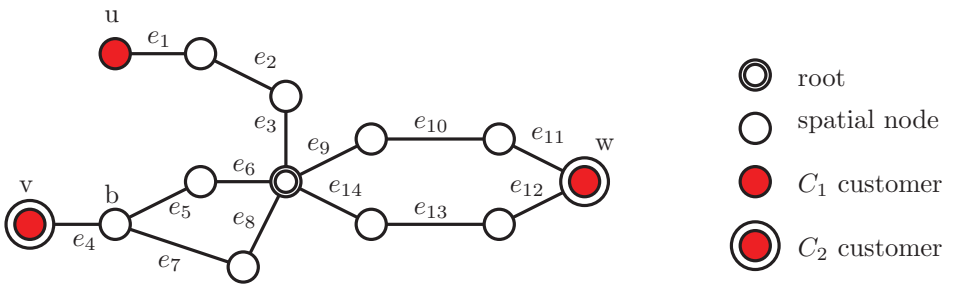
For each concrete instantiation of λ, all subproblems obtained by the Lagrangian decomposition are always solved to optimality and integrality. Therefore, $f^k \in \text{conv}(F_k)$ holds for all $k \in C$, and the abstract constraints (3.96) of our model (3.94)–(3.98) can be regarded as "ideally instantiated". Assuming we would be able to identify an optimal Lagrange vector λ^*, the lower bound obtained by (LR(λ^*)) is at least as good as the lower bound determined by an LP relaxation of the model. As already argued before, the MCF formulation from [178] is weaker than an "ideal" instantiation of the abstract model, compare also Figure 3.11. We therefore conclude that (LR(λ^*)) is stronger than the LP relaxation of the MCF formulation.

3.9 Neighborhood Structures for Improving Primal Solutions

Our algorithms make use of three types of neighborhood structures. While the first two aim to reduce the cost of a given solution, the last type consisting of two concrete neighborhood structures tries to improve a solution by removing customers from a candidate solution. Therefore, the latter is only applicable to the SST variant.

For our neighborhood structures, next to its node set $V' \subseteq V$ and its edge set $E' \subseteq E$, each candidate solution $G' = (V', E')$ is further represented by its set of feasibly connected customers $C' \subseteq C$ and the corresponding individual connections $X' = \{E'_k \mid k \in C'\}$, where E'_k denotes the set of edges used to connect customer k. Note that there may exist multiple connections to a single customer node in a solution G' in which case we store only one of them.

Furthermore, for each connection E'_k we maintain its internal structure consisting of its branch node $B(E'_k) \in V'$, edge sets $P(E'_k), Q(E'_k) \subseteq E'$ of its two paths between r and $B(E'_k)$ and finally the edge set of its branch line $L(E'_k) \subseteq E'$. Note that we



$E'_{\mathrm{u}} = \{e_1, e_2, e_3\}$	$E'_{\mathrm{v}} = \{e_4, e_5, e_6, e_7, e_8\}$	$E'_{\mathrm{w}} = \{e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$
$B(E'_{\mathrm{u}}) = u$	$B(E'_{\mathrm{v}}) = b$	$B(E'_{\mathrm{w}}) = w$
$P(E'_{\mathrm{u}}) = \{e_1, e_2, e_3\}$	$P(E'_{\mathrm{v}}) = \{e_5, e_6\}$	$P(E'_{\mathrm{w}}) = \{e_9, e_{10}, e_{11}\}$
$Q(E'_{\mathrm{u}}) = \emptyset$	$Q(E'_{\mathrm{v}}) = \{e_7, e_8\}$	$Q(E'_{\mathrm{w}}) = \{e_{12}, e_{13}, e_{14}\}$
$L(E'_{\mathrm{u}}) = \emptyset$	$L(E'_{\mathrm{v}}) = \{e_4\}$	$L(E'_{\mathrm{w}}) = \emptyset$

Figure 3.15: An exemplary candidate solution and the representation of its connections.

assume $B(E'_k) = k$ if $b_{\max}(k) = 0$ or $k \in C_1$ as well as define $P(E'_k)$ to be the "first" path of a connection, i.e. $P(E'_k)$ is used for type-1 customers while $Q(E'_k) = L(E'_k) = \emptyset$ for type-1 customers, see Figure 3.15. Finally, to allow for efficient updates of a solution with respect to connections, we maintain for each edge $e \in E'$ a list of the customers that are connected via this edge: $M_e = \{k \in C' \mid e \in E'_k\}$.

3.9.1 Connection Exchange Neighborhood

The *Connection Exchange Neighborhood* (CEN) consists of all solutions differing from the current solution G' by exactly one connection E'_k, see Algorithm 3.2. To determine the best neighboring solution for a fixed customer $k \in C'$, CEN calculates the saving due to removing the corresponding connection E'_k (which is the sum of all edge costs exclusively used to connect k). The connection to k leading to minimum additional costs is then determined by calculating the cheapest feasible connection to k in a graph with edge costs $c'_e = 0,\ \forall e \in E'' = E' \setminus \{e \in E'_k \mid M_e = \{k\}\}$ and $c'_e = c_e,\ \forall e \in E \setminus E''$, see Figure 3.16 for an exemplary move.

For type-1 customer nodes as well as type-2 customer nodes with $b_{\max}(k) = 0$, the computational complexity of finding this new connection for one specific client node k is bounded by $O(|E| + |V| \log |V|)$, whereas the worst case complexity for a single type-2 customer k with $b_{\max}(k) > 0$ is $O(b_{\max}(k)|\mathcal{E}(k)| + |E| + |V| \log |V|)$, see

Algorithm 3.2: Connection Exchange (Solution G')

$c'_e = 0, \forall e \in E'$
$c'_e = c_e, \forall e \in E \setminus E'$
$d_{\text{opt}} = 0$
forall the $k \in C'$ **do**
  $E'' = \{e \in E'_k \mid M_e = \{k\}\}$
  $c'_e = c_e, \forall e \in E''$
  $d = \sum_{e \in E''} c_e$
  $E''_k =$ shortest connection to k using edge costs c'
  $d = \sum_{e \in E''} c_e - \sum_{e \in E''_k} c'_e$
  if $d > d_{\text{opt}}$ **then**
    $d_{\text{opt}} = d$
    store solution G' with E''_k replacing E'_k as best solution
  $c'_e = 0, \forall e \in E''$
return best solution

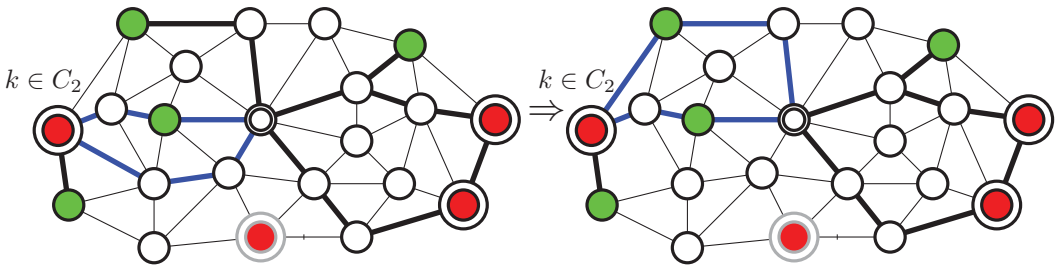


Figure 3.16: An exemplary connection exchange for customer $k \in C_2$.

Section 3.4. Thus, the overall complexity of completely searching CEN is bounded by $O(|C|(\max_{k\in C}\{b_{\max}(k)|\mathcal{E}(k)|\} + |E| + |V|\log|V|))$.

3.9.2 Key-Path Exchange Neighborhood

A *key-node* of a solution G' is a node $v \in V' \setminus C'$ with node degree $\deg_{G'}(v) \geq 3$, while a *key-path* is a path $K_{\mathrm{P}} = (V_{\mathrm{P}}, E_{\mathrm{P}})$ whose end nodes are either key-nodes, connected customer nodes $k \in C'$, or the root node r, while all other nodes are Steiner nodes $v \in V' \setminus (C' \cup \{r\})$ of degree two, i.e. $\deg_{G'}(v) = 2$. This concept of key-paths is well known for the STP and several metaheuristic methods utilizing a key-path exchange neighborhood have been proposed, see e.g. [137]. The *Key-Path Exchange Neighborhood* (KPEN) given in Algorithm 3.3 extends this concept by exchanging key-paths while respecting node- as well as $b_{\max}$-redundancy, see also Figure 3.17 for a visualization of an exemplary path exchange.

Algorithm 3.3: Key-Path Exchange (Solution G')

determine key-paths W
$d_{\mathrm{opt}} = 0$
forall the key-paths $(V_{\mathrm{P}}, E_{\mathrm{P}}) \in W$ **do**
  // actual key-path connects its end nodes m, n
  $c'_e = 0\ \forall e \in E' \setminus E_{\mathrm{P}}$
  $c'_e = c_e\ \forall e \in E_{\mathrm{P}} \cup (E \setminus E')$
  choose $e \in E_{\mathrm{P}}$ randomly
  $l_{\max} = \infty$
  forall the $k \in M'_e$ **do**
    if $e \in P(E'_k)$ **then**
      $c'_e = \infty$, $\forall e \in E$ incident to a inner node of $Q(E'_k)$
    else if $e \in Q(E'_k)$ **then**
      $c'_e = \infty$, $\forall e \in E$ incident to a inner node of $P(E'_k)$
    else if $e \in L(E'_k)$ **then**
      $l_{\max} = b_{\max}(k) - \sum_{e\in(L(E'_k)\setminus E_{\mathrm{P}})} l_e$
  $(V'_{\mathrm{P}}, E'_{\mathrm{P}})$ = shortest path from m to n using c'_e with max. length $l_{\max}$
  $d = \sum_{e\in E_{\mathrm{P}}} c_e - \sum_{e\in E_{\mathrm{P}'}} c'_e$
  if $d > d_{\mathrm{opt}}$ **then**
    $d_{\mathrm{opt}} = d$
    store solution G' with $(V_{\mathrm{P}}, E_{\mathrm{P}})$ replacing $(V_{\mathrm{P}}, E_{\mathrm{P}})$ as best solution
return best solution

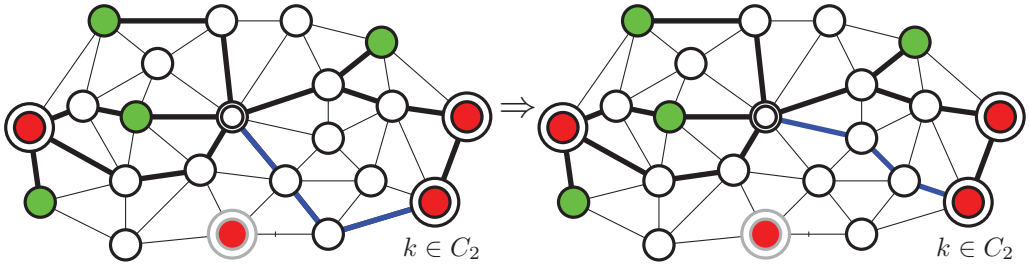


Figure 3.17: An exemplary key-path exchange between r and k.

KPEN of a candidate solution G' consists of all feasible solutions that differ from G' by at most one key-path. To ensure feasibility, after exchanging a key-path K_P, three relevant cases need to be considered. If K_P is used to connect type-1 customers only, it may simply be replaced by any other path, while if it is used in a branch line $L(E'_k)$ of a type-2 customer $k \in C_2$, the maximum length of the new path may be at most $b_{\max}(k) - \sum_{e \in (L(E'_k) \setminus E_\mathrm{P})} l_e$. Finally, if K_P is used in the first path $P(E'_k)$ of a C_2 customer k, all edges incident to "internal" nodes of its second path $Q(E'_k)$ may not be used by the new key-path to guarantee node redundancy (and vice versa for the alternate path $Q(E'_k)$). All other edges $e \in E'$ are treated as pseudo-infrastructure, i.e. $c'_e = 0$.

It is obvious that the number of key-paths $|W|$ of any solution $G' = (V', E')$ is smaller or equal to $|E'|$. Let $\overline{\mathcal{B}} = \max\{b_{\max}(k) : k \in C_2\}$ denote the maximum value of $b_{\max}(k)$, $\forall k \in C_2$, and $\overline{\mathcal{E}} = \max\{|\mathcal{E}(k)| : k \in C_2\}$ be the maximum number of edges in the $b_{\max}$-neighborhood of a type-2 customer node. Then, the worst case complexity for finding the cheapest feasible path eventually replacing a key-path $(V_\mathrm{P}, E_\mathrm{P}) \in W$ is $O(\max\{\overline{\mathcal{B}} \cdot \overline{\mathcal{E}}, |E| + |V| \log |V|\})$ and the overall complexity of searching KPEN is $O(|E'| \left(\max\{\overline{\mathcal{B}} \cdot \overline{\mathcal{E}}, |E| + |V| \log |V|\}\right))$.

However, despite its large worst case complexity, KPEN performs well in practice since number of key-paths is usually relatively small in realistic solutions.

3.9.3 Connection Remove Neighborhood

Instead of exchanging a customer's connection as in CEN, the *Connection Remove Neighborhood* (CRN) – which is given in Algorithm 3.4 – removes the connection to a single customer node $k \in C'$.

CRN of a current solution G' therefore consists of all solutions G'' where exactly one customer connected in C' is not connected anymore, i.e. $C'' \subsetneq C'$ such that

Algorithm 3.4: Connection Remove (Solution G')

$d_{\text{opt}} = 0$
forall the $k \in C'$ **do**
  $d = \sum_{e \in E'_k | M_e = \{k\}} c_e - p_k$
  if $d > d_{\text{opt}}$ **then**
    $d_{\text{opt}} = d$
    store G' without connection to k as best solution
return best solution

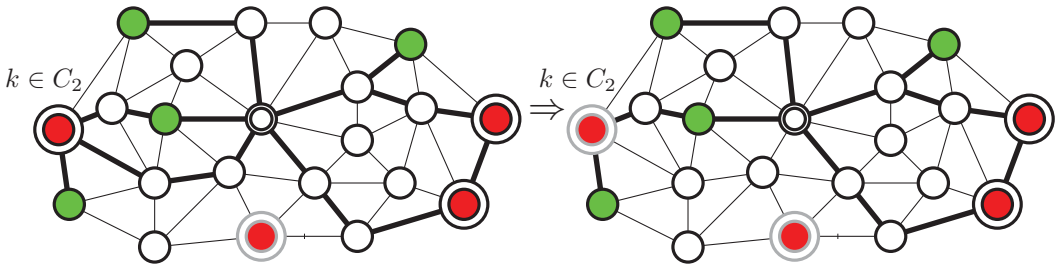


Figure 3.18: An exemplary move, removing the connection to customer k.

$|C' \setminus C''| = 1$. As a customer's connection may consist of $O(|V|)$ edges only, CRN consisting of $|C'|$ neighboring solutions can be searched in $O(|C'||V|)$ time. Figure 3.18 depicts an exemplary move, removing the connection to a type-2 customer $k \in C_2$.

3.9.4 Restricted two Connection Remove Neighborhood

CRN can be easily generalized to simultaneously remove multiple customer nodes. However, removing the connections to $l > 1$ customers at once will result in $|C'|^l$ neighboring solutions and the computational effort of searching such a neighborhood would be $O(|C'|^l|V|)$. To limit the effort, we therefore concentrate on simultaneously removing only pairs of customers $i, j \in C'$, $i \neq j$, that share at least one edge exclusively used by them, i.e. $\exists e \in E' \mid M_e = \{i, j\}$; see also Algorithm 3.5. The *Restricted two Connection Remove Neighborhood* (R2CRN) can be searched in $O(|V| \min(|E'|, |C'|^2))$; see Figure 3.19 for an exemplary move.

Algorithm 3.5: Restricted two Connection Remove Neighborhood (Solution G').

$P = \{\{u, v\} \mid \exists e \in E' : M_e = \{u, v\}\}$
$d_{\text{opt}} = 0$
forall the $\{u, v\} \in P$ **do**
  $d = \sum_{e \in E'_k | M_e \in \{\{u\},\{v\},\{u,v\}\}} c_e - p_u - p_v$
  if $d > d_{\text{opt}}$ **then**
    $d_{\text{opt}} = d$
    store G' without connection to u, v as best solution
restore best solution

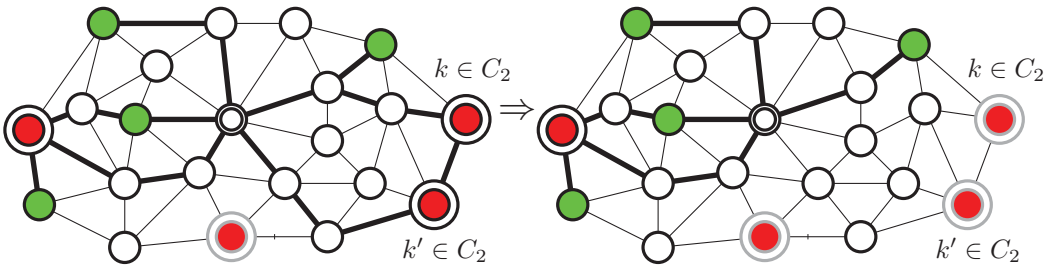


Figure 3.19: An exemplary move, removing the connections to customers k and k'.

3.10 Metaheuristics

In this section we present metaheuristic approaches utilizing the neighborhood structures explained in Section 3.9 to compute feasible solutions. After describing a construction heuristic in Section 3.10.1, we present a variable neighborhood search (VNS) with embedded variable neighborhood descent (VND) in Section 3.10.2 and – as an alternative – a GRASP/VND hybrid in Section 3.10.3.

3.10.1 Minimum Spanning Tree Augmentation Heuristic

We use a three-phase approach called *Minimum Spanning Tree Augmentation Heuristic* (MSTAH) to construct a feasible solution for a given selection of customers $C' \subseteq C$ to be connected.

Initially, a Steiner tree is computed using the minimum spanning tree (MST) heuristic from [111], see also [138]. This procedure determines a MST T_D on the *distance network*, which is the complete graph $D = (C', C' \times C')$ with node set C' and edge costs $d(u, v)$ corresponding to the costs of the cheapest paths between any $u, v \in C'$ in G. A feasible solution G'' to the Steiner tree problem is derived by further computing a MST on $G(T_D)$ which is the subgraph of G induced by all edges part of any cheapest path corresponding to an edge in T_D.

In its second phase, MSTAH augments $G'' = (V'', E'')$ by feasible connections to C_2 customers. Such connections are determined by individually calculating the cheapest feasible connection (compare Section 3.4) for all customers $k \in C_2$ in random order. All so far selected edges $e \in E''$ are considered as pseudo-infrastructure, i.e. have zero costs.

Finally, an edge minimal solution is extracted (i.e. no further edges can be deleted without violating feasibility) by greedily removing unnecessary key-paths in random order. A similar heuristic which does not consider $b_{\max}$ redundancy has been presented in [36]. Similar to MSTAH the heuristic from [36] uses the MST heuristic [111, 138] to compute a Steiner tree. As opposed to MSTAH redundancy for C_2 customers is ensured by adding a redundant route to each type-2 customer avoiding any inner node of the existing primary path using so far selected edge as pseudo-infrastructure.

Algorithm 3.6: Minimum Spanning Tree Augmentation Heuristic (Customers C')

$G'' = (V'', E'')$ = Steiner tree computed by MST heuristic
$c'_e = 0, \forall e \in E''$
$c'_e = c_e, \forall e \notin E''$
$G' = G''$
forall the $k \in C_2 \cap C'$ **do**
    f_k = cheapest connection to k using edge costs c'
    $G' = G' \cup f_k$
    update c' according to f_k
make edge minimal
determine solution structure

3.10.2 Variable Neighborhood Search

We use the general VNS scheme with VND as embedded local improvement [82] as described in Section 2.2.4. In VND, we alternate between CEN, KPEN, CRN, R2CRN in this order, with the latter two considered only in the SST variant.

Our shaking algorithm used to escape local optima modifies a solution G' by excluding a subset of its Steiner nodes as well as changing the set of connected customers C' in the SST variant: A set of $l = 1, \ldots, l_{\max} = \lfloor \frac{|C|}{2} \rfloor$ Steiner nodes $V_{\mathrm{F}} \subseteq V' \setminus C$ of the current solution G' is randomly chosen for removal. Furthermore, we select a set of l customer nodes $V_{\mathrm{C}} \subseteq C$ at random. The set of customers C'' connected in the new solution G'' is $C'' = C' \triangle V_{\mathrm{C}}$, i.e. we add those customers of V_{C} that are currently unconnected while removing the so far connected ones. Finally, we apply MSTAH using the following adapted edge costs c' with a sufficiently large value for M ($M \gg \max_{e \in E} c_e$).

$$c'_e = \begin{cases} M & \text{if } e \text{ is incident to a nodes } v \in V_{\mathrm{F}}, \\ 0 & \text{if } e \in E' \text{ and } e \text{ not incident to a node } v \in V_{\mathrm{F}}, \\ c_e & \text{else.} \end{cases}$$

Edge costs c' ensure the creation of a new solution G'' that is in general similar to G' while the Steiner nodes selected for exclusion will not be used unless there is no other option to obtain a feasible solution G''. Algorithm 3.7 summarizes the details of the VND approach.

Algorithm 3.7: VND for $b_{\max}$-SNDP.

```
l = 1
while l ≤ 4 do
    switch l do
        case 1:
            x' = CEN(x)  // see Algorithm 3.2
        case 2:
            x' = KPEN(x)  // see Algorithm 3.3
        case 3:
            x' = CRN(x)  // see Algorithm 3.4
        case 4:
            x' = R2CRN(x)  // see Algorithm 3.5
    if c(x') < c(x) then
        x = x'
        l = 1
    else
        l = l + 1
return x
```

3.10.3 Greedy Randomized Adaptive Search Procedure

As an alternative to the general VNS, we also consider a GRASP in which local search is again performed by the above mentioned VND, see Algorithm 3.7. A similar approach utilizing node- and path-based neighborhoods has been already proposed for the classical STP by Martins et al. [137]. They used a modified version of the MST heuristic [111, 138] in the construction phase. Similarly, we modify our construction heuristic MSTAH by randomizing Kruskal's algorithm for computing the MST on the distance network D. Let $d_{\max} = \max\{d(u,v) \mid \forall (u,v) \in C' \times C'\}$ and $d_{\min} = \min\{d(u,v) \mid \forall (u,v) \in C' \times C'\}$ be the maximum and minimum distances, respectively. Instead of always adding the cheapest feasible edge that connects two yet unconnected components, the randomized spanning tree construction selects the edge to be included next randomly from a restricted candidate list consisting of all feasible edges $(u,v) \in C' \times C'$ with $d(u,v) \leq d_{\min} + \alpha(d_{\max} - d_{\min})$ with $0 < \alpha \leq 1$.

3.11 Combining Lagrangian Decomposition and Variable Neighborhood Descent

As described in Section 3.8 we solve the Lagrangian dual problem of determining optimal Lagrangian multipliers λ^* by the volume algorithm. In each iteration we need to determine optimal x_e variables as well as f_e^k variables for the current set of Lagrangian multipliers λ. The latter are computed by calculating individual cheapest connections for each customer $k \in C$ and eventually choosing to connect k in case it pays off. Obviously, the graph $G' = (V', E')$ induced by the set of edges $E' = \{e \in E \mid \exists k \text{ s.t. } f_e^k = 1\}$ is a primal feasible solution. This offers multiple ways of hybridizing the Lagrangian decomposition approach with metaheuristics in order to obtain better primal solutions and reduce the gap between lower and upper bounds.

Here, we pursue two alternatives: Either we immediately try to improve promising solutions gained by the iterations of the volume algorithm, or we store the N best solutions obtained by the volume algorithm and try to improve them after termination of the volume algorithm. In both cases, we use VND with CEN, KPEN, CRN, and R2CRN in this order to generate a local optimum for a given candidate solution (CRN and R2CRN are again only considered in the SST variant). According to the classification of hybrid metaheuristics given in [156] the latter approach is a sequential hybridization with respect to the order of execution, while the former falls into the category of interleaved hybridization, see also Section 2.3.

As the time for performing VND on a candidate solution is not negligible, it is critical to apply it wisely on a well-chosen subset of candidate solutions only. In the interleaved approach, we found the following self-adaptive strategy with the exogenous parameters δ, γ, and $\beta_{\max}$ to work well. Let G' and G'_{best} be the current and so far best solutions obtained by the volume algorithm, respectively. VND is applied to G' iff $o(G') \leq (1+\beta)o(G'_{\text{best}})$. Preliminary tests indicated that a good value for β is not easy to find as it depends on the problem instance, and so we automatically adapt it after every δ iterations as follows. Let z be the ratio of how often VND has been applied during the last δ iterations of the volume algorithm. If $z < \gamma$ we set $\beta = \min(2\beta, \beta_{\max})$ while $\beta = \max(\beta/2, \beta_{\min})$ if $z > \gamma$. Concrete values for all parameters will be given in Section 3.13.

3.12 Test Instances and Environment

Real world instances from a German city [8] have been used to test our approaches. Table 3.1 details their individual characteristics, while Table 3.2 highlights the number of potential branching nodes and edges for all used values of $b_{\max}$. Note that in each experiment the value of $b_{\max}$ is identical for each type-2 customer node and we write $b_{\max} = 30$ instead of $b_{\max}(k) = 30$, $\forall k \in C_2$, in the following.

Table 3.1: Instance set characteristics.

Set	#	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert C\rvert$	$\overline{\lvert C\rvert}$	$\lvert C_1\rvert$	$\overline{\lvert C_1\rvert}$	$\lvert C_2\rvert$	$\overline{\lvert C_2\rvert}$
ClgSE-I1	25	190	377	5-8	5.9	3-5	3.8	2-3	2.1
ClgSE-I2	15	190	377	11-17	13.8	7-12	8.9	4-7	4.9
ClgSE-I3	15	190	377	8-12	9.6	5-8	6.0	3-6	3.6
ClgN1B-I1	20	2804	3082	11-14	11.8	8-11	8.5	3-4	3.3
ClgN1B-I2	19	2804	3082	7-11	9.0	3-6	4.1	4-6	5.0
ClgME-I1	25	1757	3877	6-10	7.2	4-7	5.0	2-3	2.3
ClgME-I2	15	1523	3290	11-14	12.2	8-11	8.7	3-4	3.5

All computational experiments have been performed on a single core of an Intel Xeon E5540 with 2.53GHz. IBM CPLEX 12.1 [98] has been used to solve the multi-commodity flow model (MCF) from Wagner et al. [178] as well as its linear relaxation $(\text{MCF})^{\text{LP}}$. We used SCIP 1.2.0 [2, 182] with IBM CPLEX 12.1 [98] as embedded LP solver for solving (Col) and (dCol) as well as their linear relaxations $(\text{Col})^{\text{LP}}$ and $(\text{dCol})^{\text{LP}}$, respectively.

Note that CPLEX 12.1 is approximately three times faster than SCIP 1.2.0 with

Table 3.2: Average number of potential branching nodes and edges per instance set for $b_{\max}(k) \in \{30, 50, 100\}$, $\forall k \in C_2$.

	$b_{\max} = 30$		$b_{\max} = 50$		$b_{\max} = 100$	
Set	$\overline{\mathcal{B}(k)}$	$\overline{\mathcal{E}(k)}$	$\overline{\mathcal{B}(k)}$	$\overline{\mathcal{E}(k)}$	$\overline{\mathcal{B}(k)}$	$\overline{\mathcal{E}(k)}$
ClgSE-I1	12.93	17.07	26.23	41.43	86.54	144.30
ClgSE-I2	10.23	12.58	29.47	42.33	131.01	235.36
ClgSE-I3	10.62	13.29	32.84	47.51	119.54	210.67
ClgN1B-I1	10.47	9.82	23.20	24.10	68.89	77.59
ClgN1B-I2	7.92	6.94	15.04	14.10	40.53	40.24
ClgME-I1	17.70	28.10	39.08	70.11	120.93	243.47
ClgME-I2	12.29	18.64	24.96	42.54	63.93	120.93

CPLEX 12.1 as an embedded LP solver[3]. An absolute time limit of 7200 CPU-seconds has been used for all experiments.

3.13 Computational Results

In the following we summarize all obtained computational results. First, a detailed comparison of the proposed exact models (Col), (dCol), and the MCF formulation of Wagner et al. [178] is given. Unfortunately, the undirected cut formulation from the same authors [177] does not work properly when using CPLEX 12.1 and when compiled with activated optimization flags on a 64 bit. Hence, a fair comparison to the cut formulation is not possible and it is not considered in the following.

Afterwards, we compare the results of the Lagrangian decomposition approach from Section 3.8 to its two hybrid variants as described in Section 3.11. A comparison of the VND, VNS and GRASP approaches from Section 3.10 is given before we finally compare the upper bounds obtained by each group's individual best method.

Since preliminary tests indicated identical trends for the OPT variant, we primarily concentrate on the more complex SST variant of $b_{\max}$-SNDP in the following. Furthermore, to analyze the influence of the size of the $b_{\max}$-neighborhood, we consider different values for $b_{\max}$ for each instance set.

[3] http://scip.zib.de

3.13.1 Results on Exact Models

When solving (Col) we initialize $\tilde{F}$ by all variables corresponding to connections obtained by applying MSTAH plus connections obtained from a single run of VND. For (dCol) we pursue the same strategy, but additionally need to orient each of the obtained connections. Using the method described by Chimani et al. [36] we initially orient the solutions obtained by MSTAH and VND, respectively, and afterwards adopt the oriented connections obtained in this way.

Solving (Col) and (dCol) has been further configured as follows. For (Col) we add the cheapest connection to each customer $k \in C$ to the restricted master problem in each pricing iterations if it has negative reduced costs.

Unfortunately, preliminary tests showed that solving the pricing subproblem for (dCol) by algorithms for the elementary shortest path problem with resource constraints – as discussed in Section 3.6.2 – is too time consuming already for relatively small instances. Too many labels need to be considered for each node and thus, this approach turned out to perform much worse than the MIP based approach discussed in Section 3.6.1. Hence, we do only consider the MIP based approach in the following. To speed-up the pricing for type-2 customers, we return the first found solution that prices out favorably instead of trying to find a proven optimal solution in each execution. Thus, as for (Col) we add at most one connection for every customer in each pricing iteration.

As opposed to our problem definition in Section 3.2, we allow for the root node r being a potential branching node of some type-2 customer $k \in C_2$. Otherwise, we would restrict ourself to a too small set of feasible values of $b_{\max}$. Since the MIP for solving the pricing subproblem of (dCol) does not allow this case, we additionally apply a directed variant of the length constrained shortest path algorithm in this case; compare Section 3.4.

Linear Programming Relaxations

Table 3.3 depicts the average improvement and corresponding standard deviations in percent of the LP relaxation values of (Col) and (dCol) over (MCF). Furthermore, these values are additionally given for (dCol) compared to (Col).

The results from Table 3.3 confirm the results of our theoretical comparison from Section 3.7. While the LP relaxation values of (MCF) and (Col) are – for the considered instances – equal for the OPT variant without considering $b_{\max}$-redundancy

Table 3.3: Relative LP relaxation values and corresponding standard deviations in % for (MCF), (Col), and (dCol).

Variant	Set	$\frac{(\text{Col})^{\text{LP}}-(\text{MCF})^{\text{LP}}}{(\text{MCF})^{\text{LP}}}$ [%]		$\frac{(\text{dCol})^{\text{LP}}-(\text{MCF})^{\text{LP}}}{(\text{MCF})^{\text{LP}}}$ [%]		$\frac{(\text{dCol})^{\text{LP}}-(\text{Col})^{\text{LP}}}{(\text{Col})^{\text{LP}}}$ [%]	
OPT, $b_{\max}=0$	ClgSE-I1	0.00	(0.00)	1.63	(2.38)	1.63	(2.38)
	ClgSE-I2	0.00	(0.00)	8.84	(4.08)	8.84	(4.08)
	ClgSE-I3	0.00	(0.00)	5.53	(4.55)	5.53	(4.55)
	ClgN1B-I1	0.00	(0.00)	2.78	(2.32)	2.78	(2.32)
	ClgN1B-I2	0.00	(0.00)	0.95	(0.89)	0.95	(0.89)
SST, $b_{\max}=0$	ClgSE-I1	0.05	(0.23)	1.68	(2.35)	1.63	(2.38)
	ClgSE-I2	0.14	(0.55)	9.30	(5.07)	9.13	(4.65)
	ClgSE-I3	0.88	(2.47)	8.02	(5.31)	7.09	(4.84)
	ClgN1B-I1	3.07	(6.67)	5.29	(7.1)	2.58	(2.29)
	ClgN1B-I2	2.12	(5.05)	3.09	(4.7)	1.36	(1.47)
SST, $b_{\max}=30$	ClgSE-I1	7.06	(5.07)	8.81	(5.65)	1.75	(2.36)
	ClgSE-I2	5.66	(2.63)	19.39	(5.95)	12.99	(4.66)
	ClgSE-I3	4.80	(2.89)	12.2	(5.07)	7.07	(4.01)
	ClgN1B-I1	5.88	(7.08)	9.07	(7.68)	2.72	(1.42)
	ClgN1B-I2	4.03	(5.52)	5.76	(5.44)	1.58	(1.89)
SST, $b_{\max}=50$	ClgSE-I1	9.61	(8.98)	11.88	(10.14)	2.17	(3.05)
	ClgSE-I2	5.85	(3.48)	24.17	(6.81)	17.32	(5.31)
	ClgSE-I3	6.53	(3.78)	13.04	(6.42)	6.08	(3.95)
	ClgN1B-I1	2.45	(3.16)	5.53	(3.35)	2.97	(2.00)
	ClgN1B-I2	4.21	(6.16)	5.72	(6.36)	1.65	(1.91)
SST, $b_{\max}=100$	ClgSE-I1	8.10	(11.94)	10.77	(13.70)	2.07	(2.58)
	ClgSE-I2	3.39	(2.57)	23.24	(7.27)	19.14	(4.88)
	ClgSE-I3	2.75	(2.56)	13.10	(8.16)	10.29	(6.57)
	ClgN1B-I1	2.37	(3.95)	6.07	(4.95)	3.55	(2.28)
	ClgN1B-I2	1.06	(2.03)	1.87	(1.01)	1.49	(1.18)

– i.e. $b_{\max}(k)=0$, $\forall k \in C_2$ – the values obtained from solving $(\text{Col})^{\text{LP}}$ are significantly better for all other configurations and instance sets. Furthermore, the LP relaxation values of (dCol) clearly dominate those of $(\text{Col})^{\text{LP}}$.

Tables 3.4 and 3.5 analyze the efficiency of the various approaches for using alternative dual-optimal solutions in the pricing subproblems of (Col) as proposed in Section 3.5.2. As previously described, D^* simply uses the obtained dual variable values without any modification, while D' equally splits the potential increase for each edge over all $|C|$ subproblems. $D^{(k,d)}$ refers to the fine-grained variant controlled by parameter d, while $D^{(k,d')}$ is the compromise between $D^{(k,d)}$ and D' where d is never decreased. Finally, $D^{(\text{p})}$ which is valid for the SST variant only, denotes the strategy considering each customers prize.

From Table 3.4, we conclude that all variants are able to solve the linear relaxations

Table 3.4: Median CPU-times for solving the LP relaxation of (MCF) and the various variants of (Col). Best values are marked bold.

Variant	Set	$(\text{MCF})^{\text{LP}}$	$(\text{Col})^{\text{LP}}$				
			D^*	D'	$D^{(k,d')}$	$D^{(k,d)}$	$D^{(\text{p})}$
OPT, $b_{\max} = 0$	ClgSE-I1	**0.09**	0.55	0.16	0.11	0.13	-
	ClgSE-I2	**0.34**	5.26	2.83	2.10	1.03	-
	ClgSE-I3	**0.20**	3.30	0.40	0.34	0.40	-
	ClgN1B-I1	43.55	94.48	21.57	**11.98**	13.48	-
	ClgN1B-I2	58.27	203.53	41.52	14.05	**12.19**	-
SST, $b_{\max} = 0$	ClgSE-I1	**0.10**	0.58	0.20	0.12	0.13	0.24
	ClgSE-I2	**0.35**	5.99	4.15	1.12	1.11	2.79
	ClgSE-I3	**0.19**	1.15	0.41	0.22	0.36	0.40
	ClgN1B-I1	42.82	116.25	19.26	**10.90**	16.04	25.32
	ClgN1B-I2	79.55	137.68	66.10	**13.32**	15.24	51.76
SST, $b_{\max} = 30$	ClgSE-I1	**0.15**	0.86	0.51	0.30	0.38	0.45
	ClgSE-I2	**0.86**	6.45	4.34	2.62	2.38	3.79
	ClgSE-I3	**0.33**	2.48	1.00	0.58	1.03	1.11
	ClgN1B-I1	190.48	124.61	32.63	**20.85**	31.28	41.04
	ClgN1B-I2	1070.66	291.45	76.64	**30.09**	34.72	93.97
SST, $b_{\max} = 50$	ClgSE-I1	**0.18**	1.11	0.49	0.40	0.53	0.39
	ClgSE-I2	**0.82**	6.26	4.20	3.13	4.31	4.18
	ClgSE-I3	**0.41**	3.60	1.28	1.11	2.01	1.42
	ClgN1B-I1	212.07	220.80	39.01	**24.70**	54.66	39.99
	ClgN1B-I2	1144.86	391.44	103.83	**40.02**	55.76	136.04
SST, $b_{\max} = 100$	ClgSE-I1	**0.15**	3.04	0.95	0.74	1.28	1.21
	ClgSE-I2	**0.58**	23.80	11.29	8.63	15.78	10.80
	ClgSE-I3	**0.37**	9.40	2.97	1.97	4.94	3.48
	ClgN1B-I1	214.67	540.45	98.94	**59.61**	125.14	105.93
	ClgN1B-I2	1281.95	652.77	296.17	**78.53**	104.47	338.19

of the smaller ClgS instances quite efficiently. On the one hand, $(\text{MCF})^{\text{LP}}$ usually can be solved slightly faster then $(\text{Col})^{\text{LP}}$ for these instances. On the other hand the obtained bounds due to $(\text{Col})^{\text{LP}}$ are better than those of $(\text{MCF})^{\text{LP}}$. For larger instances, $(\text{Col})^{\text{LP}}$ can be additionally solved more efficient than $(\text{MCF})^{\text{LP}}$, especially when using alternative dual-optimal solutions according to D', $D^{(k,d')}$, $D^{(k,d)}$, or $D^{(\text{p})}$. Among these, $D^{(k,d')}$ performs better than the other three.

Furthermore, we conclude that considering $b_{\max}$-redundancy yields an enormous increase in terms of necessary CPU-time for $(\text{MCF})^{\text{LP}}$, while the overhead of $(\text{Col})^{\text{LP}}$ is only moderate.

Table 3.5 compares the relative number of needed pricing iterations to solve $(\text{Col})^{\text{LP}}$, i.e. the relative number of times the restricted master problem needs to be solved, using D' as a basis. In consistency with the median CPU-times from Table 3.4,

Table 3.5: Absolute and average relative number of pricing iterations and corresponding standard deviations for solving the LP relaxation of (Col) with various variants of alternative dual-optimal solutions. Best values are marked bold.

		# of pricing iterations									
Variant	Set	D'		$\frac{D^*}{D'}$		$\frac{D^{(k,d')}}{D'}$		$\frac{D^{(k,d)}}{D'}$		$\frac{D^{(p)}}{D'}$	
OPT, $b_{\max} = 0$	ClgSE-I1	85.76	(77.39)	3.45	(1.53)	**0.99**	(0.35)	1.02	(0.57)	-	(-)
	ClgSE-I2	307.87	(230.09)	1.97	(1.02)	0.99	(0.34)	**0.69**	(0.41)	-	(-)
	ClgSE-I3	232.8	(333.95)	3.73	(1.74)	1.01	(0.36)	**0.85**	(0.52)	-	(-)
	ClgN1B-I1	381.65	(472.32)	5.83	(2.97)	0.85	(0.56)	**0.76**	(0.46)	-	(-)
	ClgN1B-I2	250.16	(212.13)	4.56	(2.25)	**0.81**	(0.38)	**0.81**	(0.61)	-	(-)
SST, $b_{\max} = 0$	ClgSE-I1	101.24	(78.41)	2.31	(1.02)	**0.79**	(0.31)	0.84	(0.46)	1.25	(0.53)
	ClgSE-I2	327.53	(227.09)	1.63	(0.83)	**0.61**	(0.24)	0.62	(0.38)	1.13	(0.29)
	ClgSE-I3	280.93	(412.87)	3.12	(1.30)	**0.77**	(0.36)	0.81	(0.46)	0.98	(0.29)
	ClgN1B-I1	332.4	(397.04)	7.61	(5.33)	**0.73**	(0.32)	0.80	(0.45)	1.49	(0.98)
	ClgN1B-I2	254.37	(174.53)	3.87	(1.87)	**0.64**	(0.37)	0.67	(0.29)	1.08	(0.35)
SST, $b_{\max} = 30$	ClgSE-I1	91.88	(84.91)	2.34	(0.70)	**0.81**	(0.34)	0.93	(0.37)	1.08	(0.3)
	ClgSE-I2	266.53	(133.01)	1.46	(0.37)	0.78	(0.32)	**0.65**	(0.21)	1.00	(0.32)
	ClgSE-I3	137.27	(177.27)	3.20	(1.43)	**0.83**	(0.33)	0.98	(0.46)	1.29	(0.42)
	ClgN1B-I1	622.65	(1015.9)	8.34	(9.49)	**0.66**	(0.44)	0.77	(0.48)	1.26	(0.53)
	ClgN1B-I2	232.79	(115.87)	4.68	(1.96)	**0.62**	(0.24)	0.66	(0.23)	1.15	(0.31)
SST, $b_{\max} = 50$	ClgSE-I1	67.56	(58.34)	2.62	(1.09)	**0.89**	(0.31)	1.02	(0.39)	0.99	(0.27)
	ClgSE-I2	191.6	(96.62)	1.50	(0.55)	0.81	(0.34)	**0.79**	(0.26)	0.96	(0.22)
	ClgSE-I3	81.8	(62.1)	3.77	(1.70)	**0.92**	(0.32)	1.15	(0.41)	1.21	(0.43)
	ClgN1B-I1	361.8	(503.84)	5.82	(4.53)	**0.70**	(0.45)	0.94	(0.69)	1.15	(0.71)
	ClgN1B-I2	239.26	(113.94)	4.43	(2.59)	**0.59**	(0.19)	0.62	(0.23)	1.24	(0.52)
SST, $b_{\max} = 100$	ClgSE-I1	49.04	(21.06)	2.95	(1.45)	**0.99**	(0.31)	1.23	(0.32)	1.16	(0.35)
	ClgSE-I2	119.20	(49.23)	2.24	(0.92)	**0.99**	(0.27)	1.16	(0.33)	1.04	(0.27)
	ClgSE-I3	72.53	(45.00)	3.96	(1.54)	**0.88**	(0.27)	1.24	(0.54)	1.24	(0.45)
	ClgN1B-I1	546.05	(853.58)	6.61	(6.63)	**0.75**	(0.46)	0.92	(0.61)	1.06	(0.45)
	ClgN1B-I2	280.58	(103.76)	3.61	(1.58)	0.55	(0.18)	**0.53**	(0.19)	1.10	(0.35)

we conclude that using D', $D^{(k,d')}$, $D^{(k,d)}$, or $D^{(\mathrm{p})}$ significantly reduces the number of needed pricing iterations. As for the CPU-times, slight advantages of $D^{(k,d')}$ over the other approaches can be observed. Note that already using D' instead of simply using the standard dual-optimal variable values – i.e. using D^* – yields a major improvement. We conclude that D', $D^{(k,d')}$, $D^{(k,d)}$, and $D^{(\mathrm{p})}$ are able to find meaningful connections already in the beginning of the column generation process and thus allow for efficiently solving the linear relaxation of (Col).

Tables 3.6 and 3.7 analyze the efficiency of the various approaches using alternative dual-optimal solutions for the directed connection formulation (dCol). As described in Section 3.6.4, the interpretations of D', $D^{(k,d')}$, $D^{(k,d)}$, and $D^{(\mathrm{p})}$ correlates to the undirected case, although some calculations are slightly different.

Table 3.6: Median CPU-times for solving the LP relaxation of (MCF) and the diverse variants of (dCol). Best values are marked bold.

			$(\text{dCol})^{\text{LP}}$				
Variant	Set	$(\text{MCF})^{\text{LP}}$	D^*	D'	$D^{(k,d')}$	$D^{(k,d)}$	$D^{(\text{p})}$
OPT, $b_{\max} = 0$	ClgSE-I1	**0.09**	28.44	5.66	5.74	6.36	-
	ClgSE-I2	**0.34**	92.30	50.30	57.87	62.16	-
	ClgSE-I3	**0.20**	70.45	9.79	8.64	19.43	-
	ClgN1B-I1	**43.55**	7200.00	3677.30	1805.73	2838.3	-
	ClgN1B-I2	**58.27**	7200.00	7200.00	7200.00	7200.00	-
SST, $b_{\max} = 0$	ClgSE-I1	**0.10**	23.31	6.36	4.12	7.10	7.46
	ClgSE-I2	**0.35**	114.82	125.52	61.06	63.80	82.06
	ClgSE-I3	**0.19**	61.93	8.51	7.95	25.98	9.37
	ClgN1B-I1	**42.82**	7200.00	1342.96	800.41	2795.7	4410.18
	ClgN1B-I2	**79.55**	7200.00	6968.09	2884.42	6499.98	7200.00
SST, $b_{\max} = 30$	ClgSE-I1	**0.15**	49.61	10.81	7.11	14.03	11.49
	ClgSE-I2	**0.86**	174.55	69.01	52.68	95.22	59.10
	ClgSE-I3	**0.33**	111.91	27.69	13.05	35.06	28.10
	ClgN1B-I1	**190.48**	7200.00	1457.13	791.07	3715.07	3055.81
	ClgN1B-I2	**1070.66**	7200.00	6821.66	3331.36	7200.00	7200.00
SST, $b_{\max} = 50$	ClgSE-I1	**0.18**	38.77	8.90	7.73	16.11	9.19
	ClgSE-I2	**0.82**	179.41	39.57	36.78	113.35	83.53
	ClgSE-I3	**0.41**	98.76	12.39	11.53	35.63	13.71
	ClgN1B-I1	**212.07**	7200.00	1171.84	842.75	4493.58	1568.81
	ClgN1B-I2	**1144.86**	7200.00	7200.00	4782.36	6739.93	7200.00
SST, $b_{\max} = 100$	ClgSE-I1	**0.15**	50.03	4.72	4.48	17.64	6.17
	ClgSE-I2	**0.58**	950.35	36.38	41.88	117.71	23.65
	ClgSE-I3	**0.37**	589.65	10.66	18.40	81.27	10.90
	ClgN1B-I1	**214.67**	7200.00	802.08	726.16	2841.21	1132.12
	ClgN1B-I2	**1281.95**	7200.00	7200.00	4463.8	7200.00	7200.00

As expected the CPU-time overhead for solving $(\text{dCol})^{\text{LP}}$ due the $\mathcal{NP}$-hard pricing subproblems for type-2 customers $k \in C_2$ is not negligible. However, similar to the previous discussion for (Col) we can observe that D', $D^{(k,d')}$, $D^{(k,d)}$, and $D^{(\text{p})}$ significantly speed-up the solution of $(\text{dCol})^{\text{LP}}$. Furthermore, the relative additional effort for solving $(\text{dCol})^{\text{LP}}$ compared to $(\text{MCF})^{\text{LP}}$ decreases when considering larger instances and $b_{\max}$-redundancy, i.e. if $b_{\max}(k) \neq 0$, $\forall k \in C_2$.

Since the LP relaxation values of (dCol) are much tighter than those of the other models, (dCol) might nevertheless outperform them due to a significantly smaller number of nodes that need to be considered in the branch-and-bound tree.

Table 3.7 details the relative number of pricing iterations needed to solve $(\text{dCol})^{\text{LP}}$ for $D^{(k,d')}$, $D^{(k,d)}$, and $D^{(\text{p})}$ in comparison to D'. Here, only those instances are considered where $(\text{dCol})^{\text{LP}}$ could be solved within the given time limit of 7200 CPU-

Table 3.7: Absolute and average relative number of pricing iterations and corresponding standard deviations for solving the LP relaxation of (dCol) with various variants of alternative dual-optimal solutions. Best values are marked bold.

		# of pricing iterations							
Variant	Set	D'		$\frac{D^{(k,d')}}{D'}$		$\frac{D^{(k,d)}}{D'}$		$\frac{D^{(\mathrm{p})}}{D'}$	
OPT, $b_{\max} = 0$	ClgSE-I1	135.13	(107.25)	0.91	(0.32)	**0.86**	(0.46)	-	(-)
	ClgSE-I2	417.6	(287.62)	1.04	(0.42)	**0.69**	(0.38)	-	(-)
	ClgSE-I3	122.07	(72.18)	**0.83**	(0.15)	1.06	(0.35)	-	(-)
	ClgN1B-I1	160.69	(69.88)	**0.88**	(0.22)	1.59	(0.51)	-	(-)
	ClgN1B-I2	185.89	(88.07)	**1.16**	(0.38)	1.25	(0.49)	-	(-)
SST, $b_{\max} = 0$	ClgSE-I1	126.64	(85.87)	0.89	(0.37)	**0.87**	(0.49)	1.05	(0.30)
	ClgSE-I2	496.57	(283.48)	0.73	(0.39)	**0.56**	(0.36)	1.08	(0.54)
	ClgSE-I3	137.93	(104.35)	**0.94**	(0.19)	1.04	(0.46)	1.25	(0.61)
	ClgN1B-I1	153.21	(43.33)	**0.84**	(0.14)	1.37	(0.28)	1.48	(0.38)
	ClgN1B-I2	258.5	(117.52)	**0.67**	(0.35)	1.27	(0.92)	1.39	(0.47)
SST, $b_{\max} = 30$	ClgSE-I1	205.68	(286.75)	**0.77**	(0.3)	0.83	(0.43)	1.06	(0.44)
	ClgSE-I2	423.67	(564.89)	0.80	(0.34)	**0.73**	(0.27)	1.13	(0.44)
	ClgSE-I3	159.40	(103.11)	**0.77**	(0.36)	0.86	(0.45)	1.22	(0.34)
	ClgN1B-I1	153.60	(91.40)	**0.96**	(0.24)	1.67	(0.55)	1.34	(0.44)
	ClgN1B-I2	267.20	(91.16)	**0.54**	(0.25)	0.83	(0.39)	0.66	(0.28)
SST, $b_{\max} = 50$	ClgSE-I1	88.72	(54.75)	**0.90**	(0.39)	1.00	(0.37)	1.06	(0.29)
	ClgSE-I2	264.00	(226.09)	**0.89**	(0.54)	0.96	(0.64)	1.26	(0.45)
	ClgSE-I3	104.20	(58.63)	**0.82**	(0.28)	0.92	(0.38)	0.99	(0.56)
	ClgN1B-I1	145.89	(71.02)	**0.82**	(0.22)	1.43	(0.48)	1.15	(0.53)
	ClgN1B-I2	238.00	(100.27)	**0.66**	(0.20)	0.90	(0.39)	1.12	(0.25)
SST, $b_{\max} = 100$	ClgSE-I1	50.83	(30.24)	**0.96**	(0.29)	1.16	(0.39)	1.19	(0.59)
	ClgSE-I2	91.20	(43.69)	**1.08**	(0.38)	1.33	(0.44)	**1.08**	(0.48)
	ClgSE-I3	113.43	(185.62)	1.46	(1.19)	**1.29**	(0.62)	1.36	(0.59)
	ClgN1B-I1	112.44	(51.13)	**0.94**	(0.27)	1.72	(0.47)	1.29	(0.45)
	ClgN1B-I2	232.00	(107.13)	**0.68**	(0.25)	0.97	(0.37)	0.81	(0.25)

seconds when using D'. We do not report on D^*, since it could solve $(\mathrm{dCol})^{\mathrm{LP}}$ for very few instances only. As for the undirected model, we conclude that the advanced adaptation strategies often significantly reduce the number of needed pricing iterations, and $D^{(k,d')}$ is the best option for solving $(\mathrm{dCol})^{\mathrm{LP}}$ too.

Solutions and Optimality Gaps

In the following, computational results for solving (Col) and (dCol) by branch-and-price are presented. Branching is performed on edge variables for (Col) and on arc variables for (dCol), respectively. We do not use any problem specific branching

rules, but trust on the branching decisions as performed by SCIP.

All results for (Col) and (dCol) have been computed using $D^{(k,d')}$ for adapting dual variable values, which has been shown to outperform the other variants. To allow for a meaningful comparison, we only report on those instances where the LP relaxation of (dCol) could be solved within the given time limit of 7200 CPU-seconds when using $D^{(k,d')}$. The corresponding number of considered instances is additionally stated in each table.

Table 3.8 shows average gaps as well as corresponding standard deviations in percent for each considered instance set and setting. We conclude that (dCol) could be solved to proven optimality whenever its linear relaxation was solved. The undirected connection formulation (Col), however, failed to find a proven optimal solution within two hours for some instances and performs slightly worse than the multi-commodity flow formulation of Wagner et al. [178] with respect to this criterion. Although the LP relaxation values of (Col) are better than those of model (MCF) and the root relaxation gaps are already quite small, a too large number of nodes needs to be considered in the branch-and-bound tree for further improving the obtained lower bound in order to proof optimality of a solution.

Table 3.9 reports median CPU-times for solving (MCF), (Col), and (dCol), respectively, for those instances where $(\text{dCol})^{\text{LP}}$ could be solved within the given time limit using alternative dual-optimal solutions according to $D^{(k,d')}$. We conclude that the performance of both connection based formulations improves compared to the MCF formulation when considering $b_{\max}$-redundancy. When taking into account that SCIP 1.2.0 with CPLEX 12.1 is slower than CPLEX 12.1 alone approximately by a factor of three[4], (dCol) can be considered as the most effective method on larger instances when $b_{\max}(k) \neq 0$, $\forall k \in C_2$. The undirected formulation (Col) is, however, typically the fastest approach for those larger instances where it needs to consider only a reasonable number of nodes in the branch-and-bound tree, i.e. on the set ClgN1B-I2.

Since, we observed from Table 3.8 that (dCol) could be solved to proven optimality whenever its linear relaxation $(\text{dCol})^{\text{LP}}$ was solved, we further analyzed for how many instances the solution to its linear relaxation is integral, i.e. is an optimal solution to the corresponding instance. As detailed in Table 3.10, solving $(\text{dCol})^{\text{LP}}$ yields a proven optimal solution to (dCol) for almost all considered instances and settings. On the contrary, most solutions of $(\text{MCF})^{\text{LP}}$ and $(\text{Col})^{\text{LP}}$ include fractional variables.

[4] http://scip.zib.de

Table 3.8: Average optimality gaps and corresponding standard deviations after 7200 CPU-seconds for instances where $(\text{dCol})^{\text{LP}}$ could be solved when using $D^{(k,d')}$. Best values are marked bold.

Variant	Set	#	(MCF)		(Col)		(dCol)	
OPT, $b_{\max} = 0$	ClgSE-I1	25	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I2	14	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I3	15	**0.00**	(0.00)	0.10	(0.39)	**0.00**	(0.00)
	ClgN1B-I1	13	**0.00**	(0.00)	0.99	(0.94)	**0.00**	(0.00)
	ClgN1B-I2	9	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
SST, $b_{\max} = 0$	ClgSE-I1	25	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I2	15	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I3	15	**0.00**	(0.00)	0.08	(0.31)	**0.00**	(0.00)
	ClgN1B-I1	16	0.02	(0.06)	0.95	(0.96)	**0.00**	(0.00)
	ClgN1B-I2	17	**0.00**	(0.00)	0.09	(0.25)	**0.00**	(0.00)
SST, $b_{\max} = 30$	ClgSE-I1	25	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I2	15	**0.00**	(0.00)	0.31	(1.21)	**0.00**	(0.00)
	ClgSE-I3	15	**0.00**	(0.00)	0.12	(0.46)	**0.00**	(0.00)
	ClgN1B-I1	15	0.32	(0.68)	1.11	(1.20)	**0.00**	(0.00)
	ClgN1B-I2	17	0.41	(1.10)	0.14	(0.32)	**0.00**	(0.00)
SST, $b_{\max} = 50$	ClgSE-I1	25	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I2	15	**0.00**	(0.00)	0.64	(1.50)	**0.00**	(0.00)
	ClgSE-I3	15	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgN1B-I1	18	0.20	(0.58)	1.41	(1.66)	**0.00**	(0.00)
	ClgN1B-I2	13	0.26	(0.95)	0.16	(0.38)	**0.00**	(0.00)
SST, $b_{\max} = 100$	ClgSE-I1	24	**0.00**	(0.00)	**0.00**	(0.00)	**0.00**	(0.00)
	ClgSE-I2	15	**0.00**	(0.00)	0.76	(1.61)	**0.00**	(0.00)
	ClgSE-I3	14	**0.00**	(0.00)	0.13	(0.40)	**0.00**	(0.00)
	ClgN1B-I1	18	1.14	(1.48)	1.78	(2.05)	**0.00**	(0.00)
	ClgN1B-I2	14	0.14	(0.54)	0.17	(0.45)	**0.00**	(0.00)

Overall, we conclude that both connection based formulations have their individual advantages. While the LP relaxation of (Col) is tighter than the one of (MCF) and can be solved efficiently, sometimes a too large number of nodes in the branch-and-bound tree needs to be considered. Thus (Col) sometimes fails to prove optimality of a solution within reasonable time. The resulting gaps are, however, relatively tight already after solving the root node.

With respect to model (dCol), we conclude that its LP relaxation is extremely tight and in particular turned out to be integral for almost all used test instances and settings. While the computational effort for solving it is not negligible, it nevertheless outperforms the other methods on medium sized instances.

Both models perform bad, when simply using the dual variable values obtained by the used LP solver. Above computational results clearly show that the usage of alter-

Table 3.9: Median CPU-times for instances where $(\text{dCol})^{\text{LP}}$ could be solved when using $D^{(k,d')}$. Best values are marked bold.

Variant	Set	#	(MCF)	(Col)	(dCol)
OPT, $b_{\max} = 0$	ClgSE-I1	25	**0.25**	0.28	5.74
	ClgSE-I2	14	**3.92**	177.88	54.46
	ClgSE-I3	15	**0.87**	2.25	8.64
	ClgN1B-I1	13	**643.28**	7200.01	1109.79
	ClgN1B-I2	9	**136.03**	241.72	3191.45
SST, $b_{\max} = 0$	ClgSE-I1	25	**0.30**	0.32	4.12
	ClgSE-I2	15	**3.63**	81.82	61.06
	ClgSE-I3	15	**0.76**	2.25	7.95
	ClgN1B-I1	16	**519.06**	7200.00	728.73
	ClgN1B-I2	17	237.52	**211.20**	2500.98
SST, $b_{\max} = 30$	ClgSE-I1	25	1.43	**0.48**	7.11
	ClgSE-I2	15	**23.87**	329.06	52.68
	ClgSE-I3	15	**2.06**	4.41	13.05
	ClgN1B-I1	15	1524.24	7200.00	**752.98**
	ClgN1B-I2	17	2322.39	**261.72**	3185.55
SST, $b_{\max} = 50$	ClgSE-I1	25	1.37	**0.70**	7.73
	ClgSE-I2	15	191.69	585.37	**36.78**
	ClgSE-I3	15	**2.78**	6.18	11.53
	ClgN1B-I1	18	2788.23	7200.00	**818.78**
	ClgN1B-I2	13	2210.25	**339.64**	3151.63
SST, $b_{\max} = 100$	ClgSE-I1	24	1.70	**1.65**	4.34
	ClgSE-I2	15	46.93	4000.79	**41.88**
	ClgSE-I3	14	**4.00**	19.41	16.25
	ClgN1B-I1	18	7156.75	7200.00	**672.97**
	ClgN1B-I2	14	2419.23	**557.36**	2445.72

native dual-optimal solutions as described in Sections 3.5.2 and 3.6.4, respectively, significantly reduces the time necessary for solving (Col) and (dCol).

We further conclude that the performance of (MCF) heavily decreases then considering $b_{\max}$-redundancy. For (Col) and (dCol) the additional computational effort increases only moderately, however.

3.13.2 Lagrangian Decomposition Approaches

We use the volume algorithm as described by Haouari and Siala [86] for approximately solving the Lagrangian dual problem, which is configured as follows. Lagrangian multipliers are initialized by $\lambda_{k,e} = c_e/|C|$, $\forall k \in C$, $\forall e \in E$, ensuring a positive lower bound already in the first iteration. The target value T is set to

Table 3.10: Number of instances per set where the solution to the linear relaxation is integral. Only those instances are considered where $(\text{dCol})^{\text{LP}}$ could be solved withing 7200 CPU-seconds using $D^{(k,d')}$.

Variant	Set	#	$(\text{MCF})^{\text{LP}}$	$(\text{Col})^{\text{LP}}$	$(\text{dCol})^{\text{LP}}$
OPT, $b_{\max} = 0$	ClgSE-I1	25	3	3	25
	ClgSE-I2	14	0	0	14
	ClgSE-I3	15	0	0	15
	ClgN1B-I1	13	0	0	11
	ClgN1B-I2	9	3	3	9
SST, $b_{\max} = 0$	ClgSE-I1	25	2	3	25
	ClgSE-I2	15	0	0	15
	ClgSE-I3	15	0	0	15
	ClgN1B-I1	16	0	0	15
	ClgN1B-I2	17	1	3	17
SST, $b_{\max} = 30$	ClgSE-I1	25	0	1	25
	ClgSE-I2	15	0	0	14
	ClgSE-I3	15	0	0	14
	ClgN1B-I1	15	0	0	14
	ClgN1B-I2	17	0	2	17
SST, $b_{\max} = 50$	ClgSE-I1	25	0	2	25
	ClgSE-I2	15	0	0	14
	ClgSE-I3	15	0	0	14
	ClgN1B-I1	18	0	0	18
	ClgN1B-I2	13	1	2	13
SST, $b_{\max} = 100$	ClgSE-I1	24	0	0	24
	ClgSE-I2	15	0	0	14
	ClgSE-I3	14	0	0	13
	ClgN1B-I1	18	0	0	18
	ClgN1B-I2	14	0	1	14

$T = 1.1 z_{\text{UB}}$ with z_{UB} being the actual upper bound unless the actual lower bound $z_{\text{LB}} > 0.9\,T$ in which case T is multiplied by 1.1. We initially set $\rho = 0.1$ and $\alpha = 0.01$. After 20 consecutive non-improving iterations, ρ is multiplied by 0.67 in case it is greater than 10^{-4} and by 1.1 in an improving iteration if $\rho < 1$. If z_{LB} did not improve by more than 1% within the last 100 iterations and if $\alpha > 10^{-5}$, we multiply α by 0.85. The volume algorithm is terminated if $\lceil z_{\text{LB}} \rceil = z_{\text{UB}}$, after 250 consecutive non improving iterations, or if the maximum time limit is reached.

For the sequential hybrid Lagrangian method (SEQ) from Section 3.11, we set $N = 30$ and $\beta_{\min} = 0.01$, $\beta_{\max} = 0.4$, $\gamma = 0.05$, and $\delta = 100$ for the interleaved hybrid (INT) where β is initially set to $\beta = 0.1$.

Furthermore, we memorize hash-values of candidate solutions which have already

been used as starting solutions to avoid unnecessary runs of VND. These hash-values are also used to ensure that the N solutions stored in the sequential approach are pairwise different.

Table 3.11 details average relative gaps and corresponding standard deviations for LD, SEQ, and INT, while Table 3.12 shows median values of their individual CPU-times.

Table 3.11: Average gaps and corresponding standard deviations in % for LD, SEQ, and INT for the SST variant of $b_{\max}$-SNDP. Best values are marked bold.

Variant	Set	LD		SEQ		INT	
SST, $b_{\max} = 0$	ClgSE-I1	2.00	(2.62)	**1.63**	(2.38)	**1.63**	(2.38)
	ClgSE-I2	15.37	(4.65)	**9.13**	(4.65)	**9.13**	(4.65)
	ClgSE-I3	9.11	(4.84)	**7.09**	(4.84)	**7.09**	(4.84)
	ClgN1B-I1	5.37	(5.76)	2.61	(2.21)	**2.56**	(2.18)
	ClgN1B-I2	2.35	(2.34)	**1.29**	(1.46)	**1.29**	(1.46)
SST, $b_{\max} = 30$	ClgSE-I1	2.53	(3.39)	1.92	(2.60)	**1.75**	(2.36)
	ClgSE-I2	21.59	(4.78)	13.35	(4.78)	**13.18**	(4.82)
	ClgSE-I3	10.92	(4.13)	7.19	(4.13)	**7.08**	(4.02)
	ClgN1B-I1	6.76	(4.52)	3.36	(2.84)	**2.91**	(1.72)
	ClgN1B-I2	3.74	(4.71)	**1.54**	(1.85)	**1.54**	(1.85)
SST, $b_{\max} = 50$	ClgSE-I1	3.27	(4.80)	**2.17**	(3.05)	**2.17**	(3.05)
	ClgSE-I2	23.58	(5.13)	16.96	(5.13)	**16.59**	(5.00)
	ClgSE-I3	9.58	(4.01)	6.19	(4.01)	**6.09**	(3.97)
	ClgN1B-I1	6.03	(3.63)	3.84	(2.57)	**2.96**	(1.95)
	ClgN1B-I2	3.01	(3.75)	**1.59**	(1.83)	**1.59**	(1.83)
SST, $b_{\max} = 100$	ClgSE-I1	2.93	(3.54)	2.23	(2.64)	**2.22**	(2.63)
	ClgSE-I2	29.12	(7.13)	20.36	(7.13)	**19.13**	(4.96)
	ClgSE-I3	14.51	(6.38)	10.40	(6.38)	**10.33**	(6.33)
	ClgN1B-I1	7.48	(4.34)	4.84	(3.43)	**3.65**	(2.26)
	ClgN1B-I2	2.76	(3.23)	**1.79**	(2.00)	**1.79**	(2.00)

We conclude that both SEQ as well as INT significantly reduce the resulting optimality gap for all used settings and instance sets. Furthermore, the gaps obtained by INT are always smaller than or equal to those of SEQ.

The average relative difference between the lower bounds of SEQ and INT, respectively, compared to LD is strictly smaller than 0.07% except for the set ClgSE-I1 in the SST variant with $b_{\max}(k) = 100$, $\forall k \in C_2$, where it is 0.18%. Since the relative difference between the lower bounds obtained by SEQ and INT is always smaller than 0.01% we conclude that a smaller gap implies the generation of better feasible solutions.

From Table 3.12 we observe that both SEQ and INT significantly increase the neces-

Table 3.12: Median CPU-times for LD, SEQ, and INT for the SST variant of $b_{\max}$-SNDP. Best values are marked bold.

Variant	Set	LD	SEQ	INT
SST, $b_{\max} = 0$	ClgSE-I1	**0.8**	1.1	1.4
	ClgSE-I2	**2.6**	4.6	7.2
	ClgSE-I3	**1.7**	2.3	3.0
	ClgN1B-I1	**30.1**	107.7	88.6
	ClgN1B-I2	**38.6**	67.5	83.2
SST, $b_{\max} = 30$	ClgSE-I1	**4.8**	7.7	8.4
	ClgSE-I2	**26.6**	32.3	85.3
	ClgSE-I3	**12.7**	15.7	34.7
	ClgN1B-I1	**104.7**	417.6	489.4
	ClgN1B-I2	**146.0**	346.2	325.4
SST, $b_{\max} = 50$	ClgSE-I1	**9.7**	13.7	20.0
	ClgSE-I2	**76.0**	137.9	349.7
	ClgSE-I3	**22.1**	49.8	69.1
	ClgN1B-I1	**209.5**	714.9	807.2
	ClgN1B-I2	**254.3**	388.2	631.3
SST, $b_{\max} = 100$	ClgSE-I1	**41.8**	57.9	70.2
	ClgSE-I2	**423.2**	797.3	2239.4
	ClgSE-I3	**118.9**	296.5	384.3
	ClgN1B-I1	**493.3**	2249.0	3043.4
	ClgN1B-I2	**714.1**	1415.3	1729.3

sary CPU-time with SEQ usually being the fastest among these two. However, since its overhead compared to SEQ is not too high, INT can be recommended among the three Lagrangian variants to compute high quality solutions with relatively tight bounds in reasonable time.

3.13.3 Metaheuristics

In the following, we compare the metaheuristic methods introduced in Section 3.10. The VNS approach is terminated after 25 iterations of the outermost, largest shaking move. For GRASP we chose $\alpha = 0.25$ and generated 30 initial solutions. Furthermore, an absolute time limit of 7200 CPU-seconds has been used for all experiments.

Table 3.13 depicts average relative improvements of the obtained solution values as well as corresponding standard deviations for VNS and GRASP compared to the simpler VND. Each experiment has been repeated 30 times. Table 3.14 reports on average CPU-times and corresponding standard deviations.

Table 3.13: Average relative improvements over VND and corresponding standard deviations in % for VNS and GRASP considering the SST variant of $b_{\max}$-SNDP for 30 runs per instance. Best values are marked bold.

Variant	Set	$\frac{\text{VND}-\text{GRASP}}{\text{GRASP}}$ [%]		$\frac{\text{VND}-\text{VNS}}{\text{VNS}}$ [%]	
SST, $b_{\max} = 0$	ClgSE-I1	0.08	(0.45)	**0.18**	(0.42)
	ClgSE-I2	**1.76**	(1.85)	1.39	(1.83)
	ClgSE-I3	**1.42**	(2.72)	1.34	(2.74)
	ClgN1B-I1	1.05	(3.22)	**3.57**	(8.33)
	ClgN1B-I2	1.19	(4.61)	**5.05**	(9.59)
	ClgME-I1	**0.52**	(1.13)	0.49	(0.98)
	ClgME-I2	1.55	(3.56)	**1.57**	(3.60)
SST, $b_{\max} = 30$	ClgSE-I1	-0.15	(1.40)	**0.49**	(0.99)
	ClgSE-I2	2.22	(4.79)	**2.90**	(3.55)
	ClgSE-I3	1.05	(3.24)	**1.97**	(2.46)
	ClgN1B-I1	0.41	(6.36)	**3.21**	(7.88)
	ClgN1B-I2	-0.81	(2.98)	**3.64**	(8.85)
	ClgME-I1	-0.20	(1.38)	**0.43**	(0.79)
	ClgME-I2	1.18	(3.71)	**1.96**	(3.45)
SST, $b_{\max} = 50$	ClgSE-I1	1.22	(3.84)	**2.12**	(4.71)
	ClgSE-I2	1.73	(3.86)	**1.75**	(2.32)
	ClgSE-I3	1.07	(3.50)	**1.92**	(3.93)
	ClgN1B-I1	0.86	(6.82)	**3.90**	(8.38)
	ClgN1B-I2	-0.32	(3.15)	**4.11**	(10.61)
	ClgME-I1	0.06	(2.34)	**0.69**	(2.18)
	ClgME-I2	0.46	(3.95)	**2.02**	(3.35)
SST, $b_{\max} = 100$	ClgSE-I1	**2.71**	(7.23)	1.52	(6.03)
	ClgSE-I2	1.20	(2.64)	**1.35**	(2.47)
	ClgSE-I3	0.32	(1.69)	**0.59**	(1.16)
	ClgN1B-I1	2.67	(10.74)	**3.89**	(9.91)
	ClgN1B-I2	-1.23	(4.82)	**3.73**	(10.55)
	ClgME-I1	-0.40	(1.28)	**0.39**	(0.74)
	ClgME-I2	2.04	(5.99)	**3.45**	(5.83)

We conclude that both metaheuristics outperform the simpler VND with respect to the obtained solutions in the majority of cases. While the solutions obtained by the GRASP approach are sometimes worse than those of VND, this is obviously impossible for the VNS approach. With respect to the average improvement over VND, the VNS clearly outperforms GRASP in 24 out of 28 cases while its runtime is approximately equal.

Table 3.14: Average CPU-times in seconds and corresponding standard deviations for GRASP and VNS for the SST variant of $b_{\max}$-SNDP. 30 runs per instance. Best values are marked bold.

Variant	Set	GRASP		VNS	
SST, $b_{\max} = 0$	ClgSE-I1	**0.16**	(0.06)	**0.16**	(0.06)
	ClgSE-I2	**0.68**	(0.23)	1.37	(0.70)
	ClgSE-I3	**0.39**	(0.16)	0.57	(0.26)
	ClgN1B-I1	**7.30**	(2.40)	12.17	(4.52)
	ClgN1B-I2	14.30	(3.99)	**11.51**	(4.11)
	ClgME-I1	5.45	(1.89)	**4.82**	(2.28)
	ClgME-I2	**6.98**	(2.21)	13.05	(6.40)
SST, $b_{\max} = 30$	ClgSE-I1	**1.34**	(0.54)	1.53	(0.91)
	ClgSE-I2	**5.55**	(2.20)	13.84	(8.27)
	ClgSE-I3	**3.29**	(1.35)	5.80	(3.25)
	ClgN1B-I1	**22.37**	(7.81)	35.68	(17.82)
	ClgN1B-I2	39.43	(11.69)	**33.98**	(14.59)
	ClgME-I1	19.79	(7.11)	**17.81**	(7.90)
	ClgME-I2	**19.96**	(8.50)	32.41	(14.37)
SST, $b_{\max} = 50$	ClgSE-I1	**2.22**	(0.84)	2.90	(1.39)
	ClgSE-I2	**10.54**	(4.39)	25.44	(13.34)
	ClgSE-I3	**5.90**	(2.15)	10.83	(5.71)
	ClgN1B-I1	**37.48**	(11.60)	60.72	(25.42)
	ClgN1B-I2	65.49	(22.31)	**60.27**	(27.56)
	ClgME-I1	36.29	(13.35)	**34.77**	(17.78)
	ClgME-I2	**33.66**	(12.37)	61.73	(31.75)
SST, $b_{\max} = 100$	ClgSE-I1	**5.10**	(2.13)	7.10	(3.22)
	ClgSE-I2	**24.58**	(9.89)	65.17	(35.61)
	ClgSE-I3	**13.63**	(5.33)	25.09	(10.71)
	ClgN1B-I1	**99.24**	(38.98)	157.64	(92.94)
	ClgN1B-I2	144.13	(40.86)	**129.85**	(53.71)
	ClgME-I1	107.73	(39.54)	**88.54**	(40.68)
	ClgME-I2	**78.28**	(35.64)	134.62	(69.91)

3.13.4 Overall Comparison

In the following, we compare the objective values of the solutions derived by INT and VNS to those of (dCol) for a representative subset of the so far considered configurations. Table 3.15 reports average relative objective values solutions as well as corresponding standard deviations in percent.

Overall, we conclude that each method is able to compute high quality solutions. The interleaved Lagrangian hybrid approach yields slightly better primal solutions than VNS and has the advantage of additionally providing a lower bound and thus a gap on the maximum distance to an optimal solution. VNS, however, can be used

Table 3.15: Average relative objective values for INT and VNS compared to (dCol) and corresponding standard deviations in %.

Variant	Set	$\frac{\text{INT}-(\text{dCol})}{(\text{dCol})}$ [%]		$\frac{\text{VNS}-(\text{dCol})}{(\text{dCol})}$ [%]	
SST, $b_{\max} = 0$	ClgSE-I1	0.000	(0.000)	0.059	(0.204)
	ClgSE-I2	0.000	(0.000)	0.581	(0.757)
	ClgSE-I3	0.000	(0.000)	0.556	(1.049)
	ClgN1B-I1	0.000	(0.000)	0.051	(0.116)
	ClgN1B-I2	0.000	(0.000)	0.538	(0.674)
SST, $b_{\max} = 30$	ClgSE-I1	0.002	(0.010)	0.161	(0.657)
	ClgSE-I2	0.104	(0.293)	0.998	(1.386)
	ClgSE-I3	0.011	(0.032)	0.447	(0.663)
	ClgN1B-I1	0.004	(0.016)	0.137	(0.226)
	ClgN1B-I2	0.012	(0.049)	0.881	(0.840)
SST, $b_{\max} = 100$	ClgSE-I1	0.104	(0.041)	2.334	(5.886)
	ClgSE-I2	0.105	(0.216)	0.448	(0.512)
	ClgSE-I3	0.102	(0.295)	0.236	(0.382)
	ClgN1B-I1	0.015	(0.064)	0.183	(0.276)
	ClgN1B-I2	0.000	(0.000)	2.945	(3.164)

for even larger instances as it generally computes feasible solutions close to optimal ones in much shorter time than the other methods.

3.14 Conclusions and Future Work

In this chapter, the $b_{\max}$-Survivable Network Design Problem ($b_{\max}$-SNDP) which aims to efficiently extend real-world communication networks has been considered. In $b_{\max}$-SNDP a subset of all customers is redundantly connected by means of two node disjoint routes. These redundancy requirements are, however, occasionally relaxed by allowing a connection via a final non-redundant branch line that does not exceed a certain length $b_{\max}$.

In a first section, two new mixed integer programming approaches for solving $b_{\max}$-SNDP to proven optimality based on an exponential number of so-called connection variables have been presented. These can be solved by branch-and-price. One main contribution within this section is the usage of alternative dual-optimal solutions in the pricing subproblems to significantly speed up the solution of the linear relaxation of both models. By a polyhedral comparison we subsequently showed that both proposed models theoretically dominate existing ones. We further proved that the

second model, which is a directed variant of the first one, dominates its undirected counterpart.

In the second part of this chapter a Lagrangian decomposition approach for $b_{\max}$-SNDP based on an already existing multi-commodity flow formulation has been introduced. The subproblems arising in the Lagrangian dual problem have been subsequently discussed in great detail and some comments on the lower bounds that can be obtained by this approach are given.

Afterwards, a constructive heuristic for computing initial feasible solutions to $b_{\max}$-SNDP as well as three types of neighborhood structures have been introduced. These are used within a variable neighborhood search and a greedy randomized adaptive search, respectively. Using these neighborhood structures, two hybrid method combining above mentioned Lagrangian decomposition approach with variable neighborhood descent have been finally discussed.

Computational results show that both branch-and-price approaches perform reasonably well on medium sized instances. While, the undirected model yields tight optimality gaps already after relatively short time, it sometimes has problems to further raise the obtained lower bounds in order to prove optimality of a solution. For solving the linear relaxation of its directed counterpart much more computational effort is needed. The obtained solutions are, however, already integral and thus proven optimal solutions in the majority of test cases.

For the Lagrangian methods, both hybrid approaches turn out to outperform the pure Lagrangian decomposition method with respect to the obtained upper bounds and optimality gaps. The obtained upper bounds are usually optimal or close to an optimal solutions value. Among the two metaheuristic methods, the VNS approach which computes near optimal solutions very fast clearly outperforms the GRASP.

Which among the proposed methods should be used in practice depends on the considered instances. Small and medium sized instances can be solved to proven optimality in reasonable time. Thus, one of the two proposed exact methods can be recommended. However, when considering larger instance or when the necessary runtime should be kept relatively small these are not suitable. Here, if both lower and upper bounds are needed, one of the Lagrangian methods can be used, while the VNS can be recommended to compute high quality solutions for large scale instances or when small computational times are needed.

Interesting areas for further research include the development of methods based on the multilevel approach; see e.g. [179] for a survey. These might include the methods proposed in this chapter for solving smaller subproblems and can be used to tackle

very large scale instances of $b_{\max}$-SNDP. Furthermore, considering additional algorithms and methods for solving the $\mathcal{NP}$-hard pricing subproblems of the directed connection formulation might allow for solving even larger instances to proven optimality.

Chapter 4

The Capacitated Connected Facility Location Problem

4.1 Introduction

We consider a real-world network design problem with additional location aspects which occurs when extending existing fiber-optic networks. Nowadays, telecommunication companies are often confronted with rising bandwidth requirements of customers while especially in smaller cities and rural areas realizing connections entirely with fiber-optic routes (i.e. fiber-to-the-home) is often too expensive and does not pay of economically. In such situations, providers need to make a compromise between the bandwidth offered to individual customers and the resulting construction costs.

Frequently, these companies deal with such situations by extending the fiber-optic infrastructure by new routes to so-called *mediation points* that bridge the high-bandwidth network with an older lower-bandwidth network. While the original network is still used between a customer and its assigned mediation point, the newly installed high-bandwidth routes are used in the remaining network. Ensuring that the maximum distance between a customer and its mediation point is not too high, the bandwidth available for each customer can be significantly increased while avoiding too high construction costs. Depending on the network used between these mediation points and the customers, these scenarios are typically referred to as *fiber-to-the-curb* in case of a traditional copper network or *powerline* in case of using electric power transmission lines.

From an optimization point of view these scenarios can be modeled as variants of the *Connected Facility Location Problem (ConFL)* [125], where new facilities, which correspond to the above mentioned mediation points, need to be installed and connected with each other and customer nodes need to be assigned to them. However, the classical ConFL often cannot be used to model and solve real-world scenarios since it does neglect real-world constraints such as those imposed by individual client bandwidth demands and corresponding maximum assignable demands to individual facilities. Furthermore, telecommunication providers are usually interested in upgrading not necessarily all but only the most profitable subset of potential customers by additionally considering the expected return on investment for individual customers.

To overcome these shortages, our model to which we refer as the *rooted Prize Collecting Capacitated Connected Facility Location Problem (CConFL)* resembles a prize collecting variant of ConFL and additionally considers capacity constraints on potential facility locations.

In this chapter, we formally introduce CConFL in Section 4.2 and review previous and related work in Section 4.3. Afterwards, we present mixed integer programming based approaches for solving CConFL to proven optimality based on multi-commodity flows in Section 4.4, a branch-and-cut approach based on directed connectivity cuts in Section 4.5, and a branch-and-cut-and-price approach additionally involving an exponential number of so-called pattern variables in Section 4.6. Theoretical comparisons of the corresponding polyhedra of all presented formulations are given in Section 4.7.

Furthermore, we describe a Lagrangian decomposition (LD) approach based on one of the previously presented MIP formulations in Section 4.8 and detail a Lagrangian heuristic to derive feasible solutions in Section 4.9. Section 4.10 is dedicated to local search and very large scale neighborhood search based methods for improving the obtained solutions. Test instances and computational results are discussed in Sections 4.11 and 4.12, before we finally draw conclusions and outline potential future work in Section 4.13.

The approaches presented in this chapter have been published in [118, 121, 119]. Furthermore, in [117], we presented a metaheuristic hybrid for CConFL but using a slightly different objective function.

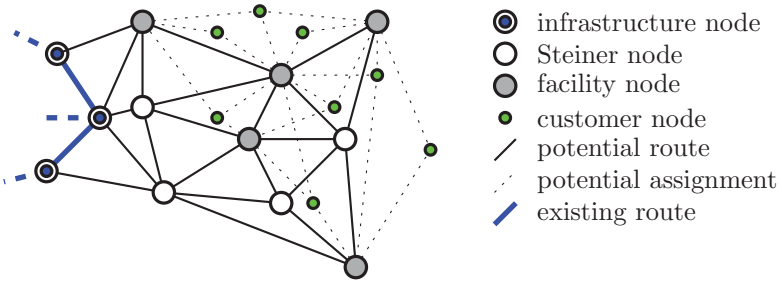


Figure 4.1: Original problem instance.

4.2 Problem Definition

Formally, an instance of CConFL is given by an undirected connected graph $G^{\mathrm{o}} = (V^{\mathrm{o}}, E^{\mathrm{o}})$ with a connected subgraph $G_{\mathrm{I}} = (V_{\mathrm{I}}, E_{\mathrm{I}})$, $V_{\mathrm{I}} \subsetneq V^{\mathrm{o}}$, $E_{\mathrm{I}} \subsetneq E^{\mathrm{o}}$ representing the existing fiber-optic infrastructure, see Figure 4.1.

Each edge $e = (u, v) \in E^{\mathrm{o}}$ has associated costs $c_e^{\mathrm{o}} \geq 0$ corresponding to the costs of installing a new route between u and v. Potential facility locations (mediation points) $F^{\mathrm{o}} \subseteq V^{\mathrm{o}} \setminus V_{\mathrm{I}}$ are given with associated costs $f_i \geq 0$ for installing them (*opening costs*) and maximum assignable demands $D_i \in \mathbb{N}_0$, $\forall i \in F^{\mathrm{o}}$. All remaining nodes $v \in V^{\mathrm{o}} \setminus (V_{\mathrm{I}} \cup F^{\mathrm{o}})$ are Steiner nodes that may be used in a solution. Note that each facility node might also be used as a Steiner node when no customer is assigned to it, in which case its opening costs need not to be paid. Furthermore, we are given a set of potential customers C^{o} with individual demands $d_k \in \mathbb{N}_0$ and prizes $p_k \geq 0$, $\forall k \in C^{\mathrm{o}}$, the latter corresponding to the expected profit when supplying customer k. Finally, costs $a_{i,k} \geq 0$ for assigning the complete demand of customer $k \in C^{\mathrm{o}}$ to a potential facility location $i \in F^{\mathrm{o}}$ are given (*assignment costs*). If a client k cannot be assigned to facility i we assume here for simplicity $a_{i,k} = \infty$.

During preprocessing we shrink the existing fiber-optic infrastructure $G_{\mathrm{I}} = (V_{\mathrm{I}}, E_{\mathrm{I}})$ into a single root node r, yielding a reduced graph $G = (V, E)$ with node set $V = (V^{\mathrm{o}} \cup \{r\}) \setminus V_{\mathrm{I}}$ and edge set $E = \{(u, v) \in E^{\mathrm{o}} \mid u, v \notin E_{\mathrm{I}}\} \cup \{(r, v) \mid \exists (u, v) \in E^{\mathrm{o}} : u \in V_{\mathrm{I}} \wedge v \notin V_{\mathrm{I}}\}$; see Figure 4.2 for such a rooted problem instance. Edge costs $c_e \geq 0$ are defined as

$$c_e = \begin{cases} c_e^{\mathrm{o}} & \text{if } u, v \in V^{\mathrm{o}} \setminus V_{\mathrm{I}} \\ \min_{f=(w,v) \in E^{\mathrm{o}} \mid w \in V_{\mathrm{I}}} c_f^{\mathrm{o}} & \text{otherwise} \end{cases} \qquad \forall e = (u, v) \in E.$$

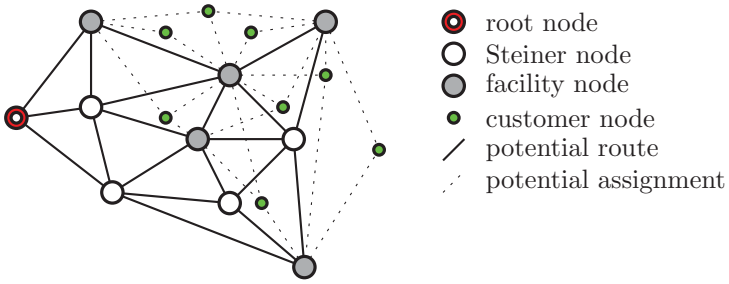


Figure 4.2: Rooted problem instance.

Furthermore, we remove all possibly existing assignment possibilities between customers $k \in C^{\mathrm{o}}$ and facilities $i \in F^{\mathrm{o}}$ where $a_{i,k} \geq p_k$ by setting $a_{i,k} = \infty$. In case strict inequality holds – i.e. $a_{i,k} > p_k$ – such an a assignment cannot be part of an optimal solution as it does not pay off, while at least one optimal solution not including the assignment between i and k does exist if $a_{i,k} = p_k$.

Customers with no remaining assignment possibilities are entirely removed. Similarly, some potential facilities $i \in F^{\mathrm{o}}$ that cannot be profitable can be identified by solving a 0–1 knapsack problem for each facility with knapsack size D_i, and an item with weight d_k and profit $p_k - a_{i,k}$ for each assignable customer. A facility can be removed if the profit of the optimal solution to this knapsack problem does not exceed the facility's opening costs f_i. If solving these knapsack problems for all the facilities is too time-consuming, an option is to only solve the corresponding linear programming relaxations and to use the hereby obtained upper bounds to the optimal solutions' profits.

We denote by $C \subseteq C^{\mathrm{o}}$ and $F \subseteq F^{\mathrm{o}}$ ($F \subseteq V$) the resulting, possibly reduced sets of potential customers and facility locations. Furthermore, $C_i = \{k \in C \mid a_{i,k} < p_k\}$ denotes the set of customers that may be assigned to facility $i \in F$ and $F_k = \{i \in F \mid k \in C_i\}$ the set of potential facilities a customer $k \in C$ may be assigned to.

As depicted in Figure 4.3, a solution to CConFL $S = (R_S, T_S, F_S, C_S, \alpha_S)$ consists of a set of opened facilities $F_S \subseteq F$ connected to each other as well as to the root node r by a Steiner tree (R_S, T_S), $R_S \subseteq V$, $T_S \subseteq E$. $C_S \subseteq C$ is the set of customers feasibly (i.e. respecting the capacity constraints) assigned to facilities F_S, whereas the actual mapping between customers and facilities is described by $\alpha_S : C_S \to F_S$.

Since we are considering a single source variant of the connected facility location problem, each customer may be assigned to at most one facility. The objective function of CConFL can be stated as

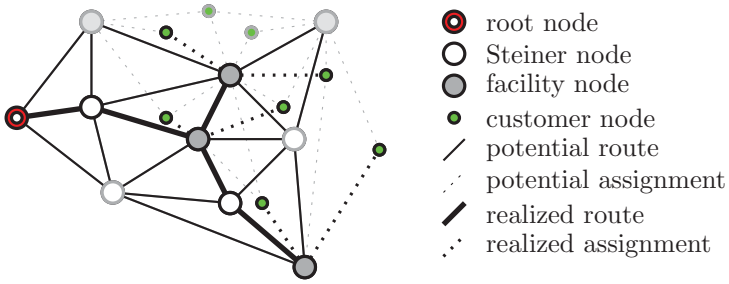


Figure 4.3: An exemplary solution to CConFL.

$$c(S) = \sum_{e \in T_S} c_e + \sum_{i \in F_S} f_i + \sum_{k \in C_S} a_{\alpha_S(k),k} + \sum_{k \in C} p_k - \sum_{k \in C_S} p_k \tag{4.1}$$

$$= \sum_{e \in T_S} c_e + \sum_{i \in F_S} f_i + \sum_{k \in C_S} a_{\alpha_S(k),k} + \sum_{k \in C \setminus C_S} p_k \tag{4.2}$$

An optimal solution S^* (i.e. a most profitable one) is given by the minimal objective value, i.e. $c(S^*) \leq c(S)$ for all feasible solutions S. Note that we add the profits lost – i.e. the profits of uncovered customers – instead of subtracting the collected profits in equation (4.2), ensuring a nonnegative objective value for any feasible solution. Since CConFL combines the (prize collecting) Steiner tree problem (STP) on a graph with the single source capacitated facility location problem (SSCFLP), which are both strongly $\mathcal{NP}$-hard [103, 42], CConFL is strongly $\mathcal{NP}$-hard, too.

4.3 Related Work

Karger and Minkoff [101] considered the so-called maybecast problem which can be modeled as a connected facility location problem and described a constant factor approximation for their problem. The name connected facility location has been introduced by Gupta et al. [81] in their work on virtual private networks.

Since then several authors proposed approximation algorithms for diverse variants of ConFL. Swamy and Kumar [166] presented a primal-dual algorithm with an approximation ratio of 8.55 which is also a factor 4.55 approximation for the so called rent-or-buy problem, a variant of ConFL where no opening costs are given and facilities may be opened at all nodes. By considering an LP rounding technique, Hasan

et al. [89] improved their method to a factor 8.29 approximation algorithm for the case of edge costs obeying the triangle inequality and a factor seven approximation in case all opening costs are equal. Recently, a randomized approximation algorithm with an expected approximation ratio of four, which can be derandomized with a resulting approximation factor of 4.23, has been presented by Eisenbrand et al. [57].

Ljubić [125] described a branch-and-cut approach based on directed connection cuts as well as a hybrid metaheuristic combining variable neighborhood search (VNS) with reactive tabu search for the rooted variant of ConFL. Tomazic and Ljubić [169] considered the unrooted version of ConFL and presented a greedy randomized adaptive search procedure. Furthermore, they transformed the problem to the minimum Steiner arborescence problem and solved it by an exact branch-and-cut method. Ten different integer programming formulations for ConFL have been presented by Gollowitzer and Ljubić [77]. Next to computational results on their models, they further ranked them by comparing the various polyhedra. The same authors subsequently discussed a large number of models for a hop constrained variant of ConFL [126, 127]. Bardossy and Raghavan [155, 14] combined dual ascent with local search to derive lower and upper bounds for a more general variant of ConFL.

We presented two VNS variants for a version of CConFL without assignment and opening costs in [117]. To the best of our knowledge our concrete variant of the connected facility location problem, which contains many of the previously discussed problem variants as special cases, has not been considered so far.

A closely related problem is the Steiner tree star (STS) problem, where opening costs for facilities included in the Steiner tree must be paid even if no customers are assigned to them. Exact methods for the STS problem have been described by Lee et al. [115, 114], while Xu et al. [181] presented a tabu search metaheuristic. A generalized variant of the STS problem, where customers nodes and potential facilities are not necessarily disjoint, has been described by Khuller and Zhu [108].

Furthermore, literature on the (prize collecting) Steiner tree problem (STP) on a graph, as well as on the (single source) capacitated facility location problem (SSCFLP) can be considered as relevant, since CConFL is composed from these two problems, see e.g. [180] for a survey on the STP and [6] for a recent work on the SSCFLP with a comprehensive list of further references on that topic.

4.4 Multi-Commodity Flow Formulations for CConFL

CConFL can be modeled as a mixed integer program (MIP) based on directed multi-commodity flows in two rather obvious ways. While our first model ($\mathrm{dMCF_f}$) presented in Section 4.4.1 is based on sending one unit of flow to each potential facility location, model ($\mathrm{dMCF_c}$) presented in Section 4.4.2 sends flow to each potential customer.

For an easier presentation we define an extended graph $G' = (V', E')$ combining G with the set of potential customers C as additional nodes and potential assignments between facilities and customers as additional edges (*assignment edges*). Formally, G' is given by its node set $V' = V \cup C$ and its edge set $E' = E \cup \{(i,j) \mid i \in F \wedge j \in C_i\}$. Edge costs $c'_e \geq 0$ are defined by

$$c'_e = \begin{cases} c_e & \text{if } e \in E \\ a_{i,k} & \text{otherwise} \end{cases} \qquad \forall e = (i,k) \in E'. \tag{4.3}$$

4.4.1 A Facility Oriented Model

Let $A_r = \{(r,v) \mid (r,v) \in E\}$ denote the set of directed edges, i.e. arcs, going out from the root node r and $A'_i = \{(u,v),(v,u) \mid (u,v) \in E \wedge u,v \notin \{r,i\}\}$, $\forall i \in F$, the set containing two oppositely directed arcs for each pair of nodes $u,v \in V \setminus \{r,i\}$ that are connected by an edge in G. Let $A_i^- = \{(v,i) \mid (v,i) \in E\}$ be the set of ingoing arcs for each facility $i \in F$. We can now define the set of arcs relevant for connecting a facility $i \in F$ to the root node as $A_i = A_r \cup A'_i \cup A_i^-$. In model ($\mathrm{dMCF_f}$) (4.4)–(4.14) we use decision variables $x_e \in \{0,1\}$, $\forall e \in E'$, indicating whether an edge is used in a solution (in which case $x_e = 1$) or not and variables $y_k \in \{0,1\}$, $\forall k \in C$, to specify whether a customer is feasibly assigned to an opened facility ($y_k = 1$) or not. Furthermore, to specify whether an arc is used in the connection to a potential facility we use flow variables $s^i_{u,v} \in [0,1]$, $\forall i \in F$, $\forall (u,v) \in A_i$, and design variables $z_i \in [0,1]$, $\forall i \in F$, to indicate if a potential facility is opened ($z_i = 1$).

$$(\mathrm{dMCF_f})\ \min \sum_{e\in E'} c'_e x_e + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k(1-y_k) \tag{4.4}$$

$$\text{s.t.} \sum_{(u,v)\in A_i} s^i_{u,v} - \sum_{(v,u)\in A_i} s^i_{v,u} = \begin{cases} -z_i & \text{if } v = r \\ z_i & \text{if } v = i \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in F,\ \forall v \in V \tag{4.5}$$

$$s^i_{u,v} + s^i_{v,u} \le x_{u,v} \qquad \forall i \in F,\ \forall (u,v) \in E \mid u,v \neq r \tag{4.6}$$

$$s^i_{r,v} \le x_{r,v} \qquad \forall i \in F,\ \forall (r,v) \in E \tag{4.7}$$

$$x_{i,k} \le z_i \qquad \forall (i,k) \in E' \mid k \in C \tag{4.8}$$

$$\sum_{k\in C_i} d_k x_{i,k} \le D_i z_i \qquad \forall i \in F \tag{4.9}$$

$$\sum_{i\in F_k} x_{i,k} \ge y_k \qquad \forall k \in C \tag{4.10}$$

$$0 \le s^i_{u,v} \le 1 \qquad \forall i \in F,\ \forall (u,v) \in A_i \tag{4.11}$$

$$0 \le z_i \le 1 \qquad \forall i \in F \tag{4.12}$$

$$x_e \in \{0,1\} \qquad \forall e \in E' \tag{4.13}$$

$$y_k \in \{0,1\} \qquad \forall k \in C \tag{4.14}$$

The objective function (4.4) unifies assignment and edge costs by using the concept of the extended graph G' but otherwise corresponds to function (4.2). Constraints (4.5) are the usual flow conservation constraints, inequalities (4.6) and (4.7) link variables $s^i_{u,v}$ and x_e, and inequalities (4.8) ensure that a facility is opened if an incident assignment edge is used. Inequalities (4.9) are the capacity constraints for each facility $i \in F$, while inequalities (4.10) ensure that a customer's prize can only be earned if the customer is connected to a facility. Note that for variables z_i and $s^i_{u,v}$ only lower and upper bounds are defined in (4.11) and (4.12). They will automatically become integer due to constraints (4.5), (4.6), and (4.8).

4.4.2 A Customer oriented model

Model ($\mathrm{dMCF_c}$) (4.15)–(4.24) sends one unit of flow to each potential customer, but otherwise is similar to model ($\mathrm{dMCF_f}$). Thus we define the set of relevant arcs $A_k = A_r \cup A' \cup A^-_k$ for each customer $k \in C$, where A_r is the set of arcs going out from the root node as defined in Section 4.4.1, $A' = \{(u,v),(v,u) \mid (u,v) \in E \wedge u,v \neq r\}$, and $A^-_k = \{(i,k) \mid (i,k) \in E'\}$.

$$(\mathrm{dMCF_c})\ \min \sum_{e\in E'} c'_e x_e + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k(1-y_k) \tag{4.15}$$

$$\text{s.t.} \sum_{(u,v)\in A_k} s^k_{u,v} - \sum_{(v,u)\in A_k} s^k_{v,u} = \begin{cases} -y_k & \text{if } v = r \\ y_k & \text{if } v = k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in C,\ \forall v \in E' \tag{4.16}$$

$$s^k_{u,v} + s^k_{v,u} \le x_{u,v} \qquad \forall k \in C,\ \forall (u,v) \in E' \mid u,v \neq r \tag{4.17}$$

$$s^k_{r,v} \le x_{r,v} \qquad \forall k \in C,\ \forall (r,v) \in E \tag{4.18}$$

$$x_{i,k} \le z_i \qquad \forall i \in F,\ \forall k \in C_i \tag{4.19}$$

$$\sum_{k\in C_i} d_k x_{i,k} \le D_i z_i \qquad \forall i \in F \tag{4.20}$$

$$0 \le s^k_{u,v} \le 1 \qquad \forall k \in C,\ \forall (u,v) \in A_k \tag{4.21}$$

$$0 \le z_i \le 1 \qquad \forall i \in F \tag{4.22}$$

$$x_e \in \{0,1\} \qquad \forall e \in E' \tag{4.23}$$

$$y_k \in \{0,1\} \qquad \forall k \in C \tag{4.24}$$

Here, constraints (4.16) resemble the flow conservation constraints for each customer $k \in C$ and similarly to ($\mathrm{dMCF_f}$) inequalities (4.17), (4.18), and (4.19) link variables x with y and x with z, respectively. While the capacity constraints (4.20) are identical to those of formulation ($\mathrm{dMCF_f}$), we do not need explicit linking constraints between variables x and y in model ($\mathrm{dMCF_c}$) since they are implicitly included in the flow conservation constraints. Note that for variables z_i and $s^k_{u,v}$ only lower and upper bounds are defined in (4.21) and (4.22). They will automatically become integer by the same arguments as for model ($\mathrm{dMCF_f}$).

4.5 Branch-and-Cut for CConFL

In this section we present another exact approach for CConFL. Model (dCut) involves an exponential number of constraints and can be solved by dynamically including them on demand at each node of the search tree, i.e. by branch-and-cut. (dCut) is based on so-called directed connection cuts. It is well known that such models often outperform multi-commodity flow based models like the ones presented in the last two sections from a computational point of view. As will be shown in Section 4.7 our directed cut model (dCut) is also theoretically stronger than both previously presented flow models.

Similarly to the extended graph introduced earlier, for model (dCut) we define a directed extended graph (V', A') combining G with all potential customers, i.e. $V' = V \cup C$. Its arc set A' consists of one arc going out of the root node for each edge in G adjacent to r, while all other edges of G are replaced by two oppositely directed arcs. Furthermore, A' contains one assignment arc (i, k) for each potential assignment between a facility $i \in F$ and a customer $k \in C_i$. Arc costs $c'_{u,v}$, $\forall (u, v) \in A'$, are defined as

$$c'_{u,v} = \begin{cases} c_e & \text{if } e = (u, v) \in E \\ a_{u,v} & \text{otherwise.} \end{cases} . \tag{4.25}$$

Model (dCut) uses variables $z_i \in \{0, 1\}$, $\forall i \in F$, indicating whether or not a facility is opened, variables $y_k \in \{0, 1\}$, $\forall k \in C$, denoting if a customer is supplied or not, and variables $x_{i,j} \in \{0, 1\}$, $\forall (i, j) \in A'$, specifying whether or not an arc is used.

$$\text{(dCut)} \quad \min \sum_{(u,v)\in A'} c'_{u,v} x_{u,v} + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k (1 - y_k) \tag{4.26}$$

$$\text{s.t.} \quad x_{u,v} + x_{v,u} \leq 1 \qquad \forall e = (u, v) \in E \mid u, v \neq r \tag{4.27}$$

$$x_{i,k} \leq z_i \qquad \forall i \in F,\ \forall k \in C_i \tag{4.28}$$

$$\sum_{i\in F_k} x_{i,k} \geq y_k \qquad \forall k \in C \tag{4.29}$$

$$\sum_{k\in C_i} d_k x_{i,k} \leq D_i z_i \qquad \forall i \in F \tag{4.30}$$

$$\sum_{(u,v)\in\delta^+(W)} x_{u,v} \geq z_i \qquad \forall i \in F,\ \forall W \subsetneq V \mid r \in W \wedge i \notin W \tag{4.31}$$

$$\sum_{(u,v)\in\delta^+(W)} x_{u,v} + \sum_{i\in F_k \cap W} x_{i,k} \geq y_k \qquad \forall k \in C,\ \forall W \subsetneq V \mid r \in W \tag{4.32}$$

$$x_{u,v} \in \{0, 1\} \qquad \forall (u, v) \in A' \tag{4.33}$$

$$z_i \in \{0, 1\} \qquad \forall i \in F \tag{4.34}$$

$$y_k \in \{0, 1\} \qquad \forall k \in C \tag{4.35}$$

Due to using (V', A'), assignment costs are represented as arc costs in the objective function (4.26). Constraints (4.27) ensure that at most one out of each pair of oppositely directed arcs between two nodes is chosen, linking constraints (4.28) guarantee that an assignment arc might only be used if the corresponding facility is opened, while inequalities (4.29) ensure that a customers prize can only be earned if

it is assigned to a facility by an assignment arc. Constraints (4.30) are the capacity constraints for each facility. In inequalities (4.31) and (4.32) which resemble the directed connection cuts for facilities and customers, respectively, we denote by $\delta^+(W) = \{(u,v) \in A' \mid u,v \in V \wedge u \in W \wedge v \notin W\}$ the set of arcs going out of node set W, i.e. the cutset of W. Since customer nodes have only incoming arcs, we need not consider other customer nodes than k for a directed connection cut to $k \in C$ in inequalities (4.32). Note that the directed connection cuts for customers (4.32) do only strengthen the LP relaxation of model (dCut), i.e. removing inequalities (4.32) would also yield a valid model for CConFL.

Since the number of connectivity constraints (4.31) and (4.32) is exponentially large, we dynamically identify violated inequalities during runtime. As already mentioned in Section 2.1.1 the minimum capacity of a cut between two nodes u and v is equivalent to a maximum flow between them. Thus, we use an implementation of the push-relabel method for the maximum flow problem by Cherkassky and Goldberg [33] for identifying violated cut inequalities.

4.6 Branch-and-Cut-and-Price for CConFL

Model (dBCP) presented in this section considers whole profitable assignment patterns between customers and facilities instead of taking into account each potential assignment individually. We consider the set of all feasible and profitable assignment patterns Ω_i for facility $i \in F$ and denote by $\Omega = \bigcup_{i \in F} \Omega_i$ the total set of such assignment patterns. By $\Omega(k) \subseteq \Omega$, $k \in C$, we denote the set of patterns connecting customer k. Each pattern $\omega \in \Omega$ assigns a set of customers $\mathcal{C}(\omega) = \{k \in C \mid \omega \in \Omega(k)\}$ to a dedicated facility $\mathcal{F}(\omega) \in F$, with $\mathcal{F}(\omega) = i \in F \Leftrightarrow \omega \in \Omega_i$. Furthermore, Ω does only contain valid and profitable patterns, i.e. $\mathcal{C}(\omega) \subseteq C_{\mathcal{F}(\omega)}$, $\sum_{k \in \mathcal{C}(\omega)} d_k \leq D_{\mathcal{F}(\omega)}$, and $\sum_{k \in \mathcal{C}(\omega)} p_k - a_{\mathcal{F}(\omega),k} > f_{\mathcal{F}(\omega)}$, $\forall \omega \in \Omega$. As (dCut), model (dBCP) uses variables $z_i \in \{0,1\}$, $\forall i \in F$, indicating opened respectively closed facilities, and variables $y_k \in \{0,1\}$, $\forall k \in C$, denoting if a customer is connected. Variables $\gamma_\omega \in \{0,1\}$, $\forall \omega \in \Omega$, denote whether a pattern is realized or not. Since these pattern variables do implicitly model assignments between facilities and customers, we need not consider corresponding assignment arcs in variables $x_{u,v} \in \{0,1\}$, $\forall (u,v) \in A = \{(u,v),(v,u) \mid (u,v) \in E \wedge u,v \neq r\} \cup \{(r,v) \mid (r,v) \in E\}$, which indicate whether an arc is used in the Steiner tree connecting open facilities and the root node.

$$\text{(dBCP)} \quad \min \sum_{(u,v)\in A} c'_{u,v}x_{u,v} + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k(1-y_k) +$$

$$+ \sum_{\omega\in\Omega} \sum_{k\in\mathcal{C}(\omega)} a_{\mathcal{F}(\omega),k}\gamma_\omega \tag{4.36}$$

$$\text{s.t.} \quad \sum_{\omega\in\Omega_i} \gamma_\omega \le z_i \qquad \forall i \in F \tag{4.37}$$

$$\sum_{\omega\in\Omega(k)} \gamma_\omega \ge y_k \qquad \forall k \in C \tag{4.38}$$

$$x_{u,v} + x_{v,u} \le 1 \qquad \forall (u,v) \in E \mid u,v \neq r \tag{4.39}$$

$$\sum_{(u,v)\in\delta^+(W)} x_{u,v} \ge z_i \qquad \forall i \in F,\ \forall W \subsetneq V \mid r \in W \wedge i \notin W \tag{4.40}$$

$$\sum_{(u,v)\in\delta^+(W)} x_{u,v} +$$

$$+ \sum_{i\in F_k\cap W} \sum_{\omega\in\Omega_i\cap\Omega(k)} \gamma_\omega \ge y_k \qquad \forall k \in C,\ \forall W \subsetneq V \mid r \in W \tag{4.41}$$

$$z_i \in \{0,1\} \qquad \forall i \in F \tag{4.42}$$

$$y_k \in \{0,1\} \qquad \forall k \in C \tag{4.43}$$

$$x_{u,v} \in \{0,1\} \qquad \forall (u,v) \in A \tag{4.44}$$

$$\gamma_\omega \in \{0,1\} \qquad \forall \omega \in \Omega \tag{4.45}$$

Constraints (4.37) and (4.38) are the coupling constraints between assignment patterns and facilities respectively customers. As for model (dCut), constraints (4.39) ensure that at most one arc of each pair of oppositely directed arcs can be used, while constraints (4.40) are the directed connection cuts for facilities. Constraints (4.41) – which are included to strengthen the LP relaxation of model (dBCP) – resemble the directed connectivity cuts for customers. They need to be partly expressed in terms of pattern variables, since no variables explicitly modeling assignments between facilities and customers are included in (dBCP).

As for model (dCut), connectivity cuts for facilities as well as for customers are added as cutting planes to the model on demand only. Note that variables z_i, $\forall i \in F$, as well as y_k, $\forall k \in C$, are declared as binary due to our branching strategy – see Section 4.6.1 – while defining them as continuous would also yield a valid model.

Since Ω contains exponentially many variables, we cannot solve (dBCP) directly by branch-and-cut. Thus, as usual in column generation approaches – see e.g. [16, 49] for

introductions to column generation and branch-and-price – we consider the reduced master problem (RMP) containing only a small subset of variables $\tilde{\Omega} \subsetneq \Omega$ where constraints (4.42)–(4.45) are replaced by their continuous relaxations. After solving this RMP, we search for new pattern variables that price out favorably in the pricing problem. If at least one such column is found, it is added to RMP which in turn is resolved. This process is repeated until no further columns can be added.

Let $\mu_i \leq 0$, $\forall i \in F$, be the dual variables associated to constraints (4.37), $\pi_k \geq 0$, $\forall k \in C$, the dual variables associated to constraints (4.38), and $\lambda_{k,W} \geq 0$, $\forall k \in C$, $\forall W \subsetneq V \mid r \in W$, the dual variables associated to the customers connection constraints (4.41). Let $W(i,k) = \{W \subseteq V \mid r, i \in W\}$, $\forall i \in F$, $\forall k \in C_i$, denote the set of all subsets of V including the root node and at least one facility to which a customer k can be assigned.

When solving RMP, we obtain optimal dual variable values μ_i^*, π_k^*, and $\lambda_{k,W}^*$, defining reduced costs $\bar{c}_\omega$ for variables $\omega \in \Omega \setminus \tilde{\Omega}$:

$$\bar{c}_\omega = \sum_{k \in \mathcal{C}(\omega)} a_{\mathcal{F}(\omega),k} - \mu_{\mathcal{F}(\omega)} - \sum_{k \in \mathcal{C}(\omega)} \pi_k - \sum_{k \in \mathcal{C}(\omega)} \sum_{Q \in W(\mathcal{F}(\omega),k)} \lambda_{k,Q} \tag{4.46}$$

$$= -\mu_{\mathcal{F}(\omega)} - \sum_{k \in \mathcal{C}(\omega)} \left(\pi_k - a_{\mathcal{F}(\omega),k} + \sum_{Q \in W(\mathcal{F}(\omega),k)} \lambda_{k,Q} \right). \tag{4.47}$$

The pricing problem is to find a pattern $\omega^* \in \Omega \setminus \tilde{\Omega}$ yielding minimum reduced costs, i.e.

$$\omega^* = \operatorname{argmin}_{\omega \in \Omega \setminus \tilde{\Omega}} \{\bar{c}_\omega\}.$$

In other words, we need to find a feasible assignment ω between some customers $\mathcal{C}(\omega)$ and a facility $\mathcal{F}(\omega)$ yielding negative reduced costs $\bar{c}_\omega$ or prove that no such assignment exists.

Thus, we need to solve a binary knapsack problem for each facility $i \in F$, with one item for each customer $k \in C_i$ assignable to i, demand d_k, and profit $\pi_k - a_{i,k} + \sum_{Q \in W(i,k)} \lambda_{k,Q}$, where we obviously need not consider items with negative or zero profit. The total capacity of the knapsack is D_i. If $|\mu_i|$ is smaller than the total profit of the optimal solution to such a knapsack problem, the corresponding pattern variable has negative reduced costs, in which case it is added to RMP.

4.6.1 Branching in Branch-and-Price

Branching on the exponentially large set of variables γ_ω, $\forall \omega \in \Omega$, is not a viable option since it would lead to strong asymmetries in the partitioning of the search space. Thus next to variables z_i, $\forall i \in F$, variables $x_{u,v}$, $\forall (u,v) \in A$, and variables y_k, $\forall k \in C$, we accomplish branching by decisions on assignments between facilities and customers. Integrality on one such assignment between a facility $i \in F$ and a customer $k \in C_i$ can be achieved by adding either branching constraint (4.48) or (4.49) to the model if $\sum_{\omega \in \tilde{\Omega}(k) \cap \tilde{\Omega}_i} \gamma_\omega$ is fractional.

$$\sum_{\omega \in \tilde{\Omega}(k) \cap \tilde{\Omega}_i} \gamma_\omega = 0 \tag{4.48}$$

$$\sum_{\omega \in \tilde{\Omega}(k) \cap \tilde{\Omega}_i} \gamma_\omega = 1 \tag{4.49}$$

For each included branching constraint, we need to consider its dual variable value in the pricing problem when solving a knapsack problem with an item corresponding to an assignment fixed due to an already included branching constraint. Adding such additional terms in the pricing problem eventually modifies an item's profit but does not affect the structure of the pricing problem, i.e. the approach is robust.

Lemma 15 proves that any solution S' to the LP relaxation of (dBCP) (denoted by $(\text{dBCP})^{\text{LP}}$) for which – according to above mentioned branching rules – no further branching can be accomplished represents a feasible solution to CConFL, i.e. eventually existing pattern variables with fractional values can be replaced by pattern variables with integral values while maintaining all assignments between facilities and customers.

Lemma 15 *Consider a solution S' to $(\text{dBCP})^{\text{LP}}$ and an arbitrary facility $i \in F$. Let $\Omega' = \{\omega \in \tilde{\Omega}_i \mid \gamma_\omega^{\text{LP}} \neq 0\}$ denote the set of active patterns for i in S', and $C' = \{k \in C \mid \exists \omega \in \Omega'(k)\}$ denote the set of customers assigned to i in S'. Furthermore, assume that $\sum_{\omega \in \Omega'(k)} \gamma_\omega = 1$, $\forall k \in C'$. Then $\zeta \in \Omega_i$ exists such that $C' = \mathcal{C}(\zeta)$.*

Proof Let $\zeta \in \Omega_i$ denote the single variable replacing all variables $\omega \in \Omega'$, i.e. $\mathcal{C}(\zeta) = C'$. Due to the implicit integrality of each assignment between i and a customer $k \in C'$ we only need to prove that ζ does not violate the capacity constraints. Due to constraints (4.37) the following inequality holds:

$$D_\zeta = \sum_{k \in C'} d_k = \sum_{\omega \in \Omega'} \gamma_\omega^{\mathrm{LP}} \sum_{k \in \mathcal{C}(\omega)} d_k \leq \sum_{\omega \in \Omega'} \gamma_\omega^{\mathrm{LP}} D_i = D_i \sum_{\omega \in \Omega'} \gamma_\omega^{\mathrm{LP}} \leq D_i.$$

4.7 Polyhedral Comparison

In the following, we compare the polyhedra corresponding to the sets of feasible solutions of the LP relaxations to the four models presented in the last sections. Hereby, we denote by $\mathcal{P}_{\mathrm{dMCF_f}}$ the polyhedron corresponding to the set of feasible solutions of the linear relaxation of model ($\mathrm{dMCF_f}$). Similarly, $\mathcal{P}_{\mathrm{dMCF_c}}$ denotes the polyhedron induced by the LP relaxation of model ($\mathrm{dMCF_c}$), $\mathcal{P}_{\mathrm{dCut}}$ those of model (dCut), and $\mathcal{P}_{\mathrm{dBCP}}$ the polyhedron corresponding to the LP relaxation of model (dBCP). Furthermore, superscript LP denotes the linear programming relaxation of a model, e.g. $(\mathrm{dMCF_f})^{\mathrm{LP}}$ denotes the LP relaxation of model ($\mathrm{dMCF_f}$).

Lemma 16 ($\mathrm{dMCF_f}$) *does not dominate* ($\mathrm{dMCF_c}$), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}})$.

Proof Consider a fractional solution $S' = (R'_S, T'_S, F'_S, C'_S, \alpha'_S)$ corresponding to the example given in Figure 4.4. S' can be feasibly described in the LP relaxation of our facility oriented model using the variable values as indicated in the figure, i.e. $S' \in (\mathrm{dMCF_f})^{\mathrm{LP}}$. Here, the corresponding flow to each facility with value $\frac{1}{3}$ is routed over two disjoint paths. For feasible solutions of model $(\mathrm{dMCF_c})^{\mathrm{LP}}$, $\sum_{(r,u) \in A_k} s^k_{r,u} \geq y_k$, $\forall k \in C$, must hold due to the flow conservation constraints. Since $\sum_{(r,u) \in A_k} s^k_{r,u} \leq \sum_{(r,u) \in E} x_{r,u} = \frac{1}{3}$, but $y_k = 1$, $\forall k \in \{0,1\}$, we conclude that $S' \notin (\mathrm{dMCF_c})^{\mathrm{LP}}$.

Lemma 17 ($\mathrm{dMCF_c}$) *does not dominate* ($\mathrm{dMCF_f}$), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}})$.

Proof We consider a fractional solution $S'' = (R''_S, T''_S, F''_S, C''_S, \alpha''_S)$ corresponding to Figure 4.5. Since the capacity constraints as well as all linking constraints are met and the corresponding flow to each of the two customer is routed over two disjoint paths, where each fractional value $s^k_{u,v}$ is set to $\frac{1}{2}$, $S'' \in (\mathrm{dMCF_c})^{\mathrm{LP}}$. For feasible solutions of model $(\mathrm{dMCF_f})^{\mathrm{LP}}$, $\sum_{(u,i) \in A_i} s^i_{u,i} \geq z_i$, $\forall i \in F$, must hold due to the flow conservation constraints. Since $\sum_{(u,i) \in A_i} s^i_{u,i} \leq \sum_{(u,i) \in E} x_{u,v} = \frac{1}{2}$ but $z_i = 1$, $\forall i \in \{0,1\}$, we conclude that $S'' \notin (\mathrm{dMCF_f})^{\mathrm{LP}}$.

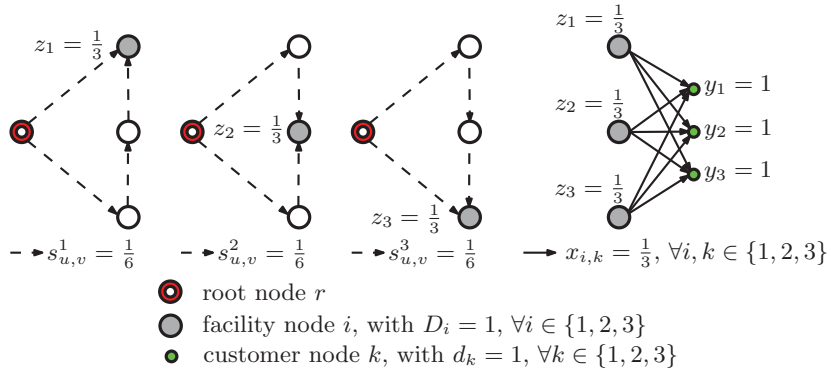


Figure 4.4: Feasible LP solution of ($\mathrm{dMCF_f}$) which is infeasible for ($\mathrm{dMCF_c}$).

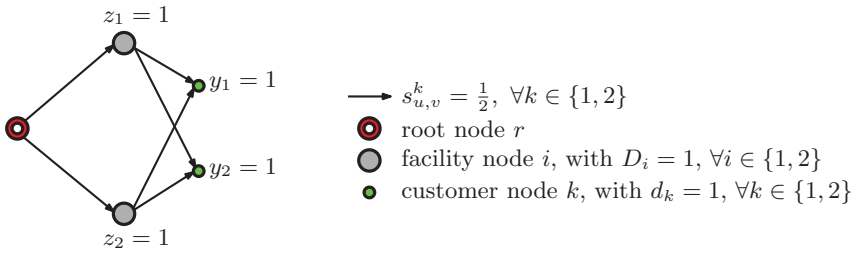


Figure 4.5: Feasible LP solution of ($\mathrm{dMCF_c}$) which is infeasible for ($\mathrm{dMCF_f}$).

Theorem 18 *None of the formulations* ($\mathrm{dMCF_c}$) *and* ($\mathrm{dMCF_f}$) *dominates the other, i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}})$ *and* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}})$.

Proof Theorem 18 immediately follows due to Lemmas 16 and 17.

Lemma 19 (dCut) *dominates* ($\mathrm{dMCF_f}$), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dCut}}) \subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}})$.

Proof ($\mathrm{dMCF_f}$) differs from (dCut) by modeling connections to facilities by multi-commodity flow constraints instead of directed connection cuts (4.31) whereas (dCut) additionally contains directed connection cuts for customers (4.32). The max-flow min-cut theorem [58] implies that for an arbitrary facility $i \in F$ with $\sum_{(u,v)\in\delta^+(W)} x^{\mathrm{LP}}_{u,v} \geq z^{\mathrm{LP}}_i$, $\forall W \subsetneq V \mid r \in W \wedge i \notin W$, a feasible flow of value z^{LP}_i from the root node to i exists; compare [132]. Thus any solution to $(\mathrm{dCut})^{\mathrm{LP}}$ is valid for $(\mathrm{dMCF_f})^{\mathrm{LP}}$.

Lemma 20 (dCut) *dominates* ($\mathrm{dMCF_c}$), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dCut}}) \subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}})$.

Proof ($\mathrm{dMCF_c}$) differs from (dCut) by modeling connections to customers by multi-commodity flow constraints instead of directed connection cuts (4.32) whereas (dCut) additionally contains directed connection cuts for facilities. Thus, as for Lemma 19 the max-flow min-cut argument also holds for the flow to customers.

Theorem 21 (dCut) *strictly dominates* ($\mathrm{dMCF_f}$) *and* ($\mathrm{dMCF_c}$), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dCut}}) \subsetneq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}})$ *and* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dCut}}) \subsetneq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}})$.

Proof Since none of the multi-commodity flow formulations dominates the other, i.e. $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}})$ and $proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_f}}) \not\subseteq proj_{x,y,z}(\mathcal{P}_{\mathrm{dMCF_c}})$, Theorem 21 follows from Lemmas 19 and 20.

Theorem 22 (dBCP) *strictly dominates* (dCut), *i.e.* $proj_{x,y,z}(\mathcal{P}_{\mathrm{dBCP}}) \subsetneq proj_{x,y,z}(\mathcal{P}_{\mathrm{dCut}})$.

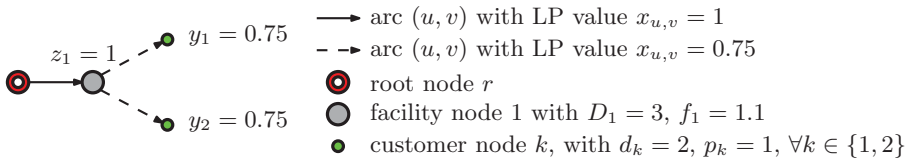


Figure 4.6: Feasible LP solution of (dCut) which is infeasible for (dBCP).

Proof Consider a fractional solution S' according to the example given in Figure 4.6 assuming zero costs for all included arcs. As can be easily seen S' is valid for $(\text{dCut})^{\text{LP}}$. For describing S' in the space of (dBCP), each assignment pattern ω can only contain one of the customers. However, since those patterns do not pay off – i.e. the collected profit is smaller than the facilities' opening costs f_1 – $\omega \notin \Omega$ and thus $S' \notin (\text{dBCP})^{\text{LP}}$.

Now, we consider a solution $S^{\text{bcp}} \in (\text{dBCP})^{\text{LP}}$ and denote by $\gamma_\omega^{\text{bcp}}$, $\forall \omega \in \Omega$, $x_{u,v}^{\text{bcp}}$, $\forall (u, v) \in A$, z_i^{bcp}, $\forall i \in F$, and y_k^{bcp}, $\forall k \in C$, the values of all variables of S^{bcp}. Using equations (4.50)–(4.53) we transform these values to the space of (dCut), where superscript cut denotes a value with respect to $(\text{dCut})^{\text{LP}}$ and S^{cut} the corresponding solution to $(\text{dCut})^{\text{LP}}$.

$$z_i^{\text{cut}} = z_i^{\text{bcp}} \qquad \forall i \in F \tag{4.50}$$

$$y_k^{\text{cut}} = y_k^{\text{bcp}} \qquad \forall k \in C \tag{4.51}$$

$$x_{u,v}^{\text{cut}} = x_{u,v}^{\text{bcp}} \qquad \forall (u, v) \in A \tag{4.52}$$

$$x_{i,k}^{\text{cut}} = \sum_{\omega \in \Omega_i \cap \Omega(k)} \gamma_\omega^{\text{bcp}} \qquad \forall i \in F,\ \forall k \in C_i \tag{4.53}$$

To show that $S^{\text{cut}} \in (\text{dCut})^{\text{LP}}$ and thus $(\text{dBCP})^{\text{LP}} \subseteq (\text{dCut})^{\text{LP}}$ we consider each set of constraints from (dCut) in turn. S^{cut} obviously does not violate constraints (4.27), since (4.39) identically models them in (dBCP). Validity of constraints (4.28) follows from above mentioned transformation rules and constraints (4.37):

$$x_{i,k}^{\text{cut}} = \sum_{\omega \in \Omega_i \cap \Omega(k)} \gamma_\omega^{\text{bcp}} \leq \sum_{\omega \in \Omega_i} \gamma_\omega^{\text{bcp}} \leq z_i^{\text{bcp}} = z_i^{\text{cut}}.$$

Using our transformation rules and constraints (4.38) the following inequality ensures that S^{cut} does not violate constraints (4.29):

$$y_k^{\text{cut}} = y_k^{\text{bcp}} \leq \sum_{\omega \in \Omega(k)} \gamma_\omega^{\text{bcp}} = \sum_{i \in F_k} \sum_{\omega \in \Omega_i \cap \Omega(k)} \gamma_\omega^{\text{bcp}} = \sum_{i \in F_k} x_{i,k}^{\text{cut}}.$$

Using constraints (4.37) and the fact that the total demand of a single pattern $\omega \in \Omega_i$ does not exceed the maximum assignable demand of its facility $i \in F$, the validity of the capacity constraints (4.30) is ensured as follows:

$$\begin{aligned}\sum_{k \in C_i} d_k x_{i,k}^{\text{cut}} = \sum_{k \in C_i} d_k \sum_{\omega \in \Omega_i \cap \Omega(k)} \gamma_\omega^{\text{bcp}} = \sum_{\omega \in \Omega_i} \gamma_\omega^{\text{bcp}} \sum_{k \in \mathcal{C}(\omega)} d_k \leq \\ \leq \sum_{\omega \in \Omega_i} \gamma_\omega^{\text{bcp}} D_i \leq D_i z_i^{\text{bcp}} = D_i z_i^{\text{cut}}.\end{aligned}$$

Since directed connection cuts for facilities are identically included in both formulations and the validity of customer connection cuts (4.32) immediately follows by substituting $\sum_{\omega \in \Omega_i \cap \Omega(k)} \gamma_\omega$ by $x_{i,k}$ in the customer connection cuts (4.41) of (dBCP), we conclude that $S^{\text{cut}} \in (\text{dCut})^{\text{LP}}$.

4.8 Lagrangian Decomposition

Since Lagrangian relaxation based approaches have proven to be quite successful for the Steiner tree problem [11] as well as for the capacitated facility location problem [93] and CConFL is composed of these two problems it is quite natural to decompose CConFL by means of Lagrangian relaxation. Model (4.54)–(4.62) which we will relax in the following, is a more abstractly written, undirected variant of model (dMCF_c). As previously, binary variables x_e, $\forall e \in E'$, indicate if an edge e is part of the solution, variables $z_i \in \{0,1\}$, $\forall i \in F$, specify if a facility i is opened, and variables $y_k \in \{0,1\}$, $\forall k \in C$, if a customer k is feasibly assigned to an open facility. Similarly to the flow variables of model (dMCF_c), we use variables $s_e^k \in \{0,1\}$, $\forall k \in C$, $\forall e \in E'$, to indicate if an edge $e \in E'$ is part of the unique path from the root node r to a connected customer k. Finally $P_k \in \{0,1\}^{|E'|}$ denotes the set of incidence vectors corresponding to these simple paths from r to $k \in C$ using exactly one assignment edge $(i,k) \in E' \setminus E$.

$$\min \quad \sum_{e\in E'} c'_e x_e + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k(1-y_k) \tag{4.54}$$

$$\text{s.t.} \quad s^k_e \le x_e \qquad \forall k \in C,\ \forall e \in E' \tag{4.55}$$

$$s^k \in P_k \text{ if } y_k = 1 \qquad \forall k \in C \tag{4.56}$$

$$x_{i,k} \le z_i \qquad \forall i \in F,\ \forall k \in C_i \tag{4.57}$$

$$\sum_{k\in C_i} d_k x_{i,k} \le D_i z_i \qquad \forall i \in F \tag{4.58}$$

$$s^k_e \in \{0,1\} \qquad \forall k \in C,\ \forall e \in E' \tag{4.59}$$

$$x_e \in \{0,1\} \qquad \forall e \in E' \tag{4.60}$$

$$z_i \in \{0,1\} \qquad \forall i \in F \tag{4.61}$$

$$y_k \in \{0,1\} \qquad \forall k \in C \tag{4.62}$$

Note that we use x_e in the objective function (4.54) as well as in inequalities (4.55) and (4.60) when considering graph as well as assignment edges, while $x_{i,k}$ is used to denote assignment edges only in inequalities (4.57) and (4.58).

We relax inequalities (4.55) linking variables s and x in a classical Lagrangian fashion by adding corresponding terms weighted with nonnegative Lagrangian multipliers $\pi_{k,e} \ge 0$, $\forall k \in C,\ \forall e \in E'$, to the objective function. This yields the parameterized model (LD(π)). See for example [18] for a general introduction to Lagrangian relaxation.

$$\begin{aligned}(\mathrm{LD}(\pi))\ \min & \sum_{e\in E'} c'_e x_e + \sum_{i\in F} f_i z_i + \sum_{k\in C} p_k(1-y_k) + \sum_{k\in C}\sum_{e\in E'} \pi_{k,e}\cdot(s^k_e - x_e) = \\ & = \sum_{k\in C} p_k + \sum_{k\in C}\left(\sum_{e\in E'} \pi_{k,e} s^k_e - p_k y_k\right) + \\ & + \sum_{e\in E'}\left(c'_e - \sum_{k\in C}\pi_{k,e}\right) x_e + \sum_{i\in F} f_i z_i \\ \text{s.t.} \quad & (4.56)\text{–}(4.62)\end{aligned}$$

(LD(π)) decomposes into independent subproblems (LD$_{s,y}(\pi)$) for determining variables s^k_e, $\forall k \in C$, $\forall e \in E'$, and y_k, $\forall k \in C$, subproblem (LD$_x(\pi)$) for determining variables x_e, $\forall e \in E$, and subproblem (LD$_{x,z}(\pi)$) to determine variables x_e,

$\forall e \in E' \setminus E$, and z_i, $\forall i \in F$. We consider these subproblems and their solving in the following in detail.

$$
\begin{aligned}
(\mathrm{LD}_{s,y}(\pi)) \quad \min \quad & \sum_{k \in C} p_k + \sum_{k \in C} \left(\sum_{e \in E'} \pi_{k,e} s_e^k - p_k y_k \right) && && (4.63)\\
\text{s.t.} \quad & s^k \in P_k \text{ if } y_k = 1 && \forall k \in C && (4.64)\\
& s_e^k \in \{0,1\} && \forall k \in C,\ \forall e \in E' && (4.65)\\
& y_k \in \{0,1\} && \forall k \in C && (4.66)
\end{aligned}
$$

$(\mathrm{LD}_{s,y}(\pi))$ consists of $|C|$ independent shortest path problems. Thus it can be solved for customer $k \in C$ by computing the cheapest path with respect to edge costs $\pi_{k,e}$ from the root to customer node k which includes exactly one assignment edge $(i,k) \in E' \setminus E$, i.e. we need to determine the corresponding incidence vector $q \in P_k$. If the total costs of this path are smaller than the customers prize p_k, y_k as well as the corresponding path variables s_e^k, $\forall e \in E' \mid q_e = 1$, are set to one. Since, all edge costs $\pi_{k,e}$ are nonnegative we use $|C|$ runs of Dijkstras' algorithm [50], resulting in a total time-complexity of $O(|C|(|E| + |V| \log |V|))$ for solving $(\mathrm{LD}_{s,y}(\pi))$ when using a binary heap implementation of Dijkstras' algorithm.

$$
\begin{aligned}
(\mathrm{LD}_{x}(\pi)) \quad \min \quad & \sum_{e \in E} \left(c_e - \sum_{k \in C} \pi_{k,e} \right) x_e && && (4.67)\\
\text{s.t.} \quad & x_e \in \{0,1\} && \forall e \in E && (4.68)
\end{aligned}
$$

$(\mathrm{LD}_{x}(\pi))$, can be trivially solved by inspection in time $O(|C||E|)$. Variables x_e, $\forall e \in E$, are set to one if $c_e < \sum_{k \in C} \pi_{k,e}$, and to zero otherwise.

$$
\begin{aligned}
(\mathrm{LD}_{x,z}(\pi)) \quad \min \quad & \sum_{i \in F} f_i z_i + \sum_{\substack{e=(i,k) \in E' \mid \\ i \in F \wedge k \in C_i}} \left(c'_{i,k} - \sum_{k \in C} \pi_{k,e} \right) x_{i,k} && && (4.69)\\
\text{s.t.} \quad & \sum_{k \in C_i} d_k x_{i,k} \leq D_i z_i && \forall i \in F && (4.70)\\
& x_{i,k} \leq z_i && \forall i \in F,\ \forall k \in C_i && (4.71)\\
& z_i \in \{0,1\} && \forall i \in F && (4.72)\\
& x_e \in \{0,1\} && \forall e \in E' \setminus E && (4.73)
\end{aligned}
$$

Model ($\mathrm{LD}_{x,z}(\pi)$) resembles $|F|$ 0–1 knapsack problems, one for each facility $i \in F$. In such a knapsack problem for facility $i \in F$, we are given the total knapsack capacity D_i, and one item for each potential assignment $e = (i,k) \in E' \setminus E$, with profit $\sum_{k \in C} \pi_{k,e} - c'_e$ and weight d_k. Obviously, we can neglect all items with negative or zero profit. Let χ_i^* denote the optimal solution to the knapsack problem of facility $i \in F$, and $o(\chi_i^*)$ the according objective value (i.e. the total profit). z_i and all variables x_e corresponding to items used in χ_i^* are set to one if $o(\chi_i^*) > f_i$. Although the knapsack problem is weakly $\mathcal{NP}$-hard [69], several algorithms capable of solving large instances relatively quickly are known, see e.g. [104, 147]. In our implementation we use the Combo algorithm[1] of Martello et al. [136]. Since ($\mathrm{LD}_{x,z}(\pi)$) does not possess the integrality property, we may be able to determine better lower bounds than by a simpler LP relaxation of model (4.54)–(4.62).

In the Lagrangian dual problem, we aim to maximize the resulting lower bound by determining optimal Lagrangian multipliers π^*. Since this maximization problem is convex and piecewise linear, we can approximately solve it using subgradient-like methods. We use the volume algorithm [13], which is an extension of the classic subgradient method [65], for solving the Lagrangian dual. While the volume algorithm has been reported to be more efficient in a number of other applications [11, 86], it sometimes might converge too quickly – see e.g. [29] – thus leading to poorer lower bounds than other subgradient-based algorithms. However, preliminary tests in our scenario indicated that the volume algorithm usually yields better lower bounds than the classic subgradient method. Therefore, even though a comparison of various existing methods for solving the Lagrangian dual would be interesting in future, we decided to focus on using the volume algorithm.

4.9 Primal Heuristic

Applying the volume algorithm [13] to approximately solve the Lagrangian dual problem, we compute integer values for variables s_e^k, x_e, z_i, and y_k in each iteration. The solution to ($\mathrm{LD}_{s,y}(\pi)$) does connect a subset of customers with the root node, but the subgraph induced by these paths might contain redundant edges or violate capacity constraints. The solution to ($\mathrm{LD}_{x,z}(\pi)$), however, does open some facilities and assigns customers to them respecting the capacity constraints, but does not take into account whether those facilities are connected to the root node. Furthermore, customers may be assigned to multiple facilities due to ($\mathrm{LD}_{x,z}(\pi)$).

[1] http://www.diku.dk/~pisinger/codes.html

To create a feasible solution $S = (R_S, T_S, F_S, C_S, \alpha_S)$ using the solutions to $(\mathrm{LD}_{s,y}(\pi))$ and $(\mathrm{LD}_{x,z}(\pi))$ we apply the Lagrangian heuristic (LH) presented in Algorithm 4.1.

Algorithm 4.1 initially declares all facilities as open whose corresponding nodes are part of a path to some customer $k \in C$ due to the actual solution to $(\mathrm{LD}_{s,y}(\pi))$, i.e. $F_S = \{i \in F \mid \exists k \in C : s^k_{i,k} = 1\}$.

In a second phase the Steiner tree (R_S, F_S) connecting these facilities $i \in F_S$ is created. Let $W_{i,k} = \{e \in E \mid s^k_e = 1\}$, $\forall k \in C'_i$, with $C'_i = \{k \in C \mid s^k_{i,k} = 1\}$ be the set of customers connected to the root node r via facility i, and $W_i = \mathrm{argmin}_{W_{i,k} | k \in C'_i}\{\sum_{e \in W_{i,k}} c_e\}$ be the cheapest of these subpaths for each open facility $i \in F_S$. After initializing the Steiner tree to consist of the root node only – i.e. $R_S = \{r\}$, $T_S = \emptyset$ – all facilities $i \in F_S$ are considered in increasing order w.r.t. the costs $\sum_{e \in W_i} c_e$ of the cheapest path W_i connecting them. We connect each considered facility $i \in F$ to the so far constructed Steiner tree by adding the necessary subpath $W' \subseteq W_i$ with $W' = \{(v_0 = i, v_1), (v_1, v_2), \ldots (v_l, v_m)\}$, $(v_a, v_b) \in W_i$, $0 \leq a, b \leq m$, $v_i \notin R_S$, $0 \leq i \leq l$, $v_m \in R_S$, to the Steiner tree, i.e. $R_S = R_S \cup \{v_0, v_1 \ldots, v_l\}$, and $T_S = T_S \cup W'$.

After connecting facility $i \in F_S$ the optimal subset of customers $C''_i \subseteq C'_i$ which are connected by paths via i is assigned to facility i. If assigning all these customers C'_i would exceed the maximum demand D_i assignable to i, we use the Combo algorithm [136] to solve the corresponding 0–1 knapsack problem, while simply all customers $k \in C'_i$ are assigned to i if $\sum_{k \in C'_i} d_k \leq D_i$.

In the third phase of Algorithm 4.1 the so far created solution is further improved by assigning additional customers. Here, we first consider the set of assignments $\mathcal{A}$ between customers and open facilities $i \in F_S$ from the solution to $(\mathrm{LD}_{x,z}(\pi))$, i.e. $\mathcal{A} = \{(i,k) \mid i \in F_S \wedge k \in C \wedge x_{i,k} = 1\}$, in decreasing order w.r.t. their efficiency values $\frac{p_k - c'_{i,k}}{d_k}$. Each considered assignment (i,k) is added to S if the corresponding customer has not yet been assigned, i.e. $k \notin C_S$, and the facility's capacity constraint will not be exceeded, i.e. $d_k + \sum_{k' \in C_S | \alpha_S(k') = i} d_{k'} \leq D_i$. Subsequently, further assignments are added to S using an identical greedy strategy for all remaining possible assignments to facilities $i \in F_S$.

Finally, we further improve the obtained solution S using the neighborhood structures described in Section 4.10 in case S is better than the so far best solution S' derived by LH before applying these improvements.

Algorithm 4.1: Primal Heuristic (Solution S', variable values s_e^k, x_e, z_i, y_k)

// Phase 1: open facilities
$F_S = \{i \in F \mid \exists k \in C : s_{i,k}^k = 1\}$
// Phase 2: construct Steiner tree (R_S, T_S) and assign initial customers
$R_S = \{r\}$
$T_S = \emptyset$
forall the $i \in F_S$ **do**
  $C_i' = \{k \in C \mid s_{i,k}^k = 1\}$
  $W_{i,k} = \{e \in E \mid s_e^k = 1\}, \forall k \in C_i'$
  $W_i = \text{argmin}_{W_{i,k} \mid k \in C_i'} \{\sum_{e \in W_{i,k}} c_e\}$
forall the $i \in F_S$ in increasing order of $\sum_{e \in W_i} c_e$ **do**
  if $\sum_{k \in C_i'} d_k \leq D_i$ **then**
    $C_i'' = C_i'$
  else
    determine optimal assignable subset $C_i'' \subseteq C_i'$ using Combo algorithm
  $C_S = C_S \cup C_i''$
  $\alpha_S(k) = i, \forall k \in C_i''$
// Phase 3: assign additional customers
$\mathcal{A} = \{(i,k) \mid i \in F_S \wedge k \in C \setminus C_S \wedge x_{i,k} = 1\}$
forall the $(i,k) \in \mathcal{A}$ in decreasing order w.r.t. efficiency $\frac{p_k - c'_{i,k}}{d_k}$ **do**
  if $k \notin C_S \wedge d_k + \sum_{k' \in C_S \mid \alpha_S(k') = i} d_{k'} \leq D_i$ **then**
    $C_S = C_S \cup \{k\}$
    $\alpha_S(k) = i$
$\mathcal{A}' = \{(i,k) \mid i \in F_S \wedge k \in C \setminus C_S \wedge x_{i,k} = 0\}$
forall the $(i,k) \in \mathcal{A}'$ in decreasing order w.r.t. efficiency $\frac{p_k - c'_{i,k}}{d_k}$ **do**
  if $k \notin C_S \wedge d_k + \sum_{k' \in C_S \mid \alpha_S(k') = i} d_{k'} \leq D_i$ **then**
    $C_S = C_S \cup \{k\}$
    $\alpha_S(k) = i$
// Phase 4: primal improvement
if $c(S) \leq c(S')$ **then**
  $S' = S$
  Primal Improvement(S) *// see Algorithm 4.2*

4.10 Solution Improvement

Representing solutions by means of open facilities and computing the Steiner tree connecting them as well as assigning customers to them during the solution decoding process has been the usual approach taken in metaheuristics for variants of ConFL so far [117, 125, 169]. In our case, modifying the set of open facilities is quite expensive w.r.t. computational time, since determining the optimal connecting Steiner tree as well as assigning the optimal clients are $\mathcal{NP}$-hard problems. Using some heuristic for decoding a solution after adapting the set of open facilities and subsequently trying to improve those aspects is an option for a pure metaheuristic but is likely to be also too time consuming in case of an intertwined hybrid approach in which the primal improvement procedure is repeatedly applied to solutions derived within the course of the volume algorithm.

We therefore decided to concentrate on improving a solution by means of its Steiner tree and its assigned customers, but do not modify the set of open facilities generated by our Lagrangian heuristic. Diversity by means of open facilities is ensured in our approach due to the fact that we generate one initial solution in each iteration of the volume algorithm. As shown by Algorithm 4.2, we use one neighborhood structure for each of the remaining solution aspects: a path exchange neighborhood – see Section 4.10.1 – for reducing the costs of the connecting Steiner tree and either a simple swap neighborhood – see Section 4.10.2 – or a very large scale neighborhood – see Section 4.10.3 – for improving facility customer assignments. Both neighborhoods are searched using a best improvement strategy. Finally, we remove non-profitable parts from S using strong pruning as described in [140].

Algorithm 4.2: Primal Improvement(Solution S)

```
Key Path Improve(S) // see Algorithm 4.3
switch improvement mode do
    case simple:
        Customer Swap Improve(S) // see Algorithm 4.4
    case advanced:
        Very Large Scale Neighborhood Search(S) // see Algorithm 4.5
prune solution
```

It is further worth mentioning that since the improved solution aspects are independent one could easily apply the corresponding neighborhoods in parallel instead of our sequential approach to reduce the total runtime.

4.10.1 Key Path Improvement

For the Steiner tree problem in graphs, the concept of so-called *key nodes* – also called *crucial nodes* – of a solution, which are all customer nodes as well as all Steiner nodes of degree greater than or equal to three is well known. Voß [175] was the first who considered representing a solution to STP by its key nodes – although he did not yet use the term key nodes – and trying to improve it by means of replacing the paths between those key nodes. Since then this type of neighborhood structure has been successfully used in several approaches for the STP – see e.g. [137, 174] – as well as some of its generalizations, see e.g. [116].

For a solution S to CConFL the set of key nodes $\mathcal{K} = \{r\} \cup F_S \cup \{v \in R_S \mid \deg_S(v) \geq 3\}$ is given by the root node, all open facilities as well as all Steiner nodes of degree greater than or equal to three in S. A *key path* $(\mathcal{V}, \mathcal{E})$ of solution S is a non-empty path in S between two key nodes $u, v \in \mathcal{K}$ containing no other key node, i.e. $\mathcal{V} \cap \mathcal{K} = \{u, v\}$. Our *Key-Path Improvement* procedure as given in Algorithm 4.3 considers each such key path $(\mathcal{V}, \mathcal{E}) \in \tilde{P}(S)$ from the set of all key paths $\tilde{P}(S)$ of solution S and replaces it by the shortest connection between its end nodes using the remaining solution edges as infrastructure (i.e. zero edge costs are assumed for them); see Figure 4.7 for an exemplary move.

Algorithm 4.3: Key Path Improvement (Solution S)

repeat
- $c'_e = \begin{cases} 0 & \text{if } e \in T_S \\ c_e & \text{else} \end{cases}, \forall e \in E$
- $\delta = 0$
- **forall the** key paths $\mathcal{P} = (\mathcal{V}, \mathcal{E}) \in \tilde{P}(S)$ **do**
  - *// key (end) nodes of $\mathcal{P}$ are u and v*
  - $c'_e = c_e, \forall e \in \mathcal{E}$
  - find shortest path $\mathcal{P}' = (\mathcal{V}', \mathcal{E}')$ between u and v w.r.t. c'
  - $\delta' = \sum_{e \in \mathcal{E}'} c'_e - \sum_{e \in \mathcal{E}} c_e$
  - **if** $\delta' < \delta$ **then**
    - $\delta = \delta'$
    - store replacement of $\mathcal{P}$ by $\mathcal{P}'$ as best move
  - $c'_e = 0, \forall e \in \mathcal{E}$
- **if** $\delta < 0$ **then**
  - apply best move

until $\delta \geq 0$

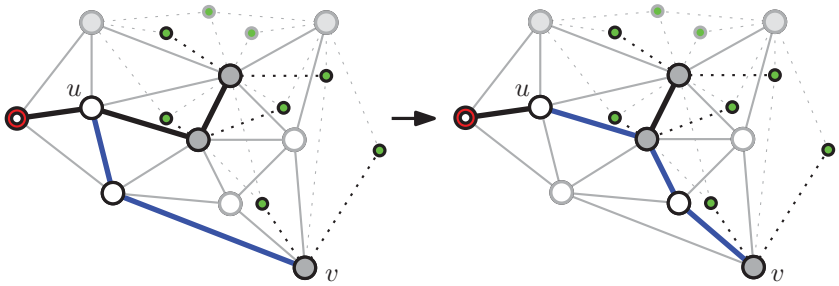


Figure 4.7: An exemplary key path exchange move between key nodes u and v.

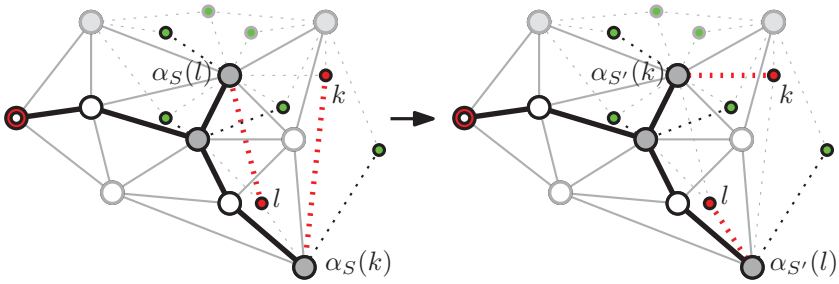


Figure 4.8: An exemplary move swapping the assignments of customers k and l.

4.10.2 Customer Swap Neighborhood

The *Customer Swap Neighborhood* focuses on realized assignments between facilities and customers. It consists of all solutions S' differing from a solution S by swapping the assignment of exactly two customer nodes. Formally, each swap move transforms a solution S with $\alpha_S(k) = i$ and $\alpha_S(l) = j$ for customers $k, l \in C_S$ and facilities $i, j \in F_S$, into a solution S' where $\alpha_{S'}(k) = j$ and $\alpha_{S'}(l) = i$; see Figure 4.8 for an exemplary move. This customer swap neighborhood can be searched in $O(|C_S|^2)$ by Algorithm 4.4. It has already been used by Contreras et al. [41] for the SSCFLP.

4.10.3 Very Large Scale Neighborhood Search

Small neighborhoods as the customer swap neighborhood described above can be searched relatively fast but often yield rather poor local optima only. Recently, *very large scale neighborhood* (VLSN) search approaches have been considered for various problems to overcome limitations of simple standard neighborhood structures. If

Algorithm 4.4: Customer Swap (Solution S)

repeat
  $\delta = 0$
  $r_i = D_i - \sum_{j \in C' | \alpha_S(j) = i} d_j, \forall i \in F_S$
  forall the $l \in C_S$ **do**
    forall the $k \in C_S$ **do**
      if $k \in C_{\alpha_S(l)} \wedge l \in C_{\alpha_S(k)}$ **then**
        if $\alpha_S(l) \neq \alpha_S(k)$ **then**
          if $d_l \leq r_{\alpha_S(k)} + d_k \wedge d_k \leq r_{\alpha_S(l)} + d_l$ **then**
            $\delta' = -a_{\alpha_S(k),k} - a_{\alpha_S(l),l} + a_{\alpha_S(k),l} + a_{\alpha_S(l),k}$
            if $\delta' < \delta$ **then**
              $\delta = \delta'$
              store current move as best
  if $\delta < 0$ **then**
    apply best move
until $\delta \geq 0$

such large neighborhoods can be efficiently searched they often lead to superior solutions, since they allow for covering larger areas of a problem's search space; see e.g. [5, 34] for surveys on this topic.

Ahuja et al. [6] proposed very large scale neighborhoods for the single source capacitated facility location problem (SSCFLP) based on the exchange of an arbitrary number of customers and showed how to efficiently search them via shortest path calculations on a so-called improvement graph. Since CConFL contains a special variant of SSCFLP where some customers may be unassigned, in the following we generalize their work on *single-customer multi-exchanges* to be applicable to our problem variant.

To formally introduce single-customer cyclic and path exchanges, we define the remaining capacity of each facility $i \in F$ w.r.t. a solution S as

$$r_S(i) = \begin{cases} D_i - \sum_{k \in C_S | \alpha_S(k) = i} d_k & \text{if } i \in F_S \\ D_i & \text{otherwise} \end{cases}, \ \forall i \in F.$$

Furthermore, by $\mathcal{F}(k) \in F_S$, $\forall k \in C_S$, we denote the facility $i \in F_S$ customer k is assigned to in S.

Analogously to Ahuja et al. [6], we define a *single-customer cyclic exchange* w.r.t. solution S as a sequence $R = (k_1, k_2, \ldots, k_q)$, $k_i \neq k_j \in C$, $1 \leq i \neq j \leq q$, such that each pair of currently assigned customers $k, t \in C_S$, $k \neq t$, from R is assigned to different facilities, i.e. $\mathcal{F}(k) \neq \mathcal{F}(t)$. Furthermore, no two consecutive customers of R may be currently unassigned, i.e. $k_i \in C_S \vee k_{i+1} \in C_S$, $i = 1, \ldots, q-1$, and $k_1 \in C_S \vee k_q \in C_S$.

Each such sequence R defines a move from an actual solution S to a solution S' by releasing each assigned customer $k_i \in C_S$ from its facility $\mathcal{F}(k_i)$, $1 \leq i \leq q$, and subsequently assigning k_i to the facility of its successor k_{i+1} in case $k_{i+1} \in C_S$, $1 \leq i \leq q-1$. Finally, k_q is assigned to $\mathcal{F}(k_1)$ if $k_1 \in C_S$. A single-customer cyclic exchange is feasible if customers may be assigned to the corresponding facilities and no capacity condition is exceeded.

Similarly a *single-customer path exchange* w.r.t. a solution S is a sequence $P = (k_1, k_2, \ldots, k_{q-1}, w)$ of customers $k_i \in C$, $1 \leq i \leq q-1$, and one facility $w \in F$ as last element of the sequence with $w \neq \mathcal{F}(k_i) \neq \mathcal{F}(k_j)$, $k_i, k_j \in C_S$, $1 \leq i \neq j \leq q-1$. Thus, as for cyclic exchanges, each assigned customer $k_i \in C_S$, $i = 1, \ldots, q-1$, is released and customers k_j, $j = 1, \ldots, q-2$ are assigned to their successors' facilities $\mathcal{F}(k_{j+1})$ if $k_{j+1} \in C_S$. Finally, instead of interpreting the sequence as a cycle by eventually assigning the last customer to the first customer's original facility, k_{q-1} is simply assigned to w. As for cyclic exchanges, a path exchange is feasible, if all assignment rules as well as capacity constraints are respected.

Since applying a path exchange move might induce opening a facility and/or closing one, we also need to determine corresponding changes in the costs w.r.t. the Steiner tree in order to decide whether the corresponding move is actually improving solution S. Since computing the exact additional costs or savings would mean to re-compute a Steiner tree for each facility $k \in F$, we apply a faster shortest path heuristic that returns an upper bound for additional costs and a lower bound for savings, respectively. Thus, using these heuristic values $\zeta(i)$, $\forall i \in F$, we might miss some improving moves but can be sure that no non-improving moves are considered as improving. To determine, $\zeta(i)$, $\forall i \in F$, we compute the shortest path tree from r treating all solution edges as infrastructure, i.e. we use modified edge costs $c'_e = 0$, $\forall e \in T_S$ and $c'_e = c_e$, $\forall e \in E \setminus T_S$. For facilities $i \in F \setminus F_S$, $\zeta(i) = \sum_{e \in Q(i)} c'_e$, where $Q(i)$ denotes the edge set of the cheapest path from r to i w.r.t. edge costs c', is obviously an upper bound for the additional connection costs of facility i. Furthermore, for open facilities $i \in F_S$ we set $\zeta(i) = -\sum_{e \in Q(i) \setminus \left(\bigcup_{j \in F_S \setminus \{i\}} Q(j)\right)} c_e$, since we can obviously remove all edges $e \in Q(i) \setminus \left(\bigcup_{j \in F_S \setminus \{i\}} Q(j)\right)$ from a solution after closing facility i.

For SSCFLP, Ahuja et al. [6] showed that improving path and cyclic exchanges correspond to negative subset disjoint cycles in a correspondingly defined improvement graph. Thus, in the following we show how to maintain this correlation between cycles and improving moves for our problem variant, i.e. how to define the improvement graph.

Improvement Graph:

For each solution S to CConFL, we define the corresponding improvement graph $I(S) = (N(S), M(S))$. The node set $N(S) = N^{\mathrm{a}}(S) \cup N^{\mathrm{u}}(S) \cup N^{\mathrm{p}}(S) \cup \{\mathrm{o}\}$ is the disjoint union of *assigned regular nodes* $u_k \in N^{\mathrm{a}}(S)$, $\forall k \in C_S$, *unassigned regular nodes* $v_k \in N^{\mathrm{u}}(S)$, $\forall k \in C \setminus C_S$, *pseudo nodes* $w_i \in N^{\mathrm{p}}(S)$, $\forall i \in F$, and an origin node o. The origin node o and its adjacent arcs are included to model path exchanges by means of cycles in $I(S)$, see also [6].

The set of arcs $M(S)$ is the disjoint union of

- arcs $M^{(\mathrm{a,a})}(S)$ between assigned regular nodes,
- arcs $M^{(\mathrm{a,u})}(S)$ from assigned to unassigned regular nodes,
- arcs $M^{(\mathrm{u,a})}(S)$ from unassigned to assigned regular nodes,
- arcs $M^{(\mathrm{a,p})}(S)$ from assigned regular to pseudo nodes,
- arcs $M^{(\mathrm{u,p})}(S)$ from unassigned regular to pseudo nodes,
- arcs $M^{(\mathrm{p,o})}(S)$ from pseudo nodes to the origin,
- arcs $M^{(\mathrm{o,a})}(S)$ from the origin to assigned regular nodes, and
- arcs $M^{(\mathrm{o,u})}(S)$ from the origin to unassigned regular nodes.

Next, we will describe these arcs as well as their costs $\gamma_{i,j}$, $\forall (i,j) \in M(S)$, corresponding to the resulting changes of the objective value formally as well as w.r.t. their interpretation.

Arcs $(u_k, u_l) \in M^{(\mathrm{a,a})}(S)$ denote releasing customer $l \in C_S$ from $i = \mathcal{F}(l)$ and in turn assigning customer $k \in C_S$ to facility i, leading to arc costs $\gamma_{u_k,u_l} = a_{i,k} - a_{i,l}$. Since, we must ensure that k can be assigned to $\mathcal{F}(l)$ as well as that capacity constraints are respected, the corresponding arc set is defined as $M^{(\mathrm{a,a})}(S) = \{(u_k, u_l) \mid u_k, u_l \in N^{\mathrm{a}}(S) : \mathcal{F}(l) \in F_k \wedge \mathcal{F}(k) \neq \mathcal{F}(l) \wedge r_S(\mathcal{F}(l)) + d_l \geq d_k\}$. Each arc $(u_k, v_l) \in M^{(\mathrm{a,u})}(S) = \{(u_k, v_l) \mid u_k \in N^{\mathrm{a}}(S), v_l \in N^{\mathrm{u}}(S)\}$, with corresponding costs $\gamma_{u_k,v_l} = p_k$ from an assigned to an unassigned regular node, models releasing customer k. Arcs $(v_k, u_l) \in M^{(\mathrm{u,a})}(S) = \{(v_k, u_l) \mid v_k \in N^{\mathrm{u}}(S), u_l \in N^{\mathrm{a}}(S) : \mathcal{F}(l) \in F_k \wedge r_S(\mathcal{F}(l)) + d_l \geq d_k\}$ with costs $\gamma_{u_k,v_l} = a_{\mathcal{F}(l),k} - a_{\mathcal{F}(l),l} - p_k$ indicate releasing l from

$i = \mathcal{F}(l)$ and subsequently assigning the previously unassigned customer k to facility $i \in F_S$.

$M^{(\mathrm{a,p})}$ consists of one arc (u_k, w_i) from each each assigned regular node to each pseudo node if the corresponding customer k can be assigned to facility i, i.e. $M^{(\mathrm{a,p})}(S) = \{(u_k, w_i) \mid u_k \in N^{\mathrm{a}}(S), w_i \in N^{\mathrm{p}}(S) : i \neq \mathcal{F}(k) \wedge i \in F_k \wedge r_S(i) \geq d_k\}$. Since eventually occurring facility opening costs will be considered by arcs going out of w_i, costs $\gamma_{u_k,w_i} = a_{i,k}$ are given by the costs of assigning customer k to facility i. To allow for assigning currently unassigned customers $k \in F \setminus F_S$ to some facility $i \in F$ without previously releasing another customer from i, we include arcs $(v_k, w_i) \in M^{(\mathrm{u,p})}(S) = \{(v_k, w_i) \mid v_k \in N^{\mathrm{u}}(S), w_i \in N^{\mathrm{p}}(S) : i \in F_k \wedge r_S(i) \geq d_k\}$. As we additionally earn a customers prize here, arc $(v_k, w_i) \in M^{(\mathrm{u,p})}(S)$ has costs $\gamma_{v_k,w_i} = a_{i,k} - p_k$.

To model path exchanges as cycles in the graph, we further need to include arcs from each pseudo node to the origin and arcs from the origin to assigned as well as unassigned regular nodes. Arcs $M^{(\mathrm{p,o})}(S) = \{(w_i, \mathrm{o}) \mid w_i \in N^{\mathrm{p}}(S)\}$ model eventually occurring opening and connection costs of facility $i \in F$, i.e.

$$\gamma_{w_i,\mathrm{o}} = \begin{cases} 0 & \text{if } i \in F_S \\ f_i + \zeta_i & \text{otherwise} \end{cases}, \ \forall (w_i, \mathrm{o}) \in M^{(\mathrm{p,o})}.$$

Using an arc $(\mathrm{o}, u_k) \in M^{(\mathrm{o,a})}(S) = \{(\mathrm{o}, u_k) \mid u_k \in N^{\mathrm{a}}(S)\}$ from the origin node o to some assigned regular node u_k releases customer k from its facility, yielding arc costs

$$\gamma_{\mathrm{o},u_k} = \begin{cases} -a_{\mathcal{F}(k),k} & \text{if } \exists l \neq k \in C_S : \mathcal{F}(k) = \mathcal{F}(l) \\ -a_{\mathcal{F}(k),k} - f_{\mathcal{F}(k)} + \zeta_{\mathcal{F}(k)} & \text{otherwise} \end{cases}, \ \forall (\mathrm{o}, u_k) \in M^{(\mathrm{o,a})}.$$

Finally, arcs $(\mathrm{o}, v_k) \in M^{(\mathrm{o,u})}(S) = \{(\mathrm{o}, v_k) \mid v_k \in N^{\mathrm{u}}(S)\}$ from the origin to some unassigned regular node are included for allowing to assign a new customer without previously releasing another one. Consequently, these arcs have zero costs, i.e. $\gamma_{\mathrm{o},v_k} = 0$, $\forall (\mathrm{o}, v_k) \in M^{(\mathrm{o,u})}$.

Searching for improving moves:

Generalizing the definition given in [6] we call a directed cycle $(u_1, \ldots, u_q)$, $u_i \in N(S)$, $i = 1, \ldots, q$, of $I(S)$ subset disjoint, if each of its assigned regular nodes and pseudo nodes are associated with different facility locations. If the total edge costs of such a cycle are negative, it is called negative cost subset disjoint. Since only feasible

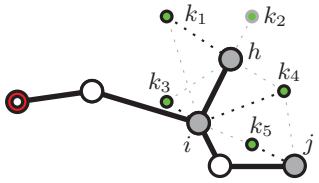


Figure 4.9: An exemplary solution S to CConFL.

arcs w.r.t. assignment rules and capacity conditions are included in $I(S)$, and edge costs reflect changes in the objective value, those negative cost subset disjoint cycles correspond to improving path and cyclic exchange moves. However, if such a cycle does induce opening facility $i \in F \setminus F_S$ as well as closing a facility $j \in F_S$, a cycle's cost might not be equal to the actual cost changes when applying the move since the additional costs/savings ζ due to adapting the Steiner tree have been computed independently for each facility. Since opening and connecting a new facility and assigning only one customer to it does only rarely pay off, this special case is rather unlikely to occur in practice. Therefore, we simply check whether a found cycle does simultaneously open and close two facilities and add eventually occurring additional connection costs before deciding whether this cycle is an improving one.

Thomson and Orlin [168] proved that deciding whether a graph contains a negative subset disjoint cycle is $\mathcal{NP}$-hard. Subsequently, Ahuja et al. [7] proposed an effective heuristic for finding negative cost subset disjoint cycles based on the label correcting algorithm for the shortest path problem. This heuristic has already been used for the SSCFLP [6] and in practice rarely fails to find existing negative cost subset disjoint cycles if started once from each regular node. As shown in Algorithm 4.5, we search the neighborhood defined by the set of single customer path and cyclic exchanges using a best improvement strategy, adopting the heuristic of Ahuja et al. [6] to find negative subset disjoint cycles which is also started from every regular node.

Figure 4.10 depicts an exemplary improvement graph $I(S) = (N(S), M(S))$ with respect to a solution S as shown in Figure 4.9 assuming that each clients demand is equal to one, while each facility's maximum assignable demand is two.

Figure 4.11 shows an exemplary feasible cyclic exchange $R = (k_1, k_4, k_5, k_2)$ with respect to solution S. Thus after applying R, customer k_2 will be assigned to facility h, k_1 to i, k_4 to j, and finally k_5 will be unassigned. Since $k_3 \notin R$ it will still be assigned to facility i.

An exemplary path exchange $P = (k_2, k_1, k_4, j)$ is shown in Figure 4.12. Here, k_2 will

Algorithm 4.5: Very Large Scale Neighborhood Search(Solution S)

repeat
- $\delta = 0$
- construct improvement graph
- **forall the** $k \in C$ **do**
  - *// start heuristic from regular node corresponding to k*
  - heuristically find negative cost subset disjoint cycle $\mathcal{C}$
  - $\delta = \sum_{(u,v)\in\mathcal{C}} \gamma_{u,v}$
  - **if** $\mathcal{C}$ induces closing facility $i \in F_S$ and opening $j \in F \setminus F_S$ **then**
    - $Q = Q(i) \setminus \left(\bigcup_{l\in F_S\setminus\{i\}} Q(l)\right)$
    - $\delta = \delta + \sum_{e\in Q\cap Q(j)} c_e$
  - **if** $\delta < \delta'$ **then**
    - $\delta = \delta'$
    - store current cycle as best move
- **if** $\delta < 0$ **then**
  - apply best move

until $\delta \geq 0$

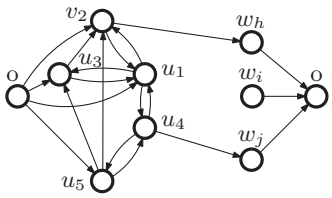


Figure 4.10: Improvement graph $I(S) = (N(S), M(S))$ corresponding to the solution given in Figure 4.9.

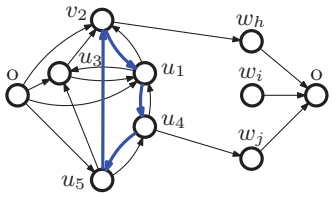


Figure 4.11: An exemplary cyclic exchange $R = (k_1, k_4, k_5, k_2)$.

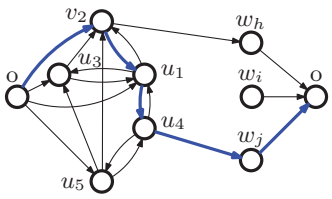


Figure 4.12: An exemplary path exchange $P = (k_2, k_1, k_4, j)$.

be assigned to facility h, k_1 to i, and k_4 to j after applying the corresponding move, while k_3 and k_5 will remain assigned to their facilities i and j since $k_3, k_5 \notin P$.

Note that the origin node o is duplicated in Figures 4.10, 4.11, and 4.12 to keep them simple.

4.11 Test Instances and Environment

For ConFL, Ljubić [125] combined benchmark instances for the STP with instances for uncapacitated facility location. Similarly, we created instances for CConFL[2] by combining STP instances from the OR-library[3] with instances for the SSCFLP created with the instance generator of Kratica et al. [112][4].

The node with index one in the STP instance is chosen as root node, while $|F|$ other nodes are randomly chosen as potential facility locations. Customers with associated demands, assignment costs as well the maximum assignable demands and opening costs for each facility are given by the SSCFLP instance. Next, we need to choose reasonable customer prizes, high enough to ensure that some customers will be supplied while avoiding creating relatively easy instances by setting them too high. For each customer $k \in C$, we randomly select its prize $p_k \in \mathbb{N}_0$ from the interval $\left[\overline{a}(k), a_{\max}(k) + \overline{f}\right]$, where $\overline{a}(k) = \frac{\sum_{i \in F_k} a_{i,k}}{|F_k|}$ denotes the average assignment costs of customer k, $a_{\max}(k) = \max_{i \in F_k}\{a_{i,k}\}$ the maximum assignment costs of customer k, and $\overline{f} = \frac{\sum_{i \in F} f_i}{|F|}$ the average facility opening costs. This ensures that each customer may be assigned to the majority of potential facilities in a profitable way. In particular it turned out that no customers or facilities are completely removed from an instance during preprocessing. Finally, degree-one and degree-two filtering [17] is

[2] available at http://www.ads.tuwien.ac.at/people/mleitner/cconfl/instances.tar.gz
[3] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html
[4] http://alas.matf.bg.ac.yu/~kratica/instances/splp_gen_w32.zip

applied to remove some Steiner nodes and edges. A more detailed description on the individual characteristics of all resulting instances is given in Tables 4.2 and 4.3.

We performed all computational experiments on a single core of an Intel Core 2 Quad with 2.83GHz. IBM CPLEX 12.1 [98] has been used for directly solving (dMCF_f), (dMCF_f), and (dCut) as well as their LP relaxations $(\text{dMCF}_\text{c})^{\text{LP}}$, $(\text{dMCF}_\text{f})^{\text{LP}}$, and $(\text{dCut})^{\text{LP}}$. SCIP 1.2.0 [2, 182] with IBM CPLEX 12.1 [98] as embedded LP solver has been used for solving (dBCP) and its linear relaxation $(\text{dBCP})^{\text{LP}}$.

To allow for a fair comparison to our Lagrangian decomposition based approaches, we used the single threaded variant of CPLEX. A time limit of 7200 CPU-seconds has been applied whenever not explicitly mentioned.

4.12 Computational Results

In the following the obtained computational results are summarized. First, results on the four exact IP models (dMCF_c), (dMCF_f), (dCut), and (dBCP) are given. Here, we start by evaluating their LP relaxation values and corresponding runtimes, before turning to obtained bounds and optimality gaps. Afterwards, the results of three variants of the LD approach from Section 4.8 are analyzed. Finally, the two most promising methods from these different fields are further discussed and compared to each other.

4.12.1 Results on Exact Models

When solving (dCut) and (dBCP), we separate directed connection cuts for customers only if no further violated connection cuts for facilities can be found. For (dBCP), we initially set $\tilde{\Omega} = \emptyset$ and accomplish branching by considering variables z_i, $\forall i \in F$, $x_{u,v}$, $\forall (u,v) \in A$, and y_k, $\forall k \in C$, in this order before considering assignments between facilities and customers. For each set of variables, branching is performed on the most fractional variable; ties are broken at random. We did not implement problem specific primal heuristics to speed-up the solution of our models, but simply trust on the built-in heuristics of CPLEX and SCIP, respectively.

Linear Programming Relaxations

Table 4.1 compares LP relaxation values of (dMCF_f) and (dMCF_c) for small test instances using a CPU-time limit of 14400 seconds. As shown in Section 4.7, none of

these two formulations theoretically dominates the other. However, as can be seen in Table 4.1, ($\mathrm{dMCF_f}$) is on the used test instances far more efficient than ($\mathrm{dMCF_c}$) from a computational perspective. Solving the LP relaxation of model ($\mathrm{dMCF_c}$) often needs too much time already on relatively small instances. Thus, we do not consider ($\mathrm{dMCF_c}$) in all further experiments.

Table 4.1: Comparison of LP relaxation values and corresponding CPU-times in seconds for ($\mathrm{dMCF_f}$) and ($\mathrm{dMCF_c}$) on small instances (time limit 14400s).

Instance					LP value		CPU time [s]	
Name	$\lvert F\rvert$	$\lvert C\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	$(\mathrm{dMCF_f})^{\mathrm{LP}}$	$(\mathrm{dMCF_c})^{\mathrm{LP}}$	$(\mathrm{dMCF_f})^{\mathrm{LP}}$	$(\mathrm{dMCF_c})^{\mathrm{LP}}$
c10-mo75	75	75	408	908	2878.7	2852.2	94	3272
c10-mq75	75	75	405	905	7095.2	7077.3	116	1386
c10-ms75	75	75	407	907	9506.3	9479.4	194	4487
d10-mo75	75	75	771	1770	2772.6	-	484	14400
d10-mq75	75	75	775	1774	7295.0	7278.9	167	3458
d10-ms75	75	75	781	1780	10069.3	-	1103	14400
c10-mo	75	200	404	904	8153.5	8118.2	713	6450
c10-mp	75	200	403	903	14917.4	-	228	14400
c10-mq	75	200	403	903	20717.2	-	328	14400
c10-mo	200	75	435	935	2957.0	-	6229	14400
c10-mp	200	75	428	928	5444.6	5432.0	3439	11206
c10-mq	200	75	430	930	8093.5	8076.6	1931	10748

Computational results for the LP relaxations of ($\mathrm{dMCF_f}$), (dCut), and (dBCP) are summarized in Table 4.2 for instances with $|F| = |C|$ and in Table 4.3 for instances with $|F| \neq |C|$. Tables 4.2 and 4.3 also detail the used test instances. For each considered instance, its number of potential facility locations $|F|$, its number of customers $|C|$, as well as the number of nodes $|V|$ and edges $|E|$ are given. All further tables will refer to an instance by its number only, which is given in the first column of Tables 4.2 and 4.3, respectively.

We conclude that in addition to their theoretical advantages and thus better LP relaxation values, the necessary CPU time to solve the linear relaxations of (dCut) and (dBCP) is significantly smaller for all tested instances. Moreover, $(\mathrm{dBCP})^{\mathrm{LP}}$ can be solved much faster than $(\mathrm{dCut})^{\mathrm{LP}}$ for almost all instances.

Solutions and Optimality Gaps

Results on best obtained lower and upper bounds as well as corresponding gaps and needed CPU times for ($\mathrm{dMCF_f}$), (dCut), and (dBCP) are presented in Table 4.4 for instances with $|F| = |C|$ and in Table 4.5 for instances with $|F| \neq |C|$. Since ($\mathrm{dMCF_f}$) could not solve any instance to proven optimality, we do not report its

Table 4.2: Comparison of LP relaxation values and corresponding CPU-times in seconds for ($dMCF_f$), (dCut), and (dBCP) (time limit 7200s) on instances with $|F| = |C|$. Best values are marked bold.

	Instance					LP value			CPU time [s]		
Nr	Name	$\|F\|$	$\|C\|$	$\|V\|$	$\|E\|$	$(dMCF_f)^{LP}$	$(dCut)^{LP}$	$(dBCP)^{LP}$	$(dMCF_f)^{LP}$	$(dCut)^{LP}$	$(dBCP)^{LP}$
1	c10-mo75	75	75	408	908	2878.7	2912.5	**2914.8**	94	7	**5**
2	c10-mq75	75	75	405	905	7095.2	7116.2	**7119.9**	116	5	**3**
3	c10-ms75	75	75	407	907	9506.3	9533.8	**9536.7**	194	11	**3**
4	c15-mo75	75	75	500	2500	2747.5	2766.8	**2767.9**	877	94	**26**
5	c15-mq75	75	75	500	2500	7466.5	7489.0	**7493.3**	1567	30	**12**
6	c15-ms75	75	75	500	2500	9354.6	9368.5	**9371.0**	2040	16	**5**
7	d10-mo75	75	75	771	1770	2772.6	2800.5	**2802.8**	484	19	**11**
8	d10-mq75	75	75	775	1774	7295.0	7328.2	**7332.7**	167	16	**4**
9	d10-ms75	75	75	781	1780	10069.3	10112.7	**10115.1**	1103	31	**9**
10	d15-mo75	75	75	1000	5000	2641.8	2662.0	**2664.1**	2402	61	**21**
11	d15-mq75	75	75	1000	5000	-	7395.5	**7401.7**	7200	90	**27**
12	d15-ms75	75	75	1000	5000	-	9256.0	**9258.4**	7200	136	**31**
13	c10-mo100	100	100	406	906	3330.9	3363.0	**3365.3**	217	22	**15**
14	c10-mq100	100	100	406	906	9352.6	9397.7	**9403.5**	367	17	**8**
15	c10-ms100	100	100	416	916	11740.1	11781.9	**11788.7**	166	10	**8**
16	c15-mo100	100	100	500	2500	3422.6	3449.7	**3454.3**	2809	37	**15**
17	c15-mq100	100	100	500	2500	9120.5	9141.2	**9149.0**	4008	23	**8**
18	c15-ms100	100	100	500	2500	11277.0	11301.3	**11306.1**	5204	46	**13**
19	d10-mo100	100	100	788	1787	3376.7	3411.3	**3414.9**	435	26	**13**
20	d10-mq100	100	100	778	1777	9179.2	9216.7	**9223.8**	581	28	**13**
21	d10-ms100	100	100	783	1782	11049.0	11093.1	**11096.8**	603	49	**17**
22	d15-mo100	100	100	1000	5000	-	3330.7	**3335.0**	7200	80	**29**
23	d15-mq100	100	100	1000	5000	-	9183.3	**9192.4**	7200	104	**34**
24	d15-ms100	100	100	1000	5000	-	11358.2	**11362.1**	7200	102	**19**
25	c10-mo200	200	200	433	933	7116.2	7180.3	**7184.8**	353	**47**	50
26	c10-mq200	200	200	428	928	19270.3	19326.2	**19332.2**	579	**38**	47
27	c10-ms200	200	200	431	931	25190.6	25254.2	**25257.1**	1040	178	**71**
28	c15-mo200	200	200	500	2500	-	7169.8	**7173.4**	7200	137	**59**
29	c15-mq200	200	200	500	2500	-	19220.9	**19227.9**	7200	149	**61**
30	c15-ms200	200	200	500	2500	-	24717.7	**24720.2**	7200	238	**60**
31	d10-mo200	200	200	816	1815	7194.1	7249.0	**7251.6**	3273	127	**94**
32	d10-mq200	200	200	814	1813	18789.0	18866.7	**18872.6**	3791	229	**131**
33	d10-ms200	200	200	806	1805	24509.6	24567.1	**24571.2**	6624	192	**70**
34	d15-mo200	200	200	1000	5000	-	7201.6	**7206.4**	7200	795	**231**
35	d15-mq200	200	200	1000	5000	-	19528.9	**19536.4**	7200	469	**240**
36	d15-ms200	200	200	1000	5000	-	24085.2	**24088.5**	7200	429	**124**

Table 4.3: Comparison of LP relaxation values and corresponding CPU-times in seconds for (dMCF$_f$), (dCut), and (dBCP) (time limit 7200s) on instances with $|F| \neq |C|$. Best values are marked bold.

Instance						LP value			CPU time [s]		
Nr	Name	$\|F\|$	$\|C\|$	$\|V\|$	$\|E\|$	$(\text{dMCF}_f)^{LP}$	$(\text{dCut})^{LP}$	$(\text{dBCP})^{LP}$	$(\text{dMCF}_f)^{LP}$	$(\text{dCut})^{LP}$	$(\text{dBCP})^{LP}$
37	c10-mo	75	200	404	904	8153.5	8206.0	**8209.6**	713	72	**39**
38	c10-mp	75	200	403	903	14917.4	14969.5	**14972.1**	228	**10**	17
39	c10-mq	75	200	403	903	20717.2	20786.4	**20789.4**	328	**16**	22
40	c15-mo	75	200	500	2500	-	7971.9	**7975.6**	7200	42	**25**
41	c15-mp	75	200	500	2500	14493.1	14526.4	**14529.2**	5533	45	**30**
42	c15-mq	75	200	500	2500	21570.7	21611.9	**21615.1**	3574	40	**33**
43	d10-mo	75	200	775	1775	8228.0	8293.9	**8296.9**	2166	55	**49**
44	d10-mp	75	200	775	1774	14836.9	14909.7	**14911.2**	2265	45	**40**
45	d10-mq	75	200	774	1773	20834.2	20893.6	**20896.3**	1001	31	**27**
46	d15-mo	75	200	1000	5000	-	8179.8	**8184.1**	7200	221	**93**
47	d15-mp	75	200	1000	5000	-	14771.5	**14775.3**	7200	134	**54**
48	d15-mq	75	200	1000	5000	-	21459.0	**21461.7**	7200	189	**66**
49	c10-mo	200	75	435	935	2957.0	2981.7	**2984.5**	6229	285	**111**
50	c10-mp	200	75	428	928	5444.6	5480.4	**5483.7**	3439	78	**28**
51	c10-mq	200	75	430	930	8093.5	8124.2	**8129.2**	1930	37	**10**
52	c15-mo	200	75	500	2500	-	2962.3	**2965.8**	7200	67	**26**
53	c15-mp	200	75	500	2500	-	5171.1	**5174.8**	7200	243	**37**
54	c15-mq	200	75	500	2500	-	7683.2	**7689.8**	7200	62	**11**
55	d10-mo	200	75	811	1810	-	3069.6	**3073.0**	7200	421	**276**
56	d10-mp	200	75	809	1808	5377.7	5407.7	**5410.5**	5608	39	**16**
57	d10-mq	200	75	820	1819	7698.7	7735.8	**7740.0**	3620	166	**49**
58	d15-mo	200	75	1000	5000	-	2978.9	**2982.6**	7200	727	**384**
59	d15-mp	200	75	1000	5000	-	5415.1	**5419.7**	7200	748	**383**
60	d15-mq	200	75	1000	5000	-	7590.8	**7594.3**	7200	187	**27**

runtime which is equal to the time limit of 7200 seconds in each run. All lower and upper bounds are rounded to the first decimal place.

We conclude that the lower bounds obtained by (dCut) and (dBCP) are better than those of (dMCF$_f$) in all test instances. With respect to primal solution quality, we observe that (dCut) only found the trivial upper bound given by connecting none of the customers in 18 and (dBCP) in two out of 60 test instances. In these instances the upper bounds due to (dMCF$_f$), which failed to find any primal solution for four instances and additionally found the trivial solution only for another eight instances, are eventually better than or equal to those of (dCut) and (dBCP), respectively. For all other instances, the upper bounds and resulting optimality gaps of (dCut) and (dBCP) are better than those of (dMCF$_f$). Thus, both (dCut) and (dBCP) significantly outperform (dMCF$_f$).

Model (dBCP) solved 44 out of 60 test instances to proven optimality, while (dCut)

Table 4.4: Comparison of solution values and corresponding CPU-times in seconds for ($dMCF_f$), (dCut), and (dBCP) (time limit 7200s) on instances with $|F| = |C|$. Best values are marked bold.

	lower bound			upper bound			gap in %			CPU time [s]	
Nr	($dMCF_f$)	(dCut)	(dBCP)	($dMCF_f$)	(dCut)	(dBCP)	($dMCF_f$)	(dCut)	(dBCP)	(dCut)	(dBCP)
1	2880.1	2915.8	**2919.6**	2944.2	2923.7	**2919.6**	2.227	0.268	**0.000**	7200	**501**
2	7105.2	7126.0	**7128.8**	7171.3	7129.3	**7128.9**	0.930	0.047	**0.002**	7200	7200
3	9509.5	**9536.9**	**9536.9**	9578.1	**9536.9**	**9536.9**	0.721	**0.000**	**0.000**	35	**5**
4	2748.7	2767.7	**2771.2**	2833.3	2788.6	**2771.2**	3.076	0.755	**0.000**	7200	**1035**
5	7469.7	7493.8	**7495.9**	7966.2	7497.8	**7495.9**	6.646	0.053	**0.000**	7200	**195**
6	9357.7	**9373.1**	**9373.1**	10918.9	**9373.1**	**9373.1**	16.684	**0.000**	**0.000**	1627	**38**
7	2776.3	2804.9	**2807.0**	2842.2	2815.6	**2807.0**	2.374	0.382	**0.000**	7200	**107**
8	7299.7	7334.0	**7338.9**	7373.0	7351.6	**7338.9**	1.004	0.239	**0.000**	7200	**243**
9	10073.9	10115.0	**10118.0**	10233.6	10144.7	**10118.0**	1.585	0.293	**0.000**	7200	**277**
10	2645.0	2664.4	**2666.1**	3397.2	8496.0	**2666.2**	28.439	218.868	**0.002**	7200	7200
11	7380.2	**7401.7**	**7401.7**	8528.6	**7401.7**	**7401.7**	15.561	**0.000**	**0.000**	**164**	310
12	9237.4	9258.6	**9259.0**	11007.6	9355.5	**9259.0**	19.164	1.048	**0.000**	7200	**284**
13	3333.0	3364.9	**3367.0**	3380.0	3371.6	**3367.0**	1.409	0.199	**0.000**	7200	**135**
14	9359.0	**9405.0**	9404.9	9473.7	**9405.0**	**9405.0**	1.226	**0.000**	0.002	**3829**	7200
15	11746.0	**11789.3**	**11789.3**	11855.1	**11789.3**	**11789.3**	0.929	**0.000**	**0.000**	383	**13**
16	3426.5	3453.9	**3455.3**	3933.8	3467.8	**3455.3**	14.803	0.403	**0.000**	7200	**98**
17	9125.1	**9149.3**	9149.2	9739.6	**9149.4**	**9149.4**	6.735	**0.000**	0.002	**940**	7200
18	11281.4	11307.3	**11308.5**	12722.2	11308.8	**11308.5**	12.771	0.013	**0.000**	7200	**243**
19	3380.7	3415.5	**3417.3**	3483.2	3418.3	**3417.3**	3.031	0.083	**0.000**	7200	**250**
20	9185.4	9225.0	**9226.2**	9258.9	9226.4	**9226.2**	0.800	0.015	**0.000**	7200	**119**
21	11055.0	**11098.1**	**11098.1**	11197.6	**11098.1**	**11098.1**	1.290	**0.000**	**0.000**	533	**34**
22	3314.0	3333.6	**3336.0**	3862.1	3338.6	**3336.2**	16.537	0.149	**0.004**	7200	7200
23	-	9191.8	**9194.4**	23780.0	23780.0	**9194.4**	-	158.708	**0.000**	7200	**889**
24	11332.4	11362.1	**11363.9**	12715.9	11391.5	**11363.9**	12.208	0.259	**0.000**	7200	**414**
25	7123.0	7184.7	**7185.2**	7329.4	7208.3	**7185.2**	2.898	0.328	**0.000**	7200	**66**
26	19279.8	19332.4	**19335.1**	19539.8	19340.7	**19335.1**	1.349	0.043	**0.000**	7200	**854**
27	25197.3	25256.6	**25258.9**	25327.2	77024.0	**25258.9**	0.516	204.966	**0.000**	7200	**647**
28	7139.0	7173.4	**7174.3**	8383.9	7196.4	**7174.3**	17.437	0.321	**0.000**	7200	**1880**
29	19191.4	19227.6	**19229.4**	21455.8	56699.0	**19229.4**	11.799	194.884	**0.000**	7200	**378**
30	24683.6	24720.7	**24721.1**	26764.0	**24721.0**	**24721.1**	8.428	0.001	**0.000**	7200	**563**
31	7197.4	7251.5	**7252.6**	8021.9	7263.5	**7252.6**	11.455	0.165	**0.000**	7200	**941**
32	18796.9	18870.8	**18875.5**	21247.5	18916.3	**18875.5**	13.037	0.241	**0.000**	7200	**4515**
33	24517.3	**24571.6**	**24571.6**	27880.1	**24571.6**	**24571.6**	13.716	**0.000**	**0.000**	6209	**141**
34	-	7206.0	**7207.2**	-	23975.0	**7207.2**	-	232.710	**0.000**	7200	**5606**
35	-	19535.2	**19537.5**	-	19544.5	**19537.7**	-	0.048	**0.001**	7200	7200
36	-	24088.3	**24090.7**	73434.0	73434.0	**24090.7**	-	204.853	**0.000**	7200	**3742**

Table 4.5: Comparison of solution values and corresponding CPU-times in seconds for (dMCF$_f$), (dCut), and (dBCP) (time limit 7200s) on instances with $|F| \neq |C|$. Best values are marked bold.

	lower bound			upper bound			gap in %			CPU time [s]	
Nr	(dMCF$_f$)	(dCut)	(dBCP)	(dMCF$_f$)	(dCut)	(dBCP)	(dMCF$_f$)	(dCut)	(dBCP)	(dCut)	(dBCP)
37	8158.2	8209.2	**8212.0**	9181.3	8286.8	**8212.2**	12.541	0.945	**0.002**	7200	7200
38	14924.5	**14973.3**	**14973.3**	15056.9	**14973.3**	**14973.3**	0.887	**0.000**	**0.000**	1624	**96**
39	20725.5	**20791.3**	**20791.4**	20915.4	**20791.4**	**20791.4**	0.916	**0.000**	**0.000**	4017	**277**
40	7948.0	7975.7	**7976.4**	9634.2	7989.4	**7976.8**	21.215	0.173	**0.005**	7200	7200
41	14497.5	14529.1	**14529.8**	15722.6	14557.0	**14530.1**	8.450	0.192	**0.002**	7200	7200
42	21576.2	**21615.3**	**21615.4**	22973.5	**21615.3**	**21615.4**	6.476	**0.000**	**0.000**	1536	**42**
43	8234.7	**8297.4**	**8297.5**	8511.6	**8297.4**	**8297.5**	3.362	**0.000**	**0.000**	1375	**1238**
44	14842.5	14911.1	**14912.6**	15075.2	38988.0	**14912.6**	1.568	161.470	**0.000**	7200	**324**
45	20839.4	**20896.9**	**20896.9**	21044.1	**20896.9**	**20896.9**	0.982	**0.000**	**0.000**	1811	**82**
46	-	8183.5	**8185.2**	20610.0	20610.0	**8185.2**	-	151.847	**0.000**	7200	**2838**
47	14731.9	14774.9	**14776.2**	15760.3	41720.0	**14776.4**	6.981	182.371	**0.001**	7200	7200
48	-	21461.4	**21462.1**	57923.0	57923.0	**21462.3**	-	169.894	**0.001**	7200	7200
49	2957.0	2982.6	**2989.7**	7209.0	7163.0	**3016.6**	143.794	140.156	**0.898**	7200	7200
50	5448.9	5482.5	**5486.9**	5517.8	13672.0	**5486.9**	1.265	149.376	**0.000**	7200	**320**
51	8098.0	**8130.1**	**8130.1**	8203.5	**8130.1**	**8130.1**	1.304	**0.000**	**0.000**	520	**35**
52	2948.5	2965.1	**2969.6**	7189.0	3047.3	**2969.6**	143.817	2.773	**0.000**	7200	**1453**
53	5150.7	5174.2	**5179.2**	12693.0	12693.0	**5179.4**	146.433	145.311	**0.005**	7200	7200
54	7667.3	7690.1	**7693.2**	10212.3	7719.7	**7693.2**	33.193	0.386	**0.000**	7200	**443**
55	3040.5	3070.5	**3076.5**	7500.0	**7405.0**	**7405.0**	146.667	141.168	**140.699**	7200	7200
56	5381.4	5409.9	**5414.6**	5598.2	5497.5	**5414.6**	4.029	1.620	**0.000**	7200	**874**
57	7702.2	7738.8	**7743.7**	10753.2	21242.0	**7743.9**	39.612	174.489	**0.003**	7200	7200
58	-	2980.7	**2983.5**	-	7226.0	**7185.6**	-	142.423	**140.844**	7200	**6446**
59	-	5418.6	**5423.0**	13849.0	13849.0	**5423.0**	-	155.583	**0.000**	7200	**4603**
60	-	7593.9	**7598.9**	-	18640.0	**7598.9**	-	145.461	**0.000**	7200	**3795**

could only solve 14 instances. For the remaining instances, the resulting optimality gap of (dBCP) exceeded 0.01% for only three instances. Thus we conclude that, next to its theoretical strength and tight lower bounds, (dBCP) allows for deriving high quality primal solutions relatively easily and significantly outperforms all other considered models. Furthermore, one can observe that the instances with $|F| = 200$ and $|C| = 50$ – i.e. instances 49–60 – seem to be particularly hard. While (dBCP) is able to provide reasonable results on most of them, (dMCF$_f$) and (dCut) often fail to compute meaningful primal solutions already for those instances where the underlying STP instance is relatively small.

Finally, we need to mention that due to numerical issues (differences between the used solvers) the obtained optimal solution values of (dCut) and (dBCP) slightly differ for three instances in the last shown digit (instances 30, 42, and 43). Further-

more, solving (dBCP) for instance 58 has been interrupted since the memory limit was reached.

4.12.2 Lagrangian Decomposition Approaches

We use the volume algorithm as described by Haouari and Siala [86] with the following settings for approximately solving the Lagrangian dual problem. Lagrangian multipliers are initialized by $\pi_{k,e} = c'_e$ for assignment edges $e \in E' \setminus E$ and by $\pi_{k,e} = c_e/|C|$ for edges $e \in E$. The target value T is initially set to 1.2 and multiplied by 1.1 in case $z_{\mathrm{LB}} > 0.9 z_{\mathrm{UB}}$ where z_{UB} and z_{LB} denote the so far best upper and lower bounds, respectively. ρ is initialized with 0.1 and multiplied by 0.67 after 20 non-improving iterations in case $\rho > 10^{-4}$ and by 1.5 in each improving iteration if $\rho < 5$ and if $\bar{v} \cdot v^t \geq 0$. Instead of computing λ_{OPT} as suggested in [86], we always use $\lambda = \lambda_{\mathrm{MAX}}$ which we initialize with 0.01. After every 100 iterations we multiply λ_{MAX} by 0.85 in case the lower bound did improve less than 1% and if $\lambda_{\mathrm{MAX}} > 10^{-5}$. The volume algorithm is terminated after 250 consecutive non-improving iterations or if the time limit is reached.

Computational results comparing three variants of our Lagrangian decompositions approach are summarized in Table 4.6 for instances with $|F| = |C|$, and in Table 4.7 for instances with $|F| \neq |C|$. Here, LD denotes the pure Lagrangian decomposition approach applying the Lagrangian heuristic presented in Section 4.9 without any further primal improvement, while LDS corresponds to the variant applying the simpler primal improvement, i.e. considering the key path and customer swap neighborhoods, and LDV applies the VLSN search instead of the customer swap improvement, see also Algorithm 4.2. Best obtained lower and upper bounds as well as tightest gap values among the three variants are marked bold for each instance. Values on bounds and gaps have been rounded to the first decimal place; no decimal places are given for runtimes.

Comparing Tables 4.6 and 4.7 to Tables 4.1, 4.2, and 4.3 with respect to the lower bounds, we conclude that the lower bounds of our LD approaches are usually slightly better than or approximately equal to the LP relaxation values of ($\mathrm{dMCF_c}$), at least for those small instances where $(\mathrm{dMCF_c})^{\mathrm{LP}}$ could be solved. Thus, the lower bounds of LD, LDS, and LDV which do not differ significantly, are usually worse than those of ($\mathrm{dMCF_f}$), (dCut), and (dBCP).

LDV clearly outperforms LD and LDS with respect to the primal solution quality, i.e. the resulting upper bounds. For instances with $|F| = |C|$, see Table 4.6, LDV produced the best results for 29 out of 36 instances, while LDS is the winner on only seven instances. Similarly, for instances with $|F| \neq |C|$, LDV produced better upper

Table 4.6: Comparison of solution values and corresponding CPU-times in seconds for LD, LDS, and LDV (time limit 7200s) on instances with $|F| = |C|$. Best values are marked bold.

	lower bound			upper bound			gap in %			CPU time [s]		
Nr	LD	LDS	LDV	LD	LDS	LDV	LD	LDS	LDV	LD	LDS	LDV
1	2851.9	2851.7	**2852.1**	2988.2	2962.4	**2938.4**	4.8	3.9	**3.0**	107	106	**81**
2	**7079.2**	7078.4	7076.9	7239.1	7177.4	**7158.0**	2.3	1.4	**1.1**	130	101	**73**
3	**9479.9**	9478.9	**9479.9**	9629.8	9581.1	**9554.5**	1.6	1.1	**0.8**	193	**106**	175
4	2737.5	2738.2	**2738.3**	2855.2	2815.3	**2793.4**	4.3	2.8	**2.0**	133	155	**127**
5	7457.2	**7457.6**	7456.8	7576.1	7541.4	**7505.3**	1.6	1.1	**0.6**	169	177	**143**
6	9341.7	9342.1	**9343.0**	9487.5	9408.6	**9390.8**	1.6	0.7	**0.5**	302	**185**	202
7	2741.6	**2741.8**	2741.5	2921.9	2849.3	**2830.7**	6.6	3.9	**3.3**	**224**	228	244
8	7280.9	**7281.5**	7281.3	7432.9	**7358.7**	7359.5	2.1	**1.1**	**1.1**	**175**	184	215
9	**10019.0**	10018.9	10018.9	10257.3	10213.2	**10167.4**	2.4	1.9	**1.5**	242	218	**158**
10	2636.6	**2636.9**	2636.8	2743.2	**2697.1**	2699.6	4.0	**2.3**	2.4	**251**	268	306
11	**7370.8**	7370.1	7369.0	7473.5	**7433.8**	7445.2	1.4	**0.9**	1.0	380	239	**147**
12	9221.6	**9222.2**	9221.1	9334.3	**9292.5**	9311.3	1.2	**0.8**	1.0	298	549	**237**
13	**3303.2**	3297.7	3302.3	3486.1	3437.3	**3406.3**	5.5	4.2	**3.1**	**222**	470	300
14	9322.5	**9322.6**	9322.5	9610.5	9491.5	**9460.4**	3.1	1.8	**1.5**	250	234	**228**
15	**11697.9**	11693.2	11696.8	11979.1	**11896.3**	11899.4	2.4	**1.7**	**1.7**	288	**207**	243
16	**3413.8**	3413.6	3413.7	3562.4	3542.6	**3493.6**	4.4	3.8	**2.3**	314	295	**209**
17	**9118.6**	9113.3	9117.2	9331.9	9214.8	**9192.4**	2.3	1.1	**0.8**	270	205	**173**
18	11264.0	11263.6	**11264.6**	11533.2	11426.2	**11379.1**	2.4	1.4	**1.0**	411	**329**	386
19	3337.8	3339.6	**3340.2**	3540.9	3474.6	**3461.1**	6.1	4.0	**3.6**	358	465	**253**
20	9129.6	9128.1	**9130.3**	9406.5	9374.9	**9261.6**	3.0	2.7	**1.4**	400	**278**	486
21	11000.4	**11001.3**	11000.7	11348.6	11234.3	**11161.8**	3.2	2.1	**1.5**	330	292	**247**
22	3297.8	3298.6	**3298.8**	3454.6	3424.1	**3369.8**	4.8	3.8	**2.2**	704	551	**404**
23	9149.5	**9150.4**	9149.8	9422.2	**9232.0**	9256.4	3.0	**0.9**	1.2	457	**373**	492
24	**11309.2**	11307.7	11307.3	11549.3	**11398.4**	11413.0	2.1	**0.8**	0.9	521	645	**369**
25	7052.9	**7053.3**	7052.5	7440.1	7325.2	**7269.6**	5.5	3.9	**3.1**	4302	7201	**2174**
26	19211.7	**19213.2**	19211.8	19673.0	19574.3	**19436.1**	2.4	1.9	**1.2**	**3978**	4409	5855
27	**25115.3**	25114.6	25115.1	25680.7	25627.1	**25306.5**	2.3	2.0	**0.8**	7201	4354	**4327**
28	**7108.8**	7105.5	7106.8	7427.4	7450.5	**7252.3**	4.5	4.9	**2.0**	3797	**3142**	4129
29	**19171.1**	19171.0	19171.0	19495.3	19326.5	**19290.1**	1.7	0.8	**0.6**	**3459**	4961	4823
30	24654.4	**24655.8**	24655.2	25302.2	25003.3	**24854.6**	2.6	1.4	**0.8**	4662	**4197**	4899
31	7107.3	7106.9	**7107.9**	7599.7	7448.3	**7331.6**	6.9	4.8	**3.1**	5498	**4612**	7201
32	18720.8	**18720.9**	18720.5	19214.6	19025.8	**18971.6**	2.6	1.6	**1.3**	5900	4770	**3119**
33	24426.7	24425.9	**24426.9**	24856.3	24730.5	**24696.9**	1.8	1.2	**1.1**	6681	**4340**	4686
34	**7129.0**	7126.9	7127.8	7448.1	7381.0	**7329.1**	4.5	3.6	**2.8**	**5159**	5822	5174
35	19457.0	19455.3	**19457.1**	19880.4	19772.2	**19606.4**	2.2	1.6	**0.8**	**4743**	5081	7201
36	24007.7	24008.3	**24009.4**	24539.0	24201.9	**24198.9**	2.2	**0.8**	**0.8**	**4666**	5835	6050

Table 4.7: Comparison of solution values and corresponding CPU-times in seconds for LD, LDS, and LDV (time limit 7200s) on instances with $|F| \neq |C|$. Best values are marked bold.

	lower bound			upper bound			gap in %			CPU time [s]		
Nr	LD	LDS	LDV	LD	LDS	LDV	LD	LDS	LDV	LD	LDS	LDV
37	8117.6	**8118.5**	8116.9	8630.1	8442.8	**8258.6**	6.3	4.0	**1.7**	**591**	735	1458
38	**14882.6**	14881.6	14882.2	15407.4	15225.5	**15059.9**	3.5	2.3	**1.2**	707	**651**	1672
39	20681.3	**20681.6**	20681.1	21150.6	20957.2	**20844.9**	2.3	1.3	**0.8**	**901**	1387	1468
40	7935.1	**7935.4**	7935.3	8166.6	8068.3	**8049.9**	2.9	1.7	**1.4**	870	**839**	1609
41	**14483.2**	**14483.2**	**14483.2**	14809.9	14643.5	**14608.0**	2.3	1.1	**0.9**	**919**	952	977
42	**21561.4**	**21561.4**	21561.3	21940.2	21770.5	**21662.6**	1.8	1.0	**0.5**	**931**	984	2183
43	**8183.6**	8181.9	8182.7	8717.3	8604.5	**8427.4**	6.5	5.2	**3.0**	853	**791**	1421
44	14779.3	14778.0	**14779.4**	15271.8	15181.1	**14975.6**	3.3	2.7	**1.3**	915	**857**	1532
45	20766.9	20766.6	**20767.2**	21221.4	21081.6	**21009.1**	2.2	1.5	**1.2**	1052	**1034**	2476
46	8127.5	8128.2	**8128.5**	8451.4	8401.2	**8262.5**	4.0	3.4	**1.6**	1528	**1325**	2268
47	14717.0	14717.6	**14718.5**	15115.2	14893.5	**14839.6**	2.7	1.2	**0.8**	1164	1202	**1090**
48	21407.6	21407.0	**21407.8**	21741.0	21588.7	**21526.7**	1.6	0.8	**0.6**	1419	1672	**1329**
49	**2952.0**	2951.1	2949.8	3321.4	3179.6	**3044.6**	12.5	7.7	**3.2**	484	**434**	509
50	5434.5	**5434.9**	**5434.9**	5668.1	5638.7	**5506.6**	4.3	3.7	**1.3**	556	**502**	579
51	8080.2	8080.3	**8080.9**	8208.3	**8189.2**	8290.7	1.6	**1.3**	2.6	**438**	494	460
52	2947.3	2946.0	**2947.4**	3312.9	3187.0	**3156.9**	12.4	8.2	**7.1**	475	437	**350**
53	**5153.6**	5152.6	5151.3	5446.2	5296.4	**5262.2**	5.7	2.8	**2.2**	561	627	**455**
54	7666.4	7668.3	**7668.4**	7837.4	7810.8	**7789.5**	2.2	1.9	**1.6**	509	533	**398**
55	3026.3	3028.4	**3028.5**	3887.1	3971.5	**3825.4**	28.4	31.1	**26.3**	461	454	**381**
56	5361.8	**5361.9**	5360.9	5777.4	5657.6	**5594.5**	7.8	5.5	**4.4**	628	**540**	558
57	7675.5	**7676.7**	7676.4	8127.2	7890.6	**7827.8**	5.9	2.8	**2.0**	**449**	533	546
58	2950.1	**2951.6**	2951.1	3633.4	3265.0	**3236.2**	23.2	10.6	**9.7**	775	722	**500**
59	5387.3	**5387.8**	5387.2	5828.8	5625.1	**5486.3**	8.2	4.4	**1.8**	680	730	**621**
60	7564.3	7563.6	**7571.7**	8040.7	7805.4	**7704.4**	6.3	3.2	**1.8**	663	599	**471**

bounds than the other two approaches in 23 out of 24 cases, while LDS performed best with respect to primal solution quality on a single instance only.

Due to applying primal improvement only to a relatively small, but highly promising subset of candidate solutions derived by our Lagrangian heuristic, the overhead of LDS and LDV usually is only moderate. Sometimes LDS or LDV are even faster than LD since a better upper bound eventually found in an early iteration of the volume algorithm does influence Lagrangian multipliers and the whole process of approximately solving the Lagrangian dual. Even though LDV tends to need more time than LDS for larger instances, no clear advantages with respect to runtime can be observed for one of these two approaches. We conclude that LDS yields the best results among the three variants based on Lagrangian relaxation and does not require excessive runtimes.

4.12.3 Overall Comparison

In the following, the performance of (dBCP) and LDV which showed to outperform the other methods from their area will be further investigated. A detailed comparison of relative upper bounds and runtimes of (dBCP) and LDV is given in Tables 4.8 and 4.9, respectively. Here, instances are grouped by the size of the underlying SSCFLP instance in Tables 4.8 and by the size of the original STP instance in Table 4.9.

Table 4.8: Relative solution values and corresponding CPU-times for (dBCP) and LDV grouped by SSCFLP instance size (12 instances per set).

Instance Set			relative upper bound $\frac{\text{LDV}-\text{(dBCP)}}{\text{(dBCP)}}$ in %			relative CPU-time $\frac{\text{LDV}}{\text{(dBCP)}}$		
Nr.	$\lvert F\rvert$	$\lvert C\rvert$	minimum	median	maximum	minimum	median	maximum
1–12	75	75	0.12	0.53	1.25	0.01	0.65	35.65
13–24	100	100	0.38	0.65	1.28	0.02	1.30	19.29
25–36	200	200	0.19	0.52	1.69	0.69	6.77	33.32
37–48	75	200	0.22	0.54	1.57	0.14	0.97	52.12
49–60	200	75	-54.96	1.21	6.31	0.05	0.13	13.09

Table 4.9: Relative solution values and corresponding CPU-times for (dBCP) and LDV grouped by STP instance size (15 instances per set).

Instance Set			relative upper bound $\frac{\text{LDV}-\text{(dBCP)}}{\text{(dBCP)}}$ in %			relative CPU-time $\frac{\text{LDV}}{\text{(dBCP)}}$		
Name	$\lvert V^o\rvert$	$\lvert E^o\rvert$	minimum	median	maximum	minimum	median	maximum
c10-*	500	1000	0.18	0.58	1.97	0.01	5.31	35.65
c15-*	500	2500	0.12	0.62	6.31	0.02	0.9	52.12
d10-*	1000	2000	-48.34	0.54	3.32	0.05	1.15	33.32
d15-*	1000	5000	-54.96	0.59	1.69	0.04	0.47	1.62

Since (dBCP) successfully solved the majority of instances to proven optimality it dominates LDV with respect to obtained upper bounds. Thus, the gaps due to LDV are usually larger than those of (dBCP), but exceeded 4.4% only for three instances with $|F| = 200$ and $|C| = 75$, which seem to be particularly hard and are smaller than or equal to 2% for 70% of all tested instances. When the instances get larger (dBCP) often needs longer than LDV and completely failed to compute meaningful solutions for two out of 60 instances.

Thus, while (dBCP) has the potential to compute superior solutions, LDV can be regarded as the more stable and if the instances get more difficult also faster ap-

proach. Furthermore, if runtime is of special relevance LDV can be easily parallelized by simply solving all independent subproblems of the Lagrangian dual in parallel.

Overall, (dBCP) can be recommended for medium sized instances given enough runtime, while (a parallel variant of) LDV should be used to approximately solve even larger instances or when keeping the runtime small is more important than reducing the optimality gap by a few percent.

4.13 Conclusions and Outlook

In this chapter, we considered a generalized variant of the rooted connected facility location problem with capacity constraints and customer prizes where only the most profitable client subset shall be supplied.

In a first section, four different mixed integer programming models for solving CConFL to proven optimality have been presented. While the first two are multi-commodity flow based models including a polynomial number of constraints and variables only, the third model involves an exponential number of so-called connectivity constraints, but can efficiently be solved by branch-and-cut. Finally, an alternative model incorporating an exponential number of constraints and variables has been proposed and its solution by branch-and-cut-and-price has been discussed in detail. A polyhedral comparison showed that this model is the theoretically strongest among the four considered models.

In the second main part of this chapter a Lagrangian decomposition approach for CConFL based on one of the above mentioned multi-commodity flow formulations has been introduced. After discussing the subproblems arising while approximately solving the Lagrangian dual problem, a Lagrangian heuristic to additionally obtain feasible solutions during the course of solving the Lagrangian dual problem has been discussed. Furthermore, we presented two hybrid methods combining the Lagrangian approach with local search and VLSN search.

Computational results show that the branch-and-cut-and-price approach based on the theoretically strongest exact model significantly outperforms the other three integer programming approaches. It could solve the majority of test instances to proven optimality relatively fast, and the resulting optimality gaps are usually extremely small in case the computation is aborted due to the given time limit.

For the Lagrangian methods, the hybrid approach involving VLSN search turns out to outperform the others with respect to solution quality, while only moderately increasing the necessary computation time.

Whether the branch-and-cut-and-price approach or the hybrid Lagrangian/VLSN method should be used in practice is a difficult question and depends on the considered instances. The branch-and-cut-and-price method is able to solve medium sized instances to proven optimality. On these instances, the solutions and lower bounds obtained by the LD/VLSN hybrid will typically be slightly worse. However, the LD/VLSN approach is feasible for solving even larger instances, since it can be easily parallelized as the various subproblems of the relaxed model are completely independent of each other. Furthermore, the primal improvement approach is naturally composed of two independent subproblems, i.e. a Steiner tree problem and a single source capacitated facility location problem. Parallelizing the LD/VLSN approach might in particular allow for solving real world instances, which tend to be significantly larger than the instances considered in this chapter, but on the other hand are likely to be more structured, since there will usually be a strong correlation between a facility's position and its assignable customers.

Interesting areas for further research include the development of pure metaheuristic methods for CConFL. Such metaheuristics can be used to tackle very large scale instances and might include the exact and hybrid approaches presented in this chapter for solving smaller subproblems. It might also be possible to further strengthen the proposed exact models by considering additional cutting planes, e.g. from the multiple knapsack problem [62, 63] and to further speed-up the LD/VLSN hybrid by considering alternative algorithms for solving the negative subset disjoint cycle subproblem [55].

Chapter 5

Conclusions

This thesis considered the $b_{\max}$-Survivable Network Design Problem ($b_{\max}$-SNDP) and the Capacitated Connected Facility Location Problem (CConFL). These two $\mathcal{NP}$-hard combinatorial optimization problems (COPs) are suitable to model certain real world scenarios occurring when extending communication networks on the last mile.

For solving instances of these problems, methods from different streams of combinatorial optimization have been proposed, each of which having individual advantages for instances of different sizes.

The $b_{\max}$-Survivable Network Design Problem

Two mixed integer programming models for $b_{\max}$-SNDP involving an exponential number of variables have been developed and their superiority over existing formulations from a theoretical point of view has been shown. From a computational point of view, the solution of these models could be significantly accelerated by using alternative dual-optimal solutions in the pricing subproblems while solving them by branch-and-price. Both approaches are able to derive proven optimal solutions or high quality solutions with small optimality gaps for mediums sized instances within reasonable time. In particular, they are often able to solve these instance faster than a previously existing method. Computational results further show that the linear relaxations of both models are much tighter than those of the previous formulations. In particular it turned out that solving the linear relaxation of the strongest among

the newly proposed models yields integral and thus proven optimal solutions for the majority of tested instances.

For approximately solving larger instances, while still deriving both lower and upper bounds of an optimal solutions value a Lagrangian decomposition approach has been suggested and subsequently hybridized with variable neighborhood descent. Here, with respect to the execution order a sequential and an interleaved hybrid variant has been proposed. The results from the performed computational study indicate that both hybrid variants derive high quality solutions and good lower bounds within reasonable time.

Finally, for solving large scale instances metaheuristic approaches have been considered. Based on three types of neighborhood structures, a greedy randomized adaptive search as well as a general variable neighborhood search dominating the GRASP with respect to the obtained solutions have been described.

The Capacitated Connected Facility Location Problem

Four new mixed integer programming models for CConFL have been proposed and theoretically compared to each other. Next to two multi-commodity flow based models, these include a model involving an exponential number of connectivity constraints which can be efficiently solved by branch-and-cut as well as an even stronger model including an exponential number of constraints and variables. The latter can be solved by branch-and-cut-and-price.

For large instances that cannot be solved to proven optimality, a Lagrangian decomposition approach based on one of the multi-commodity flow formulations has been suggested. After discussing a Lagrangian heuristic for deriving feasible solutions during the course of solving the Lagrangian dual, two hybrid methods combining the Lagrangian approach with local search and very large scale neighborhood search have been proposed.

Computational results show that among the MIP based methods, the branch-and-cut-and-price approach based on the theoretically strongest models could solve the majority of test instances to proven optimality relatively fast. Furthermore, its resulting optimality gaps are usually extremely small in case the computation is aborted due to a given time limit. Thus, it significantly outperforms the other three MIP based methods. Among the Lagrangian methods, the hybrid approach involving VLSN search turns out to outperform the others with respect to solution quality, while only moderately increasing the necessary computation time. Thus, this approach turns out to be suitable for instances that are too large to be solved to proven optimality by the branch-and-cut-and-price method.

Bibliography

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997.

[2] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[3] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[4] Y. Agarwal, K. Mathu, and H. M. Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19(7):731–749, 1989.

[5] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[6] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutella. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50(6):749–760, 2004.

[7] R. K. Ahuja, J. B. Orlin, and D. Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91(1):71–97, 2001.

[8] P. Bachhiesl. The OPT- and the SST-problems for real world access network design – basic definitions and test instances. Working Report 01/2005, Carinthia Tech Institute, Department of Telematics and Network Engineering, Klagenfurt, Austria, 2005.

[9] P. Bachhiesl, M. Prossegger, G. Paulus, J. Werner, and H. Stögner. Simulation and optimization of the implementation costs for the last mile of fibre optic networks. *Networks and Spatial Economics*, 3(4):467–482, 2003.

[10] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[11] L. Bahiense, F. Barahona, and O. Porto. Solving Steiner tree problems in graphs with Lagrangian relaxation. *Journal of Combinatorial Optimization*, 7(3):259–282, 2003.

[12] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[13] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000.

[14] M. G. Bardossy and S. Raghavan. Dual-based local search for the connected facility location and related problems. Technical report, Smith School of Business and Institute for Systems Research, University of Maryland, 2009.

[15] C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[16] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[17] J. E. Beasley. An algorithm for the Steiner problem in graphs. *Networks*, 14(1):147–159, 1984.

[18] J. E. Beasley. Lagrangean relaxation. In C. Reeves, editor, *Modern heuristic techniques in combinatorial problems*, pages 243–303. Blackwell Scientific Publications, 1993.

[19] J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394, 1989.

[20] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[21] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[22] H. Ben Amor, J. Desrosiers, and J. M. V. de Carvalho. Dual-optimal inequalities for stabilized column generation. *Operations Research*, 54(3):454–463, 2006.

[23] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[24] D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas, 2005.

[25] C. Blum, M. J. B. Aquilera, A. Roli, and M. Sampels, editors. *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence (SCI)*. Springer, 2008.

[26] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparision. *ACM Computing Surveys*, 35(3):268–308, 2003.

[27] C. Blum and A. Roli. Hybrid metaheuristics: An introduction. In C. Blum, M. J. B. Aquilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence (SCI)*, pages 1–30. Springer, 2008.

[28] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68, 2006.

[29] M. A. Boschetti, A. Mingozzi, and S. Ricciardelli. A dual ascent procedure for the set partitioning problem. *Discrete Optimization*, 5:735–747, 2008.

[30] S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

[31] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

[32] O. Chapovska and A. P. Punnen. Variations of the prize-collecting Steiner tree problem. *Networks*, 47(4):199–205, 2006.

[33] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):290–410, 1997.

[34] M. Chiarandini, I. Dumitrescu, and T. Stützle. Very large-scale neighborhood search: Overview and case studies on coloring problems. In C. Blum, M. J. B. Aquilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics, An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

[35] M. Chimani, M. Kandyba, I. Ljubic, and P. Mutzel. Orientation-based models for {0,1,2}-survivable network design: Theory and practice. *Mathematical Programming, Series B*, 2009. accepted.

[36] M. Chimani, M. Kandyba, and P. Mutzel. A new ILP formulation for 2-root-connected prize-collecting Steiner networks. In *15th Annual European Symposium on Algorithms (ESA'07)*, volume 4698 of *LNCS*, pages 681–692. Springer, 2007.

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

[38] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. Varela and P. Bourgine, editors, *Proceedings of ECAL'91 - First European Conference on Artifial Life*, pages 134–142. Elsevier, 1992.

[39] R. K. Congram. *Polynomially Searchable Exponential Neighbourhoods for Sequencing Problems in Combinatorial Optimization*. PhD thesis, University of Southampton, Faculty of Methematical Studies, UK, 2000.

[40] R. K. Congram, C. N. Pots, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.

[41] I. A. Contreras and J. A. Diaz. Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 157(1):73–89, 2008.

[42] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P. B. Mirchandani and R. L. Francis, editors, *Discrete Location Theory*, pages 119–171. Wiley, 1990.

[43] A. S. da Cunha, A. Lucena, N. Maculan, and M. G. C. Resende. A relax-and-cut algorithm for the prize-collecting Steiner problem in graph. *Discrete Applied Mathematics*, 157(6):1198–1217, 2009.

[44] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koppmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. Wiley, 1958.

[45] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.

[46] G. B. Dantzig and P. Wolfe. The decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[47] C. Darwin. *On the origin of species by means of natural selection of the preservation of favored races in the struggle for life*. Murray, 1859.

[48] M. P. de Aragao and E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price algorithms. In *Annals of Mathematical Programming in Rio*, pages 56–61, 2003.

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

[50] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[51] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[52] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer, 2006.

[53] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[54] C. W. Duin and S. Voß. Efficient path and vertex exchange in Steiner tree algorithms. *Networks*, 29:89–105, 1997.

[55] I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, University of Melbourne, 2002.

[56] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

[57] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1174–1183, 2008.

[58] P. Elias, A. Feinstein, and C. Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, 2(4):117–119, 1956.

[59] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[60] T. Feo and M. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

[61] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[62] C. E. Ferreira, A. Martin, and R. Weismantel. Facets for the multiple knapsack polytope. Technical Report SC 93-04, Konrad-Zuse Zentrum für Informationstechnik, 1993.

[63] C. E. Ferreira, A. Martin, and R. Weismantel. Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization*, 6:858–877, 1996.

[64] P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[65] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[67] S. Fortune, J. Hopcroft, and H. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[68] B. Fortz and M. Labbé. Polyhedral approaches to the design of survivable networks. In M. G. C. Resende and P. M. Pardolas, editors, *Handbook of Optimization in Telecommunications*, pages 367–389. Springer, 2006.

[69] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[70] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[71] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9(6):849–859, 1961.

[72] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem - part II. *Operations Research*, 11(6):863–888, 1963.

[73] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[74] F. Glover. Tabu search - part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[75] F. Glover. Tabu search - part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[76] F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Norwell, MA, 2003.

[77] S. Gollowitzer and I. Ljubić. MIP models for connected facility location: A theoretical and computational study. Technical Report 2009–07, University of Vienna, 2009.

[78] T. Gomes, J. Craveirinha, and L. Jorge. An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs. *Computers & Operations Research*, 36:1670–1682, 2009.

[79] L. Gouveia, A. Paias, and D. Sharma. Modeling and solving the rooted distance-constrained minimum spanning tree problem. *Computers & Operations Research*, 35(2):600–613, 2008.

[80] M. Gruber and G. R. Raidl. (Meta-)heuristic separation of jump cuts in a branch&cut approach for the bounded diameter minimum spanning tree problem. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics - Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*, pages 209–230. Springer, 2009.

[81] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *Proceedings of the 33rd annual ACM symposium on theory of computing*, pages 389–398, 2001.

[82] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voß, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-heuristics, Advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer Academic Publishers, 1999.

[83] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[84] P. Hansen and N. Mladenović. A tutorial on variable neighborhood search. Technical Report G-2003-46, Les Cahiers du GERAD, HEC Montreal and GERAD, Canada, 2003.

[85] M. Haouari, S. B. Jayeb, and H. D. Sherali. The prize collecting Steiner tree problem: Models and Lagrangian dual optimization approaches. *Computational Optimization and Applications*, 40(1):13–39, 2008.

[86] M. Haouari and J. C. Siala. A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Computers and Operations Research*, 33(5):1274–1288, 2006.

[87] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, 4(2):100–107, 1968.

[88] P. E. Hart, N. J. Nilsson, and B. Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Bulletin*, 37:28–29, 1972.

[89] M. K. Hasan, H. Jung, and K. Chwa. Approximation algorithms for connected facility location problems. *Journal of Combinatorial Optimization*, 16(2):155–172, 2008.

[90] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[91] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.

[92] J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1992.

[93] K. Holmberg, M. Rönnqvist, and D. Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113:544–559, 1999.

[94] K. Holmberg and D. Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 2000.

[95] B. Hu. *Hybrid Metaheuristics for Generalized Network Design Problems*. PhD thesis, Vienna University of Technology, 2008.

[96] B. Hu, M. Leitner, and G. R. Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14(5):473–499, 2008.

[97] B. Hu and G. R. Raidl. Variable neighborhood descent with self-adaptive neighborhood-ordering. In C. Cotta, A. J. Fernandez, and J. E. Gallardo, editors, *Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*, 2006.

[98] IBM. CPLEX 12.1.

[99] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer, 2005.

[100] K. Kar, M. Kodialam, and T. V. Lakshman. Routing restorable bandwidth guaranteed connections using maximum 2-route flows. *IEEE/ACM Transactions on Networking*, 11(5):772–781, 2003.

[101] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 613–623. IEEE Computer Society, 2000.

[102] N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[103] R. M. Karp. Reducibility among combinatorial problems. In E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[104] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Heidelberg, 2004.

[105] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, pages 1942–1948. IEEE, 1995.

[106] H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[107] L. Khachiyan. A polynomial algorithm in linear programming (english translation). *Soviet Mathematics Doklady*, 20:191–194, 1979.

[108] S. Khuller and A. Zhu. The general Steiner tree-star problem. *Information Processing Letters*, 84(4):215–220, 2002.

[109] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[110] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

[111] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15(2):141–145, 1981.

[112] J. Kratica, D. Tosic, V. Filipovic, and I. Ljubić. Solving the simple plant location problem by genetic algorithm. *RAIRO Operations Research*, 35:127–142, 2001.

[113] J. B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

[114] Y. Lee, S. Y. Chiu, and J. Ryan. A branch and cut algorithm for a Steiner tree-star problem. *INFORMS Journal on Computing*, 8(3):194–201, 1996.

[115] Y. Lee, L. Lu, Y. Qiu, and F. Glover. Strong formulations and cutting planes for designing digital data service networks. *Telecommunication Systems*, 2(1):261–274, 1993.

[116] M. Leitner and G. R. Raidl. Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks. In M. J. Blesa et al., editors, *Hybrid Metaheuristics 2008*, volume 5296 of *LNCS*, pages 158–174. Springer, 2008.

[117] M. Leitner and G. R. Raidl. Variable neighborhood search for a prize collecting capacity constrained connected facility location problem. In *Proceedings of the 2008 International Symposium on Applications and the Internet*, pages 233–236. IEEE Computer Society, 2008.

[118] M. Leitner and G. R. Raidl. A Lagrangian decomposition based heuristic for capacitated connected facility location. In S. Voß and M. Caserta, editors, *Proceedings of the 8th Metaheuristic International Conference (MIC 2009)*, Hamburg, Germany, 2009.

[119] M. Leitner and G. R. Raidl. Branch-and-cut-and-price for capacitated connected facility location. Technical Report TR 186–1–10–01, Vienna University of Technology, Vienna, Austria, 2010.

[120] M. Leitner and G. R. Raidl. Strong lower bounds for a survivable network design problem. In *International Symposium on Combinatorial Optimization (ISCO 2010)*, Hammamet, Tunisia, March 2010.

[121] M. Leitner and G. R. Raidl. Combining Lagrangian decomposition with very large scale neighborhoood search for capacitated connected facility location. In *Post-Conference Book of the Eight Metaheuristics International Conference – MIC 2009*. accepted 2010.

[122] M. Leitner, G. R. Raidl, and U. Pferschy. Accelerating column generation for a survivable network design problem. In M. G. Scutellà et al., editors, *Proceedings of the International Network Optimization Conference 2009*, 2009.

[123] M. Leitner, G. R. Raidl, and U. Pferschy. Branch-and-price for a survivable network design problem. Technical Report TR 186–1–10–02, Vienna University of Technology, Vienna, Austria, 2010.

[124] I. Ljubić. *Exact and Memetic Algorithms for Two Network Design Problems*. PhD thesis, Vienna University of Technology, 2004.

[125] I. Ljubić. A hybrid VNS for connected facility location. In T. Bartz-Beielstein et al., editors, *Hybrid Metaheuristics, 4th International Workshop, HM 2007*, volume 4771 of *LNCS*, pages 157–169. Springer, 2007.

[126] I. Ljubić and S. Gollowitzer. Hop constrained connected facility location. Technical Report 2009–09, University of Vienna, 2009.

[127] I. Ljubić and S. Gollowitzer. Modelling the hop constrained connected facility location problem on layered graphs. In *International Symposium on Combinatorial Optimization (ISCO 2010)*, Hammamet, Tunisia, March 2010. to appear.

[128] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming, Series B*, 105(2–3):427–449, 2006.

[129] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2003.

[130] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[131] A. Lucena and M. G. C. Resende. Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1–3):277–294, 2004.

[132] T. L. Magnanti and L. A. Wolsey. Optimal trees. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 7, pages 503–615. Elsevier, 1995.

[133] V. Maniezzo, T. Stützle, and S. Voß, editors. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*. Springer, 2009.

[134] R. E. Marsten. The use of the boxstep method in discrete optimization. *Mathematical Programming Studies*, 3:127–144, 1975.

[135] R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389405, 1975.

[136] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

[137] S. L. Martins, M. G. C. Resende, C. C. Ribeiro, and P. M. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 17(1-4):267–283, 2000.

[138] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27(3):125–128, 1988.

[139] G. Mendel. Versuche über Panzen-Hybriden (Experiments on plant hybridization). In *Verhandlungen des naturforschenden Vereins Brünn (Proceedings of the Natural History Society of Brünn)*, volume 4, pages 3–47, 1866.

[140] M. Minkoff. The prize collecting Steiner tree problem. Master's thesis, Massachusetts Institute of Technology, 2000.

[141] P. Moscato. Memetic algorithms: A short introduction. In D. Corne et al., editors, *New Ideas in Optimization*, pages 219–234. McGraw Hill, 1999.

[142] G. L. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991.

[143] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[144] C. H. Papdimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Dover Publications Inc., 1998.

[145] S. Pirkwieser and G. R. Raidl. A column generation approach for the periodic vehicle routing problem with time windows. In M. G. Scutellà et al., editors, *Proceedings of the International Network Optimization Conference 2009*, 2009.

[146] S. Pirkwieser, G. R. Raidl, and J. Puchinger. A Lagrangian decomposition/evolutionary algorithm hybrid for the knapsack constrained maximum spanning tree problem. In C. Cotta and J. van Hemert, editors, *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*, pages 69–85. Springer, 2008.

[147] D. Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45(5):758–767, 1997.

[148] M. Prandtstetter and G. R. Raidl. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research*, 191(3):1004–1022, 2008.

[149] R. C. Prim. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[150] J. Puchinger and G. R. Raidl. An evolutionary algorithm for column generation in integer programming: an effective approach for 2D bin packing. In X. Yao et. al, editor, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 642–651. Springer, 2004.

[151] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation*, volume 3562 of *LNCS*, pages 41–53. Springer, 2005.

[152] J. Puchinger and G. R. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2007.

[153] J. Puchinger and G. R. Raidl. Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. *Journal of Heuristics*, 14(5):457–472, 2008.

[154] J. Puchinger, G. R. Raidl, and S. Pirkwieser. Metaboosting: Enhancing integer programming techniques by metaheuristics. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*, pages 71–102. Springer, 2009.

[155] S. Raghavan and M. G. Bardossy. Dual based heuristics for the connected facility location problem. In M. G. Scutellà et al., editors, *Proceedings of the International Network Optimization Conference 2009*, 2009.

[156] G. R. Raidl. A unified view on hybrid metaheuristics. In F. Almeida et al., editors, *Proceedings of the Hybrid Metaheuristics Workshop*, volume 4030 of *LNCS*, pages 1–12. Springer, 2006.

[157] G. R. Raidl and J. Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In C. Blum, M. J. B. Aquilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence (SCI)*, pages 31–62. Springer, 2008.

[158] G. R. Raidl, J. Puchinger, and C. Blum. Metaheuristic hybrids. In M. Gendreau and J. Y. Potvin, editors, *Handbook of Metaheuristics, 2nd edition*. Springer, 2010. to appear.

[159] I. Rechenberg. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, 1965.

[160] M. G. C. Resende and P. M. Pardolos, editors. *Handbook of Optimization in Telecommunications*. Springer, 2006.

[161] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[162] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[163] A. Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

[164] M. Stoer. *Design of Survivable Networks*, volume 1531 of *LNCS*. Springer, 1992.

[165] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–335, 1984.

[166] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[167] H. Takahashi and A. Matsuyama. An approximated solution for the Steiner tree problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[168] P. M. Thompson and J. B. Orlin. The theory of cyclic transfers. Technical Report OR 200-89, Massachusetts Institute of Technology, Operations Research Center, 1989.

[169] A. Tomazic and I. Ljubić. A GRASP algorithm for the connected facility location problem. In *Proceedings of the 2008 International Symposium on Applications and the Internet*, pages 257–260. IEEE Computer Society, 2008.

[170] E. Uchoa. Reduction tests for the prize-collecting Steiner problem. *Operations Research Letters*, 34(4):437–444, 2006.

[171] E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, and M. P. de Aragao. Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation. *Mathematical Programming*, 12(2):443–472, 2008.

[172] F. Vanderbeck. Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 331–358. Springer US, 2005.

[173] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[174] M. G. A. Verhoeven and M. E. M. Severens. Parallel local search for Steiner trees in graphs. *Annals of Operations Research*, 90:185–202, 1999.

[175] S. Voß. Steiner's problem in graphs: heuristic methods. *Discrete Applied Mathematics*, 40(1):45–72, 1992.

[176] S. Voß, S. Martello, I. H. Osman, and C. Roucairol, editors. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer, 1999.

[177] D. Wagner, U. Pferschy, P. Mutzel, G. R. Raidl, and P. Bachhiesl. A directed cut model for the design of the last mile in real-world fiber optic networks. In B. Fortz, editor, *Proceedings of the International Network Optimization Conference 2007*, pages 103/1–6, Spa, Belgium, 2007.

[178] D. Wagner, G. R. Raidl, U. Pferschy, P. Mutzel, and P. Bachhiesl. A multicommodity flow approach for the design of the last mile in real-world fiber optic networks. In K.-H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006*, pages 197–202. Springer, 2007.

[179] C. Walshaw. Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In C. Blum, M. J. B. Aquilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence (SCI)*, pages 261–289. Springer, 2008.

[180] P. Winter. Steiner problem in networks: a survey. *Networks*, 17(2):129–167, 1987.

[181] J. Xu, S. Y. Chiu, and F. Glover. Tabu search for dynamic routing communications network design. *Telecommunication Systems*, 8(1):55–77, 1997.

[182] ZIB. SCIP 1.2.0. http://scip.zib.de.

Appendix A

Curriculum Vitae

Personal Information

- Name: Markus Leitner
- Date and place of birth: November 2, 1979, Ried im Innkreis, Austria
- Nationality: Austria
- Resident: Vienna, Austria
- Family status: Married to Mag. Romana Leitner, no children
- Languages: German (native), English (fluent)

Education

- since 10/2006:
  PhD (Doctorate) studies in computer science at the Vienna University of Technology, Austria, under the supervision of Günther R. Raidl and Ulrich Pferschy.
- 10/2000 – 06/2006:
  Studies of computer science at the Vienna University of Technology, Austria; graduation to "Diplom-Ingenieur" (MSc equivalent) with distinction.
- 09/1994 – 06/1999:
  Secondary College for Electronics (Special Training Focus Technical Computer Science) Braunau am Inn, Austria; school leaving examination passed with distinction.

Professional Activities

- since 09/2009:
  Research and teaching assistant, Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
- 07/2007 – 08/2009:
  Research assistant, School of Telematics / Network Engineering, Carinthia University of Applied Sciences, Klagenfurt, Austria; collaborating in the FFG research project "Netquest": Simulation and Optimization of Access Networks, see *http://www.fh-kaernten.at/netquest*
- 10/2006 – 08/2009:
  Research assistant, Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
- 08/2005 – 06/2007:
  Part time employment, Siemens AG Austria (PSE), Vienna, Austria; collaborating in the Celtic research project "Madeira": Network Management based on distributed paradigms, see *http://www.celtic-madeira.org*
- 2001 – 2004:
  Various internships and part time jobs as programmer and tutor
- 10/1999 – 09/2000:
  Compulsory community service (instead of military service) at "Lebenshilfe Oberösterreich", Regau, Austria

International Organisatorial Activities

- Reviewer for ICNAAM 2009, 7th International Conference on Numerical Analysis and Applied Mathematics
- Reviewer for Workshop Heuristic Problem Solving at Eurocast 2009 – 12th International Conference on Computer Aided Systems Theory
- Program committee member of Workshop on Heuristic Methods for the Design, Deployment, and Reliability of Networks (HEUNET 2008) at SAINT 2008 – The IEEE/IPSJ Symposium on Applications and the Internet.

Honors and Awards

- 02/2008: Celtic excellence award for project "Madeira".

List of Publications

Journal Articles

1. B. Hu, M. Leitner, and G. R. Raidl.
   The Generalized Minimum Edge Biconnected Network Problem: Efficient Neighborhood Structures for Variable Neighborhood Search. Networks, 55(3):257–275, 2010.
2. B. Hu, M. Leitner, and G. R. Raidl.
   Combining Variable Neighborhood Search with Integer Linear Programming for the Generalized Minimum Spanning Tree Problem. Journal of Heuristics, 14(5):473–499, 2008.

Book Chapters

3. M. Leitner, G. R. Raidl.
   Combining Lagrangian Decomposition with Very Large Scale Neighborhood Search for Capacitated Connected Facility Location. Post-Conference Book of the Eight Metaheuristics International Conference - MIC 2009, accepted 2010.

Conference and Workshop Proceedings

4. M. Leitner and G. R. Raidl.
   Strong Lower Bounds for a Survivable Network Design Problem. In Proceedings of International Symposium on Combinatorial Optimization (ISCO 2010), Hammamet, Tunisia, March 2010.
5. M. Leitner and G. R. Raidl.
   A Lagrangian Decomposition Based Heuristic for Capacitated Connected Facility Location. In S. Voß and M. Caserta, editors, Proceedings of the 8th Metaheuristic International Conference (MIC 2009), Hamburg, Germany, July 2009.
6. M. Leitner, G. R. Raidl, and U. Pferschy.
   Accelerating Column Generation for a Survivable Network Design Problem. In M. G. Scutellá et al., editors, Proceedings of the International Network Optimization Conference 2009, Pisa, Italy, April 2009.

7. M. Leitner and G. R. Raidl.
   Lagrangian Decomposition, Metaheuristics, and Hybrid Approaches for the Design of the Last Mile in Fiber Optic Networks. In M. J. Blesa et al., editors, Hybrid Metaheuristics 2008, volume 5296 of LNCS, pages 158–174, Malaga, Spain, October 2008. Springer-Verlag Berlin Heidelberg.

8. M. Leitner and G. R. Raidl.
   Variable Neighborhood Search for a Prize Collecting Capacity Constrained Connected Facility Location Problem. In Proceedings of the 2008 International Symposium on Applications and the Internet, SAINT 2008, pages 233-236, Turku, Finland, 2008. IEEE Computer Society.

9. G. Paulus, N. Prunner, C. Rauter, M. Prossegger, M. Leitner, J. Werner, and K. Rossegger.
   Entwicklung von kostenoptimierten räumlichen Szenarien für den strategischen Ausbau der Glasfasernetz-Infrastruktur am Beispiel eines Multi-Utility Unternehmens. 2. Forschungsforum der österreichischen Fachhochschulen (FFH 2008), 2008.

10. M. Leitner, B. Hu, and G. R. Raidl.
    Variable Neighborhood Search for the Generalized Minimum Edge Biconnected Network Problem. In B. Fortz, editor, Proceedings of the International Network Optimization Conference 2007, pages 69/1–6, Spa, Belgium, 2007.

11. L. Fallon, D. Parker, M. Zach, M. Leitner, and S. Collins.
    Self-Forming Network Management Topologies in the Madeira Management System. In A. K. Bandara et al., editors, Inter-Domain Management, volume 4543 of LNCS, pages 61–72. Springer-Verlag Berlin Heidelberg, 2007.

12. M. Leitner, P. Leitner, M. Zach, S. Collins, and C. Fahy.
    Fault Management based on peer-to-peer paradigms: A case study report from the CELTIC project Madeira. Proceedings of the 10th IFIP/IEEE Symposium on Integrated Management, pages 697–700, Munich, Germany, 2007.

13. B. Hu, M. Leitner, and G. R. Raidl.
    Computing Generalized Minimum Spanning Trees with Variable Neighborhood Search. In P. Hansen et al., editors, Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search, Tenerife, Spain, 2005.

Thesis

14. M. Leitner.
    Solving Two Generalized Network Design Problems with Exact and Heuristic

Methods. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, May 2006; supervised by G. R. Raidl and B. Hu.

Research Reports

15. M. Leitner, G. R. Raidl, and U. Pferschy
Branch-and-Price for a Survivable Network Design Problem. Technical Report TR 186-1-10-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2010.
16. M. Leitner and G. R. Raidl.
Branch-and-Cut-and-Price for Capacitated Connected Facility Location. Technical Report TR 186-1-10-01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2010.
17. M. Leitner and G. R. Raidl.
Combining Lagrangian Decomposition with Very Large Scale Neighborhood Search for Capacitated Connected Facility Location. Technical Report TR 186-1-09-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2009.
18. B. Hu, M. Leitner, and G. R. Raidl.
The Generalized Minimum Edge Biconnected Network Problem: Efficient Neighborhood Structures for Variable Neighborhood Search. Technical Report TR 186-1-07-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2007.
19. B. Hu, M. Leitner, and G. R. Raidl.
Combining Variable Neighborhood Search with Integer Linear Programming for the Generalized Minimum Spanning Tree Problem. Technical Report TR 186-1-06-01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2006.