



Jannik Matuschke

Network flows and network design

in theory and practice

Dissertation

TU Berlin

February 2014

Network flows and network design in theory and practice

vorgelegt von
Dipl. Math. Jannik Matuschke
Meschede

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss
Vorsitzender: Prof. Dr. Etienne Emmrich
Berichter: Prof. Dr. Martin Skutella
Prof. Dr. Britta Peis
Prof. Dr. Thomas McCormick

Tag der wissenschaftlichen Aussprache: 10. Dezember 2013

Berlin 2014
D83

Acknowledgements

This thesis is the outcome of many exciting years of work at COGA, TU Berlin. I very much enjoyed the fantastic atmosphere within the group and I am happy to have many of its former and current members as friends. In the following, I want to thank everybody who was involved in the creation of this thesis.

First and foremost, I am indebted to my advisors, Martin Skutella and Britta Peis, for their trust and support throughout the years and for the many things I learned from them, both as a scientist and as a person. I thank Martin for the immense freedom and independence he gave me in choosing my research topics. I thank Britta for her positive attitude and exceptional optimism, which helped me overcoming many obstacles. I am also grateful to Tom McCormick for kindly agreeing to referee this thesis.

Financially, this work has been supported by the European Regional Development Fund as part of the research project *MultiTrans*, by the German Research Foundation as part of the Priority Program *Algorithm Engineering* (SPP 1307), and by the Berlin Mathematical School, which supported many of my research stays and conference trips.

Many results in this thesis are fruits of *MultiTrans*, a research project for developing new approaches to optimization in transport logistics. Working with Tobias Harks, Felix König, Alexander Richter, and Jens Schulz as fellow project members was an invaluable experience and a great pleasure. I am also grateful to Cristina Hayden and Lars Stolletz from our project partner 4flow AG for many fruitful discussions on the topic of transport logistics, and to Michael Bastubbe and Hendrik Lüthen, who helped with the implementation of the software as student assistants of *MultiTrans*.

Beyond the project work, I am deeply indebted to Andreas Bley, Yann Disser, Jan-Philipp Kappmeier, Benjamin Müller, and Britta Peis, with whom I worked on the more theoretical topics of this thesis. Furthermore, I am grateful to Kristóf Bérczi and Júlia Pap, who introduced me to the topic of graph orientation and pointed me to the open problem that led to the results in Chapter 5.

Ashwin Arulselvan, Ágnes Cseh, Wiebke Höhn, and José Verschae are the silent guardians and watchful protectors of this thesis, who captured many errors and provided several helpful suggestions. Wiebke also has to be given credit as lead designer and L^AT_EX advisor of this thesis. My special thanks go to Dorothea Kiefer, Gabriele Klink, and Ralf Hoffmann for their restless efforts that make life at COGA so much easier.

During the time I worked on this thesis, I was lucky to visit many places around the world. Everywhere I experienced great hospitality. In particular, I want to say “Köszönöm!” to Erika and Kristóf and “¡Gracias!” to José and Natalia for their friendship and the wonderful time I had in Budapest and Santiago.

Finally, I want to thank Wiebke for being source of motivation and calming influence at the same time, and my parents for their unshakable belief in me and their unconditional support throughout my whole life.

Berlin, September 2013

Jannik Matuschke

Contents

1	Introduction	1
1.1	Preliminaries	3
1.2	Network flows	9
1.3	Network design	15
2	Abstract flows over time	19
2.1	Introduction to abstract flows	20
2.2	Time expansion of abstract networks	23
2.3	Construction of a maximum abstract flow over time	25
2.4	Storage at intermediate elements and the structure of abstract networks	29
2.5	Proof of abstract max flow/min cut over time	31
2.6	Conclusion	32
3	An integrated approach to tactical transportation planning	35
3.1	Introduction to transportation planning	37
3.2	Mathematical model	42
3.3	Tariff selection subproblem	50
3.4	Path-based local search	56
3.5	Mixed integer programming approaches	60
3.6	Computational study	64
3.7	Conclusion	70
4	Approximating combined location and network design problems	75
4.1	Introduction to combined facility location and network design	78
4.2	Capacitated location routing	84
4.3	Facility location with capacitated and length-bounded tree connections	110
4.4	Conclusion	124
5	Degree-constrained orientations of embedded graphs	127
5.1	Introduction to graph orientation	128
5.2	Orientations with fixed in-degrees	131
5.3	Orientations with upper and lower bounds	135
5.4	Conclusion	139
	Notation index	141
	Subject index	143
	References	147

Chapter 1

Introduction

Networks play a vital role in our world. We exchange messages through communication networks, enormous flows of material are shipped through world-wide logistics networks, and even our society is organized in social networks on various levels, both professionally and privately. These are only a few examples from a seemingly infinite list of networks affecting our daily lives.

In all these examples, networks represent the central infrastructure. The task of planning and modifying this infrastructure is known as *network design*. On an abstract level, a *network* consists of a set of *nodes* and a set of *links* that connect these nodes. Using this terminology, network design problems ask for a set of links to be installed in the network in order to fulfill certain problem-specific requirements, often related to connectivity of the nodes. The first application of optimization algorithms for network design dates back at least to the 1920s, when Borůvka [Bor26] studied the classic *minimum spanning tree problem* in the context of planning the electrification of south-western Moravia [NN12]. Since then, the field has evolved rapidly from simple connectivity problems to ever more general network design models that are studied in literature and applied in practice, e.g., in logistics [MW84, Cra00] and telecommunication [GMS95, SS98].

One of the most important features of networks is their ability of modeling the transportation of objects or information. A *network flow* assigns values to the links of the network indicating quantities moving from one node to another. Historically, network flows were first studied in the context of transport logistics [Sch02], starting with Tolstói's study of transportation problems in the Soviet railway network [Tol30]. Network flow theory also has proven useful as a device for obtaining important structural insights, many of them related to Ford and Fulkerson's seminal max flow/min cut theorem [FF56], which establishes that the maximum value of a flow between two nodes equals the minimum capacity of a cut separating the nodes. Today, network flows are a pervasive concept in combinatorial optimization and its application areas, such as production planning [Eva77, Sha93], traffic modeling [DS69, Mag84], evacuation management [DGK⁺10], and many others, as, e.g., described in [AMO93].

Network design and network flow theory each constitute interesting fields of research on their own, but they are also closely connected. In fact, several of the above applications are based on *capacitated network design* models, combinations of network design and network flow. While these models are extremely flexible and can be adjusted to fit many real-world applications, they also impose considerable computational challenges.

In this thesis, we investigate several network design and network flow problems and devise algorithms for their solution, aiming both for theoretical insights and practical

applicability. Our results cover structural theorems on the properties of networks and their generalizations (Chapters 2 and 5) as well as models and algorithms for concrete optimization problems in logistics and telecommunication (Chapters 3 and 4).

Outline

The thesis consists of five chapters, covering different topics from the theory and practice of network flows and network design. We shortly outline the contents here.

Chapter 1: Introduction. In the remainder of this chapter, we provide the preliminary concepts and notions this thesis is based on. In particular, we give short introductions to network flows and network design, the two central themes of this thesis. Literature concerned with more specific topics is covered within the respective chapters.

Chapter 2: Abstract flows over time. In Chapter 2, we introduce and investigate *abstract flows over time*, which are based on a generalization of classic network flows proposed by Hoffman [Hof74]. In this generalization, the underlying network is replaced by an abstract system of ‘paths’ that only fulfills a certain switching axiom—an abstraction of the behavior of crossing paths in the network. We show that the presence of this axiom alone suffices to ensure max flow/min cut properties in the time expansion of the system. This yields the max flow/min cut theorem for abstract flows over time as our main result in this chapter. Besides gaining new insight into the structural properties of networks, our research is motivated by an interest in solving general *dynamic packing problems*, i.e., packing problems with a component of time.

Chapter 3: An integrated approach to tactical transportation planning in logistics networks. In Chapter 3, we propose a new model for *transportation planning*, i.e., the optimization of freight transportation in logistics networks. The model is based on a multi-dimensional capacitated network design formulation that precisely captures the structure of transportation tariffs offered by logistics carriers in practice. It employs a cyclic expansion of the network in order to integrate inventory decisions and frequencies of shipments, two important aspects of transportation planning. In addition to the model, we propose various heuristic methods for solving the resulting optimization problem, most notably a local search procedure based on path decomposition of network flows and an aggregated mixed integer programming formulation. Our algorithms rely on subroutines that optimize the tariff selection on individual links in the network—a task that constitutes an optimization problem of own interest. The model has been developed in close collaboration with logistics experts at 4flow AG, a logistics consultancy company. We evaluate the model and our algorithms on a broad set of instances based on real-world logistics networks from ongoing and recent customer projects of 4flow AG. The computational study shows that most of our solutions are within 10% of optimality.

Chapter 4: Approximating combined location and network design problems. In Chapter 4, we discuss problems that combine location decisions, as known from the classic facility location problem, with network design. We investigate an algorithmic framework for combining multiple lower bounds into approximation algorithms for such integrated problems. We use this framework to derive approximation algorithms for two practically relevant optimization problems. The first is *capacitated location routing*, an

important problem in transport logistics that combines facility location and capacitated vehicle routing. We obtain constant factor approximation guarantees for several variants of this problem, and we also evaluate our algorithm in a computational study on instances from the literature and additional large-scale randomly generated instances. It turns out that the performance of our algorithm in practice exceeds the theoretical worst-case approximation guarantee by far. The second problem we study in this chapter is *facility location with capacitated and length-bounded trees*. It is motivated by the design of optical access networks in telecommunication. We derive bifactor approximation algorithms for the problem that provide different approximation factors for length bound and solution cost, with improved factors for two important special cases.

Chapter 5: Degree-constrained orientations of embedded graphs. In Chapter 5, we study the problem of orienting the edges of an embedded graph in such a way that the resulting digraph fulfills given in-degree specifications both for the vertices and for the faces of the embedding. This *primal-dual orientation problem* was first proposed by Frank [Fra10] for the case of planar graphs, in conjunction with the question for a good characterization of the existence of such orientations. We answer this question by showing that a planar embedding has a feasible orientation if and only if the primal and dual in-degree specifications induce a partition of the edges into two sets and the resulting orientations of these sets are compatible. This implies that the solution is unique if it exists and that it can be constructed by combining a primally feasible orientation and a dually feasible orientation. For the general case of arbitrary embeddings, we show that the number of feasible orientations is bounded by 2^{2g} , where g is the genus of the embedding. Our proof also yields a fixed-parameter algorithm for determining all feasible orientations in time $\mathcal{O}(2^{2g}|E|^2 + |E|^3)$. In contrast to these positive results, however, we also show that the problem becomes *NP*-complete even for a fixed genus if only upper and lower bounds on the in-degrees are specified instead of exact values.

1.1 Preliminaries

In this section, we introduce some of the general concepts and notations used throughout this thesis, e.g., concerning algorithms and graphs. While we assume the reader to be familiar with these subjects, we give pointers to textbooks and standard literature introducing the corresponding topics. More specific notions that are only relevant for a particular topic are introduced in the corresponding chapter. An introduction to all topics discussed in this section can be found in the book by Korte and Vygen [KV12]; for more comprehensive information also see the book by Schrijver [Sch03].

Notation index. Most of the notation used in this thesis is standard in combinatorial optimization literature. A concise list of the occurring symbols and references to their definitions can be found in the notation index at the end of this thesis.

1.1.1 Algorithms and complexity theory

The design and analysis of *algorithms* constitutes a central part of research in combinatorial optimization. We give a short overview of the different concepts of optimization algorithms occurring in this thesis.

Polynomial time algorithms and complexity theory

Since algorithms are supposed to efficiently solve problems, the notion of *running time* plays a crucial role. Closely related is the concept of *complexity theory*, which aims at classifying problems according to their computational tractability. We sketch the basic definitions and refer the unfamiliar reader to the book by Wegener [Weg05] for a comprehensive introduction.

Encoding size and running time. Let $\langle I \rangle$ denote the *encoding size* of an instance I of a problem, i.e., the number of bits needed to store the information defining that instance. The running time of an algorithm is typically measured as the *worst-case performance* depending on the size of the input, i.e., as the function $f(n)$ specifying the maximum number of elementary operations it needs to solve any given instance I of the problem with $\langle I \rangle \leq n$. In theoretical computer science literature, polynomial running time is the commonly accepted threshold to efficiency. When stating a more precise bound on the running time, constant factors are often ignored, using the \mathcal{O} -notation: For two functions $f, g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$, we write $f = \mathcal{O}(g)$ if there is a $c \in \mathbb{R}_+$ with $f(n) \leq cg(n)$ for all $n \in \mathbb{R}_+$.

Complexity classes and hardness. Unfortunately, for many fundamental problems no exact polynomial time algorithm has been found to this point despite intense efforts. While it also appears to be very hard to rigorously rule out the existence of such algorithms, complexity theory gives a tool for classifying the *hardness* of a problem by linking the existence of a polynomial time algorithm for the problem to the collapse of certain complexity classes. Two basic complexity classes are P , the set of decision problems that allow for a polynomial time algorithm, and NP , the set of decision problems that have a polynomial size certificate for 'yes'-instances. An optimization problem is *NP-hard*, if any problem in NP can be reduced to this problem. In this case, a polynomial time algorithm for the problem implies $P = NP$, a condition which is commonly considered unlikely to be fulfilled. Other typical implications of this type occurring throughout this thesis are $NP \subseteq DTIME(n^{\text{polylog}(n)})$ or $NP \subseteq ZTIME(n^{\text{polylog}(n)})$, implying the existence of an algorithm with deterministic or expected quasi-polynomial time, respectively, for every problem in NP . An extensive list of NP -hardness results can be found in the book by Garey and Johnson [GJ79].

Minimum weighted set cover. As an example for an NP -hard optimization problem, we introduce the *minimum weighted set cover problem*, which will appear several times throughout this thesis.

Problem: Minimum weighted set cover

Input: A ground set E , a family of sets $\mathcal{S} \subseteq 2^E$, and weights $w \in \mathbb{Q}_+^{\mathcal{S}}$.

Task: Find a subset $\mathcal{S}' \subseteq \mathcal{S}$ such that $E = \bigcup_{S \in \mathcal{S}'} S$, minimizing the cost $\sum_{S \in \mathcal{S}'} c(S)$.

This problem is already NP -hard when every element appears in at most two sets—a special case that is known as *vertex cover problem*; see Karp's list of NP -complete problems [Kar72].

Other algorithmic concepts

While hardness results as described above make the existence of exact polynomial time algorithms for some problems unlikely, there are several other concepts that allow us to achieve both theoretically meaningful results and practical solution methods for such problems.

Approximation algorithms. An α -*approximation algorithm* for a minimization¹ problem is a polynomial time algorithm that, given instance I of the problem, returns a feasible solution S_I of I such that

$$c(S_I) \leq \alpha OPT(I)$$

where $c(S_I)$ is the cost of solution S_I and $OPT(I)$ is the cost of an optimal solution of I . The *approximation factor* α can be a constant or a function depending on a parameter of the instance. A *polynomial time approximation scheme* (PTAS) is an algorithm that computes for any given $\varepsilon > 0$ a $(1 + \varepsilon)$ -approximate solution in time polynomial in the input size for fixed values of ε . The approximation scheme is a *fully polynomially time approximation scheme* (FPTAS), if its running time is polynomial in the input size and $\frac{1}{\varepsilon}$. As for exact algorithms, hardness results can also relate to the existence of approximation algorithms. For example, the set cover problem introduced above does not allow for an approximation algorithm with a factor better than $\ln |E|$, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$, as shown by Feige [Fei98]. For a detailed introduction to approximation algorithms see the books by Hochbaum [Hoc96], Vazirani [Vaz01], and Williamson and Shmoys [WS11].

Fixed-parameter algorithms. In many applications, it is a valid assumption that certain parameters of the input stay small, e.g., the number of different cable types installed in a communication network will typically be very limited even while the size of the network itself might be enormous. A problem is *fixed-parameter tractable* (FPT) in a parameter p depending on the instance, if there is an algorithm that solves the problem in time $f(p(I)) \cdot \phi(|I|)$, where $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a function and ϕ is a polynomial function. There also is a whole hierarchy of complexity classes related to fixed-parameter tractability and a concept for formalizing data reductions in preprocessing, called *kernelization*—however, we will not make use of these in this thesis. For an introduction to the topic see the book by Niedermeier [Nie06].

Heuristics. Not every practically successful algorithm can be cast into a rigorous theoretical framework. *Heuristics* are algorithms without proven polynomial running time or a priori approximation guarantee. Besides specialized heuristics designed to solve a particular problem, *meta-heuristics*, in particular *local search* and its generalizations like *tabu search* and *genetic algorithms*, have been devised to solve general optimization problems without deeper knowledge of the particular problem structure. Instead of deriving mathematical proofs on the performance of the algorithm, the suitability of a heuristic for a practical application is verified by extensive computational studies. Examples for

¹The definition for maximization problems is analogous. However, we will only encounter approximation algorithms for minimization problems throughout this thesis.

very successful specialized heuristics include Dantzig's *simplex method* for linear programming² [Dan51b] and the *Lin-Kernighan heuristic* for TSP [Hel00]. See the book by Aarts and Lenstra [AL97] for an introduction to local search techniques.

1.1.2 Linear and integer programming

Linear and integer programming are very general techniques for modeling and solving optimization problems. They ask for maximizing or minimizing a linear objective subject to a system of linear inequalities, where the solution is required to be integral in the case of integer programming.

Problem: Linear programming (LP)

Input: A matrix $A \in \mathbb{Q}^{m \times n}$, vectors $b \in \mathbb{Q}^m$ and $c \in \mathbb{Q}^n$.

Task: Find $x^* \in P := \{x \in \mathbb{Q}^n : Ax \leq b, x \geq 0\}$ maximizing $c^T x^*$, or decide that the maximum is infinite, or that $P = \emptyset$.

An integer program (IP) additionally requires x to be integral. If only some of the variables are required to be integral, the problem is a mixed integer program (MIP).

Both linear and integer programming take a central place in combinatorial optimization and they are also closely tied to polyhedral combinatorics. Thanks to their versatility and the availability of advanced and efficient solvers, they are widely spread in practice. Many theoretical insights obtained in the study of linear and integer programming are now central parts of solver packages like CPLEX [IBM], Gurobi [Gur], or SCIP [Ach09]. For a comprehensive overview of the theoretic foundations see the textbook by Schrijver [Sch98]. We will shortly introduce those concepts used within this thesis.

Algorithms. Several polynomial time algorithms for linear programming have been developed, the first being the *ellipsoid method* by Khachiyan [Kha80]. Still, the most widely used algorithm for solving linear programs in practice remains the *simplex method* by Dantzig [Dan51b], despite the fact that instances with exponential running time are known for all variants in use. Besides this, *interior point methods* provide a way to efficiently solve linear programs in practice while maintaining theoretically proven polynomial running time as first shown by Karmarkar [Kar84]. In contrast, integer programming is known to be *NP-hard*. In practice, *branch and bound algorithms*, combined with cutting plane methods are able to solve even large mixed integer programs in reasonable time; see, e.g., the historical survey by Bixby [Bix12]. In many cases, this can be further improved by exploiting problem-specific structures, e.g., for setting up column generation [FF58a, DW60, LD05] or employing decomposition techniques like Benders decomposition [Ben62], which is particularly useful for capacitated network design problems; also see Section 3.1.2.

²While the simplex method is both an exact algorithm providing many additional benefits such as the computation of a basic solution and a useful tool for obtaining structural insights in linear programming and polyhedral theory, it also falls under the general definition of a heuristic as an algorithm without polynomial running time.

LP duality. One of the most fundamental concepts in linear programming is *duality*; see, e.g., [Sch98]. The duality theorem of linear programming, which was first proven by von Neumann [vN47], states that

$$\max \{c^T x : Ax \leq b, x \geq 0\} = \min \{b^T y : A^T y \geq c, y \geq 0\}$$

where the two LPs occurring in the equation are known as *primal* and *dual* program. LP duality can be used to prove various other duality theorems such as the max flow/min cut theorem in network flow theory, it is useful in the design of approximation algorithms—so-called primal-dual schemes—and it also enables decomposition techniques such as Benders decomposition mentioned above.

The equivalence of optimization and separation. The *separation problem* corresponding to an LP asks whether a given vector x is a feasible solution to this LP or, if this is not the case, for a constraint violated by x . Based on Khachiyan’s work on the ellipsoid method [Kha80], Grötschel, Lovasz, and Schrijver [GLS88] showed that the optimal solution of an LP can be found by solving the separation problem for the same LP for a polynomial number of input vectors and vice versa. As a consequence, LPs can be solved in time polynomial in the number of variables, even when there are exponentially many constraints, as long as these constraints can be separated efficiently.

LP relaxation. Many hard optimization problems can be formulated as integer or mixed integer programs. While no polynomial time algorithm is known for finding optimal solutions to these formulations, one can obtain a lower bound (in the case of minimization) on the value of the optimal solution by relaxing the integrality condition and solving the resulting LP. Such lower bounds can be used for obtaining a priori guarantees for approximation algorithms, for pruning the tree of a branch and bound procedure, or for giving a posteriori guarantees for heuristic solutions.

Total dual integrality. A system of linear inequalities $Ax \leq b, x \geq 0$ with $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Z}^m$ is *totally dual integral* (TDI), if the dual program $\min\{b^T y : A^T y \geq c, y \geq 0\}$ has an integral optimal solution for every $c \in \mathbb{Z}^n$ for which an optimal solution exists, implying that also the primal program $\max\{c^T x : Ax \leq b, x \geq 0\}$ has an integral optimal solution for every $c \in \mathbb{Z}^n$; see [Sch98] for details. Such classes of LPs are particularly interesting, as solving the linear program is equivalent to solving the corresponding integer program in this case. The concept of total dual integrality goes back to Hoffman [Hof74] and was later formalized by Edmonds and Giles [EG77]. An important special case of the concept is the max flow/min cut theorem of Ford and Fulkerson [FF56].

1.1.3 Graphs

Networks are the central structure occurring in this thesis. Their representation relies on the standard notions of graphs and digraphs introduced in this section.

Undirected graphs. A *graph* $G = (V, E)$ consists of a set of *vertices* V and a set of *edges* E . Each edge $e \in E$ is associated with an unordered pair of vertices $\psi(e) = \{v, w\}$, called the *end points* of e . If there is no ambiguity, we identify e with $\psi(e)$. An edge e is a *loop*, if its endpoints are identical, i.e., $\psi(e) = \{v, v\}$. Two edges e, f are *parallel*

if $\psi(e) = \psi(f)$. An edge e and a vertex v are *incident*, if $v \in \psi(e)$. For a set of edges F , we define $V(F) := \bigcup_{e \in F} \psi(e)$ to be the set of vertices incident to the edges in F .

Directed graphs. A *directed graph* (also known as *digraph*) $D = (V, A)$ consists of a set of vertices V , which are also referred to as *nodes* in this context, and a set of *arcs* A . Every arc $a \in A$ has a start node $\text{tail}(a) \in V$ and an end node $\text{head}(a) \in V$. If there is no ambiguity, we identify a with the tuple $(\text{tail}(a), \text{head}(a))$. An arc $a \in A$ is a *loop* if $\text{tail}(a) = \text{head}(a)$. Two arcs $a, b \in A$ are *parallel*, if $\text{tail}(a) = \text{tail}(b)$ and $\text{head}(a) = \text{head}(b)$. They are *anti-parallel*, if $\text{tail}(a) = \text{head}(b)$ and $\text{head}(a) = \text{tail}(b)$.

Subgraphs. Let $G = (V, E)$ be a graph. For a set of vertices $S \subseteq V$, we define the set of edges $E[S] := \{e \in E : \psi(e) \subseteq S\}$ with both endpoints in S . A *subgraph* of G is graph $G' = (S, F)$ with $S \subseteq V$ and $F \subseteq E[V']$. In particular, we define $G[S] := (S, E[S])$ to be the *subgraph induced by S* . The notions of subgraph and induced subgraph for a digraph are defined in analogy to the undirected case.

From directed to undirected graphs and vice versa. The *underlying undirected graph* of a digraph $D = (V, A)$ is the graph $U(D) = (V, U(A))$, where $U(A) = \{e_a : a \in A\}$ with $\psi(e_a) = \{\text{tail}(a), \text{head}(a)\}$. Conversely, for an undirected graph $G = (V, E)$, the corresponding *bidirected digraph* $B(G) = (V, B(E))$ is constructed by replacing each edge e with two oppositely directed arcs a_e^+ and a_e^- , i.e., $B(E) = \{a_e^+, a_e^- : e \in E\}$ with $\text{tail}(a_e^+) = \text{head}(a_e^-)$, $\text{head}(a_e^+) = \text{tail}(a_e^-)$, and $\psi(e) = \{\text{tail}(a_e^+), \text{head}(a_e^+)\}$ for all $e \in E$.

Walks, paths, and cycles. Let $G = (V, E)$ be a graph and let $s, t \in V$. A sequence of edges (e_0, \dots, e_k) is an *s-t-walk*, if there is a sequence of vertices (v_0, \dots, v_{k+1}) such that $\psi(e_i) = \{v_i, v_{i+1}\}$ for $i \in [k]$. An *s-t-walk* is an *s-t-path*, if the edges e_0, \dots, e_k are pairwise distinct. An *s-t-walk* is *closed*, if $s = t$. A closed walk is a *cycle*, if the edges e_0, \dots, e_k are pairwise distinct. An *s-t-path* or cycle is *simple*, if the vertices v_0, \dots, v_k are pairwise distinct.

A sequence of arcs (a_0, \dots, a_k) in a digraph $D = (V, A)$ is an *s-t-walk*, *s-t-path*, or cycle, if it corresponds to an *s-t-walk*, *s-t-path* or cycle in $U(D)$, respectively. It is *directed*, if $\text{head}(a_i) = \text{tail}(a_{i+1})$ for all $i \in [k-1]$.

Connected components. A graph $G = (V, E)$ is *connected* if there is a *v-w-walk* for all $v, w \in V$. A digraph $D = (V, E)$ is *strongly connected* if there is a directed *v-w-walk* and a directed *w-v-walk* for all $v, w \in V$. A (*strongly*) *connected component* of a (di-)graph is a maximal (strongly) connected subgraph.

Cuts and degrees. Let $G = (V, E)$ be a graph and $S \subseteq V$. The *cut induced by S* is the set

$$\delta_G(S) := \{e \in E : \psi(e) \cap S \neq \emptyset \text{ and } \psi(e) \cap V \setminus S \neq \emptyset\}.$$

A cut is *simple*, if both $G[S]$ and $G[V \setminus S]$ are connected. Let $D = (V, A)$ be a digraph and $S \subseteq V$. We define

$$\delta_D^+(S) := \{a \in A : \text{tail}(a) \in S \text{ and } \text{head}(a) \in V \setminus S\}$$

and

$$\delta_D^-(S) := \{a \in A : \text{tail}(a) \in V \setminus S \text{ and } \text{head}(a) \in S\}.$$

The cut induced by S is $\delta_D(S) = \delta_D^+(S) \cup \delta_D^-(S)$. The cut is *directed*, if $\delta_D^+(S) = \emptyset$ or $\delta_D^-(S) = \emptyset$. We omit the subscript G and D , respectively, if there is no ambiguity. For $v \in V$, we will also write $\delta(v)$, $\delta^+(v)$, and $\delta^-(v)$ for $\delta(\{v\})$, $\delta^+(\{v\})$, and $\delta^-(\{v\})$, respectively. The *degree* of a vertex $v \in V$ is $|\delta(v)|$, its *out-degree* is $|\delta^+(v)|$, and its *in-degree* is $|\delta^-(v)|$.

Trees. Let $G = (V, E)$ be a graph. A tree is a set of edges $T \subseteq E$ such that the subgraph $(V(T), T)$ is connected and contains no cycles. A tree T *spans* a set of vertices S if $S \subseteq V(T)$. A *spanning tree* of G is a tree T such $V(T) = V$. Let $D = (V, A)$ be a digraph. An *out-tree* in D rooted at r is a set of arcs $T \subseteq A$ such that the subgraph $(V(T), T)$ is connected, $|\delta^-(r) \cap T| = 0$, and $|\delta^-(v) \cap T| = 1$ for every vertex $v \in V(T) \setminus \{r\}$.

1.2 Network flows

Network flows are one of the two main themes of this thesis. We give a short introduction to the topic, with a focus on those definitions and results that are used throughout the thesis, specifically the *max flow/min cut theorem*, *minimum cost flows*, and *flows over time*. Besides the concepts that are introduced here, there are of course many more variants of network flows, such as *generalized flows* or *abstract flows*. The latter will play a central role in Chapter 2 of this thesis and will be discussed extensively in that chapter. For more details on network flows, see the comprehensive textbook by Ahuja, Magnanti, and Orlin [AMO93].

1.2.1 Basic definitions

Flow conservation. Let $D = (V, A)$ be a digraph. For $v \in V$ and $x \in \mathbb{Q}_+^A$ define the *excess* of v with respect to x by

$$\text{ex}(x, v) := \sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a).$$

For sets $S, T \subset V$, an *S - T -flow* is a vector $x \in \mathbb{Q}_+^A$ such that $\text{ex}(x, v) \geq 0$ for all $v \in V \setminus S$ and $\text{ex}(x, v) \leq 0$ for all $v \in V \setminus T$. The vertices in S are called *sources*, the vertices in T are called *sinks*. If $S = \{s\}$ and $T = \{t\}$ for some vertices $s, t \in V$, we call x an *s - t -flow*. Note that all vertices that are neither sources nor sinks have to fulfill *flow conservation*, i.e., their excess must be 0. For $b \in \mathbb{Q}_+^V$, a *b -flow* is a vector $x \in \mathbb{Q}_+^A$ such that $\text{ex}(x, v) + b(v) = 0$ for all $v \in V$.

Flow decomposition. Any *S - T -flow* $x \in \mathbb{Q}_+^A$ can be alternatively represented by a decomposition into flow on paths from S to T and flow on cycles. Let \mathcal{P} be the set of simple directed *s - t -paths* in D for vertices $s \in S$ and $t \in T$, and let \mathcal{C} be the set of simple cycles in D . A *decomposition* of x is a vector $\tilde{x} \in \mathbb{Q}_+^{\mathcal{P} \cup \mathcal{C}}$ with $\sum_{P \in \mathcal{P} \cup \mathcal{C}: a \in P} \tilde{x}(P) = x(a)$. It is a well-known fact that for every flow x a decomposition of size at most $|A|$ can be found in time $\mathcal{O}(|V||A|)$.

1.2.2 The maximum flow problem and max flow/min cut

One of the most fundamental problems in network flow theory is the *maximum flow problem*. It asks for a flow of maximum value from a source to a sink without violating given capacities on the arcs. The problem was extensively studied by Ford and Fulkerson [FF56, FF57, FF62], who contributed many important concepts such as residual networks, augmenting paths, and the famous max flow/min cut theorem—results that now build the fundamentals of network flow theory.

Problem: Maximum flow

Input: A digraph $D = (V, A)$, a source $s \in V$, a sink $t \in V$, and capacities $u \in \mathbb{Q}_+^A$.

Task: Find an s - t -flow $x \in \mathbb{Q}_+^A$ such that $x(a) \leq u(a)$ for all $a \in A$, maximizing the flow value $\text{ex}(x, t)$.

Path formulation and minimum cuts. Using flow decomposition, the maximum flow problem can alternatively be stated in terms of a *path formulation*—note that flow on cycles does not contribute to the value of the flow and can therefore be ignored. The resulting linear program reads

$$\begin{aligned} \max \quad & \sum_{P \in \mathcal{P}} x(P) \\ \text{s.t.} \quad & \sum_{P \in \mathcal{P}: a \in P} x(P) \leq u(a) \quad \forall a \in A \\ & x(P) \geq 0 \quad \forall P \in \mathcal{P} \end{aligned}$$

where \mathcal{P} is the set of simple directed s - t -paths in D . The dual of this program is

$$\begin{aligned} \min \quad & \sum_{A \in A} u(a)y(a) \\ \text{s.t.} \quad & \sum_{a \in P} y(a) \geq 1 \quad \forall P \in \mathcal{P} \\ & y(a) \geq 0 \quad \forall a \in A. \end{aligned}$$

Note that every binary solution $y \in \{0, 1\}^A$ of the dual corresponds to a set $C \subseteq A$ such that $C \cap P \neq \emptyset$ for every s - t -path $P \in \mathcal{P}$. Such a set C separating s from t is called an s - t -cut and $\sum_{a \in C} u(a)$ is called the *capacity* of C . Note that every inclusionwise minimal s - t -cut corresponds to a cut $\delta^+(S)$ induced by a set of vertices $S \subseteq V \setminus \{t\}$ with $s \in S$ as introduced in the previous section.

Max flow/min cut. Ford and Fulkerson's [FF56, FF57] famous max flow/min cut theorem states that the maximum value of an s - t -flow is equal to the minimum capacity of an s - t -cut, i.e., the dual LP described above always has an integral optimal solution—note that this also implies the existence of an integral maximum s - t -flow in case of integral capacities. Ford and Fulkerson's proof of the theorem is constructive, yielding an algorithm for constructing both a maximum flow and a minimum cut. The theorem establishes a

connection between network flows and structural results in graph theory, e.g., by generalizing Menger's theorem [Men27] and the König-Egerváry theorem [Kön31]. It also enables modeling connectivity requirements in network design problems both by using flow formulations as well as cut constraints.

The residual network. Many flow algorithms use the so-called *residual network*. The residual network of a flow x with respect to capacities u is the network $D_{u,x} = (V, A_{u,x})$ that contains an arc $a^+ \in A_{u,x}$ for every arc $a \in A$ with $x(a) < u(a)$ and an arc $a^- \in A_{u,x}$ for every arc $a \in A$ with $x(a) > 0$. The arc a^+ is parallel to a , the arc a^- is anti-parallel to a . Note that both a^+ and a^- might exist for the same arc $a \in A$. Ford and Fulkerson [FF57] established the following optimality criterion for the maximum flow problem: An s - t -flow x has maximum value if and only if there is no directed s - t -path in $D_{u,x}$.

Efficient algorithms for the maximum flow problem. Ford and Fulkerson [FF56, FF57] proposed an *augmenting path algorithm* for solving the maximum flow problem. This algorithm iteratively increases the value of the flow by sending flow along an s - t -path in the residual network until no such path exists anymore. While this algorithm in its most basic form does not run in polynomial time, several polynomial time algorithms have been developed from the idea of augmenting flow along paths, e.g., by Dinic [Din70] and Edmonds and Karp [EK72]. Other notable solution techniques include the preflow-push algorithm by Goldberg and Tarjan [GT88] and the capacity scaling algorithm by Fujishige [Fuj03]. Recently, Orlin [Orl13] showed that the maximum flow problem can be solved in time $\mathcal{O}(|V||E|)$. There are more efficient algorithms for special cases. For example, Borradaile and Klein [BK09] showed that a maximum flow in a planar graph can be found in time $\mathcal{O}(|E| \log |E|)$. Another example are networks with unit capacities, where Dinic's algorithm achieves a running time of $\mathcal{O}(\min\{|V|^{\frac{2}{3}}|E|, |E|^{\frac{3}{2}}\})$, as shown by Even and Tarjan [ET75].

1.2.3 Minimum cost flow and transportation problems

As already pointed out earlier, network flows were first considered in the context of transport logistics. Of course, the main goal of optimization in logistics is to minimize cost. A simple and well-studied model that considers network flows with linear costs is the *minimum cost flow problem*.

Problem: Minimum cost flow

Input: A digraph $D = (V, A)$, node balances $b \in \mathbb{Q}^V$, capacities $u \in \mathbb{Q}_+^A$, and costs $c \in \mathbb{Q}^A$.

Task: Find a b -flow $x \in \mathbb{Q}_+^A$ with $x(a) \leq u(a)$ for all $a \in A$, minimizing the cost $\sum_{a \in A} c(a)x(a)$, or state that there is no such flow.

The special case of the minimum cost flow problem where $u = \infty$ and D is a bipartite graph with the sources on one side and the sinks on the other is known as the *transportation problem*.

It can be shown that any instance of the minimum cost flow problem can be transformed into an equivalent instance of the transportation problem. The transportation problem was formally introduced by Hitchcock [Hit41], but already appeared in a study by Tolstoï [Tol30], who investigated freight transportation in the Soviet railway system; also see the historical notes by Schrijver [Sch02].

Algorithms. Tolstoï [Tol30] already made use of the following optimality criterion—although he did not prove it formally: A b -flow x is an optimal solution if and only if there is no directed cycle of negative cost in the residual network induced by x . This motivates the *negative cycle canceling algorithm*, which, starting at some b -flow, iteratively sends flow along a cycle of negative weight in the residual network, until no such cycle exists anymore. Goldberg and Tarjan [GT89] showed that choosing a cycle of minimum mean weight in every iteration yields a strongly polynomial time algorithm for solving the minimum cost flow problem. Another approach for finding a minimum cost flow is the *successive shortest path algorithm*, which was independently discovered by Jewell [Jew58], Iri [Iri60], and Busaker and Gowen [BG61]. Starting at a flow with arbitrary excess that does not induce any negative residual cycles, it iteratively sends flow along a shortest path from a source to a sink in the residual network. While this method only runs in pseudo-polynomial time, it is interesting because of its simplicity and practical efficiency. It also inspires our heuristics for the tactical transportation planning problem in Chapter 3. Probably the most popular algorithm for solving minimum cost flow problems in practice is the *network simplex method*, a specialization of the simplex method for network flows suggested by Dantzig [Dan51a].

Non-linear costs. Of course, non-linear cost functions also occur very frequently in practice, prominently including convex, concave, and fixed costs. Convex cost functions can be approximately modeled using parallel arcs of increasing cost; see Chapter 14 of [AMO93] for details. Models and algorithms for flows with concave cost functions are discussed, e.g., by Guisewite and Pardalos [GP90]. Concave costs are also closely related to fixed costs. These in turn are commonly modeled using capacitated network design formulations. We therefore discuss flow problems with fixed charges in Section 1.3.

1.2.4 Multi-commodity flows

In the flow problems discussed above, all flow was of the same type, i.e., the demand of any sink could be satisfied using the supply of any source. We now consider the case that several flows of different commodities move through the same network sharing the same capacities. We will restrict ourselves to the *minimum cost multi-commodity flow problem*, as this most closely resembles our usage of the multi-commodity flows in the transportation planning model in Chapter 3 and in the LP relaxation for location routing in Chapter 4.

Minimum cost multi-commodity flows. Let $D = (V, A)$ be a digraph. Given a set K of commodities and a vector $b \in \mathbb{Q}^{K \times V}$, a *multi-commodity b -flow* in D is a vector $x \in \mathbb{Q}_+^{K \times A}$ such that $x_i \in \mathbb{Q}_+^A$ is a b_i -flow for all $i \in K$.

Problem: Minimum cost multi-commodity flow

Input: A digraph $D = (V, A)$, a set of commodities K , node balances $b \in \mathbb{Q}^{K \times V}$, capacities $u \in \mathbb{Q}_+^A$, and costs $c \in \mathbb{Q}^{K \times A}$.

Task: Find a multi-commodity b -flow $x \in \mathbb{Q}_+^{K \times A}$ fulfilling the constraint $\sum_{i \in K} x_i(a) \leq u(a)$ for all $a \in A$, minimizing the cost $\sum_{i \in K} \sum_{a \in A} c_i(a)x_i(a)$, or state that there is no such flow.

Unfortunately, many of the results known for single-commodity flows fail to hold for the case of multiple commodities. In particular, there is no equivalent to the max flow/min cut theorem (beyond standard LP duality) or to the negative cycle criterion, and the existence of an integral optimal flow in case of integral capacities is no longer guaranteed.

Algorithms. Clearly, the minimum cost multi-commodity flow problem can be formulated as an LP and thus it can be solved in polynomial time. However, no exact combinatorial algorithm is known for the problem. Garg and Könemann [GK07] proposed an FPTAS based on the path formulation of the problem and its dual, which due to its efficiency is also used as a subroutine in many software packages in practice, e.g., in the context of VLSI design [Vyg04].

1.2.5 Flows over time

In many applications of network flows, time plays a crucial role. The concept of *flows over time* was already investigated by Ford and Fulkerson [FF58b], who showed how to model transit times and time-dependent flow rates within a so-called time-expanded network and how to construct a maximum flow over time by temporally repeating a static minimum cost flow. Since then, numerous results on different variants of flow over time problems have emerged. For a comprehensive overview, we refer the reader to the introductory article by Skutella [Sku09].

Flows over time and the related concept of time-expanded networks, which is described in detail below, will play an important role at several places throughout this thesis. In Chapter 2, we will generalize Ford and Fulkerson's max flow/min cut over time theorem and the corresponding algorithmic techniques to the setting of abstract flows, showing that all their results in fact rely on a very simple switching axiom fulfilled by the system of s - t -paths in a network. In Chapter 3, we propose a model for tactical transportation planning, using cyclic holdover arcs for modeling the frequency of shipments. In Chapter 4, we use a condensed layered graph, similar to a time-expanded network, to reduce a depth-bounded tree problem to a directed Steiner tree problem. We give a short overview of these concepts in the following.

Time horizon, transit times and flow conservation. As in the static network flow problems discussed before, we are given a network consisting of a digraph $D = (V, A)$ with capacities $u \in \mathbb{Q}_+^A$ on the arcs. Different from the above, however, we now consider a period of time, starting at time 0 and ending at the *time horizon* $T \in \mathbb{Z}_+$. We discretize the time from 0 to T into intervals $[0, 1), \dots, [T-1, T)$ and identify each interval with its starting time, yielding the set $\mathcal{T} = \{0, \dots, T-1\}$. In addition, every arc is given a *transit*

time $\tau(a) \in \mathbb{Z}_+$, specifying how long it takes for flow to traverse the arc. Similar to static flows, we define the *excess* of node $v \in V$ at time $\theta \in \mathcal{T}$ with respect to $x \in \mathbb{Q}_+^{A \times \mathcal{T}}$ by

$$\text{ex}(x, v, \theta) := \sum_{\xi=0}^{\theta} \left(\sum_{a \in \delta^-(v) : \tau(a) \leq \xi} x(a, \xi - \tau(a)) - \sum_{a \in \delta^+(v)} x(a, \xi) \right).$$

An *s-t-flow over time* is a vector $x \in \mathbb{Q}_+^{A \times \mathcal{T}}$ such that $\text{ex}(x, v, \theta) \leq 0$ for all $v \in V \setminus \{s\}$, and $\text{ex}(x, v, \theta) \geq 0$ for all $v \in V \setminus \{t\}$ at any point in time $\theta \in \mathcal{T}$.

Problem: Maximum flow over time

Input: A digraph $D = (V, A)$, a source $s \in V$, a sink $t \in V$, capacities $u \in \mathbb{Q}_+^A$, transit times $\tau \in \mathbb{Z}_+^A$, and a time horizon $T \in \mathbb{Z}_+$.

Task: Find an *s-t-flow over time* x of maximum value $\text{ex}(x, t, T - 1)$.

Temporally repeated flows. Ford and Fulkerson [FF58b] showed that the maximum flow over time problem can be solved in polynomial time by reducing it to a minimum cost flow problem in the underlying static network. The resulting static *s-t-flow* can be transformed into an *s-t-flow over time* by computing a path decomposition and temporally repeating the flow on each path as long as possible within the time horizon. They also showed how to construct a corresponding *cut over time* from the residual network induced by the static flow, proving the *max flow/min cut theorem for flows over time*.

The time-expanded network. Ford and Fulkerson [FF58b] also observed that flows over time can be modeled by so-called *time-expanded networks* which are constructed from multiple copies of the underlying static network D as follows. For every node $v \in V$ of the static network, the time-expanded network $D_T = (V_T, A_T)$ contains T copies of the node, labeled by $v_0, \dots, v_{T-1} \in V_T$. For every arc $a \in A$ of the static network with $\text{tail}(a) = v$ and $\text{head}(a) = w$, the time-expanded network contains the arcs $a_0, \dots, a_{T-1-\tau(a)} \in A_T$ with $\text{tail}(a_i) = v_i$, $\text{head}(a_i) = w_{i+\tau(a)}$ and capacities $u(a_i) = u(a)$. The time-expanded network can be used to reduce flow over time problems to static network flow problems. For example, it is easy to see that every *s-t-flow over time* in D corresponds to a static $\{s_0, \dots, s_{T-1}\}$ - $\{t_0, \dots, t_{T-1}\}$ -flow in D_T and vice versa. Note, however, that the size of the time-expanded network is linear in T and thus exponential in the input size.

Storage at intermediate nodes and holdover arcs. In some contexts, it may be possible to *store* flow at intermediate nodes, i.e., we allow $\text{ex}(x, v, \theta) > 0$ for some or all vertices $v \in V$ and $\theta \in [T - 2]$. This can also be modeled in the time-expanded network by introducing additional *holdover arcs* from v_i to v_{i+1} for the corresponding vertices and all $i \in [T - 2]$. The optimality of the temporally repeated flow for the maximum flow over time problem implies that adding the possibility of storage at intermediate nodes does not have any effect on the value of an optimal solution in this case. However, this is not true for all flow over time problems; see the discussion in Section 2.2 for more details.

Further results. After Ford and Fulkerson’s seminal work on the maximum flow over time problem, many other concepts, such as arc costs and multiple commodities have been transferred to flows over time, and the corresponding problems have been studied extensively. Both the min-cost flow over time and the multi-commodity flow over time problem have been shown to be *NP*-hard by Klinz and Woeginger [KW04] and Hall Hippler, and Skutella [HHS07], respectively. Remarkable results in the area of flows over time also include the polynomial-time algorithm for the transshipment over time problem by Hoppe and Tardos [HT00], the condensed time-expanded networks by Fleischer and Skutella [FS07, FS03], and the recent approximation results for earliest arrival flows by Groß et al. [GKSS12].

Discrete vs. continuous model. The model of flows over time discussed above uses a *discrete* notion of time. Alternatively, flows over time can also be modeled in a *continuous* time setting. In this setting, the flow rate on each arc $a \in A$ is specified as a Lebesgue-integrable function of the time $x(a, \cdot) : [0, T) \rightarrow \mathbb{R}_+$. The excess of a node v at time θ then is defined to be the integral of the flow rates

$$\text{ex}(x, v, \theta) := \int_{\xi=0}^{\theta} \left(\sum_{a \in \delta^-(v) : \tau(a) \leq \xi} x(a, \xi - \tau(a)) - \sum_{a \in \delta^+(v)} x(a, \xi) \right) d\xi.$$

The definitions of *s-t*-flows and other concepts are then based on this notion of excess, in analogy to the discrete model. Discrete and continuous versions of many flow over time problems were shown to be equivalent by Fleischer and Tardos [FT98]. Throughout this thesis, we will restrict to the discrete model of time.

1.3 Network design

The second main theme of this thesis is network design. In general, a network design problem asks for a minimum cost subgraph of a given graph fulfilling certain constraints—usually specifying connectivity requirements. Network design problems typically differ in the type of connectivity constraints they impose, whether the graph is directed or undirected, whether the edges or arcs are capacitated, and whether multiple copies of the same edge or arc are allowed or not. Each of these criteria has a significant impact on the algorithmic tractability of the corresponding problem. Accordingly, network design has been approached with a very broad set of algorithmic techniques, ranging from approximation algorithms to mixed integer programming formulations and combinatorial heuristics. There is little unifying literature that covers all of these approaches and all classes of network design problems. We refer the reader to the surveys by Magnanti and Wong [MW84] and Crainic [Cra00] for applications of network design, by Kortsarz and Nutov [KN07] and Chuzhoy et al. [CGNS08] for approximability results, by Grötschel, Monma, and Stoer [GMS95] and Gendron, Crainic, and Frangioni [GCF99] for MIP formulations, polyhedral results, and heuristics—the latter topic will also be covered in more detail in Section 3.1.2.

In this section, we discuss two problems that play an important role at various points in this thesis, the *Steiner tree problem* and the *fixed-charge network flow problem*. They are canonical examples for two large and important problem classes: uncapacitated and

capacitated network design problems. In practice, most uncapacitated problems—at least in undirected graphs—are considered to be manageable, while capacitated problems earned a reputation of being a great computational challenge [GCF99]. This is also reflected by the theoretical hardness results that we review in this section.

1.3.1 Uncapacitated network design: Steiner trees and networks

The *Steiner tree problem* is one of the most fundamental network design problems. It asks for a minimum cost tree connecting a certain set of vertices, called *terminals*, possibly spanning some additional vertices, called *Steiner vertices*.

Problem: Steiner tree

Input: A graph $G = (V, E)$, costs $c \in \mathbb{Q}_+^E$, and a set of *terminals* $S \subseteq V$.

Task: Find a tree $T \subseteq E$ with $S \subseteq V(T)$, minimizing $c(T)$.

The special case of the Steiner tree problem with $S = V$ is known as *minimum spanning tree problem*.

The problem is named after the Swiss mathematician Jakob Steiner, who investigated the problem of connecting three points in the plane with straight lines by adding an additional point, so as to minimize the total length of the lines; see, e.g., [GP68]. Despite their relatively simple structure, Steiner tree problems play an important role in many applications such as infrastructure planning [Bor26] or chip design [KV08]. In this thesis, they will occur as an important subproblem in Chapter 4. For more details on the topic, we refer to Chapters 6 and 20 of [KV12] and the book by Prömel and Steger [PS02].

Minimum spanning trees. Spanning trees, i.e., trees spanning all vertices of a graph, are a fundamental structure in graph theory. A minimum spanning tree of a weighted graph can be found in time $\mathcal{O}(|E| + |V| \log |V|)$ using a simple greedy algorithm that goes back to Borůvka [Bor26], Jarník [Jar30], and Prim [Pri57]. The *Steiner ratio* denotes the ratio between the minimum cost of a spanning tree on a set of terminals and the minimum cost of a Steiner tree on the same terminals, assuming a complete graph with metric edge costs. Gilbert and Pollak [GP68] showed a tight upper bound of 2 for the Steiner ratio.

Algorithms for the Steiner tree problem. The Steiner tree problem is known to be *NP*-hard [Kar72]. However, several approximation algorithms exist. Kou, Markowsky, and Berman [KMB81] proposed to compute a tree in the metric closure of the graph spanning exactly the terminals, which by the result from [GP68] yields a 2-approximation. A considerably more involved algorithm by Byrka et al. [BGRS10] achieves the currently best known approximation ratio of $\ln 4$. The problem is also fixed-parameter tractable in the number of terminals, as shown by Dreyfus and Wagner [DW71]. Several heuristic approaches for solving Steiner tree problems in practice are reviewed by Winter [Win87] and Voß [Voß92].

Steiner trees in directed graphs. The Steiner tree problem can also be formulated in directed graphs, with an additional node specified as root of the tree.

Problem: Directed Steiner tree problem

Input: A digraph $D = (V, A)$, costs $c \in \mathbb{Q}_+^A$, a root $r \in V$, and a set of nodes $S \subseteq V$.

Task: Find an out-tree $T \subseteq A$ rooted at r with $S \subseteq V(T)$, minimizing $c(T)$.

The special case of the directed Steiner tree problem with $S = V$ is known as *minimum cost arborescence problem*.

The minimum cost arborescence problem can be solved in polynomial time using Edmonds' branching algorithm [Edm67]. However, there is a considerable jump in complexity when turning to the directed Steiner tree problem: Halperin and Krauthgamer [HK03] showed that the problem does not allow for a $\log^{2-\varepsilon} |S|$ -approximation for any $\varepsilon > 0$, unless $NP \subseteq ZTIME(n^{\text{polylog}(n)})$. On the positive side, Charikar et al. [CCC⁺99] devised a quasi-polynomial time algorithm with an approximation guarantee of $\mathcal{O}(\log^2 |S|)$. We will make use of their result in the context of *shallow-light trees*, which are closely related to the directed Steiner tree problem in a so-called layered graph; see Section 4.3.1 for details. As in the undirected case, the directed Steiner tree problem is fixed-parameter tractable in the number of terminals; also see the article by Guo, Niedermeier and Suchý [GNS11] for further results on the fixed-parameter tractability of directed Steiner tree problems. For solving the problem in practice, a dual-ascent method was proposed by Wong [Won84].

Steiner forests and networks. A straightforward generalization of the Steiner tree problem is the *Steiner forest problem*, which asks for a minimum cost forest connecting specified pairs of terminals. The *Steiner network problem* generalizes this further by imposing the requirement of connecting each pair of terminals with a given number of edge-disjoint paths. This latter problem—also known under the name *survivable network design*—is motivated by the construction of reliable networks in telecommunication that are robust against individual link failures. This application and many polyhedral and heuristic results are discussed by Grötschel, Monma, and Stoer [GMS95]. With respect to approximability, the primal-dual algorithm by Goemans and Williamson [GW95] for the Steiner forest problem and the iterative rounding approach by Jain [Jai01] for the Steiner network problem both achieve an approximation factor of 2 in undirected graphs. In contrast, the *directed* version of the Steiner forest problem is very hard to approximate.

Problem: Directed Steiner forest problem

Input: A digraph $D = (V, A)$, costs $c \in \mathbb{Q}_+^A$, and a set of node pairs $\{(s_0, t_0), \dots, (s_k, t_k)\}$.

Task: Find a set of edges $T \subseteq A$ such that T contains a directed s_i - t_i -path for every $i \in [k]$, minimizing $c(T)$.

Dodis and Khanna [DK99] showed that there is no $2^{\log^{1-\varepsilon} k}$ -approximation for the directed Steiner forest problem for any $\varepsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$. Feldmann and Ruhl [FR99] derived an exact $\mathcal{O}(|E||V|^{4k-2} + |V|^{4k-1} \log |V|)$ -time algorithm for the

problem. Note, however, that this is not a fixed-parameter algorithm in the sense of the definition given in Section 1.1.1. In fact, Guo, Niedermeier, and Suchý [GNS11] established $W[1]$ -hardness for the directed Steiner forest problem parameterized both by the number of terminals and the arc costs, making the existence of fixed-parameter algorithms for the problem unlikely.

1.3.2 Capacitated network design: Fixed-charge network flows

The Steiner tree problem and its generalizations impose simple connectivity requirements for the terminals. Every link installed in the network contributes the same unit capacity, and the connecting paths of different node pairs can use this capacity independently from one another. This is different in *capacitated network design*, where the flow running between the terminals has to share the capacity of the installed links, and the capacities provided by each link can be different. A fundamental version of capacitated network design is the *fixed-charge network flow problem*.

Problem: Fixed-charge network flow

Input: A digraph $D = (V, A)$, a set of commodities K , node balances $b \in \mathbb{Q}^{K \times V}$, fixed costs $c \in \mathbb{Q}_+^A$, linear costs $c_i \in \mathbb{Q}^A$ for each $i \in K$, and capacities $u \in \mathbb{Q}_+^A$.

Task: Find numbers $y \in \mathbb{Z}_+^A$ and a multi-commodity b -flow $x \in \mathbb{Q}_+^{K \times A}$ such that $\sum_{i \in K} x_i(a) \leq u(a)y(a)$ for all $a \in A$, minimizing the cost $\sum_{a \in A} c(a)y(a) + \sum_{i \in K} c_i(a)x_i(a)$.

The fixed-charge network flow problem generalizes the minimum cost multi-commodity flow problem by adding fixed costs. This can also be used to model piecewise linear concave cost functions; see Section 3.2.4. There are numerous closely related variants of capacitated network design problems, e.g., with upper bounds on the number of copies of each arc installed in the network.

Fixed-charge network flows and other capacitated network design problems are a versatile tool in the planning of communication and transportation networks. Due to their great significance for practical applications, most generic mixed integer programming solvers include specialized methods for detecting and capacitated network design structures strengthening the corresponding formulations; see the thesis of Raack [Raa12] for recent progress in this direction. In Chapter 3, we will devise a new model for transportation planning that is based on a generalized version of the fixed-charge network flow problem. We will therefore discuss mixed integer programming formulations and heuristic methods for solving capacitated network design problems in detail in that chapter.

Complexity. Clearly, the fixed-charge network flow problem generalizes the directed Steiner forest problem and thus inherits all hardness results mentioned above. When each arc may only be installed once in the network, the $2^{\log^{1-\epsilon} |V|}$ -hardness of approximation even holds for the single-commodity case [CCKK11]. Also in undirected graphs, fixed-charge network flow remains hard to approximate: Chuzoy et al. [CGNS08] showed that there is no approximation better than $\Omega(\log \log |V|)$ even for the single-commodity case of the problem, unless $NP \subseteq DTIME(n^{\log \log n})$.

Chapter 2

Abstract flows over time

In this chapter, we study a generalization of network flows called *abstract flows*. This model replaces the underlying network structure by an abstract system of linearly ordered sets fulfilling a simple switching axiom. Abstract flows were introduced by Hoffman [Hof74] to investigate minimal structural requirements for obtaining max flow/min cut results. We extend his results by introducing a notion of time, generalizing Ford and Fulkerson’s concept of *flows over time* [FF58b]. Using the maximum abstract flow algorithm of McCormick [McC96], we show how maximum flows and minimum cuts over time can still be computed in the abstract setting.

Publication remark: The results presented in this chapter are joint work with Jan-Philipp W. Kappmeier and Britta Peis [KMP12].

Ford and Fulkerson’s max flow/min cut theorem [FF56] is among the most influential results in combinatorial optimization. Understanding the driving forces behind this result is of great interest, not only for its fundamental importance to network flow theory itself but also due to the implied structural connection between cuts and connectivity in networks. Hoffman [Hof74] observed that the original proof of the theorem does not use the underlying network structure directly but only exploits one particular property of the path system, the so-called *switching axiom*: Whenever two paths P and Q intersect, there must be another path that is contained in the beginning of P and the end of Q . Hoffman succeeded in showing that a generalized version of the maximum flow problem defined on any set system fulfilling the switching axiom, called *abstract network*, still is totally dual integral (TDI). His structural results were later complemented by the combinatorial primal-dual algorithms of McCormick [McC96] and Martens and McCormick [MM08].

The high level of abstraction in Hoffman’s model leads to the question whether his results are restricted to the classic maximum flow problem or whether they extend to other variants of network flows. In this chapter, we introduce and investigate *abstract flows over time* and show how a temporally repeated abstract flow and a corresponding minimum cut can be computed by solving a single static weighted abstract flow problem—which in our case can be done even when accessing the abstract network through a very limited oracle. This immediately leads to the max flow/min cut theorem for abstract flows over time as our main result in this chapter.

Although our construction resembles that of Ford and Fulkerson’s original result on (non-abstract) flows over time [FF58b], the proof turns out to be considerably more involved and we will need to take a detour via a relaxed version of abstract flows over time

that also considers storage of flow at intermediate elements. However, our results also imply that this relaxation is not necessary and that there always is an optimal solution that does not make use of storage at intermediate nodes. In the course of our proof, we also establish some interesting structural properties of abstract networks, showing that the relatively modest switching axiom of abstract path systems already captures many essential properties of classic networks.

Dynamic packing problems. Besides yielding new insights into the mechanics behind max flow/min cut for flows over time and the generality of Hoffman’s model, our research on abstract flows over time is also motivated by a second perspective. Network flows comprise a special class of packing problems: We try to pack the capacitated arcs of the graph by assigning flow values to the source-sink-paths. Accordingly, flows over time can be seen as a *dynamic packing problem*, i.e., a packing problem with a temporal component, where solutions may vary over time and a decision taken at some point in time may impact the state of the solution at later points as well. Considering the importance of time in many applications of combinatorial optimization and the large impact of Ford and Fulkerson’s initial results on flows over time, which spawned a whole new theory in this area, one now might ask how the concept of time can be extended to other packing problems. Abstract flows appear to be an ideal first candidate for investigating this question. This impression is amplified by further abstractions based on uncrossing axioms that have been inspired by the concept of abstract flows—corresponding TDI results have been established, e.g., for lattice polyhedra by Hoffman and Schwartz [HS78], Gröffin and Hoffman [GH82], and Hoffman [Hof78], as well as for switchdec polyhedra by Gaillard [Gai97]; also see the survey by Schrijver [Sch84].

Chapter outline

Section 2.1 introduces Hoffman’s model of abstract flows in detail and discusses the related literature. In Section 2.2, we extend this model by conducting a time expansion. We point out differences from the time-expanded network for classic network flows by Ford and Fulkerson [FF58b]. Section 2.3 then explains how to construct a maximum temporally repeated abstract flow and a corresponding minimum abstract cut of the same value. We argue how to compute both the flow and the cut using a standard oracle for accessing abstract networks. In order to validate the feasibility of the abstract cut over time, we prove some interesting structural properties of abstract networks in Section 2.4. Using these results, we can finally show in Section 2.5 that the cut actually intersects all temporal paths, completing the proof of the abstract max flow/min cut over time theorem.

2.1 Introduction to abstract flows

In this section we give a short introduction to Hoffman’s model of abstract flows and the corresponding structural and algorithmic results.

Abstract networks. An *abstract path system* consists of a ground set E of *elements* and a family of *paths* $\mathcal{P} \subseteq 2^E$. For every $P \in \mathcal{P}$ there is a linear order $<_P$ of the elements in

P . We introduce the following notation for $P \in \mathcal{P}$ and $e \in P$:

$$\begin{aligned} [P, e] &:= \{p \in P : p \leq_P e\} & [e, P] &:= \{p \in P : p \geq_P e\} \\ (P, e) &:= \{p \in P : p <_P e\} & (e, P) &:= \{p \in P : p >_P e\} \end{aligned}$$

An abstract path system is an *abstract network*, if the *switching axiom* is fulfilled: For every $P, Q \in \mathcal{P}$ and every $e \in P \cap Q$, there is a path

$$P \times_e Q \subseteq [P, e] \cup [e, Q].$$

Weighted abstract flows and cuts. An *abstract flow* is an assignment $x \in \mathbb{Q}_+^{\mathcal{P}}$ of flow values to the paths of the abstract network. Given an abstract network with capacities for all elements, the *abstract flow problem* asks for an abstract flow such as to maximize the total flow value while not violating the capacity of any element. This problem can be generalized further by introducing a weight function that specifies the “reward” per unit of flow sent along each path.

Problem: Weighted abstract flow

Input: An abstract network (E, \mathcal{P}) with capacities $u \in \mathbb{Q}_+^E$ and weights $r \in \mathbb{Q}_+^{\mathcal{P}}$.

Task: Find an abstract flow $x \in \mathbb{Q}_+^{\mathcal{P}}$ with $\sum_{P \in \mathcal{P}: e \in P} x(P) \leq u(e)$ for all $e \in E$, maximizing $\sum_{P \in \mathcal{P}} r(P)x(P)$.

The special case of $r \equiv 1$ is called (*unweighted*) *abstract flow problem*.

The weighted abstract flow problem corresponds to the following packing LP.

$$\begin{aligned} \text{[WAF]} \quad & \max \sum_{P \in \mathcal{P}} r(P)x(P) \\ & \text{s.t.} \quad \sum_{P \in \mathcal{P}: e \in P} x(P) \leq u(e) \quad \forall e \in E \\ & \quad \quad \quad x(P) \geq 0 \quad \forall P \in \mathcal{P} \end{aligned}$$

The dual of this LP is the *weighted abstract cut problem*, a covering problem that assigns a value to every element so as to cover every path according to its weight.

$$\begin{aligned} \text{[WAC]} \quad & \min \sum_{e \in E} u(e)y(e) \\ & \text{s.t.} \quad \sum_{e \in P} y(e) \geq r(P) \quad \forall P \in \mathcal{P} \\ & \quad \quad \quad y(e) \geq 0 \quad \forall e \in E. \end{aligned}$$

Oracles. From the viewpoint of complexity theory, we are interested in algorithms whose running times are polynomial in the size of the ground set. We will thus assume E to be given explicitly while the abstract network can be accessed by the following oracle.

Oracle \mathbb{O}_{path} : Given $Q \subseteq E$, return (P, \prec_P) for some path $P \in \mathcal{P}$ with $P \subseteq Q$, or verify that no path is contained in Q .¹

Note that using \mathbb{O}_{path} , we can also determine a possible choice for $P \times_e Q$ for intersecting paths $P, Q \in \mathcal{P}$ by asking the oracle for a path in $[P, e] \cup [e, Q]$. In general, also the weight function r is given by an oracle. The weights occurring in this thesis however will always be of the form $r(P) = T - \sum_{e \in P} \tau(e)$ for some constant T and a vector $\tau \in \mathbb{Z}^E$ and are thus easily computable.

Supermodularity. It is easy to check that both [WAF] and [WAC] can have fractional optimal solutions, even if all capacities and weights are integral. In fact, when allowing arbitrary weight functions, any packing problem on an arbitrary set system can be modeled as a maximum abstract flow problem by setting the weight of undesired paths to zero. We therefore turn our attention to supermodular weight functions. A function $r : \mathcal{P} \rightarrow \mathbb{Q}$ is *supermodular* if

$$r(P \times_e Q) + r(Q \times_e P) \geq r(P) + r(Q)$$

for every $P, Q \in \mathcal{P}$ and $e \in P \cap Q$. An especially interesting class of supermodular functions are functions of the type $r(P) = T - \sum_{e \in P} \tau(e)$ for some $T \in \mathbb{Q}_+$ and $\tau \in \mathbb{Q}_+^E$. We will encounter functions of this type in Section 2.3.

Previous results. Hoffman [Hof74] showed that for every integral supermodular weight function, the abstract cut LP [WAC] is totally dual integral. This implies the max flow/min cut theorem for abstract flows, which generalizes Ford and Fulkerson's original result in two ways: On the one hand, the switching axiom represents a significant abstraction of the underlying structure, allowing for more general settings than only classic networks. On the other hand, supermodular weight functions lead to weighted cuts, i.e., elements can appear multiple times in the cut. We will later see an example for the usefulness of such weights in the context of temporally repeated flows, which also yields an intuitive interpretation of the cut values as the times each element is part of the cut. Hoffman's structural result was extended by McCormick [McC96], who presented a combinatorial algorithm that solves the unweighted version ($r \equiv 1$) of the abstract flow problem in time polynomial in $|E|$ and the encoding size of the capacities using the oracle \mathbb{O}_{path} . Later, Martens and McCormick [MM08] presented a combinatorial primal-dual algorithm for the case of general supermodular weights using a separation oracle for the weighted abstract cut LP [WAC].

Abstract networks vs. classic networks. While these results indicate that the switching axiom is the essential force behind max flow/min cut and similar total dual integrality results for flow problems in networks, we want to close this section by pointing out an example that shows how abstract networks actually may differ from classic networks. In classic networks, if two paths P and Q both intersect a third path R , then there also must be a path from the beginning of P to the end of Q or the other way around. The following example shows that this is not true in abstract networks, even in cases where the switching axiom preserves the order of intersecting abstract paths.

¹The oracle \mathbb{O}_{path} is equivalent to the one used by McCormick [McC96], who states that the idea for this oracle actually goes back to Hoffman.

Example 2.1 Consider the abstract network (E, \mathcal{P}) with $E = \{1, 2, 3, 4, a, b, c, d\}$ and paths $\mathcal{P} = \{(1, 2, 3, 4), (a, 2, c), (b, 3, d), (1, c), (1, d), (a, 4), (b, 4)\}$. Although both $(a, 2, c)$ and $(b, 3, d)$ intersect the path $(1, 2, 3, 4)$, there is neither a path that starts with a and ends with d nor one that starts with b and ends with c .

2.2 Time expansion of abstract networks

Time plays an important role in many application areas of network flows. Flow rates can vary over time, and flow also takes time to travel within the network. One concept to capture these temporal effects is the time-expanded network introduced by Ford and Fulkerson [FF62] as described in Section 1.2.5. Recall that a time-expanded network contains multiple copies of the nodes of the underlying static network, one for each point in time, with arcs connecting copies of nodes according to their travel time. We extend this concept to the world of abstract flows by introducing the time expansion of an abstract network. In the spirit of Ford and Fulkerson's idea, we will introduce multiple copies of the abstract network. In contrast to the classic model however, instead of copies of individual arcs, whole paths will be introduced.

The *time expansion* of an abstract network consists of a (static) abstract network with capacities $u \in \mathbb{R}_+^E$, transit times $\tau \in \mathbb{Z}_+^E$ and a *time horizon* $T \in \mathbb{Z}_+$. The time from 0 to T is discretized into T intervals $[0, 1), \dots, [T-1, T)$ which we identify with the set of their starting times $\mathcal{T} := \{0, \dots, T-1\}$. For each interval, a copy of the ground set E is introduced, i.e., the time-expanded ground set is $E_T := E \times \mathcal{T}$. A *temporal path* is denoted by P_t , where P is a path of the underlying static abstract network and $t \in \mathcal{T}$ specifies the starting time of the path. Flow sent along the temporal path P_t enters element e at time

$$\gamma(P_t, e) := t + \sum_{p \in (P, e)} \tau(p)$$

which is the time it needs for traversing all preceding elements plus the initial offset of the path. Accordingly, we identify P_t with the set of its temporal elements by defining

$$P_t := \{(e, \gamma(P_t, e)) \in E_T : e \in P\}.$$

The *arrival time* of the temporal path P_t is $t + \sum_{e \in P} \tau(e)$, i.e., the time at which the flow arrives at the end of the path. Since all flow is supposed to arrive at its destination within the time horizon, we only allow copies of paths with a maximum arrival time of $T-1$, which is the final element of \mathcal{T} . Thus, the set of temporal paths is defined by

$$\mathcal{P}_T := \left\{ P_t : P \in \mathcal{P}, t \in \mathcal{T}, t + \sum_{p \in P} \tau(p) < T \right\}.$$

Abstract flows and cuts over time. An abstract flow over time is an assignment $x \in \mathbb{Q}_+^{\mathcal{P}_T}$ of flow values to the temporal paths such that

$$\sum_{P_t \in \mathcal{P}_T : (e, \theta) \in P_t} x(P_t) \leq u(e)$$

for all $(e, \theta) \in E_T$, i.e., the capacity of every element at every point in time is respected.

Problem: Maximum abstract flow over time

Input: An abstract network (E, \mathcal{P}) , capacities $u \in \mathbb{Q}_+^E$, transit times $\tau \in \mathbb{Z}_+^E$, a time horizon $T \in \mathbb{Z}_+$.

Task: Find an abstract flow over time x of maximum flow value $\sum_{P_t \in \mathcal{P}_T} x(P_t)$.

An *abstract cut over time* is a subset $C \subseteq E_T$ of the time-expanded ground set that covers every temporal path, i.e., $P_t \cap C \neq \emptyset$ for all $P_t \in \mathcal{P}_T$. The capacity of such a cut is $\sum_{(e,\theta) \in C} u(e)$. In analogy to the static case, the maximum value of an abstract flow over time is bounded by the capacity of an abstract cut over time.

Lemma 2.2 *Let x be an abstract flow over time and let C be an abstract cut over time. Then $\sum_{P_t \in \mathcal{P}_T} x(P_t) \leq \sum_{(e,\theta) \in C} u(e)$.*

Proof. As the cut contains an element of every temporal path and the capacity constraints are respected at every point in time, we get

$$\sum_{P_t \in \mathcal{P}_T} x(P_t) \leq \sum_{(e,\theta) \in C} \sum_{P_t: (e,\theta) \in P_t} x(P_t) \leq \sum_{(e,\theta) \in C} u(e). \quad \square$$

Time expansion of an abstract network vs. time-expanded network. While the time expansion of abstract networks as defined above is similar to the notion of a *time-expanded network* as defined by Ford and Fulkerson [FF62] for classic network flows, the two definitions are not quite identical. Time-expanded networks are based on the arc formulation of network flows. They are constructed by introducing copies of both the nodes and the arcs of the underlying static network and adjusting the end points of the arcs according to their transit times. By construction, the resulting structure is guaranteed to be a network again. Unfortunately, there is no correspondence to the arc flow formulation for abstract flows—their definition is inherently tied to the path system, which does not allow for local concepts such as flow conservation at a particular element. Our model of time expansion therefore introduces copies of each abstract path as a whole. In contrast to time-expanded networks, the time expansion of an abstract network is *not* an abstract network in general, as can be seen in the following example.

Example 2.3 Let $E = \{s, a, b, t\}$ and $\mathcal{P} = \{P, Q, R, S\}$ with paths $P = (s, a, b, t)$, $Q = (s, b, a, t)$, $R = (s, a, t)$, and $S = (s, b, t)$. It is easy to verify that \mathcal{P} in fact fulfills the switching axiom. Assume all elements have unit transit times, i.e., $\tau \equiv 1$. The temporal paths P_0 and Q_1 intersect at element $(b, 2)$. However, there is no temporal path in \mathcal{P}_T that can be constructed from the elements $\{(s, 0), (a, 1), (b, 2), (a, 3), (t, 4)\}$. Thus, the time expansion violates the switching axiom.

Statement of the main theorems

In view of Example 2.3, it is not even clear whether max flow/min cut results are still valid in the context of abstract flows over time or how far existing algorithms for abstract flow problems can be applied to the time expansion of the abstract network. Fortunately, our investigations in the following sections will show that both max flow/min cut and efficient algorithms can still be achieved for abstract flows over time.

Theorem 2.4 (Abstract max flow/min cut over time) *The value of a maximum abstract flow over time equals the capacity of a minimum abstract cut over time.*

In fact, our proof of Theorem 2.4 is constructive and it implies that both a maximum flow and a minimum cut over time can be computed by solving a single (static) weighted abstract flow problem.

Theorem 2.5 *A maximum abstract flow and a minimum abstract cut over time can be computed in time $\phi(|E|, \langle u \rangle, \log(T)) \cdot \text{TIME}(\mathbb{O}_{\text{path}}, \mathcal{P})$, where ϕ is a polynomial, $\langle u \rangle$ is the encoding size of u , and $\text{TIME}(\mathbb{O}_{\text{path}}, \mathcal{P})$ denotes the time needed by a call of the oracle \mathbb{O}_{path} for the abstract network \mathcal{P} .*

Our proof of the above theorems involves constructing an abstract cut over time. In order to show feasibility of this cut, we will have to introduce the possibility of storage at intermediate elements as an important device in our proof in Section 2.4. Storage of flow at intermediate nodes plays an interesting role in the field of flows over time. In some settings, such as the maximum flow over time problem or the NP-hard minimum cost flow over time problem, there always exist optimal solutions that do not wait at intermediate nodes; see the articles by Ford and Fulkerson [FF58b] and Fleischer and Skutella [FS03], respectively. This is not true in other settings: e.g., for multi-commodity flows over time, the decision of allowing flow storage at intermediate nodes has an influence on the value of the solution and also on the complexity; see the article by Hall, Hippler, and Skutella [HHS07] for an overview and by Groß and Skutella [GS12] for a recent technique designed to cope with the absence of intermediate storage. In the context of abstract flows over time, our results imply that the possibility of storage has no influence on the problem, as we prove in Section 2.5 that the temporally repeated solution constructed in Section 2.3 is optimal even if intermediate storage is allowed.

Theorem 2.6 *The value of an abstract flow over time with storage at intermediate elements is not larger than the value of a maximum abstract flow over time without storage.*

Remark 2.7 As pointed out in the Section 1.2.5, flows over time allow for both a *discrete* and a *continuous* model of time. While we restrict ourselves to the discrete model in this thesis, we remark that all our results can easily be transferred to the corresponding continuous-time version of abstract flows over time. In fact, our proofs do not make any use of the discretization of time, and any occurring sequence of consecutive discrete points in time t_1, \dots, t_k can naturally be replaced by the continuous interval $[t_1, t_k + 1)$.

2.3 Construction of a maximum abstract flow over time and a minimum abstract cut over time

The number of paths created by applying the time expansion is linear in T and can thus be exponential in the size of the input. Hence, even encoding a solution in the straightforward way may result in an exponentially sized output. Ford and Fulkerson [FF58b] resolved this problem for the classic (non-abstract) flow over time problem by introducing a so-called temporally repeated flow, i.e., a flow over time constructed by temporally repeating a static flow pattern. In this section, we will translate the concept of temporally repeated flows to the abstract setting and show how to construct a maximum temporally repeated abstract flow by solving a static weighted abstract flow problem.

Temporally repeated flows

A *temporally repeated abstract flow* is an abstract flow over time x_T that is constructed from a static abstract flow x by setting $x_T(P_t) := x(P)$ for every $P \in \mathcal{P}$ and every $t \in \mathcal{T}$ with $0 \leq t < T - \sum_{e \in P} \tau(e)$. In other words, the static flow on each path is repeatedly sent as long as possible before the time horizon is reached. It is easy to check that feasibility of the underlying static flow implies feasibility of the temporally repeated flow.

Lemma 2.8 *A temporally repeated abstract flow x_T derived from a feasible abstract flow x is a feasible abstract flow over time.*

Proof. We only need to verify that x_T obeys the capacity restrictions for every $e \in E$ and every $\theta \in \mathcal{T}$. Observe that $(e, \theta) \in P_t$ if and only if $e \in P$ and $\theta = t + \sum_{p \in P} \tau(p)$. As the second summand on the right hand side is constant for a fixed $P \in \mathcal{P}$, there is at most one value of $t \in \mathcal{T}$ for which $(e, \theta) \in P_t$. Thus

$$\sum_{P_t \in \mathcal{P}_T : (e, \theta) \in P_t} x_T(P_t) \leq \sum_{P \in \mathcal{P} : e \in P} x(P) \leq u(e)$$

for all $(e, \theta) \in E_T$. □

In order to construct a maximum temporally repeated abstract flow, we first observe that flow can be sent along path $P \in \mathcal{P}$ up to time $r(P) := T - \sum_{e \in P} \tau(e)$, i.e., the flow value $x(P)$ is repeated $r(P)$ times. Thus, the total flow value of the temporally repeated flow x_T resulting from the static flow x is $\sum_{P \in \mathcal{P}} r(P)x(P)$ and a maximum temporally repeated flow corresponds to a static abstract flow that is maximum with respect to the weights $r(P)$. It is not hard to observe that the weight function defined in this way is supermodular.

Lemma 2.9 *The weight function $r(P) := T - \sum_{e \in P} \tau(e)$ is supermodular.*

Proof. By definition of r we have

$$\begin{aligned} r(P \times_e Q) + r(Q \times_e P) &= T - \sum_{e \in P \times_e Q} \tau(e) + T - \sum_{e \in Q \times_e P} \tau(e) \\ &\geq 2T - \left(\sum_{e \in [P, e]} \tau(e) + \sum_{e \in (e, Q)} \tau(e) \right) - \left(\sum_{e \in [Q, e]} \tau(e) + \sum_{e \in (e, P)} \tau(e) \right) \\ &= 2T - \sum_{e \in P} \tau(e) - \sum_{e \in Q} \tau(e) \\ &= r(P) + r(Q). \end{aligned} \quad \square$$

Let x^* be a (static) abstract flow of maximum weight with respect to r , and let x_T^* be the corresponding temporally repeated flow—at the end of this section, we will argue how to compute x^* using the oracle \mathbb{O}_{path} and the algorithm from [McC96]. We will show that the value of x_T^* is not only maximum among the temporally repeated abstract flows but also among *all* abstract flows over time, i.e., it is an optimal solution to the maximum abstract flow over time problem. To this end, we construct an abstract cut over time whose capacity matches the flow value of x_T^* .

Constructing an abstract cut over time

Let y^* be an optimal solution to the dual of the static weighted abstract flow problem with the weights $r(P)$ used to construct the temporally repeated flow. Note that by [Hof74] and Lemma 2.9, we can assume y^* to be integral. We will interpret the values y^* as the number of time steps for which element e is contained in the cut. We define the time at which $e \in E$ enters the cut by setting

$$\alpha(e) := \min_{P \in \mathcal{P}} \sum_{p \in (P, e)} (\tau(p) + y^*(p))$$

and define

$$C := \{(e, \theta) \in E_T : \alpha(e) \leq \theta < \alpha(e) + y^*(e)\}.$$

Theorem 2.10 *C is an abstract cut over time.*

The proof of Theorem 2.10 involves some additional results on the structure of abstract networks, which we will elaborate on in the following sections. Using LP duality, Theorem 2.10 immediately leads to the following corollary, which in turn implies Theorem 2.4.

Corollary 2.11 *The temporally repeated abstract flow x_T^* is a maximum abstract flow over time, and C is a minimum abstract cut over time whose capacity is equal to the flow value.*

Proof. We observe that by LP duality,

$$\sum_{(e, \theta) \in C} u(e) = \sum_{e \in E} u(e) y^*(e) = \sum_{P \in \mathcal{P}} r(P) x^*(P) = \sum_{P_t \in \mathcal{P}_T} x_T^*(P_t)$$

and thus the capacity of C equals the flow value of x_T^* . \square

Remark 2.12 Our proof of Theorem 2.10 will make use of the fact that our definition of C is based on the “shortest path labels” α . Unfortunately, in the case of abstract networks these labels need *not* satisfy the triangle inequality, i.e., there might be elements $e, f \in E$ such that f is the immediate successor of e in an abstract path P , but $\alpha(e) + \tau(e) + y^*(e) < \alpha(f)$; see Example 2.13. This makes the proof considerably more involved than the corresponding proof for classic networks.

Example 2.13 Consider the abstract network with ground set $E = \{a, b, c, d, z\}$ and path set $\mathcal{P} = \{(a, c, z), (b, c, d, z), (a, z), (b, z)\}$. Suppose $T = 2$, $\tau(b) = 1$, $y^*(z) = 2$, and $\tau(e) = 0$ for all $e \in E \setminus \{b\}$ and $y^*(e) = 0$ for all $e \in E \setminus \{z\}$. Observe that in this case $\alpha(d) = 1 > \alpha(c) + \tau(c) + y^*(c)$.

Note that for the static abstract flow problem, the first two paths in the above example would be redundant, as an optimal solution can always restrict to the latter two. However, it is not immediately clear that this is true for abstract flows over time; additional elements on a path can delay flow, which in principle could help to avoid bottlenecks and achieve larger flow values. Yet we will show later that this is not the case and delaying flow is never beneficial.

Complexity of constructing x_T^* and C

We close this section by arguing how to compute the maximum weighted abstract flow x^* , the corresponding dual solution y^* and the values $\alpha(e)$ for elements occurring in the cut while only accessing the abstract network using the oracle \mathbb{O}_{path} . This suffices to implicitly construct x_T^* and C and thus proves Theorem 2.5.

Lemma 2.14 *The values of x_T^* , y^* , and α restricted to the support of y^* can be computed in time $\phi(|E|, \langle u \rangle, \log(T)) \cdot \text{TIME}(\mathbb{O}_{\text{path}}, \mathcal{P})$, where ϕ is a polynomial, $\langle u \rangle$ is the encoding size of u , and $\text{TIME}(\mathbb{O}_{\text{path}}, \mathcal{P})$ denotes the time needed by a call of the oracle \mathbb{O}_{path} for the abstract network \mathcal{P} .*

Proof. First observe that for any $u' \in \mathbb{Q}_+^E$, the corresponding *unweighted* abstract cut problem can be solved using the algorithm of McCormick [McC96] with a polynomial number of oracle calls. By the equivalence of optimization and separation [GLS88], we can thus decide whether a given $y \in \mathbb{Q}_+^E$ fulfills $\sum_{e \in P} y(e) \geq 1$ for all $P \in \mathcal{P}$, or find a violated path $P \in \mathcal{P}$ with $\sum_{e \in P} y(e) < 1$. Because $\sum_{e \in P} y(e) \geq r(P)$ if and only if $\sum_{e \in P} (y(e) + \tau(e))/T \geq 1$, we can solve the separation problem of the minimum *weighted* abstract cut problem with respect to r as well. As a result, we can compute x^* and y^* with a polynomial number of oracle calls, either by once more applying the equivalence of optimization and separation and LP duality, or by using the combinatorial algorithm of Martens and McCormick [MM08].

For any $P \in \mathcal{P}$ with $x^*(P) > 0$ and any $e \in P$ define

$$\alpha_P(e) := \sum_{p \in (P, e)} (\tau(p) + y^*(p)).$$

We will show that $y^*(e) > 0$ implies $\alpha(e) = \alpha_P(e)$ for all paths $P \in \mathcal{P}$ with $e \in P$ and $x^*(P) > 0$. Therefore, α can be computed by only considering the support of x^* . Let $e \in E$ with $y^*(e) > 0$. By complementary slackness, $\sum_{P \in \mathcal{P}: e \in P} x^*(P) = u(e)$. Thus there is a $P \in \mathcal{P}$ with $e \in P$ and $x^*(P) > 0$ (unless $u(e) = 0$, in which case we can ignore e). Again by complementary slackness, $\sum_{p \in P} y^*(p) = r(P)$. By contradiction assume there is a path $Q \in \mathcal{P}$ with $\alpha_Q(e) < \alpha_P(e)$. Let $S := Q \times_e P$. Then

$$\sum_{s \in S} (\tau(s) + y^*(s)) \leq \alpha_Q(e) + \sum_{p \in [e, P]} (\tau(p) + y^*(p)) < \sum_{p \in P} (\tau(p) + y^*(p)) = T$$

contradicting the feasibility of y^* . □

Remark 2.15 The usage of the equivalence of optimization and separation in the above proof can be omitted if the oracle for accessing the abstract network is slightly more powerful:

Oracle \mathbb{O}_{sep} : Given $y \in \mathbb{Q}_+^E$, return $(P, <_P)$ for some $P \in \mathcal{P}$ with $\sum_{e \in P} y(e) < 1$, or verify that $\sum_{e \in P} y(e) \geq 1$ for all $p \in \mathcal{P}$.

Note that \mathbb{O}_{path} is equivalent to the restriction of \mathbb{O}_{sep} to integral y . Although \mathbb{O}_{sep} is not a separation oracle for arbitrary supermodular weight functions, using the argument in the proof of Lemma 2.14 it works as separation oracle for all weight functions of the type $r(P) = T - \sum_{e \in P} \tau(e)$ for some $T \in \mathbb{Z}_+$ and $\tau \in \mathbb{Z}_+^E$. Therefore, we can directly apply the combinatorial algorithm from [MM08] for computing x^* and y^* .

2.4 Storage at intermediate elements and the structure of abstract networks

In order to prove that the set C constructed in the preceding section actually covers all temporal paths, we need to ensure that the switching operation \times preserves the order of the intersecting paths. In this section we show how this can be done without loss of generality. We start by showing a weaker version of this statement, asserting that we can always choose the path resulting from an application of \times in such a way that the two subpaths used for its construction are not mixed.²

Lemma 2.16 *Let $P, Q \in \mathcal{P}$, $e \in P \cap Q$, then there is a path $R \subseteq [P, e] \cup [e, Q]$ such that $a <_R b$ for any $a \in R \cap [P, e]$ and $b \in R \setminus [P, e]$.*

Proof. Let $P, Q \in \mathcal{P}$ and $e \in P \cap Q$. Let R to be a path contained in $[P, e] \cup [e, Q]$ such that $|R \setminus [P, e]|$ is minimal. By contradiction assume the existence of $a \in R \cap [P, e]$ and $b \in R \setminus [P, e]$ with $b <_R a$. Let $R' := P \times_a R$. Observe that $R' \subset [P, e] \cup [e, Q]$ and furthermore $R' \setminus [P, e] \subset R \setminus [P, e]$ as $b \notin R'$. This contradicts the choice of R . \square

As a result of Lemma 2.16, the following assumption is without loss of generality.

Assumption A If $a \in P \times_e Q \cap [P, e]$ and $b \in P \times_e Q \setminus [P, e]$, then $a <_{P \times_e Q} b$.

In order to show that \times actually preserves the internal order of P and Q , we will—temporally—extend our model of time expansion by allowing flow to deliberately delay its traversal at intermediate elements.

Storage at intermediate elements

A *temporal path with intermediate storage* is denoted by P_σ , where $P \in \mathcal{P}$ is a path of the underlying static abstract network and $\sigma : P \rightarrow \mathcal{T}$ specifies the storage time $\sigma(e)$ before traversing element $e \in P$. Flow sent along P_σ enters element e at time

$$\gamma(P_\sigma, e) := \sum_{p \in (P, e)} (\sigma(p) + \tau(p)) + \sigma(e)$$

which is the time it needs for traversing all preceding elements and the time it spends waiting at those elements and at e itself. Accordingly, we identify P_σ with the set of its temporal elements by defining

$$P_\sigma := \{(e, \gamma(P_\sigma, e)) \in E_T : e \in P\}.$$

The set of all temporal paths with intermediate storage is denoted by

$$\mathcal{P}_T^* := \{P_\sigma : P \in \mathcal{P}, \sigma \in \mathcal{T}^P, \sum_{e \in P} (\sigma(e) + \tau(e)) < T\}.$$

We will identify $P_t \in \mathcal{P}_T$ with $P_{(t, 0, \dots, 0)} \in \mathcal{P}_T^*$. Note that the maximum abstract flow over time problem with storage at intermediate elements is a relaxation of maximum abstract flow over time without storage, and that in particular the temporally repeated

²This property of the switching operation was first noted by Orlin in personal communication with McCormick [McC13].

abstract flow x_T^* defined in Section 2.3 is a feasible solution to this relaxation. We will show that C actually covers all paths in \mathcal{P}_T^* , and thus x_T^* is optimal even if storage is allowed. This implies that the relaxation does not have any effect on the value of the optimal solution.

However, the extension of the model allows us to delete certain paths from the network. Observe that if Q is a strict subset of P , and $<_Q$ is identical to the restriction of $<_P$ to Q , then there always is an optimal abstract flow over time that does not use any copy of P , since the flow can wait at intermediate elements and use Q instead. Thus we can safely erase P from the base network in this case—note that this does not violate the switching axiom, as Q can always replace P as switching choice. Hence, if we allow storage at intermediate elements, the following assumption is without loss of generality.

Assumption B If $Q \subset P$ then there are $a, b \in Q$ with $a <_P b$ and $b <_Q a$.

In the remainder of this section, we show that Assumption B implies the following lemma. As a consequence of the lemma, we can assume the switching operation to preserve order; see Corollary 2.18.

Lemma 2.17 *There are no paths $P, Q \in \mathcal{P}$ such that $Q \subset P$.*

Proof. By contradiction assume there are $P, Q \in \mathcal{P}$ with $Q \subset P$. Let P^* be such that $|P^*|$ is minimal among all possible choices of such a P .

For $Q \subset P^*$ define $b(Q) \in Q$ to be the maximal element with respect to $<_Q$ such that $p <_{P^*} b(Q)$ for all $p \in (Q, b(Q))$, i.e., until element $b(Q)$, the order of Q is identical to that of P . By Assumption B, $b(Q)$ cannot be the last element of Q . So let $a(Q) \in Q$ be the successor of $b(Q)$ in Q . Note that this implies $a(Q) <_{P^*} b(Q)$ by definition of $b(Q)$. Among all paths $Q \subset P^*$, choose Q^* such that $b^* := b(Q^*)$ is maximal with respect to $<_{P^*}$. Let $a^* := a(Q^*)$.

Now let $R := Q^* \times_{b^*} P^*$. Note that $a^* \notin R$, as $a^* >_{Q^*} b^*$ and $a^* <_{P^*} b^*$. Therefore $R \subset P^*$. We now claim that $<_R$ is identical to $<_{Q^*}$ on the (Q^*, b^*) -part of R .

Claim Let $c, d \in R \cap (Q^*, b^*)$. Then $c <_{Q^*} d$ if and only if $c <_R d$.

In order to see the claim is true, assume $c <_{Q^*} d$ but $d <_R c$ and let $R' := R \times_d Q^*$. Note that $c \notin R'$ and by Assumption A, we have chosen R such that $[R, d] \subset Q^*$. This implies $R' \subset Q^* \subset P^*$, which contradicts the choice of P^* , proving the claim.

By definition of $b(Q^*)$, the order $<_{Q^*}$ is identical to $<_{P^*}$ on (Q^*, b^*) and thus the claim implies that $<_R$ is identical to $<_{P^*}$ on the (Q^*, b^*) -part of R . Therefore $a(R)$ and $b(R)$ cannot be both in the (Q^*, b^*) -part of R . Thus, $a(R) \in [b^*, P^*]$, which by $a(R) <_{P^*} b(R)$ implies that $b(R) \in (b^*, P^*)$. However, this means $b(R) >_{P^*} b^*$, contradicting our choice of Q^* maximizing b^* . This proves the lemma. \square

Corollary 2.18 *Let $R := P \times_e Q$.*

- If $a, b \in R \cap [P, e]$ and $a <_P b$, then $a <_R b$.
- If $a, b \in R \setminus [P, e]$ and $a <_Q b$, then $a <_R b$.

Proof.

- By contradiction assume $a, b \in [P, e] \cap R$ and $a <_P b$ but $b <_R a$. Then, by Assumption A, there is no $c \in R \setminus [P, e]$ with $c <_R a$. This means $[R, b] \subseteq P$ and thus $R \times_b P \subseteq P \setminus \{a\}$, contradicting Lemma 2.17.

- By contradiction assume $a, b \in R \setminus [P, e]$ and $a <_Q b$ but $b <_R a$. Then, by Assumption A, there is no $c \in R \cap [P, e]$ with $c >_R b$. This means $[a, R] \subseteq Q$ and thus $Q \times_a R \subseteq Q \setminus \{b\}$, contradicting Lemma 2.17. \square

2.5 Proof of abstract max flow/min cut over time

We will show that C not only covers all paths in \mathcal{P}_T but even those paths that use storage at intermediate elements, implying optimality of the constructed temporally repeated abstract flow for the relaxation of the problem. We are thus allowed to use the results from Section 2.4 in the proof.

Theorem 2.10a $C \cap P_\sigma \neq \emptyset$ for every $P_\sigma \in \mathcal{P}_T^*$.

Proof. By contradiction assume there is a path that is not covered by C . Among all uncovered paths choose $P_\sigma \in \mathcal{P}_T^*$ such that the length $\sum_{e \in P} (\tau(e) + y^*(e))$ is minimal. We will show that there is an uncovered path R whose length is strictly shorter, yielding a contradiction.

Let $\bar{e} \in P$ be maximal with respect to $<_P$ among all $e \in P$ with $\gamma(P_\sigma, e) \geq \alpha(e)$. Note that such an element exists because the first element of P fulfills this inequality. By construction, P_σ arrives at \bar{e} after the element has entered the cut. Note that, as P_σ is not covered by the cut, the path must actually arrive at \bar{e} after it has left the cut again, i.e., $\gamma(P_\sigma, \bar{e}) \geq \alpha(\bar{e}) + y^*(\bar{e})$. Adding $\tau(\bar{e})$ to both sides of this inequality yields

$$\sum_{e \in [P, \bar{e}]} (\sigma(e) + \tau(e)) \geq \alpha(\bar{e}) + y^*(\bar{e}) + \tau(\bar{e}). \quad (2.1)$$

Now let $Q \in \mathcal{P}$ be a path with $\sum_{e \in (Q, \bar{e})} (\tau(e) + y^*(e)) = \alpha(\bar{e})$, and let

$$R := Q \times_{\bar{e}} P.$$

In order to show that R actually contradicts our choice of P , we first argue that \bar{e} cannot be the final element of P . Assume this was the case. Then $Q \times_{\bar{e}} P \subseteq [Q, \bar{e}]$ and thus $\alpha(\bar{e}) + \tau(\bar{e}) + y^*(\bar{e}) = \sum_{e \in [Q, \bar{e}]} (\tau(e) + y^*(e)) \geq T$ by feasibility of y^* . Combining this with (2.1) yields $\sum_{e \in P} (\sigma(e) + \tau(e)) \geq T$, a contradiction to $P_\sigma \in \mathcal{P}_T^*$.

Thus, \bar{e} is not the final element of P and we let e' be the successor of \bar{e} on P . Observe that the choice of \bar{e} and the definition of α imply

$$\gamma(P_\sigma, e') < \alpha(e') \leq \sum_{e \in [P, \bar{e}]} (\tau(e) + y^*(e)). \quad (2.2)$$

Note that the left hand side of (2.1) is bounded from above by $\gamma(P_\sigma, e')$ and thus combining (2.1) and (2.2) yields $\alpha(\bar{e}) < \sum_{e \in (P, \bar{e})} (\tau(e) + y^*(e))$. Therefore

$$\sum_{e \in R} (\tau(e) + y^*(e)) \leq \alpha(\bar{e}) + \sum_{e \in [\bar{e}, P]} (\tau(e) + y^*(e)) < \sum_{e \in P} (\tau(e) + y^*(e)).$$

Now let $s := \sum_{e \in [Q, \bar{e}]} y^*(e) + \sum_{e \in [Q, \bar{e}] \setminus R} \tau(e)$ and $\sigma' := (s, 0, \dots, 0) \in \mathcal{T}^R$. We will show that the temporal path $R_{\sigma'}$ is not covered by C , which contradicts the choice of P_σ as uncovered path minimizing the length with respect to $\tau + y^*$.

Let $f \in R$. We show $(f, \gamma(R_{\sigma'}, f)) \notin C$. Note that $\gamma(R_{\sigma'}, f) = s + \sum_{e \in (R, f)} \tau(e)$. Our results from Section 2.4 further imply that R consists of two parts: The first part containing elements from $[Q, \bar{e}]$ in the same order as $<_Q$, the second part containing elements from (\bar{e}, P) , in the same order as $<_P$.

- If $f \in [Q, \bar{e}]$, then

$$\gamma(R_{\sigma'}, f) \geq \sum_{e \in (Q, f)} (\tau(e) + y^*(e)) + y^*(f) \geq \alpha(f) + y^*(f).$$

So $R_{\sigma'}$ reaches f after it has left the cut in this case.

- If $f \in R \setminus [Q, \bar{e}]$, then

$$\begin{aligned} \gamma(R_{\sigma'}, f) &= \sum_{e \in [Q, \bar{e}]} (\tau(e) + y^*(e)) + \sum_{e \in (\bar{e}, R) \cap (R, f)} \tau(e) \leq \alpha(\bar{e}) + \tau(\bar{e}) + y^*(\bar{e}) + \sum_{e \in (\bar{e}, P) \cap (P, f)} \tau(e) \\ &\leq \gamma(P_{\sigma}, \bar{e}) + \sum_{e \in [\bar{e}, P] \cap (P, f)} (\tau(e) + \sigma(e)) \leq \sum_{e \in (P, f)} (\tau(e) + \sigma(e)) + \sigma(f) \end{aligned}$$

where the penultimate inequality follows from $\gamma(P_{\sigma}, \bar{e}) \geq \alpha(\bar{e}) + y^*(\bar{e})$. Note that the last term of the inequality is equal to $\gamma(P_{\sigma}, f)$, and thus strictly less than $\alpha(f)$ due to the choice of \bar{e} and the fact that $f >_P \bar{e}$. So $R_{\sigma'}$ reaches f before it enters the cut in this case.

This concludes the proof. □

2.6 Conclusion

In this chapter, we introduced and investigated abstract flows over time, an extension of flows over time that can be viewed as a first approach towards more general dynamic packing problems. Our main result shows that the max flow/min cut result of Ford and Fulkerson still is valid in Hoffman's setting of abstract flows, emphasizing the robustness of the concept. Our proofs rely exclusively on the switching axiom for abstract networks, showing how thoroughly this abstraction actually captures the essential forces behind total dual integrality in network-based packing problems.

Open problems and future research

Some interesting questions concerning abstract flows have already been posed in [McC96] and most of them have remained open. In addition, we would like to point out two further directions of future research that have been brought up by our study.

Dynamic packing problems. As we laid out in the introduction, our research on abstract flows over time originated from a project to further the understanding of “dynamic” packing problems, i.e., packing problems with a temporal component, where solutions may vary over time and a decisions taken at some point in time may impact the state of the solution at later points as well. Canonical candidates for future investigations are, e.g., matching problems, knapsack problems, and packing integer programs in general. Some preliminary results for matchings over time have already been established

in [BKM⁺12]. Similar packing problems over time have also been investigated by Adjashvili, Bosio, and Weismantel [ABW12], who provide complexity results and an approximation algorithm for the case where the underlying structure is an independence system.

Shortest abstract paths. Our results, specifically the proof of Lemma 2.14, reveal an interesting implication of McCormick’s abstract flow algorithm [McC96] and the equivalence of optimization and separation [GLS88]: Given an abstract network (\mathcal{P}, E) with non-negative weights $w \in \mathbb{Q}_+^E$ on the elements, a *shortest abstract path*, i.e., a path $P \in \mathcal{P}$ minimizing $\sum_{e \in P} w(e)$ can be found in time polynomial in E , $\langle w \rangle$, and $\text{TIME}(\mathbb{O}_{\text{path}}, \mathcal{P})$, even when the abstract network can only be accessed by the oracle \mathbb{O}_{path} (note that the weighted abstract flow algorithm of Martens and McCormick [MM08] needs a much stronger separation oracle). This directly leads to the question of whether there is a combinatorial—possibly strongly polynomial—algorithm for the shortest abstract path problem.

Chapter 3

An integrated approach to tactical transportation planning in logistics networks

In this chapter, we introduce a new model for the optimization of freight transportation in logistics networks. The main features of our approach include accurately modeled tariff structures and the integration of spatial and temporal consolidation effects via a cyclic expansion of the network. We propose various heuristic methods for solving the resulting capacitated network design problem, most notably a local search procedure based on path decomposition of network flows and an aggregated mixed integer programming formulation that also provides lower bounds on the value of the optimal solution. In a computational study based on data from our project partner 4flow AG, we show that most of our solutions are within a single-digit percentage of the optimum.

Publication remark: The results presented in this chapter are joint work with Tobias Harks, Felix G. König, Alexander Richter, and Jens Schulz [HKM⁺].

Modeling and optimizing the transportation of goods in a network is a central application of network flow theory. While early models assumed transportation costs to be linear, at present, capacitated network design formulations—which combine network flows with network design—allow for much more precise replication of the highly involved tariff systems offered by logistics carriers in practice. *Transportation tariffs* are complex cost functions, depending on different properties of the shipment often in a non-linear or even non-continuous way. Most tariffs, however, exhibit an “economies of scale” structure, i.e., the per-unit shipping cost decreases with increasing size of the payload. This observation motivates the *consolidation* of shipments, i.e., the aggregation of several small shipments into a larger one. Consolidation may occur over space as well as over time. In *spatial* consolidation, material flows of different origins are combined at an intermediate node and forwarded jointly to the next. In *temporal* consolidation, material is kept in inventory at a node for some time in order for more flow to accumulate, thereby enabling a larger outbound shipment. This leads to a complex tradeoff between efficient usage of economies of scale, short transportation routes and low inventory cost. The increasing world-wide shipping volumes and the availability of comprehensive data and network analytics fosters an interest in more precise models and optimization techniques that address this tradeoff in an integrated fashion.

In this chapter of the thesis, we introduce a new model for *tactical transportation planning*, the task of optimizing inventory levels, material flows, and tariff choices for freight transportation in logistics networks. Our model integrates temporal delivery

patterns and inventory decisions into a multi-commodity flow formulation with a very general set of cost functions that captures many important cases of transportation tariffs used in logistics networks today. We also present algorithms for solving the optimization problem resulting from this formulation and generating lower bounds on the value of the optimal solution. The model has been developed in close collaboration with logistics experts at 4flow AG, a logistics consultancy company serving small, medium-sized and global customers from a broad spectrum of industries. Our algorithmic methods were evaluated on a library of transportation networks obtained from recent and ongoing customer projects of 4flow AG.

Chapter outline

A general description of transportation planning—the logistical task that is the subject of our model—is given in Section 3.1. This section also introduces literature covering transportation models and solution methods for related problems.

In Section 3.2, we introduce a new model for tactical transportation planning. The main decision variables of our model include the flow paths of commodities through the network, the choice of transportation tariffs, and inventory levels. Several graph-based gadgets enable us to formulate complex tariff systems within a capacitated network design problem. By using a cyclic expansion of the network, our model also includes the possibility for flexible delivery patterns, accurately modeling the tradeoff between inventory cost and economies of scale in transportation.

In the process of designing heuristic algorithms suitable for large-scale instances of logistics networks arising in practice, we identified the problem of selecting optimal tariffs on a single link in the network as an important subproblem that is crucial in speeding up the solution process: In order to compute cost efficient paths, our algorithms need good and fast estimates on the cost incurred by sending a particular amount of flow along a transport relation. In Section 3.3, we will propose different algorithms that provide a good balance between accuracy and speed for solving this *NP*-hard subproblem.

In Section 3.4, we then devise a local search heuristic that employs local changes on a path decomposition of flow in the network using the tariff selection subroutines mentioned above. In contrast to many local search heuristics known in the literature—that either directly modify the network design or reroute flow of a single commodity only—our approach applies a neighborhood search based on path decomposition of flow and re-routing multiple commodities simultaneously and then adjusting the network design accordingly. In order to obtain good initial solutions for our local search heuristic, we provide two successive shortest path type algorithms. By forbidding certain paths (for instance direct connections) and linearizing costs we further tune the initial solutions towards a high level of flow consolidation that will eventually be disaggregated by the local search heuristic.

In Section 3.5, we complement our heuristic approach by mixed integer programming (MIP) techniques. While our model can be naturally formulated as a MIP, this plain formulation is not suited for solving reasonably sized real-world instances due to enormous problem sizes. Instead, we propose an aggregated formulation that considerably reduces model size and still yields good lower bounds on the solution value. We combine this with efficient preprocessing techniques to tighten the relaxation and a post-processing step to improve the solution quality. Combining the LP relaxation of this strengthened and aggregated formulation with the tariff selection heuristics mentioned

earlier yields a third way of constructing initial solutions for our local search procedure.

In Section 3.6, we evaluate the performance of our different algorithmic approaches on a library of real-world instances provided by our project partner 4flow AG. The test set consists of case studies from the automotive, chemical, and retail industry with up to thousands of facilities and hundreds of commodities. We can show that most of our solutions are within a single-digit percentage of the optimum, and that our modeling and algorithmic techniques yield a cost reduction of over ten percent over the current status quo, which can result in annual savings of several millions of euros.

3.1 Introduction to transportation planning

Before we can introduce our model in Section 3.2, we give an introduction to the logistical task referred to as *transportation planning*, which is the subject of the model. We also discuss literature related to this topic.

3.1.1 Problem description

In the following, we describe the important aspects of transportation planning and introduce some terminology we will use throughout this chapter.

Levels of planning. Due to a strong variance in lead times associated with the different decisions to be made when designing transportation networks, the planning process is hierarchically structured in strategic, tactical, and operational levels [SLKSL03]. The work presented in this chapter focusses on the *tactical* level: We assume the location and product decisions to be already made, and the general design of the network to be fixed. Typical logistic decisions on the tactical level include the amount of flow between the existing nodes of the network, e.g., which customers to serve from which warehouses or suppliers, how much inventory to keep at which locations, and which transportation modes and delivery frequencies to employ on the different connections [GP03]. In contrast, strategic planning is concerned with long-term decisions such as location of facilities, while operational planning focuses on the daily tasks of operating the network such as scheduling workforce and meeting time-windows for delivery. We will discuss later in Section 3.2.5 to what extent our results can be translated to strategic and operational planning.

Transportation networks. We consider a network of *facilities*, which are of different types, e.g., production plants, warehouses, distribution centers, or retailers. Some facilities have a *supply* of, or a *demand* for certain products, also known as *commodities*. The number of commodities can be large and distinct commodities typically differ in many aspects, e.g., their weight, volume, or value. Facilities are joined by *transport relations*, and on each transport relation, different *transportation tariffs* are available corresponding to concurring offers of freight forwarders and available transportation modes. Each tariff is characterized by capacity restrictions and a cost function, describing how much of a commodity (or of some commodity mix) can be transported, and at which cost. E.g., a full truck load tariff may have the payload and footprint of a certain truck type as capacity restrictions and incur a fixed charge cost. Some facilities may be able to carry

inventory, usually with a commodity-dependent capacity and cost. *Handling cost* may result from commodities passing through facilities, such as distribution centers, regardless of whether they are moved to inventory or not.

Consolidation. Quite commonly, transportation cost includes fixed-charge costs for dispatching shipments, and the larger a shipment, the lower the effective per-unit shipping cost. Hence, a key ingredient to successful tactical planning in a logistics network is the efficient *consolidation* of material flows, i.e., the combination of smaller order amounts into larger shipments in order to utilize capacity efficiently and enable economies of scale [Çet05]. As already pointed out in the introduction, consolidation may occur over space—in form of aggregation of different shipments at intermediate hubs—as well as over time—in form of accumulation of material through lower shipment frequencies. While temporal consolidation may incur cost for holding inventory, spatial consolidation may require shipments to deviate from the shortest path to their destination. Thus, in both cases, a tradeoff has to be considered. Also note that spatial and temporal consolidation are not mutually exclusive, they may in fact occur jointly at the same node.

Temporal patterns. The interplay between inventory cost and different transportation tariffs necessitates a notion of time in planning. Since temporal details such as transport transit times or demand deadlines are commonly postponed to operational planning, the goal in tactical optimization is a cyclic *pattern* of deliveries and inventory. The length and structure of this pattern usually follows some natural notion of rough timing, like “once every month”, “once every week” or “once every day of the week”, and in each slot of the pattern (like in one month, week or weekday), deliveries are dispatched, and inventories are replenished or depleted.

In conclusion, the outcome of tactical transportation planning as described above comprises

- the paths each commodity takes through the network, i.e., the total amount of flow for each commodity on each transport relation,
- the transportation tariffs employed on each transport relation, together with an assignment of a commodity mix to each of them,
- a cyclic pattern in which transports are executed for each tariff used on each transport relation, including the amounts shipped for each commodity in each slot of the pattern, and finally
- a pattern of inventory levels for each commodity at each node, supporting the above transport patterns.

Note that in tactical planning, the aim is not to use the results to operate the logistics network directly, as this is the subject of operational planning. Rather, tactical optimization intends to aid with decisions which have to be made with some lead time, providing the framework for efficient operation: How much throughput capacity needs to be reserved at certain distribution centers? Which logistics provider should be cooperated with on which network connections, and which available tariffs will be employed on what volume of commodities? Hence, the main purpose of many details in tactical modeling is not primarily to reflect operational reality, but much more to yield a realistic assessment of operational cost in the framework provided.

3.1.2 Related work

Mathematical optimization for logistic problems has been a vast field of research for several decades. We give an overview of models and algorithms for transportation planning.

Models for transportation planning

Transportation planning as a scientific field is part of *supply chain management* (SCM), an area that deals with “the management of flows between and among all stages of a supply chain to maximize total profitability” [CM07]. Literature on SCM is as broad and diverse as the field itself, see the textbooks by Simchi-Levi, Kaminsky, and Simchi-Levi [SLKSL03] and Chopra and Meindl [CM07]. An excellent overview of network-based optimization techniques for SCM is given by Geunes and Pardalos [GP03]. The authors review articles dealing with strategic as well as tactical and operational planning.

In one of the earliest optimization models for SCM by Geoffrion and Graves [GG74], the authors model a multi-commodity network with several plants, possible distribution center locations, and demand zones on the strategic level. While the model incorporates fixed location costs, as well as upper and lower bounds on the throughput of a distribution center, it does not consider inventory decisions and assumes transportation costs to be linear. The resulting MIP model is solved using Benders decomposition.

A strategic optimization model that incorporates the interdependence of location, transportation, and inventory decisions is described by Jayaraman [Jay98]. Here, different transportation modes can be chosen for each connection in the network. Each mode is associated with a commodity-dependent per-unit cost and a *delivery frequency*. Keeping inventory at a plant or warehouse incurs per-unit inventory cost, and the amount of inventory held results from the delivery frequencies of the outbound transportation modes used. Note that this still captures temporal consolidation rather coarsely, as theoretically, also transportation modes with low delivery frequency could carry low shipping volume, making their assumed low per-unit cost unrealistic. The model is solved using standard MIP solvers.

While the above network-wide SCM models are focussed on strategic planning and incorporate location decisions, the tactical and operational tradeoff between transportation and inventory cost lies at the heart of *dynamic lot-sizing* in inventory theory. In the basic version of dynamic lot-sizing introduced by Wagner and Whitin [WW58], different demands for a commodity at a single facility need to be met in multiple periods. In each period, an arbitrary amount can be ordered at fixed per-order cost, while per-unit inventory cost is incurred. The goal is to determine the amount ordered in each period such that all demands are met on time and the sum of order and inventory cost is minimized. Wagner and Whitin show that this problem can be solved to optimality in polynomial time by dynamic programming. This basic model has been extended in many ways since then, and most variants are computationally hard, see e.g., the literature review by Jans and Degraeve [JD07]. The practical importance of considering the trade-off between transportation and inventory cost is highlighted impressively by Burns et al. [BHBD85, BBD⁺87], who were able to reduce logistics cost by 26% in a case study for General Motors.

Generalizing lot-sizing to networks with multiple stages brings it closer to the requirements of tactical transportation planning. The first such model was introduced by Clark and Scarf [CS60] and further developed by Afentakis, Gavish, and Karmarkar [AGK84,

AG86]. An overview of more recent works can be found in [Sta03]. Most of these models, however, still make rather restrictive assumptions on the structure of the network considered and transportation costs incurred. Moreover, the quantity of material flowing between node pairs is fixed a priori in all lot-sizing models, so the possibility for more spatial consolidation at hubs is effectively ignored.

Kempkes, Koberstein, and Suhl [KKS10] propose a general model for the integrated operational planning of external and internal logistics of the last two stages of a supply chain. In their model, all costs depend on the usage of resources, such as vehicle capacities or workforce, and this dependence can be piecewise constant as well as linear and may involve multiple resources. Planning occurs over multiple however non-cyclic periods, and in particular, inventory cost is taken into account. The authors devise a flow-based construction heuristic to generate an initial feasible solution that is passed to a standard MIP solver. In order to introduce all details necessary for realistic operational planning, their model even allows for logical relations between different resources, which however significantly increases the algorithmical challenge of solving large scale instances. Accordingly, their solution approaches are validated on relatively small instances involving only five planning periods with networks of up to 25 nodes, several hundred arcs, and up to one hundred commodities.

In a more tactical context, Schöneberg, Koberstein, and Suhl [SKS10] propose a similar resource-based model for optimizing the choice of delivery profiles in *area forwarding based networks*. In such networks, suppliers are grouped into areas and each area is equipped with a consolidation center run by a logistics carrier. The main decision variables are the choices from a fixed set of delivery profiles for each supplier and the usage of vehicles on the main legs (i.e., the connections between consolidation centers and the target). The authors propose a solution method that first decomposes the model by fixing certain decisions for each possible delivery profile and then generates an initial feasible solution for the MIP solver using a two-phase construction heuristic. The approach is evaluated in the logistics network of a German truck manufacturer, achieving cost savings of up to 36% in individual areas.

The transportation model introduced in this chapter, as well as the models in [KKS10] and [SKS10] are based on capacitated network design formulations, as discussed in more detail below. An alternative approach to modeling non-linear transportation tariffs are concave-cost network flows, see [GP90] for a survey. Note, however, that also all three models mentioned above include the possibility of concave cost functions; see Section 3.2.4 for how such functions can be modeled in context of our model.

Capacitated network design

While network flow seems to be the dominant aspect in transportation planning, the fixed cost nature of transportation tariffs brings in network design decisions: We have to install sufficient capacity in the network such that all flow can be routed. In literature, such mixtures of network flow and network design are referred to as *capacitated network design* or *fixed-charge network flow*, and they are widely used for models not only in logistics but also in telecommunication and infrastructure planning; see the surveys by Magnanti and Wong [MW84] and Crainic [Cra00]. Most capacitated network design problems are not only challenging to solve in practice but have been shown to be very hard from a theoretical point of view as well; see Section 1.3 for an overview of complexity results for network design problems.

Tabu search procedures. The intrinsic hardness of capacitated network design, combined with the enormous size of instances encountered in practical applications from logistic contexts, leaves little hope for exact solution approaches that run in acceptable time. Therefore, fast combinatorial heuristics appear to be the method of choice. The current state of the art is mainly built on specialized tabu search procedures. Crainic, Gendreau, and Farvolden [CGF00] proposed a tabu search procedure based on a neighborhood in the multi-commodity flow polytope. Their algorithm has later been adapted for parallelization by Crainic and Gendreau [CG02]. A different neighborhood for tabu search was introduced by Ghamlouche, Crainic, and Gendreau [GCG03], operating on the network design and modifying it along cycles. This procedure has been refined by the same authors by supplementing it with a path relinking technique [GCG04].

Slope scaling. A different approach for solving fixed-charge network flow problems is constituted by *slope scaling*. The slope scaling procedure, first proposed by Kim and Pardalos [KP99a] for single-commodity fixed-charge network flow, iteratively solves the min-cost flow problem arising from linearizing the fixed costs according to the current solution. Crainic, Gendron, and Hernu [CGH04] generalize this technique to the case of multiple commodities, and augment it by Lagrangian perturbation and intensification/diversification mechanisms based on a long-term memory.

Benders decomposition. Among the MIP based solution techniques for capacitated network design, *Benders decomposition* [Ben62] appears to be particularly well-suited, as it separates the complex network design decisions from the well-understood network flow substructure. A survey of various applications of Benders decomposition in this context is given by Costa [Cos05]. Costa, Cordeau, and Gendron [CCG09b] investigate the relation between different classes of inequalities. In particular, the authors explain how the inequalities from (non-extreme) dual rays of the Benders framework and cut-set inequalities can be strengthened via shortest path computations to become metric inequalities. To improve the running times, Fischetti, Salvagnin, and Zanette [FSZ10] suggest to generate cuts from a minimal infeasible subsystems. While it is *NP*-hard to find such systems, they show that this task can be carried out heuristically.

Valid inequalities. While MIP formulations of capacitated network design problems usually yield relatively weak linear programming relaxations, these can be significantly improved by adding stronger valid inequalities to the formulation. Chouman, Crainic, and Gendron [CCG09a] investigate the effect of various classes of such inequalities on the efficiency of the branch and bound process. In particular, they present separation and lifting procedures for *strong capacity*, *cover*, *minimum cardinality*, *flow pack*, and *flow cover* inequalities. We will discuss some of these classes in more detail in Section 3.5.2.

Tractable instance sizes. All solution methods referenced above are designed for capacitated network design problems in general graphs. The combinatorial tabu search and slope scaling algorithms have been successfully tested on a standard benchmark set of randomly generated instances of moderate size with at most 100 nodes and 400 arcs, introduced in [CGF00]. Regarding the approaches based on mixed integer programming techniques, the above works indicate that the scope of tractable instance sizes is roughly

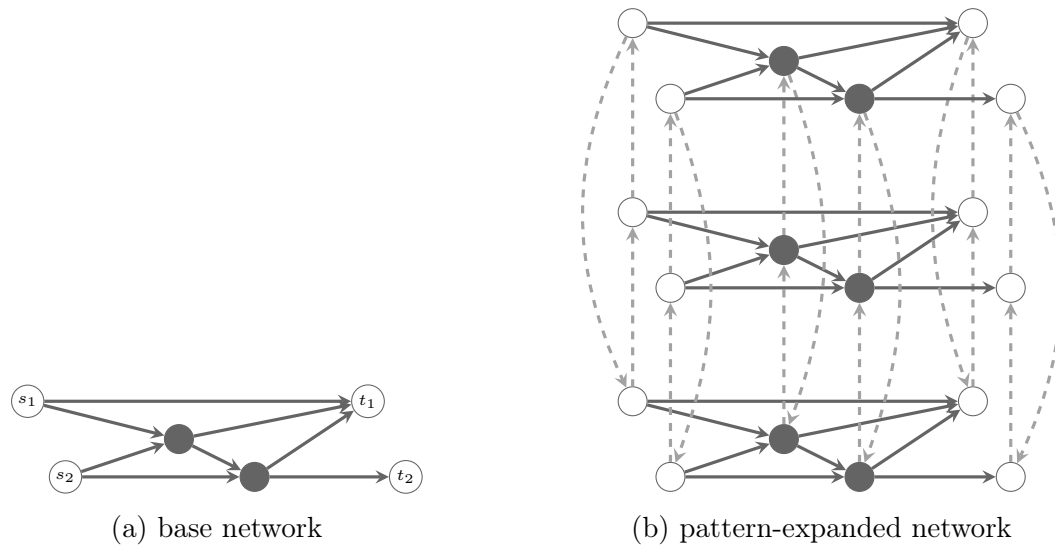


Figure 3.1: A base network and the associated pattern-expanded network with cycle length $F = 3$. Dashed arcs denote holdover arcs.

limited to 30 nodes, 500 arcs and 200 commodities, i.e., for the few larger instances reported on, the provable gaps on solution quality exceed single digits.

3.2 Mathematical model

Our model, which is formally defined as the *tactical transportation planning problem* (TTP) is at its heart based on multi-commodity network flow, with both linear and fixed-charge cost on the arcs. However, we extend the standard concepts of capacity and cost to more generality in order to reflect the requirements of logistics modeling more precisely. Moreover, we expand the underlying network significantly in order to model delivery patterns, inventory effects, and complex tariff systems. We proceed to detail all of these features in the following, describing basic concepts such as pattern expansion and the set of possible transportation tariffs in Section 3.2.3, which leads to a first formulation of our model as multi-commodity flow problem with non-linear cost functions. We then provide a reformulation of the model as a capacitated network design problem in Section 3.2.4. Finally, we discuss the advantages and inherent challenges of our model in Section 3.2.5.

3.2.1 Pattern expansion

The tradeoff between minimizing inventory cost and taking advantage of the economies of scale in transportation is of key importance in tactical logistics planning. Temporal and spatial consolidation effects regularly determine which tariff is most suitable on a connection. Consequently, even the decision which path in the network is most efficient for a commodity may ultimately depend on temporal delivery patterns. As tactical planning defines the environment for operational planning which will take place again and again over time, a solution should be a *cyclic* pattern for dispatching deliveries and replenishing

and depleting inventories. To integrate temporal and spatial consolidation together with cyclic delivery patterns, we introduce the notion of *pattern-expanded networks*, similar to the concept of time-expanded networks introduced in Section 1.2.5, but with a cyclic structure.

A pattern-expanded network denoted by \mathcal{D} has two main components: The first is the *base network* B , which comprises the physical entities of the transport network: facilities (or *nodes*) together with corresponding *transport relations* between facilities (the *arcs* of the base network). The second parameter is a cycle length F defining the number of time slots available in a period (e.g., 7 days of a week). The pattern-expanded network \mathcal{D} is now obtained from B and F by introducing F copies of B denoted by B_1, \dots, B_F and connecting copies of each node of every two adjacent networks B_i and B_{i+1} by *holdover arcs*, being directed from the nodes in B_i to those in B_{i+1} . Moreover, the nodes of the last copy B_F are also connected by holdover arcs to their corresponding copies in the first slot B_1 , yielding a cyclic network structure. If commodities are sent along holdover arcs from B_F to B_1 , this corresponds to storing commodities at the corresponding nodes at the end of a cycle, to the beginning of the next cycle. Costs can be associated with holdover arcs modeling inventory costs. In the following we will conceptually not differentiate between holdover arcs and transport arcs. We denote the set of nodes in the pattern-expanded network by \mathcal{V} and the set of all arcs of \mathcal{D} , which we will refer to as transport relations throughout the chapter, by \mathcal{R} .

The construction of the pattern-expanded network from the base network is illustrated in Figure 3.1.

3.2.2 Commodities and properties

Commodities in a logistics network can be very diverse, e.g., in their size, weight, or value, and logistic costs and transport capacities cannot be realistically assumed to be oblivious to this diversity and the resulting interdependencies when mixing commodities in transport. We introduce the concept of flexible *properties* to characterize commodities. A set of commodities K and a list of relevant properties P are parameters of our model. Each commodity $i \in K$ is assigned a per unit extent $\alpha_{ij} \in \mathbb{Q}_+$ for each property $j \in P$. The main motivation for introducing these properties is that transportation costs introduced in the next section will mostly depend on the total extent of each property of a commodity mix (rather than the specific type of commodities itself), thus reflecting the effects of consolidating goods for utilizing vehicle capacities more efficiently.

Throughout this chapter, we will denote the *aggregated properties* of a vector $x \in \mathbb{Q}_+^K$ by $\alpha(x) \in \mathbb{Q}_+^P$ with

$$\alpha_j(x) := \sum_{i \in K} \alpha_{ij} x_i.$$

Each node in the pattern-expanded network may supply or demand certain commodities. These supplies and demands are expressed by a balance vector $b(v) \in \mathbb{Q}^K$ for each node $v \in \mathcal{V}$. Note that we allow different values for distinct copies of the same node in the base network. A node with a supply ($b_i(v) > 0$) of a certain commodity $i \in K$ is called a *source* of i , and a node with a demand ($b_i(v) < 0$) is called a *sink* of i . The goal is to satisfy all demands by transporting the supplies from the sources to the sinks. Without loss of generality, we will assume $\sum_{v \in \mathcal{V}} b_i(v) = 0$ for all $i \in K$.

3.2.3 Transportation tariffs

When shipping goods on a transport relation, different *transportation tariffs* are available. For each transport relation $R \in \mathcal{R}$ we denote by $T(R)$ the set of available tariffs for transporting a flow of commodities from $\text{tail}(R)$ to $\text{head}(R)$. Each such tariff $t \in T(R)$ is associated with a cost function $C_t : \mathbb{Q}_+^K \rightarrow \mathbb{Q}_+$; a list of possible cost functions is given below. A solution of our model consists of a multi-commodity flow in the pattern-expanded network satisfying all demands, together with an assignment of the flow on each transport relation to the tariffs available on this relation.

Problem: Tactical transportation planning (TTP)

Input: A pattern-expanded network $\mathcal{D} = (\mathcal{V}, \mathcal{R})$, a set of commodities K and a set of properties P with per-unit extents $\alpha \in \mathbb{Q}_+^{K \times P}$, a balance vector $b(v) \in \mathbb{Q}_+^K$ for every node $v \in \mathcal{V}$, and a set of tariffs $T(R)$ on P and K for every $R \in \mathcal{R}$.

Task: Find a multi-commodity b -flow $x \in \mathbb{Q}_+^{K \times \mathcal{R}}$ in \mathcal{D} and a tariff assignment $\tilde{x} \in \mathbb{Q}_+^{K \times T(R)}$ for each transport relation $R \in \mathcal{R}$ such that $\sum_{t \in T(R)} \tilde{x}_i(t) = x_i(R)$ for all $i \in K$, minimizing the cost $\sum_{R \in \mathcal{R}} \sum_{t \in T(R)} C_t(\tilde{x}(t))$.

We will now present a set of cost functions that covers most tariffs occurring in current logistical applications. In the next section, we will then show how all these cost functions can also be modeled in a unified form as a capacitated network design problem.

Linear costs. In many logistical applications, commodity-dependent linear costs of the form

$$C_t(\tilde{x}) = \sum_{i \in K} c_i \cdot \tilde{x}_i$$

with cost rates $c_i \in \mathbb{Q}_+$ for each commodity occur, e.g., in the context of handling costs, in-stock and in-transit inventory costs and simple linear tariffs without interdependencies of the transported commodities.

Maximum over multiple cost rates. Tariffs can also be specified as the maximum over varying cost rates for distinct properties, i.e., when sending a shipment that rate applies for which the cost is highest. More formally, with c_j being the cost rate for property $j \in P$, the cost function is given as

$$C_t(\tilde{x}) = \max_{j \in P} c_j \cdot \sum_{i \in K} \alpha_{ij} \tilde{x}_i.$$

Note that, in contrast to the linear costs described before, these maximum cost functions capture the effect of cost savings when mixing commodities of different dimensions, e.g., light but voluminous with heavy but compact ones.

Property-dependent piecewise constant costs. Many tariffs, such as those offered by most full truck load (FTL) carriers and some less than truck load (LTL) carriers, are based on piecewise constant cost functions, i.e., they are specified by a cost $c \in \mathbb{Q}_+$ and a capacity vector $\beta \in \mathbb{Q}_+^P$ for a single shipment, yielding the function

$$C_t(\tilde{x}) = c \cdot \max_{j \in P} \left\lceil \frac{\alpha_j(\tilde{x})}{\beta_j} \right\rceil.$$

In practice, logistic carriers offer groups of such tariffs realizing different levels of discount for higher shipment volumes. We will see in Section 3.3 that finding the most cost-efficient combination of such tariffs for a given shipment volume is already an *NP*-hard problem.

Of course, linear and fixed costs can also occur at the same time, e.g., to model a transport to a distribution center which incurs fixed cost for transportation and a linear cost for handling the incoming shipment at the distribution center. We thus also allow the combination of these two cost types.

Incremental discount costs. We consider a tariff with varying cost rates depending on a single property. The cost rates are specified on intervals and decrease with increasing size of shipment, resulting in a piecewise linear and concave cost function; see Table 3.1 for an illustration. Formally, label the intervals from 0 to L . For each $\ell \in [L]$, let $c^{(\ell)} \in \mathbb{Q}_+$ be the cost rate on the interval $[\beta_j^{(\ell)}, \beta_j^{(\ell+1)})$ for the fixed property $j \in P$, with $0 = \beta_j^{(0)} < \beta_j^{(1)} < \dots < \beta_j^{(L)} < \beta_j^{(L+1)} = \infty$ and $c^{(0)} > c^{(1)} > \dots > c^{(L-1)} > c^{(L)}$. Then the cost function is

$$C_t(\tilde{x}) = \sum_{\ell=0}^L c^{(\ell)} \cdot \min \left\{ \beta_j^{(\ell+1)} - \beta_j^{(\ell)}, \left(\alpha_j(\tilde{x}) - \beta_j^{(\ell)} \right)^+ \right\}.$$

All-unit discount costs. Again we consider linear cost rates in some property $j \in P$ with several levels of decreasing per-unit cost rates. Different from the above, however, a cost rate applies to the entire transport volume as long as it lies within the corresponding interval. To ensure monotonicity, a cost cap applies whenever the cost with respect to the current rate exceeds the cost at the beginning of the next level—this corresponds to the common practice of declaring higher volumes than actually transported in such cases [CMS⁺02]. See Table 3.1 for a graphical illustration of the resulting cost function. Formally, if cost rate $c^{(\ell)}$ for $\ell \in [L]$ is applicable starting from transport volume $\beta_j^{(\ell)}$ on, the cost function is

$$C_t(\tilde{x}) = \min_{\ell \in [L]} \left(c^{(\ell)} \cdot \max \left\{ \alpha_j(\tilde{x}), \beta_j^{(\ell)} \right\} \right).$$

3.2.4 Reformulation as capacitated network design

We will now provide a different perspective to the model presented in the previous section. We introduce the concept of *containers* to model the different types of tariffs in a way that leads to a unifying description of the above model as a fixed-charge multi-commodity flow problem. This description corresponds to a natural mixed integer programming formulation, making the model accessible to MIP based solving techniques. Its compact structure demonstrates the degree of mathematical uniformity achieved by the model.

We will first present the alternative formulation of the model in full detail, and then show the equivalence to the formulation in the previous section by describing how different cost functions can be modeled using containers.

The tariff-expanded network

For each tariff on a transport relation, we introduce a *gadget* consisting of different arcs, which connects the start node of the relation with its end node. On each arc, a certain type of *container* is available, and capacities can be installed on the arc in increments of this container type. After replacing all transport relations in the pattern-expanded network by the corresponding gadgets for their tariffs, we obtain the *tariff-expanded network* $D = (V, A)$ consisting of the original nodes of the pattern-expanded network, the additional nodes introduced in the gadgets and the arcs introduced in the gadgets.

A solution to the container-based formulation of our model specifies for each arc a the integer number of containers $y(a)$ installed on a together with the arc flow values $x_i(a)$ for each commodity i . In the context of capacitated network design, the variables $y(a)$ are known as *design variables*, while the variables $x_i(a)$ are known as *flow variables*.

Each container has a capacity for every property. For each property, the capacity installed on a must be sufficient to transport the flow. More formally, the capacity constraints can be described as follows. Recall that α_{ij} denotes the per-unit extent of commodity i with respect to property j , and let $\beta_j(a)$ be the corresponding capacity of a container at arc a . Then the *capacity constraints*

$$\sum_{i \in K} \alpha_{ij} x_i(a) \leq \beta_j(a) y(a) \quad \forall j \in P \quad (3.1)$$

must hold at every arc $a \in A$. Moreover, an upper bound $u(a)$ on the number of containers installed on an arc a may be specified.

In a feasible solution, the multi-commodity flow x has to satisfy all demands. We extend the node balances introduced for the nodes in the pattern-expanded network by setting the balances for all nodes artificially introduced by tariff expansion to zero for each commodity. We thus obtain the *flow conservation constraints*

$$\sum_{a \in \delta^+(v)} x_i(a) - \sum_{a \in \delta^-(v)} x_i(a) = b_i(v) \quad \forall i \in K \quad (3.2)$$

that must be valid at every node $v \in V$ of the tariff-expanded network.

For each container installed on a , a fixed cost $c(a)$ has to be paid. Flow sent along a may furthermore incur a commodity-dependent linear cost $c_i(a)$, which may also be used to model property dependent linear costs. Thus, the total cost of a solution is

$$\sum_{a \in A} \left(c(a) y(a) + \sum_{i \in K} c_i(a) x_i(a) \right).$$

Putting all of this together, the fixed-charge multi-commodity flow problem resulting from the container formulation can be directly formulated as a MIP.

$$\begin{aligned}
[\text{TTP}_{\text{CND}}] \quad & \min \quad \sum_{i \in K} \sum_{a \in A} c_i(c) x_i(a) + \sum_{a \in A} c(a) y(a) \\
& \text{s.t.} \quad \sum_{a \in \delta^+(v)} x_i(a) - \sum_{a \in \delta^-(v)} x_i(a) = b_i(v) \quad \forall v \in V, \forall i \in K \\
& \quad \quad \quad \sum_{i \in K} \alpha_{ij} x_i(a) \leq \beta_j(a) y(a) \quad \forall a \in A, \forall j \in P \\
& \quad \quad \quad y(a) \leq u(a) \quad \forall a \in A \\
& \quad \quad \quad x_i(a) \in \mathbb{Q}_+, \quad y(a) \in \mathbb{Z}_+ \quad \forall a \in A, \forall i \in K
\end{aligned}$$

Note that a flow in the tariff-expanded network (i.e., on arcs) can be transformed into a flow in the pattern-expanded network (i.e., on transport relations) by setting the flow value $\tilde{x}(t)$ assigned to tariff $t \in T(R)$ on some transport relation R to be the amount of flow going from $\text{tail}(R)$ to $\text{head}(R)$ through the gadget corresponding to t in the tariff-expanded network—this corresponds to the total amount shipped using this tariff. Conversely, a flow in the tariff-expanded network can be obtained from a flow in the pattern-expanded network. The gadget of each tariff t will be designed to model its cost function C_t in the sense that the minimum cost incurred by the flow in the gadget—in terms of required container capacity and linear costs—equals $C_t(\tilde{x}(t))$. Therefore, the total cost of the solution in the tariff-expanded network equals the cost of the flow in the pattern-expanded network.

Modeling tariffs with containers

We now proceed to explain how containers can be used to accurately model the different types of transportation tariffs introduced in the previous section; see Table 3.1 for an overview of the more complex gadgets.

Modeling linear and piecewise constant costs. It is clear that both commodity-dependent linear costs and property-dependent piecewise constant costs are directly captured by the container concept. Linear costs are part of the definition, while piecewise constant tariff groups can be directly modeled by introducing a bundle of parallel arcs, one for each tariff in the group. The container on each arc takes the capacity and cost of the corresponding tariff.

Modeling the maximum over multiple cost rates. In order to model the maximum over multiple cost rates we need to introduce *fractional containers* to the model, i.e., the variable $y(a)$ corresponding to the number of installed copies of such a container can be fractional. We use a single gadget arc for each tariff that corresponds to maximum over multiple cost rates c_j with $j \in P$. We set the cost to $c(a) = 1$ and the capacity $\beta_j(a) = 1/c_j$ for each $j \in P$. Sending a flow of $x(a)$ through this arc requires $y(a)$ to be set to $\max_{j \in P} \alpha_j(x(a))/\beta_j(a)$, which is equal to the cost function by choice of $\beta_j(a)$. Note that introducing such fractional containers does not have significant impact on the complexity of the model. Still, for the sake of simplicity, we will assume throughout this work that all containers have to be installed in integral increments.

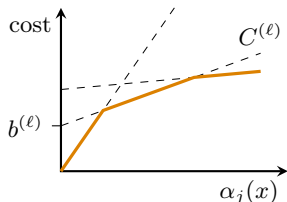
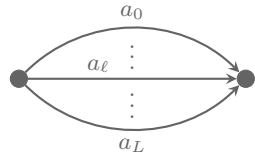
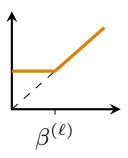
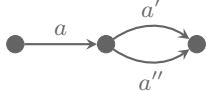
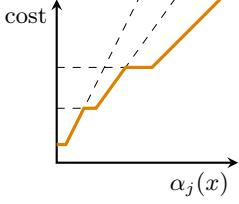
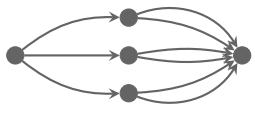
tariff	cost	gadget																
incremental discount (piecewise linear concave)	 $C(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$ $C^{(\ell)}(x) := c^{(\ell)}\alpha_j(x) + b^{(\ell)}$	<p>minimum modeled by parallel arcs</p>  $c(a_\ell) = b^{(\ell)}$ $c_i(a_\ell) = \alpha_{ij}c^{(\ell)}$																
all-unit discount	 $C^{(\ell)}(x) := c^{(\ell)} \cdot \max \{ \alpha_j(x), \beta^{(\ell)} \}$	 <table border="1" data-bbox="1013 817 1356 952"> <thead> <tr> <th></th> <th>a</th> <th>a'</th> <th>a''</th> </tr> </thead> <tbody> <tr> <td>c</td> <td>$c^{(\ell)}\beta^{(\ell)}$</td> <td>0</td> <td>0</td> </tr> <tr> <td>c_i</td> <td>0</td> <td>$\alpha_{ij}c^{(\ell)}$</td> <td>0</td> </tr> <tr> <td>β_j</td> <td>∞</td> <td>∞</td> <td>$\beta^{(\ell)}$</td> </tr> </tbody> </table>		a	a'	a''	c	$c^{(\ell)}\beta^{(\ell)}$	0	0	c_i	0	$\alpha_{ij}c^{(\ell)}$	0	β_j	∞	∞	$\beta^{(\ell)}$
		a	a'	a''														
c	$c^{(\ell)}\beta^{(\ell)}$	0	0															
c_i	0	$\alpha_{ij}c^{(\ell)}$	0															
β_j	∞	∞	$\beta^{(\ell)}$															
	 $C(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$	<p>minimum modeled by parallel gadgets</p> 																

Table 3.1: Modeling tariff systems with containers.

Modeling incremental discounts. Piecewise linear concave functions arising from incremental discount tariffs can be interpreted as the minimum of several affine linear functions. Again denoting the linear segments of the function by 0 to L with cost rates $c^{(\ell)}$ and break points $\beta^{(\ell)}$, we define

$$C^{(\ell)}(x) := c^{(\ell)}\alpha_j(x) + b^{(\ell)} \quad \text{with} \quad b^{(\ell)} := \sum_{k=0}^{\ell-1} (c^{(k)} - c^{(\ell)})(\beta_j^{(k+1)} - \beta_j^{(k)})$$

for $\ell \in [L]$. It is easy to verify that $C_i(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$; see Table 3.1 for an illustration. We now introduce a gadget of $L + 1$ parallel arcs a_0, \dots, a_L with $c(a_\ell) = b^{(\ell)}$ and $c_i(a_\ell) = \alpha_{ij}c^{(\ell)}$. Sending flow along arc a_ℓ incurs the cost $C^{(\ell)}$ and an optimal solution will always send flow along that arc which achieves the minimum cost for the transported amount.

Modeling all-unit discounts. Note that functions of the form $c^{(\ell)} \cdot \max \{ \alpha_j(x), \beta^{(\ell)} \}$ can be modeled by the following gadget; also see the corresponding figure in Table 3.1.

Introduce a series-parallel graph, consisting of a single arc a followed in series by two parallel arcs a' and a'' . We set the fixed costs $c(a) = c^{(\ell)}\beta^{(\ell)}$ and $c(a') = c(a'') = 0$. We also set the linear costs $c_i(a) = c_i(a'') = 0$ and $c_i(a') = \alpha_{ij}c^{(\ell)}$ for all $i \in K$. Capacity $\beta_j(a'')$ is set to $\beta^{(\ell)}$, all other capacities are left infinite, and we let $u(a'') = 1$ so that only one container can be installed on a'' , while the number of containers remains unbounded for all other arcs. Now, all-unit discount tariffs, which can be represented as minimum of such functions, can be modeled by introducing several of these gadgets in parallel.

3.2.5 Model characteristics

We want to close this section by discussing some characteristics and possible extensions of the TTP model introduced above.

Additional aspects of modeling. The two key ingredients of the model are a cyclic pattern expansion to incorporate inventory and frequency decisions and a characterization of commodities in terms of scalar properties that enables the precise replication of real-world transportation tariffs. We want to point out two additional general concepts that are implicitly covered by these modeling techniques and that are thus captured by our model. Firstly, the TTP model includes the possibility of omitting some holdover arcs or even some transport arcs of the base network in individual time slots, in order to model restricted operation times of transportation services or hubs. The second concept are abstract aspects of commodities, such as “needs cooling”, “is hazardous” and similar features restricting their transportation. These can be modeled by introducing a corresponding property, letting the respective commodities receive a strictly positive extent in this property and accordingly adjusting container capacities.

Computational tractability. Naturally, the generality of our model comes at the price of computational challenges. As a generalization of the directed Steiner forest problem, it does not allow for approximation factors significantly better than linear in the number of nodes [DK99]; see Section 1.3 for details.

Theorem 3.1 *For any $\varepsilon > 0$, there is no $2^{\log^{1-\varepsilon}(|V|)}$ -approximation algorithm for TTP, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$.*

Pattern and tariff expansion allow us to model temporal consolidation effects and complex tariff systems in a uniform way without further increasing the complexity from a theoretical point of view—assuming all slots in the pattern-expanded network are given explicitly. However, both expansions significantly amplify the size of the network. In the remaining sections of this chapter, we will show how to cope with this computational challenge in practice.

Strategic and operational transportation planning. Finally, we address the validity of our model in the context of strategic and operational planning. As pointed out at the beginning of Section 3.1.1, our model is aimed at the tactical level. In particular, it does not address location decisions, which are an important part of strategic planning. Fixed costs for opening and running facilities differ from fixed costs for transportation in that they affect all slots of the pattern-expanded network. Thus, incorporating such

decisions would significantly change the complexity of our model, which is designed to be solved with network flow based approaches. However, we want to point out that our model can very well be used to assist strategic planners in evaluating different network layouts: In Section 3.6, we assess solution methods that are sufficiently fast to solve multiple instances separately in a row. In the context of operational planning, the exact time of dispatchment, transit time, and arrival time of a shipment play a crucial role. We intentionally omitted these aspects in the design of the pattern-expanded network as they are not the subject of tactical planning. Of course, transit times can be included by adjusting the arcs of the pattern-expanded network in the same way as it is done in the time-expanded network of Ford and Fulkerson [FF62]. However, it still needs to be investigated how to incorporate other aspects relevant for operational planning into our model, such as scheduling personnel, vehicle usage, or loading devices and return of empties.

3.3 Tariff selection subproblem

While containers constitute a versatile tool to model various transportation tariffs as described in Section 3.2.3, the use of elaborate gadgets significantly increases the number of arcs in an instance of our model. Different algorithms may or may not be able to cope well with this challenge. In this section, we describe an approach to curb the effects of model blow-up due to tariff gadgets by encapsulating tariff selection decisions in a subordinate optimization problem, which we call the *tariff selection* subproblem (TS). While some of our algorithms for TTP introduced later will operate directly on tariff gadgets as introduced in Section 3.2.4, others will solve TS repeatedly, possibly several hundred times for each transport relation, while computing a flow pattern for all commodities through the network.

In contrast to the global perspective of the TTP model, TS constitutes a local decision limited to a single transport relation $R \in \mathcal{R}$: Given a fixed vector $\Delta \in \mathbb{Q}_+^K$ of flow to be transported on R , it asks which transportation tariffs should be selected and how the fixed demand should be distributed among the selected tariffs in order to meet flow demand at minimum cost. A solution to TS comprises a vector $x(t) \in \mathbb{Q}_+^K$ of multi-commodity flow for each tariff $t \in T(R)$ such that their sum meets the total flow demand Δ . From a network-wide perspective, solving the set of TS problems on all transport relations optimizes transport cost with respect to a given fixed multi-commodity flow in the pattern-expanded network.

Problem: Tariff selection (TS)

Input: A set K of commodities, a set P of properties, per-unit extents $\alpha \in \mathbb{Q}_+^{K \times P}$, shipping amounts $\Delta \in \mathbb{Q}_+^K$, and a set of transportation tariffs T on P and K with cost functions C_t for $t \in T$.

Task: Find a tariff assignment $\tilde{x} \in \mathbb{Q}_+^{K \times T}$ with $\sum_{t \in T} \tilde{x}_i(t) = \Delta_i$ for all $i \in K$, minimizing the cost $\sum_{t \in T} C_t(\tilde{x}(t))$.

Depending on which of the five types of tariff cost functions introduced in Section 3.2.3 are present in TS, we employ different techniques in order to solve TS. In Section 3.3.1

we devise a mixed integer programming formulation for arbitrary combinations of tariff cost functions. However, out of the different tariff cost functions, property-dependent piecewise constant costs stand out for a number of reasons. Firstly, even though they constitute the most elementary class of cost functions, finding an optimal tariff selection is already *NP*-hard when restricting to piece-wise constant costs; see Theorem 3.2. Secondly, they are a very common, if not the most common, tariff type in logistic applications. Indeed, in the real-life data for our computational study in Section 3.6, most transport relations are equipped exclusively with piecewise constant tariffs. Therefore, Section 3.3.2 is devoted to theoretic and algorithmic insights into TS for this tariff type.

Combinatorial algorithms for TTP have to solve TS as a subroutine with a very high frequency. Due to its hardness and the demand for extremely short computation times, we develop fast heuristic algorithms for piecewise constant tariffs yielding only approximate solutions as an alternative to the exact MIP approach. In particular, we propose an efficient greedy algorithm for computing solutions of decent quality within a minimum of computation time and a cost estimator, which instead of a feasible solution only outputs an estimate of the optimal cost of the given instance.

A more detailed analysis of the techniques presented in this section, together with an extensive computational study on real-world and randomly generated tariff selection instances can be found in [KMR12].

3.3.1 MIP for the general case

The introduction of tariff gadgets in Section 3.2.4 enables us to naturally formulate and solve TS as a mixed integer program. This versatile approach is especially suited when various tariff types occur together on a single transport relation, or when computational time is not a great issue, e.g., if flow paths for all commodities are already specified and TS only needs be solved once on each transport relation to optimize the tariff choice. When each tariff $t \in T(R)$ is represented by a container gadget $(V(t), A(t))$ as described in Section 3.2.3, we denote with $A(R) := \bigcup_{t \in T(R)} A(t)$ and $V(R) := \bigcup_{t \in T(R)} V(t)$ the set of all arcs and nodes, respectively, that are used to model the tariff structure on transport relation R . TS for R can then be written as

$$\begin{aligned}
\min \quad & \sum_{a \in A(R)} c(a)y(a) + \sum_{i \in K} c_i(a)x_i(a) \\
\text{s.t.} \quad & \sum_{a \in \delta^+(v)} x_i(a) - \sum_{a \in \delta^-(v)} x_i(a) = \begin{cases} \Delta_i & \text{if } v = \text{tail}(R) \\ -\Delta_i & \text{if } v = \text{head}(R) \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V(R), \forall i \in K \\
& \sum_{i \in K} \alpha_{ij}x_i(a) \leq \beta_j(a)y(a) \quad \forall a \in A(R), \forall j \in P \\
& y(a) \leq u(a) \quad \forall a \in A(R) \\
& y(a) \in \mathbb{Z}_+, x_i(a) \in \mathbb{Q}_+ \quad \forall a \in A(R), \forall i \in K.
\end{aligned}$$

As this MIP only involves the tariff choices on a single transport relation, the corresponding instances are rather small and can be solved near-optimally in reasonable time for matters of post-optimization.

3.3.2 Piecewise constant costs

When all tariffs on a transport relation are of the property-dependent piecewise constant type, the tariff-expanded counterpart of the transport relation is a bundle of parallel fixed-charge container arcs. In this case, the MIP formulation of TS can be simplified to

$$\begin{aligned}
\min \quad & \sum_{a \in A(R)} c(a)y(a) \\
\text{s.t.} \quad & \sum_{a \in A(R)} x_i(a) = \Delta_i \quad \forall i \in K \\
& \sum_{i \in K} \alpha_{ij}x_i(a) \leq \beta_j(a)y(a) \quad \forall a \in A(R), \forall j \in P \\
& y(a) \in \mathbb{Z}_+, x_i(a) \in \mathbb{Q}_+ \quad \forall a \in A(R), \forall i \in K.
\end{aligned}$$

It is not hard to see that finding an optimal tariff selection for piecewise constant cost functions is *NP*-hard, even for very restricted special cases. We give a straightforward reduction from the well-known unbounded knapsack problem, which is known to be *NP*-hard [Lue75], to TS instances with only a single property and a single commodity.

Problem: Unbounded knapsack

Input: A set of n items with values $v_1, \dots, v_n \in \mathbb{Z}_+$ and weights $w_1, \dots, w_n \in \mathbb{Z}_+$, capacity $W \in \mathbb{Z}_+$, and a desired value $V \in \mathbb{Z}_+$.

Task: Find numbers $z_1, \dots, z_n \in \mathbb{Z}_+$ such that $\sum_{i=1}^n w_i z_i \leq W$ and $\sum_{i=1}^n v_i z_i \geq V$, or decide that no such numbers exist.

Theorem 3.2 *The tariff selection problem is NP-hard, even when restricted to instances with only piecewise constant cost functions, a single property, and a single commodity.*

Proof. In the single-commodity single-property case, the above MIP reduces to $|A(R)|+1$ non-trivial constraints, and there remain single variables α_{ij} , x and β_j , which we denote by α , x and β , respectively. Every feasible solution satisfies $\alpha\Delta \leq \sum_{a \in A(R)} \beta(a)y(a)$, and conversely, if this inequality is satisfied, it is trivial to find feasible assignments $x(a)$. Hence, the MIP reduces in fact to a single non-trivial constraint.

Given an instance I_{UK} of the unbounded knapsack problem, we construct an instance I_{TS} of the above special case of TS as follows. First, for every item $i \in \{1, \dots, n\}$ of I_{UK} , define $u_i := \lceil W/w_i \rceil$ to be the maximum number of items of type i in a feasible knapsack solution. Then, for each item $i \in \{1, \dots, n\}$ introduce a corresponding arc a_i with containers of fixed cost $c(a_i) = v_i$ and capacity $\beta(a_i) = w_i$. Moreover, we set $\Delta = \sum_{i=1}^n w_i u_i - W$ and $\alpha = 1$.

We now argue that I_{UK} possesses a solution with value at least V if and only if I_{TS} can be solved with cost at most $\sum_{i=1}^n v_i u_i - V$. First assume there is a feasible solution z to I_{UK} with value at least V . We define $y(a_i) := u_i - z_i$ and observe that

$$\sum_{i=1}^n \beta(a_i)y(a_i) = \sum_{i=1}^n w_i(u_i - z_i) \geq \sum_{i=1}^n w_i u_i - W = \alpha\Delta$$

Algorithm 3.1: Greedy algorithm for tariff selection

```

Initialize  $\Delta' = \Delta$ ,  $x = 0$ ,  $y = 0$ ,  $x^* = 0$ ,  $y^* = 0$ .
while  $\Delta' \neq 0$  do
  if  $\exists a^* = \operatorname{argmin}\{c(a) : a \in A(R), \alpha(\Delta') \leq \beta(a)\}$  then
    if  $y^* = 0$  or  $c(y) + c(a^*) < c(y^*)$  then
      Set  $y^*(a) = y(a)$  and  $x^*(a) = x(a)$  for all  $a \in A(R) \setminus \{a^*\}$ .
      Set  $y^*(a^*) = y(a^*) + 1$  and  $x^*(a^*) = x(a^*) + \Delta'$ .
    Choose  $a' \in \operatorname{argmax}\{\operatorname{score}(a, \Delta') : a \in A(R)\}$ .
    Set  $d = \operatorname{fill}(a', \Delta')$ .
    Set  $k = \left\lfloor \min \left\{ \frac{\Delta'_i}{d_i} : i \in K, d_i > 0 \right\} \right\rfloor$ .
    Set  $y(a') = y(a') + k$  and  $x(a') = x(a') + kd$ .
    Set  $\Delta' = \Delta' - kd$ .
    if  $y \neq 0$  and  $c(y^*) < c(y)$  then
      return  $x^*, y^*$ 
return  $x, y$ 

```

and

$$\sum_{i=1}^n c(a_i)y(a_i) = \sum_{i=1}^n v_i(u_i - z_i) \geq \sum_{i=1}^n v_i u_i - V.$$

We omit the converse of the argument as it works analogously. \square

Greedy algorithm

We now present a generic greedy algorithm to heuristically solve instances of TS for piecewise constant cost functions. The inherent covering nature of TS—in the sense that we select containers in order to “cover” the capacity extents of a fixed flow vector—motivates us to devise a generalization of the natural greedy approach to integer programs with nonnegative data as studied for example by Dobson [Dob82]. We emphasize that our methods are specifically designed to cope well with the given practical instances: in those instances all properties and capacities are strictly positive, they are always feasible and the number of properties is small. Though some of the following methods also work with zero-valued properties or capacities and feasibility tests could be easily incorporated, we omit the explicit treatment of these issues for the sake of readability.

The greedy algorithm for tariff selection repeatedly selects a “most efficient” container $a \in A(R)$ to cover portions of, or the whole remaining shipping amount $\Delta' \in \mathbb{Q}_+^K$. In this context, the “efficiency” of a container is measured by a function $\operatorname{score}(a, \Delta')$, which reflects the ratio between cost of container type $a \in A(R)$ and the shipping amount it covers. The selected container then is packed using the function $\operatorname{fill}(a, \Delta')$, which returns a vector $d \in \mathbb{Q}_+^K$, with $d_i \leq \Delta'_i$ for all $i \in K$ and $\alpha_j(d) \leq \beta_j(a)$ for all $j \in P$, trying to ensure an “efficient” capacity usage of the container. To speed up the algorithm we can assign the computed mix of commodities d multiple times to copies of the same container, as long as $d \leq \Delta'$. The algorithm repeats until all demand is assigned, i.e., Δ' is reduced to zero. During the course of the algorithm, there might be containers large enough to cover all remaining demand, although the score method still favors a smaller container that covers only fractions and leaves demand for the next step. In such situations it is advisable to consider both container types and to branch on the computed

solution: Among all containers that suffice to cover Δ' , the algorithm picks the one with least cost and constructs a *complete solution* (x^*, y^*) to be stored separately—if it is better than any previous complete solution—and it also proceeds with the partial solution arising from selecting the container with best `score` value.

A formal listing of the greedy algorithm is given as Algorithm 3.1. To simplify notation we denote by $c(y) := \sum_{a \in A(R)} c(a)y(a)$ the selection cost of a vector $y \in \mathbb{Z}_+^{A(R)}$.

Implementation of `score` and `fill`

Algorithm 3.1 uses two subprocedures called `score` for estimating “container efficiency”, and `fill` for computing corresponding assignment $d \in \mathbb{Q}_+^K$ of commodities to the container. Several variants of different implementations of these subprocedures, including LP based methods and simple scaling techniques, have been tested in [KMR12]. For the implementation of the algorithms incorporated into the TTP solvers, we decided to base both `score` and `fill` on a two-phase greedy algorithm, which we shortly outline here. Recall that the input of both subprocedures consists of a container type $a \in A(R)$ and the vector of remaining shipping amount $\Delta' \in \mathbb{Q}_+^K$. The algorithm tries to greedily fill the container a by approximating the ray induced by its capacity vector $\beta(a)$ with the property extents $\alpha(d) \in \mathbb{Q}_+^P$ of the computed vector d . The two phases are described in detail below. The subroutine `score` only executes the first phase of this algorithm and uses the resulting vector $d \in \mathbb{Q}_+^K$ to return the score $\sum_{j \in P} \alpha_j(d)/c(a)$. Note that `score` is executed far more frequently than `fill` and thus restricting to the first phase significantly reduces computation time. Once a container is selected, `fill` returns the refined filling derived by the second phase.

Phase 1. The algorithm maintains a vector $d \in \mathbb{Q}_+^K$ of demand assigned to the container, starting with $d = 0$. It will iteratively increase d by adding a certain amount δ of a commodity $i^* \in K$. Let $\bar{\beta}(a) := \beta(a) - \alpha(d)$ denote the residual capacity of the container with respect to the currently assigned demand. At the beginning of every iteration the algorithm computes

$$\delta_i := \min \left\{ \frac{\bar{\beta}_j}{\alpha_{ij}} : j \in P, \alpha_{ij} > 0 \right\} \cup \{ \Delta'_i \}$$

for every commodity $i \in K$. Note that δ_i corresponds to the maximal assignment of commodity i that can be added to d without violating the capacities of the container. The algorithm iteratively chooses a commodity $i^* \in K$ that minimizes the Euclidian norm of the vector of slacks with respect to this assignment, i.e.,

$$i^* \in \operatorname{argmin}_{i \in K} \|\bar{\beta}(a) - \delta_i \alpha_i\|_2$$

and adds δ_{i^*} units of commodity i^* to d and subtracts the same amount from Δ' . The first phase ends when $\delta_i = 0$ for all $i \in K$.

Phase 2. The second phase tries to further reduce unused container capacities by adjusting the vector d produced in the first phase. It does so by approximating the ray induced by the capacity vector $\beta(a)$ with a conic combination of the extent vectors $\alpha_i \in \mathbb{Q}_+^P$ of the available commodities $i \in K$. More formally, we decompose the property space $\mathbb{Q}^P = \operatorname{span}(\beta(a)) + \operatorname{span}(\beta(a))^\perp$ into the linear subspace spanned by the

capacity vector $\beta(a)$ and its orthogonal complement and consider for each commodity $i \in K$ the unique decomposition of its extent vector $\alpha_i = v_i + u_i$ with $v_i \in \text{span}(\beta(a))$ and $u_i \in \text{span}(\beta(a))^\perp$. The current filling vector $d \in \mathbb{Q}_+^K$ induces the vector

$$\alpha(d) = \sum_{i \in K} d_i \alpha_i = \sum_{i \in K} d_i v_i + \sum_{i \in K} d_i u_i \in \mathbb{Q}_+^P.$$

Our goal of approximating the ray induced by $\beta(a)$ corresponds to minimizing the orthogonal deviation $\|\sum_{i \in K} d_i u_i\|_2$. For commodity $k \in K$, we define

$$\lambda_k := \sum_{i \in K} d_i \frac{u_i^T u_k}{u_k^T u_k}.$$

Note that $\lambda_k u_k$ corresponds to the projection of $\sum_{i \in K} d_i u_i$ on $\text{span}(u_k)$. We choose an $i^* \in K$ with $\lambda_{i^*} < 0$ and increase d by $\min\{-\lambda_{i^*}, \Delta'_{i^*}\}$ units of commodity i^* , which leads to a decrease of the orthogonal deviation. We iteratively augment d in this way until no additional improvement can be achieved by any commodity. Note that the resulting vector d might violate container capacities. We therefore scale d down to feasibility.

Cost estimation by covering relaxation

In many situations in which TS occurs as a subproblem in the course of an algorithm for TTP, it is not important to know which tariffs are actually utilized in a solution, but merely which cost is incurred. Examples include shortest path type algorithms where the transport relations leaving some node are to be labeled with the cost of forwarding some flow along them. In these situations, the following covering relaxation can be used to obtain considerable speed-ups while still computing reasonable cost estimates. The relaxation is based on dropping the requirement of an exact assignment of the commodities to containers. Instead, we only require the chosen containers to cover the vector of aggregated properties $\Gamma := \alpha(\Delta')$ induced by the flow vector Δ' . The result of this relaxation is the following covering integer program:

$$\begin{aligned} \min \quad & \sum_{a \in A(R)} c(a) y(a) \\ \text{s.t.} \quad & \sum_{a \in A(R)} y(a) \beta_j(a) \geq \Gamma_j \quad \forall j \in P \\ & y(a) \in \mathbb{Z}_+ \quad \forall a \in A(R) \end{aligned}$$

We can heuristically solve this problem very efficiently by adjusting Algorithm 3.1 to directly operate on the vector Γ instead of considering Δ , i.e., we reduce Γ by $\beta(a)$ for each selected container copy of type a . An appropriate scoring function can be defined by

$$\text{score}(a, \Gamma) := \frac{1}{c(a)} \min \left\{ \frac{\beta_j(a)}{\Gamma_j} : j \in P, \Gamma_j > 0 \right\}.$$

Note that a solution to the covering relaxation does not necessarily yield a feasible solution for the original TS problem. In fact, one can easily come up with counterexamples where the estimate obtained from the relaxation is arbitrarily far away from the actual optimal solution value of TS. However, these examples are of rather artificial nature, including containers with near-zero capacity in certain properties. Such cases do not occur in the instances of TS arising from our practical data.

3.4 Path-based local search

We propose a local search procedure that employs local changes on a path decomposition of flow in the pattern-expanded network using tariff selection subroutines. As described in Section 3.1.2, there are already a number of local search heuristics available for solving capacitated network design problems. Adapting those methods to multi-dimensional capacities and non-binary design variables does not suffice to cope with the large instance sizes occurring in practical application of our model: The precise replication of complex tariff structures leads to a drastically increased number of (mostly parallel) arcs, which is further amplified by the cyclic expansion of the network; to give rough numbers, the tariff-expanded networks in our computational study have 250,000 arcs on average, corresponding to a blow-up factor of 60 from an average of 4000 arcs in the base networks. This poses a great computational challenge for heuristics that operate in the tariff-expanded network without knowledge of the tariff structure. While most methods known from literature either work directly on the network design, delegating the corresponding flow computations to a subproblem, or re-route flow of a single commodity in each iteration, our approach applies a neighborhood search that is based on path decomposition of flow in the pattern-expanded network and it re-routes multiple commodities simultaneously.

In order to obtain good initial solutions for the local search algorithm that is presented in Section 3.4.3, we also provide two successive shortest path type algorithms. The first method linearizes costs by estimating the per unit cost. It is denoted by SPLC (for *shortest path with linearized cost*) and it is presented in Section 3.4.1. The second method uses a tariff selection method for the purpose of cost estimation. It is denoted by SPTS (for *shortest path with tariff selection*) and it is presented in Section 3.4.2. The SPLC method is designed with an emphasis on speed and low memory requirement, making it possible to obtain solutions of reasonable quality in short time, even for very large networks. The SPTS method, on the other hand, is more accurate in cost estimation and is therefore also used as the central subroutine for re-routing flow in our local search improving moves.

In the process of analyzing the local search procedure, we identified a certain asymmetry in its ability to find cost savings arising from consolidation and deconsolidation of flow in the network, respectively. We observed that the procedure very well detects possible cost savings resulting from splitting up paths that share a common transport relation and re-routing the flow separately. In contrast, detecting potential savings from consolidating several disjoint flow carrying paths by re-routing them along a shared subpath is not well captured. Note that such savings may only be realized by re-routing multiple paths at once. Identifying such a set of paths is an algorithmically challenging task. In order to address this issue, we adapt the two path-based algorithms to encourage consolidation by (i) forbidding direct source-sink-connections—which is well-suited for the type of logistical networks arising in our study—in the SPLC heuristic and (ii) using a partial linearization technique for SPTS. Both refinements yield considerable improvements in solution quality of the local search procedure as we will see in the computational study in Section 3.6.

3.4.1 Shortest Paths with linearized costs (SPLC)

A straightforward idea for obtaining arc costs for a shortest path computation is estimating the per unit shipping cost on each arc in the tariff-expanded network by linearizing

the fixed costs. This technique yields a highly efficient approach suited for solving even the largest occurring instances in a very small amount of time.

In each iteration, the algorithm chooses a commodity and finds a shortest path from a source to a sink with respect to arc weights $w \in \mathbb{Q}_+^A$. Whenever the algorithm encounters an arc during the shortest path computation, the residual capacity for the chosen commodity on this arc is computed and the fixed cost for that arc is divided by this capacity to obtain a linear cost rate. To make this more precise, let $k \in K$ be the commodity that is currently being routed and (x, y) be the current (partial) solution of the capacitated network design formulation [TTP_{CND}] consisting of the flow $x \in \mathbb{Q}_+^{K \times A}$ and the design choices $y \in \mathbb{Z}_+^A$. Using the notation introduced in Section 3.2.4, we compute the *residual capacity* of arc $a \in A$ for commodity k provided by the $y(a)$ containers currently installed on the arc. This capacity is defined by

$$\rho(a) := \min_{j \in P} \frac{\beta_j(a)y(a) - \alpha_j(x(a))}{\alpha_{kj}}.$$

If there is a positive residual capacity $\rho(a) > 0$, we set the capacity $r(a) := \rho(a)$ and the weight $w(a) := c_k(a)$, only considering the linear cost for shipping commodity k along a . If no residual capacity is left, i.e., $\rho(a) = 0$, then an additional container can be installed on a if $y(a) < u(a)$. In this case, we set

$$r(a) := \min_{j \in P} \frac{\beta_j(a)}{\alpha_{kj}} \quad \text{and} \quad w(a) := c_k(a) + \frac{c(a)}{r(a)}.$$

Otherwise, if $y(a) = u(a)$, we set $r(a) := 0$ and $w(a) := \infty$.

Once a shortest path P from a source to a sink of commodity k with respect to the weights w is found, the *bottleneck capacity* $r := \min_{a \in P} r(a)$ is determined and r units of commodity k are sent along the path. Note that all computations above can be carried out very efficiently and of course, instead of updating weights and capacities of all arcs in each step, these values are calculated on-demand and only updated when necessary. Technically, we flag those variables $r(a)$ and $w(a)$ as ‘invalid’ that have to be recomputed at the next encounter of arc a . A listing of SPLC is given as Algorithm 3.2.

The linearization procedure assumes optimal utilization of container capacities in the resulting flow pattern and thus favors large containers with low per unit cost rates. Since this high utilization is not always attained, the linearization often leads to suboptimal tariff choices on transport relations. The effect can be compensated by optimizing the tariff selection on each transport relation a posteriori with a tariff selection method described in Section 3.3.

Consolidation by forbidding direct connections (SPLC-F). The SPLC heuristic favors large containers with low per unit cost rates and prefers direct connections from source to sink over paths along intermediate hubs—note that the latter can only yield shorter paths with respect to the weights w , when some container with residual capacity is already installed on an intermediate arc. A simple approach for encouraging consolidation when costs are linearized is to forbid all direct connections between sources and sinks of the same commodity during the construction of the initial solution. By doing so, the heuristic is forced to route flow along intermediate hubs, and the paths intersect automatically. Unnecessary detours can be easily identified and corrected by improving moves of the local search procedure.

Algorithm 3.2: Successive shortest path algorithm with linearized costs (SPLC)

```

Initialize  $x = 0, y = 0$ .
for each commodity  $i \in K$  do
  Invalidate  $r(a)$  and  $w(a)$  for all  $a \in A$ .
  while there is a source  $s$  of  $i$  with remaining supply do
    Find path  $P$  in  $G$  from  $s$  to a sink  $t$  with  $\sum_{a \in P} w(a)$  minimum, updating
    the values of  $r(a)$  and  $w(a)$  on-demand when the previous value has been
    invalidated.
    Augment  $x$  along  $P$  by  $\min_{a \in P} r(a)$  units of commodity  $i$ , adjust  $y$ 
    accordingly.
    Invalidate  $r(a)$  and  $w(a)$  for all  $a \in P$ .

```

3.4.2 Shortest paths with tariff selection (SPTS)

The rather imprecise estimation of the actual transportation cost achieved by the linearization approach presented above might lead to weak choices of paths for optimizing the cost of the final solution. We thus propose a second strategy that employs tariff selection algorithms already during the shortest path search. Although this more sophisticated approach requires more computational effort, it still turns out to be very efficient while at the same time providing several possibilities for adjustments.

Since tariff selection methods require as input the amount of flow to be routed, these flow values $\Delta \in \mathbb{Q}_+^K$ have to be determined *before* the shortest paths computation. We implement this a priori flow computation efficiently by identifying source-sink-pairs such that the possible transport volume from source to sink is maximum (with respect to a weighted combination of the property extents). To make this more formal, let x be the current flow in the network and recall the definition of the excess of this flow at node v as

$$\text{ex}(x_k, v) := \sum_{a \in \delta^-(v)} x_k(a) - \sum_{a \in \delta^+(v)} x_k(a)$$

for $k \in K$. Using the supply/demand values $b \in \mathbb{Q}^{K \times V}$ given in the input, define for each ordered pair of nodes (s, t) in the pattern-expanded network the value $\Delta(s, t, x) \in \mathbb{Q}_+^K$ by

$$\Delta_k(s, t, x) := (\min \{b_k(s) + \text{ex}(x, s), -(b_k(t) + \text{ex}(x, t))\})^+$$

for $k \in K$. Then source s and sink t are chosen such that $\sum_{j \in P} w_j \alpha_j(\Delta(s, t, x))$ is maximum, where α are the aggregated properties of the flow as defined in Section 3.2.2 and $w \in \mathbb{Q}_+^P$ is a weight function, which is an adjustable parameter of the heuristic. A listing of SPTS is given as Algorithm 3.3.

During the shortest path computation, arc weights have to be evaluated too often to solve the tariff selection problem to optimality every time. In fact, it is sufficient to only estimate the cost using the estimator presented in Section 3.3, while the actual tariff assignment can be determined at the end of the solution process from the flow values on the transport relations in the pattern-expanded network using an exact method.

Consolidation by partial cost linearization (SPTS-L). Cost computation based on tariff selection allows for a more sophisticated approach to encourage consolidation by

Algorithm 3.3: Successive shortest path algorithm with tariff selection (SPTS)

Initialize $x = 0$.

while *not all demand has been satisfied* **do**

- Let $s, t \in \mathcal{V}$ such that $\sum_{j \in P} w_j \alpha_j(\Delta(s, t, x))$ is maximum.
- Let $\Delta = \Delta(s, t, x)$.
- Compute shortest path P in \mathcal{D} from s to t w.r.t. \tilde{c} , where $\tilde{c}(R) = \text{TS}(R, x + \Delta) - \text{TS}(R, x)$ with $\text{TS}(R, \tilde{x})$ being the estimated cost of sending flow $\tilde{x} \in \mathbb{Q}_+^K$ along transport relation $R \in \mathcal{R}$.
- Augment x along P by Δ .

Compute a tariff assignment on each transport relation for the given flow x using a tariff selection method.

taking into account the remaining, still unrouted demand. We linearize costs at inter-hub connections and also at source-hub connections (if there are fewer sources) or hub-sink connections (if there are fewer sinks) in the following way: Let \mathcal{R}' be the set of transport relations just described, and let Δ be the amount of flow to be routed in the current iteration. Let $\Delta^+ \in \mathbb{Q}_+^K$ be the sum of all supply not yet routed in the current solution, and define

$$M := \min \left\{ \frac{\alpha_j(\Delta^+)}{\alpha_j(\Delta)} : j \in P, \alpha_j(\Delta) > 0 \right\}.$$

Now, if $R \in \mathcal{R}'$ we compute

$$\tilde{c}(R) := \min \left\{ \text{TS}(R, x + \Delta) - \text{TS}(R, x), \frac{\text{TS}(R, \Delta^+)}{M} \right\}$$

in Algorithm 3.3, where $\text{TS}(R, x)$ denotes the estimated cost needed to transport the flow x along transport relation R , computed by one of the algorithms from Section 3.3. If the minimum is attained by the second argument, the arc cost $\tilde{c}(R)$ anticipates future consolidation on this transport relation by assuming all remaining demand will be send along R and scaling the cost down according to the proportion of Δ^+ represented by Δ . If $R \notin \mathcal{R}'$, we compute $\tilde{c}(R)$ as described originally in Algorithm 3.3. Our computational results in Section 3.6 show that this mixture of linearization at presumably strongly utilized transport relations and precise cost estimation for weakly utilized transport relations yields significant improvements in solution quality.

3.4.3 Path-based local search

In the following we introduce a local search algorithm that re-routes flow along paths with the aim of improving feasible solutions. The algorithm maintains a path decomposition of the flow of the current solution in the pattern-expanded network. It moves from one solution to another by replacing one or multiple paths of the decomposition with paths of lower cost.

The general outline of an improving move is the following: When removing a path P in \mathcal{D} with flow value $x(P) \in \mathbb{Q}_+^K$ from the solution, for each transport relation R of the path, the flow $x(R)$ is decreased by $x(P)$ and the tariff selection of R is adapted accordingly, using the greedy tariff selection heuristic presented in Section 3.3. After removing a set of paths, the resulting partial solution is completed again by computing

new paths using the SPTS heuristic introduced in Section 3.4.2. The move is accepted if the total cost of the solution decreases, and reverted otherwise.

The procedure uses two variants of improving moves: *Type A* moves simply remove a single path at a time. This way, only small amounts of flow are re-routed in one move and the assignment of sources to sinks is left unaffected. In contrast, *Type B* moves consider groups of paths sharing the same transport relation. All flow passing this transport relation is removed and routed anew, which means that multiple paths can be replaced at once and the assignment of sources to sinks might be altered.

The local search procedure performs improving moves in alternating phases of Type A and B. This allows us to re-compute the path decomposition at the beginning of each phase, adapted to the type of movement. In both cases the paths of the decomposition are constructed in a depth-first search (DFS) manner: At a node in the DFS tree for each incident arc R we compute the maximal flow vector $\Delta(R)$ that could be assigned to a path proceeding on that arc and choose an arc greedily so as to maximize a suitably defined weight function of that flow vector. For Type A phases, the DFS starts at a source and continues along the arc that maximizes a weighted combination of the properties of $\Delta(R)$. In contrast, the decomposition for Type B phases facilitates a bidirectional DFS starting at heavily used transport relations and chooses arcs that maximize the savings resulting from reducing their flow. In both cases, due to flow conservation we either close cycles (which can immediately be removed from the solution) or find a source-sink path, which we add to the path decomposition.

The two phases are repeated alternately until the relative improvement achieved by both of them falls below a specified threshold or the time limit is reached. At the end of the procedure, a final improvement phase is conducted by identifying and eliminating weakly utilized containers in the tariff-expanded network and again re-routing the corresponding flow using a variation of Type B moves.

3.5 Mixed integer programming approaches

In this section, we discuss mixed integer programming techniques that supplement the combinatorial heuristics presented in the previous section, not only yielding high quality solutions but also providing lower bounds for assessing solution quality.

The plain MIP formulation presented in Section 3.2.4 is not suited for solving TTP instances in practice as it involves too many variables and constraints when applied to large logistics networks occurring in practice. Instead, we propose an aggregated formulation that considerably reduces model size and still yields good lower bounds on the value of the optimal solution in Section 3.5.1. We then combine this with efficient preprocessing techniques to tighten the relaxation in Section 3.5.2. In Section 3.5.3, we describe how to use solutions to the LP relaxation of this strengthened aggregated formulation as initial solutions for the local search introduced in the previous section. Finally, a post-processing step that improves solution quality is presented in Section 3.5.4. During this post-processing step, tariff selection decisions are locally optimized on all transport relations that connect a given pair of nodes in different slots of the pattern-expanded network.

3.5.1 Aggregated MIP formulation (AMIP)

As mentioned above, the plain MIP formulation [TTP_{CND}] introduced in Section 3.2.4 suffers from the enormous sizes of resulting instances. In particular, the introduction of tariff gadgets results in a high number of—mostly parallel—arcs, each of which is associated with $|K|$ flow variables and $|P|$ capacity constraints. We make use of this parallel structure and propose an aggregated formulation that still reflects the original tariff structures while significantly reducing the number of flow variables and capacity constraints. The aggregation is set up as follows. For each pair of nodes $v, w \in V$ let $A(v, w)$ be the set of arcs from v to w in the tariff-expanded network. For each $i \in K$, we replace the flow variables $x_i(a)$ of all arcs $a \in A(v, w)$ by a single flow variable $x_i(v, w) \in \mathbb{Q}_+^K$. For each $j \in P$, we replace the capacity constraints of the arcs in $A(v, w)$ with respect to j by a single constraint

$$\sum_{i \in K} \alpha_{ij} x_i(v, w) \leq \sum_{a \in A(v, w)} \beta_j(a) y(a).$$

Clearly, the resulting MIP is a relaxation of the original TTP instance, as we can construct a feasible solution of the relaxation from a feasible solution of the original formulation by setting $x_i(v, w) := \sum_{a \in A(v, w)} x_i(a)$ and adopting the values of all design variables. Conversely, each solution of the relaxation induces a flow on the transport relations of the pattern-expanded network. These flow values yield a tariff selection subproblem on each transport relation; see Section 3.3. Computational experiments based on real-world data reveal that by applying a tariff selection heuristic on each relation, we can derive feasible solutions of the original model with only a minimal increase in cost. On the other hand, given the typically high number of parallel arcs between each pair of nodes in TTP instances—20 on average in our test sets—the aggregation drastically reduces the number of variables and constraints, resulting in a considerable boost in efficiency for branch and bound solution methods.

3.5.2 Preprocessing

Although tariff aggregation greatly helps to reduce problem sizes, the MIP models arising from TTP instances based on realistic data still suffer from numeric instability and weak lower bounds. We address these issues in the following with two preprocessing steps that can be applied to strengthen the aggregated formulation.

Strengthened container inequalities

As already discussed in Section 3.1.2, MIP formulations of capacitated network design problems can be considerably strengthened by adding valid inequalities. Among the valid inequalities used in literature are strong capacity and minimum cardinality inequalities. The natural extensions of these inequalities to TTP, however, did not turn out to be very effective for the instances in our computational study. Instead, we propose a method to bound the total extent of capacity used within individual containers. Before we describe these *strengthened container inequalities* in detail, we give some reasons for the failure of the known inequalities mentioned above.

Strong capacity inequalities. These inequalities state that $x_i(a) \leq b_i y(a)$ for all $i \in K$ and all $a \in A$, where $b_i := \sum_{v \in V: b_i(v) > 0} b_i(v)$ is the total supply of commodity $i \in K$.

While Chouman, Crainic, and Gendron [CCG09a] report on the positive impact of strong capacity inequalities on the integrality gap in their computational experiments, it is also easy to see that the strong capacity inequality for commodity i at arc a can only strengthen the original formulation if $\alpha_{ij} < \frac{\beta_j(a)}{b_i}$ for all $j \in P$. In typical TTP instances arising in practice, total demands within the network are much larger than individual transport capacities and the inequalities remained mostly ineffective.

Minimum cardinality inequalities. These inequalities require the number of containers installed on a cut induced by a set of nodes $S \subset V$ to be at least as large as the minimum number of containers required to transport the excessive demand $(\sum_{v \in S} b_i(v))_{i \in K}^+$ within S across the cut. As already observed by Chouman et al. [CCG09a], these inequalities are weak if the magnitudes of the capacities vary widely, as it is typically the case for logistics tariffs that are modeled within TTP instances. Their suggested improvements cannot be applied in our case as their model contains only binary design variables whereas ours are integer. In the following, however, we show how to strengthen our capacity inequalities using similar ideas.

Strengthened container inequalities. Solutions to the LP relaxation of TTP provide weak lower bounds for the following reason: When considering a flow carrying transport relation, LP solutions tend to set the variable of the largest container to the minimal fraction needed to grant capacities for the flow on this transport relation. These fractions are unfortunately very small, which means that they do not reflect the cost that would be incurred in an integer solution. The idea is to restrict container capacities without affecting the cost of an optimal integer solution. This is possible, if for a given transport relation $R \in \mathcal{R}$ an upper bound $\Gamma(R)$ on the flow $x(R)$ in any optimal solution is known. Useful upper bounds can be derived for transport relations incident to node sets $S \subset V$ with either $\delta^+(S) = \emptyset$ or $\delta^-(S) = \emptyset$. Given an upper bound $\Gamma(R)$, we can replace for every $a \in A(R)$ and every $j \in P$ the capacity $\beta_j(a)$ by $\beta_j(a) - s_j$, where s_j is the result of solving

$$\begin{aligned} \min \quad & s_j \\ \text{s.t.} \quad & \sum_{i \in K} \alpha_{ij'} x_i(a) + s_{j'} = \beta_{j'}(a) \quad \forall j' \in P \\ & 0 \leq x_i(a) \leq \Gamma_i(R) \quad \forall i \in K \\ & s_{j'} \geq 0 \quad \forall j' \in P. \end{aligned}$$

In a preprocessing routine we solve these linear programs for each property j of each fixed charge container e on each transport relation R for which reasonable upper bounds $\Gamma(R)$ can be computed.

Commodity scaling

During initial computational experiments, we could observe numerical difficulties while solving LP relaxations of large problem instances: The LP solving steps suffer from basis singularities and sometimes even numerical infeasibility. One reason for these difficulties lies in the diversity of properties for different commodities. The capacity inequalities involve many flow variables with property coefficients varying in magnitudes of 10^6 for our test instances. In order to attenuate the effects, we apply the following scaling steps.

For each commodity $i \in K$ we determine a scaling factor $p_i > 0$ and obtain scaled values $\tilde{b}_i(v)$ and $\tilde{\alpha}_{ij}$, defined by $\tilde{b}_i(v) := b_i(v)/p_i$ and $\tilde{\alpha}_{ij} := p_i \alpha_{ij}$ for each $j \in P$. The scaled problem instance is equivalent to the non-scaled one in the sense that feasible flow values $\tilde{x}_i(a)$ obtained for the scaled problem can be scaled back to obtain feasible flow values $x_i(a) = p_i \tilde{x}_i(a)$ for the original problem. We chose the scaling factors p_i for each commodity in such a way that among the resulting coefficients $\tilde{\alpha}_{ij}$ for $j \in P$ the smallest such coefficient has the magnitude 10^{-1} . The improved numeric stability of the constraint system significantly speeds up the LP solution process.

3.5.3 Initial solutions from aggregated LP relaxation (ALP)

In Section 3.4, we discussed the importance of properly chosen initial solutions for the local search procedure, and devised two ways to encourage consolidation of flow during the construction of the initial solution by shortest path type algorithms. Alternatively, we can obtain initial solutions from the LP relaxation of the aggregated MIP formulation by applying tariff selection heuristics to the multi-commodity flow in the pattern-expanded network induced by the aggregated LP solution.

Note that in this case, strengthened container inequalities as described above also boost consolidation in the solution process. In fact, the effect of the strengthened inequalities is strongest on arcs that are reachable from few sources or sinks only (such as direct source-sink connections). This implicitly encourages flow to take detours on paths along intermediate hubs, where less strong container inequalities permit lower costs in the LP relaxation. Since inappropriately consolidated flow can be efficiently disaggregated by the local search algorithm, initial solutions constructed from the LP relaxation lead to high quality final solutions as we shall see in Section 3.6.

3.5.4 Pattern optimization subproblem

In the tariff selection subproblem considered in Section 3.3, we fixed the amount of flow passing a given transport relation and optimized the tariff selection with respect to this given flow value. This idea can be extended by considering all transport relations that connect a given pair of nodes in different slots of the pattern-expanded network. More formally, for some node $v \in B$ in the base network and a cycle length F , let v_1, \dots, v_F be the copies of v created in the pattern expansion step, with $v_i \in V(B_i)$ for $i \in \{1, \dots, F\}$. We consider the *pattern optimization subproblem* induced by a fixed pair of nodes $s, t \in B$. To this end, we define

$$\mathcal{V}(s, t) := \bigcup_{i=1}^F \{s_i, t_i\} \quad \text{and} \quad \mathcal{R}(s, t) := \{R \in \mathcal{R} : \text{tail}(R), \text{head}(R) \in \mathcal{V}(s, t)\}.$$

Given a feasible solution to the whole TTP instance with flow values $\bar{x}(R)$ for $R \in \mathcal{R}$, we consider a locally restricted instance of TTP, fixing the flow values on all transport relations in $\mathcal{R} \setminus \mathcal{R}(s, t)$ and optimizing the flow $(x(R))_{R \in \mathcal{R}(s, t)}$ in the subnetwork of the

pattern-expanded network induced by the copies of s and t , i.e.,

$$\begin{aligned}
\min \quad & \sum_{R \in \mathcal{R}(s,t)} \sum_{t \in T(R)} C_t(\tilde{x}(t)) \\
\text{s.t.} \quad & \sum_{R \in \delta_{\mathcal{R}(s,t)}^+(v)} x_i(R) - \sum_{R \in \delta_{\mathcal{R}(s,t)}^-(v)} x_i(R) = \bar{b}_i(v) \quad \forall v \in \mathcal{V}(s,t), \forall i \in K \\
& \sum_{t \in T(R)} \tilde{x}_i(t) = x_i(R) \quad \forall R \in \mathcal{R}(s,t), \forall i \in K \\
& \tilde{x}(t) \geq 0 \quad \forall t \in T(R), \forall R \in \mathcal{R}(s,t)
\end{aligned}$$

where $\bar{b}(v) := \sum_{R \in \delta_{\mathcal{R}(s,t)}^+(v)} \bar{x}(R) - \sum_{R \in \delta_{\mathcal{R}(s,t)}^-(v)} \bar{x}(R)$.

Using the tariff gadgets from Section 3.2.4, this restricted instance of TTP can be formulated as a mixed integer program. It contains only a small fraction of the decision variables present in the whole instance. In fact, these restricted instances can be solved to near-optimality very quickly using a standard MIP solver. We thus iteratively optimize these subproblems arising for all pairs of adjacent nodes with flow carrying transport relations in between them.

Note that in contrast to the tariff selection subproblem, solving the pattern optimization subproblem for one pair of nodes may affect the subproblem of other, non-disjoint pairs of nodes, as holdover arcs of a common node appear in each of the problems as variables. Consequently, the order in which the node pairs are considered plays an important role. We sort the node pairs non-increasingly with respect to the total flow in the subnetwork affected by the pattern optimization for each pair, scalarized by a weighted sum of the property extents, i.e., $\sum_{j \in P} w_j \alpha_j (\sum_{R \in \mathcal{R}(s,t)} \bar{x}(R))$, using the same weights $w \in \mathbb{Q}_+^P$ as provided for local search and SPTS heuristic. This reflects the potential of the corresponding node pair for cost savings and leads to an ‘‘important pairs first’’ order, which is also useful when the pattern optimization process cannot be carried out on all node pairs due to time constraints.

3.6 Computational study

We verify the TTP model and the algorithmic approaches presented in the preceding sections by conducting a computational study based on real-world data provided by our project partner 4flow AG.

3.6.1 Instance sets

The benchmark library consists of 145 instances aggregated from four recent and on-going customer projects of 4flow AG in three different industries, two from the automotive industry, one from the chemical industry, and one from home appliances retail. We denote the corresponding sets by auto1, auto2, chemical, and retail, respectively. All base networks correspond to European supply chains in which goods are transported according to full truck load (FTL) or less than truck load (LTL) tariffs. The networks share a layered graph structure. More specifically, the nodes of the base network are partitioned into an ordered set of layers, with the lowest layer containing all sources, and the highest layer containing all sinks. In addition, there is a fixed number (varying from

instance set	nodes in base network			$ K $	transport relations/arcs		
	sources	sinks	hubs		base	pattern	tariff
auto1 (36)	35	6	7	162	335	2296	76653
auto2 (18)	34	3	4	117	186	1364	29264
chemical (50)	7	244	19	101	6601	41222	239238
retail (41)	4	177	26	307	5665	35229	511064

Table 3.2: Average sizes of the instances per set: the number of sources, sinks, and hubs in the base network, the number of commodities, and the number of arcs in the base network, the pattern-expanded network, and the tariff-expanded network. The number of instances in each set is given in parentheses after the name of the set.

one to three) of intermediate hub layers. There is a transport relation between every pair of nodes from distinct layers, directed towards the higher layer. However, transport relations between nodes of the same layer are not present. Pattern expansion has been conducted with a cycle length of six slots—one slot corresponds to two months of a year. All tariffs are of piecewise constant type, depending on the same two properties, weight and volume, in every instance.

While the automotive instances represent production networks with a high number of sources and a low number of sinks, the instances from chemical industry and retail are based on distribution networks with a high number of sinks but only few sources. Another notable difference lies in the number of different tariffs available on each transport relation, which is considerably lower for instances from the chemical set as compared to those of the others with an average of 6 vs. 20 tariffs per transport relation, respectively. Table 3.2 shows the average values of key parameters of the instances within each set.

A complete overview of the characteristics of all instances is given in Tables 3.5 to 3.7 at the end of this chapter.

3.6.2 Algorithms and implementation details

We implemented and tested different variants of the algorithms presented in Sections 3.4 and 3.5 in order to determine good parameter settings and combinations. In long term planning, computation time plays a minor role and the fine-tuned aggregated MIP formulation combined with the path-based local search and pattern optimization with generous time limits can be used. When evaluating multiple scenarios in a row—e.g., in the context of strategic planning, as indicated in Section 3.2.5—however, computation time becomes more significant. Our industrial partner suggested a time limit of 30 min for this application. We thus also tested approaches designed for time-efficiency without sacrificing too much solution quality. Overall, the following algorithms were tested on all 145 instances of the benchmark library.

MIP-plain: branch and bound using the plain MIP formulation [TTP_{CND}] for comparison purposes (Section 3.2)

SPLC: local search using initial solutions from the shortest path heuristic with linearized cost (Section 3.4.1)

SPLC-F: same as SPLC, but with forbidden direct connections (Section 3.4.1)

SPTS-L: local search using initial solutions from the shortest path heuristic with tariff selection and partial linearization (Section 3.4.2)

AMIP-H: branch and bound using the aggregated MIP formulation with integrated local search (Section 3.5.1)

ALP: local search using initial solutions derived from the LP relaxation of the aggregated formulation (Section 3.5.3)

All algorithms have been implemented in C++ and compiled with gcc 4.5.0 on open-SUSE 11.3 Linux with kernel 2.6.32.19-0.2. Computations have been performed on cluster nodes with two DualCore-Opteron 2218 processors (2.6 GHz, 64 bit) and 16 GB of memory using CPLEX 12.1 for MIP and LP solving. Since the heuristic approaches have not been adapted to support concurrent computations, we limited the number of threads for the CPLEX solver to one to ensure comparability of the results.

In the following we first elaborate on the interplay of the branch and bound framework and the local search heuristic and then describe the exact settings for the variants of the local search procedure.

Branch and bound frameworks

Our tests involved different MIP formulations, which we implemented in CPLEX. For a direct comparison with our algorithms, we tested a plain MIP model (MIP-plain) based on the capacitated network design formulation [TTP_{CND}] presented in Section 3.2.4. We also tested the aggregated MIP formulation from Section 3.5.1 combined with the preprocessing methods described in Section 3.5.2 and callbacks to our combinatorial heuristics. The resulting algorithm is denoted by AMIP-H and details of the implementation are given below. In order to obtain reasonably tight lower bounds, we also ran the aggregated MIP formulation without heuristic callbacks; this setting is referred to as AMIP-B below. We invoked a time limit of 2 h for the branch and bound process, and an extra time of 1 h for applying local search and pattern optimization each.

When solving the aggregated and preprocessed MIP formulation with the branch and bound framework, we improve feasible integer solutions and promising fractional LP solutions obtained during the search of the branch and bound tree using our heuristics. Note that these solutions induce a flow on the transport relations of the pattern-expanded network, which can be turned into a feasible TTP solution by solving the tariff selection subproblem on each transport relation. We further improve the solution by applying the local search heuristic and pattern optimization with a time limit of 300 s. As this procedure incurs a significant computational effort, we require at least 1500 branch and bound nodes to be processed between two successive calls of the heuristics. Furthermore, we use the cost estimator presented in Section 3.3.2 in order to evaluate the potential of a given LP solution to improve on the currently best solution: Only if the estimated total cost is within 8 % to the best known solution, we compute the corresponding TTP solution. We also apply the procedure to all integral solutions found by the MIP solver.

Local search procedure

We tested the local search algorithm described in Section 3.4.3 using initial solutions constructed by the heuristics mentioned above. For intermediate tariff selection, we employed both the estimator and the two-phase greedy method described Section 3.3. In addition, pattern-optimization is performed on the final solution using the non-aggregated MIP formulation, as described in Section 3.5.4.

	auto1	auto2	chemical	retail	all
solver	(31/36)	(18/18)	(48/50)	(30/41)	(127/145)
ALP	-17.81	-6.87	-21.61	-10.44	-15.96
AMIP-B	17.48	0.61	12.38	4.84	10.18

Table 3.3: Average improvement of the lower bound over that achieved by MIP-plain in percent. The numbers in parentheses indicate the share of instances for which MIP-plain was able to obtain a lower bound in the corresponding set.

Computation time of the starting heuristics ALP, SPLC and SPLC-F turned out to be almost negligible, and we invoked a total solution time of 30 min (including pattern optimization) in this case. Unfortunately, the more sophisticated SPTS-L solver turned out to cause considerably more computational effort. Here we invoked the same time limits as for the branch and bound approaches.

Recall that for fine-tuning the path decomposition of the local search procedure and the SPTS heuristic, an additional parameter is specified: a weight function on the properties of the model that reflects the importance of properties. For our test instances, the (physical) weight of shipments occurs to be the dominant property. We thus choose the weight function to be an indicator function on weight.

3.6.3 Results

We now elaborate on the results of our computational experiments, starting with the effect of aggregation on the lower bounds. We then analyze solution quality and the impact of the choice of initial solutions, local search, and pattern optimization. We close by comparing our approach to a reference solution on an additional instance.

Influence of aggregation and preprocessing on lower bounds. Table 3.3 shows the average improvement on lower bounds achieved by the aggregation and our preprocessing techniques over that computed by the plain MIP formulation. In fact, we observed that especially for large instances, MIP-plain suffers from numerical instabilities and degeneracy that lead to solving times of thousands of seconds already for the LP relaxation at the root node. In some cases, the initial cut generation rounds for the root node do not terminate within given time limits. In turn, the efficiency of initial cuts greatly benefits from our preprocessing techniques—fewer cuts achieve a much better lower bound when using AMIP-B. Not surprisingly, the lower bounds derived by the strengthened aggregated LP (ALP) are of low quality, with an decrease of more than 15% on average towards the values obtained by MIP-plain. In a set-by-set comparison, the aggregated AMIP-B framework achieves an average improvement over MIP-plain of more than 10%, and of up to 17% on average on set auto1, while MIP-plain is only competitive on the comparatively small instances of the auto2 set. Apparently, the loss in exactness caused by the aggregation is more than compensated by the boost in efficiency of the branch and bound procedure achieved by the smaller size of the formulation and its increased numerical stability.

Solution quality. Figure 3.2 and Table 3.4 show the gaps of the solutions obtained by the respective algorithms to the lower bound computed by AMIP-B. Throughout the automotive and retail instance sets, the average gap to the lower bound is within

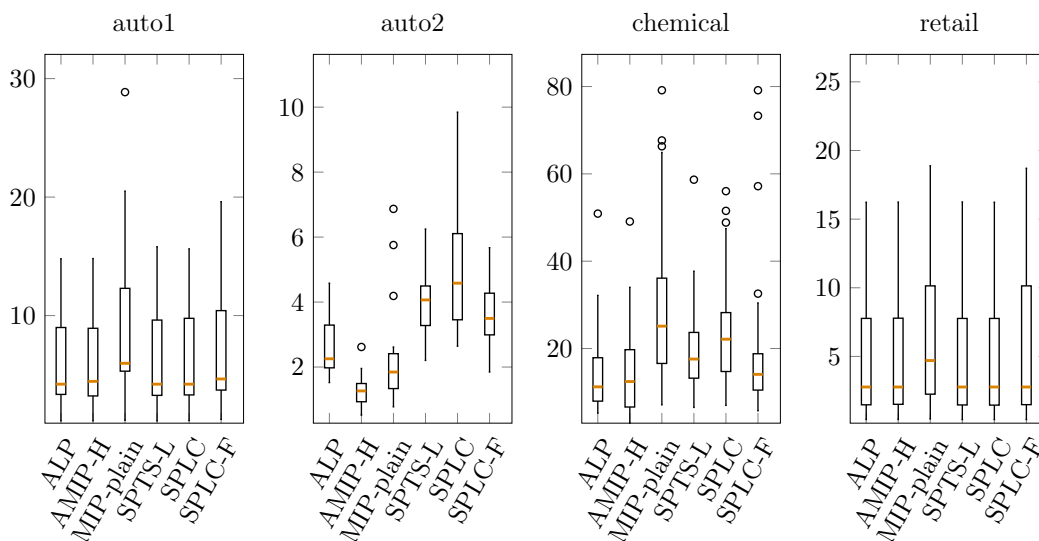


Figure 3.2: Gaps of solutions to best found lower bounds in percent. The diagrams show the distribution of the gaps for the respective algorithms within each test set. The mark inside each box represents the median of the gaps, the box boundaries represent lower and upper quartiles and the whisker ends show the minimum and maximum obtained gap, respectively, apart from possible outliers that are marked with a circle (four extreme outliers of MIP-plain and three of SPLC-F have been removed from the retail set for better readability). The diagrams have been plotted following the suggestions in [FHI89].

solver	auto1 (36)	auto2 (18)	chemical (50)	retail (41)	all (145)
MIP-plain	9.09 (1)	2.35 (3)	29.18 (0)	13.33 (1)	16.38 (5)
ALP	6.22 (24)	2.63 (0)	13.92 (17)	4.74 (40)	8.01 (81)
AMIP-H	6.12 (26)	1.26 (16)	14.61 (24)	4.75 (38)	8.06 (104)
SPLC	6.51 (17)	5.07 (0)	23.54 (0)	4.75 (37)	11.71 (54)
SPLC-F	6.90 (11)	3.65 (0)	18.08 (9)	10.70 (27)	11.43 (47)
SPTS-L	6.57 (19)	4.15 (0)	19.44 (1)	4.74 (39)	10.19 (59)

Table 3.4: Average gaps to best known lower bound in percent. The numbers in parentheses indicate the number of best solutions achieved by the solver.

a single-digit percentage. The local search with LP starting solution and the AMIP-H framework achieve the lowest costs, while the performance of approaches with shortest path-based initial solutions is weaker and varies depending on the instance set. We infer that the more holistic LP approach captures the multi-commodity flow nature of our problem better than the iterative path approaches.

AMIP-H attains near-optimality on auto2, outperforming ALP on this set. Apparently, the small instance sizes in this set benefit the branch and bound process. The gaps are considerably weaker on the instances of the chemical set. The instances of this set are much bigger with respect to the number of arcs and sinks in the base network than those from the other sets, which presumably also affects the MIP framework’s ability to produce tight lower bounds.

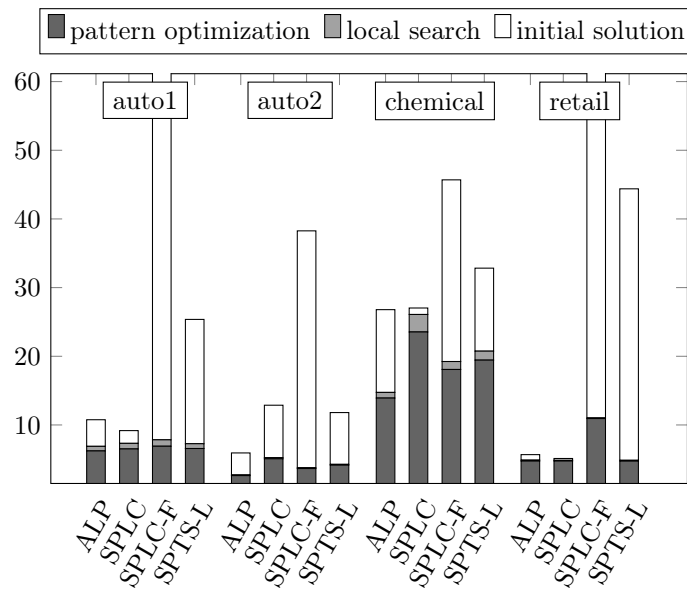


Figure 3.3: Performance of fast solvers with time limit 1800 s. Average gaps to best known lower bounds in percent are shown for the initial solution (top of the white area), after local search (top of the light area), and after pattern optimization (top of the dark area).

Performance of local search and impact of initial solutions. The results in Table 3.4 and Figure 3.2 show that the choice of the initial solution clearly affects the performance of the local search procedure. In fact, on many instances, the initially expensive flow patterns of the consolidation enforcing start heuristics lead to better final solutions than those obtained from solutions with low consolidation provided by SPLC for comparison; see Figure 3.3. However, the effectiveness of the combinatorial starting heuristics strongly depends on the structure and size of the instance. In contrast, ALP consistently shows best results among the fast solvers, on par with the AMIP-H framework—which takes considerably more computational effort.

Impact of pattern optimization. Figure 3.3 reveals that the effect of pattern optimization is almost negligible on the sets auto2 and retail, and still relatively weak on the set auto1. However, for the instances of the chemical set, the effect is significantly stronger. This better performance can be explained by the less granular tariff structure in this instance set, resulting in smaller subproblems while at the same time increasing the importance of temporal consolidation.

Purely combinatorial heuristics. In order to provide solutions independent of third party software and licenses, we also evaluated purely combinatorial variants of the local search heuristic with path-based initial solutions: After replacing MIP-based tariff selection algorithms with greedy heuristic and omitting pattern optimization, the approaches still produced good solutions with a mild increase in cost of at most 3% on average.

Comparison with reference solution. Due to confidentiality restrictions it was not possible to obtain reference solutions or current network costs for the instances presented

above. Instead, a direct comparison with an instance of a European cross-docking network from a recent customer project has been conducted in cooperation with 4flow AG. The base network consists of 228 sources, 545 sinks, 5 hubs, 5857 arcs, resulting in a tariff-expanded network with 209304 arcs. It is fully connected in contrast to the layered structure observed so far. On this instance, the AMIP-H framework obtained a solution with 1.2% gap to optimality. We compared this against a solution obtained with a standard software for supply chain design at project start operating on a conventional model and optimizing flow routes and delivery frequencies in two separate phases. Our solution constitutes a 14% improvement, which, if applied on an annual basis, results in savings of up to 1.6 million Euro.

3.7 Conclusion

The tactical transportation planning model presented in this chapter integrates the important aspects of tactical logistics network optimization in practice: realistic transportation tariffs, cyclic delivery patterns, and inventory costs. Several algorithmic techniques have been devised to address the challenges associated with the specific instance structure induced by our model. These methods have proven to be successful in tests on a broad set of instances derived from real-world logistics networks.

The performance of our algorithms relies to a great extent on the successful isolation of the tariff selection subproblem. We devise a variety of exact and heuristic methods to efficiently solve this problem, satisfying the different needs for speed and exactness of the solution procedure resulting from the various contexts of solving the subproblem. Using these subroutines, we propose a local search procedure that simultaneously re-routes flow of multiple commodities. Equipping the local search with different types of initial solutions, such as multi-commodity flow patterns derived from a strengthened LP relaxation or from combinatorial path-based approaches, yields solutions that are within a single digit percent of the optimum on average. Our algorithms can be used both in connection with standard MIP solvers, or as purely combinatorial algorithms, yielding competitive solutions without usage of third-party software.

Open problems and future research

Currently, our algorithmic toolkit is being integrated into a supply chain software package developed by 4flow AG. The collaboration with 4flow AG also sparked a second project investigating the possibility of incorporating robustness aspects into transportation models. In addition, our model opens several directions for future research.

Strategic and operational planning. While the model presented here aims at transportation planning on the tactical level, we have already discussed possible generalizations to strategic and operational settings in Section 3.2.5. As remarked there, incorporating aspects such as facility costs in the strategic context, or time-windows in the operational context poses new algorithmic challenges for solving the model. Augmenting the model by such features and devising suitable solution methods is an interesting subject for future work on the topic of transportation planning.

Efficient algorithms for multi-commodity flows with multiple capacities. Benders decomposition is one of the most successful techniques for solving capacitated network design problems. Initial experiments with this approach in the context of our model, however, have revealed the LP subproblem to be a severe computational bottleneck. Devising an efficient combinatorial algorithm for solving this subproblem, which corresponds to a multi-commodity flow problem with multiple capacities, is an interesting task for future research. A natural candidate for such a method would be an extension of the Garg-Könemann algorithm [GK07] to multiple capacities.

Approximation algorithms for the tariff selection problem. The tariff selection problem introduced in Section 3.3 constitutes a central subproblem of the TTP model, with great importance to our algorithmic methods. Generalizing classic covering problems by the aspect of assignment, this subproblem is also interesting on its own right. In [KMR12], we provide initial insights on the approximability of the tariff selection problem, including hardness of approximation with a factor better than logarithmic. The reduction, however, requires the number of properties to be unbounded. It remains an open question whether there exists a constant factor approximation algorithm for the special case that the number of properties is constant. This is particularly interesting as the number of properties is typically very small in practice.

instance	nodes in base network			$ K $	transport relations/arcs		
	sources	sinks	facilities		base	pattern	tariff
auto1_1	108	10	127	917	1109	7416	253614
auto1_1QA	60	8	77	349	701	4668	160290
auto1_1QB	13	6	28	153	206	1404	47136
auto1_1QC	47	7	63	196	559	3732	127830
auto1_1QD	22	9	40	218	267	1842	61116
auto1_1QE	8	7	24	214	168	1152	38448
auto1_1QF	18	8	35	136	224	1554	51282
auto1_1SA	30	3	37	282	146	1098	33510
auto1_1SB	25	2	31	71	126	942	28914
auto1_1SC	59	4	67	402	294	2166	67434
auto1_1SD	39	3	51	244	417	2808	95382
auto1_1SE	34	2	40	331	171	1266	39228
auto1_3	92	10	111	348	970	6486	221826
auto1_3QA	47	8	64	129	571	3810	130572
auto1_3QB	12	6	27	50	197	1344	45078
auto1_3QC	35	7	51	79	438	2934	100170
auto1_3QD	20	9	38	93	262	1800	59964
auto1_3QE	8	7	24	73	168	1152	38448
auto1_3QF	17	8	34	53	220	1524	50364
auto1_3SA	29	3	36	107	143	1074	32820
auto1_3SB	18	2	24	32	95	714	21804
auto1_3SC	48	4	56	148	242	1788	55512
auto1_3SD	35	3	47	102	378	2550	86466
auto1_3SE	30	2	36	116	151	1122	34644
auto1_4	92	10	111	257	970	6486	221826
auto1_4QA	47	8	64	105	571	3810	130572
auto1_4QB	12	6	27	35	197	1344	45078
auto1_4QC	35	7	51	70	438	2934	100170
auto1_4QD	20	9	38	63	262	1800	59964
auto1_4QE	8	7	24	47	168	1152	38448
auto1_4QF	17	8	34	42	220	1524	50364
auto1_4SA	29	3	36	71	143	1074	32820
auto1_4SB	18	2	24	27	95	714	21804
auto1_4SC	48	4	56	113	242	1788	55512
auto1_4SD	35	3	47	79	378	2550	86466
auto1_4SE	30	2	36	86	151	1122	34644
auto2_1	64	4	72	383	355	2562	13872
auto2_1A	17	4	24	64	88	672	27054
auto2_1B	29	4	37	188	172	1254	14346
auto2_1C	18	4	25	131	91	696	55812
auto2_1S0511	36	2	42	131	190	1392	29892
auto2_1S0710	42	2	48	256	220	1608	34608
auto2_2	64	4	72	161	355	2562	13872
auto2_2A	17	4	24	27	88	672	27054
auto2_2B	29	4	37	88	172	1254	14346
auto2_2C	18	4	25	46	91	696	55812
auto2_2S0511	36	2	42	64	190	1392	29892
auto2_2S0710	42	2	48	100	220	1608	34608
auto2_3	64	4	72	153	355	2562	13872
auto2_3A	17	4	24	26	88	672	27054
auto2_3B	29	4	37	83	172	1254	14346
auto2_3C	18	4	25	44	91	696	55812
auto2_3S0511	36	2	42	61	190	1392	29892
auto2_3S0710	42	2	48	95	220	1608	34608

Table 3.5: Characteristics of the auto1 and auto2 instances.

instance	nodes in base network				transport relations/arcs		
	sources	sinks	facilities	$ K $	base	pattern	tariff
chemieTN1	2	844	890	90	38140	234180	1378380
chemieTN2	2	34	48	25	469	3102	17172
chemieTN3	2	59	70	41	613	4098	22488
chemieTN4	2	205	229	64	4776	30030	173310
chemieTN5	2	62	71	36	516	3522	19002
chemieTN6	2	99	109	47	914	6138	33558
chemieTN7	2	122	136	42	1616	10512	58992
chemieTN8	2	58	67	26	483	3300	17790
chemieTN9	2	195	204	56	1596	10800	58680
chemieTN10	6	108	128	88	1741	11214	63444
chemieTN11	6	152	173	120	2573	16476	93666
chemieTN12	6	648	673	243	13318	83946	483486
chemieTN13	6	160	178	79	2204	14292	80412
chemieTN14	6	201	220	105	2961	19086	107916
chemieTN15	6	326	353	107	7400	46518	268518
chemieTN16	6	211	229	84	2860	18534	104334
chemieTN17	6	454	471	154	5694	36990	207810
chemieTN18	1	407	452	29	18359	112866	663636
chemieTN19	3	1247	1294	51	56463	346542	2040432
chemieTN20	3	70	78	23	445	3138	16488
chemieTN21	3	75	92	33	1184	7656	43176
chemieTN22	3	407	434	45	10315	64494	373944
chemieTN23	3	72	84	24	757	5046	27756
chemieTN24	3	124	133	37	908	6246	33486
chemieTN25	3	185	197	34	1903	12600	69690
chemieTN26	3	95	107	31	996	6618	36498
chemieTN27	3	208	216	41	1301	9102	48132
chemieTN28	12	408	453	248	14526	89874	525654
chemieTN29	12	174	209	145	4553	28572	165162
chemieTN30	12	234	274	213	7279	45318	263688
chemieTN31	11	90	124	78	2456	15480	89160
chemieTN32	12	84	115	108	1966	12486	71466
chemieTN33	12	113	147	161	2949	18576	107046
chemieTN34	12	121	151	122	2586	16422	94002
chemieTN35	12	238	272	151	5897	37014	213924
chemieTN36	11	63	96	71	1732	10968	62928
chemieTN37	12	117	149	115	2787	17616	101226
chemieTN38	12	58	90	59	1486	9456	54036
chemieTN39	11	33	64	37	933	5982	33972
chemieTN40	12	84	114	104	1882	11976	68436
chemieTN41	12	997	1041	373	33993	210204	1229994
chemieTN42	12	672	705	269	15551	97536	564066
chemieTN43	12	39	75	36	1290	8190	46890
chemieTN44	12	338	369	205	7209	45468	261738
chemieTN45	12	294	328	201	7224	45312	262032
chemieTN46	11	44	71	45	947	6108	34518
chemieTN47	12	514	548	200	12388	77616	449256
chemieTN48	12	322	364	156	10489	65118	379788
chemieTN49	12	292	330	151	8337	52002	302112
chemieTN50	11	30	66	27	1061	6762	38592

Table 3.6: Characteristics of the chemical instances.

instance	nodes in base network				transport relations/arcs		
	sources	sinks	facilities	$ K $	base	pattern	tariff
handel1	7	702	748	1468	27008	166536	2435208
handel2	7	702	748	316	27008	166536	2435208
handel3	7	702	748	100	27008	166536	2435208
handelTN1	7	60	106	257	2691	16782	242826
handelTN2	7	62	108	143	2485	15558	224298
handelTN3	4	74	117	104	2718	17010	245322
handelTN4	7	75	121	223	3293	20484	297096
handelTN5	7	72	118	182	3144	19572	283668
handelTN6	6	76	121	177	3227	20088	291156
handelTN7	7	65	111	175	2815	17556	254016
handelTN8	7	78	124	234	3162	19716	285324
handelTN9	4	72	115	161	2658	16638	239910
handelTN10	7	68	114	266	2951	18390	266274
handelTN11	2	121	135	142	1605	10440	145260
handelTN12	2	186	198	227	2082	13680	188568
handelTN13	3	269	289	494	5032	31926	454614
handelTN14	2	182	196	210	2401	15582	217266
handelTN15	3	174	193	372	3034	19362	274218
handelTN16	2	212	222	219	1943	12990	176202
handelTN17	2	57	67	78	530	3582	48102
handelTN18	2	62	72	77	581	3918	52722
handelTN19	2	74	87	90	912	5994	82602
handelTN20	2	62	74	118	708	4692	64164
handelTN21	2	72	84	108	820	5424	74304
handelTN22	2	52	64	84	594	3948	53844
handelTN23	3	65	81	121	956	6222	86526
handelTN24	3	78	94	187	1221	7890	110454
handelTN25	3	72	92	159	1381	8838	124842
handelTN26	3	54	70	164	803	5238	72690
handelTN27	3	60	79	176	1085	6984	98124
handelTN28	3	46	65	100	833	5388	75360
handelTN29	3	68	87	180	1212	7794	109602
handelTN30	3	57	81	115	1324	8430	119646
handelTN31	3	72	96	88	1652	10488	149256
handelTN32	3	65	89	116	1496	9510	135174
handelTN33	7	359	405	669	13952	86142	1258110
handelTN34	7	343	389	911	13323	82272	1201404
handelTN35	7	344	390	900	13948	86028	1257660
handelTN36	7	494	540	1214	19720	121560	1778040
handelTN37	4	413	456	724	15966	98532	1439676
handelTN38	3	340	382	744	12971	80118	1169682

Table 3.7: Characteristics of the retail instances.

Chapter 4

Approximating combined location and network design problems

In this chapter, we investigate optimization problems that combine facility location with vehicle routing or network design problems. We discuss a general framework for combining approximation techniques based on different lower bounds to obtain algorithms for such integrated location and network design problems. We use this framework to derive approximation algorithms for *capacitated location routing*, an important problem in transport logistics, and *facility location with capacitated and length-bounded trees*, a problem motivated by the design of optical access networks in telecommunication. We also present computational results indicating that the performance of our algorithm in practice clearly exceeds the theoretical guarantee.

Publication remark: The results presented in Section 4.2 are joint work with Tobias Harks and Felix G. König [HKM13]. The results presented in Section 4.3 are joint work with Andreas Bley and Benjamin Müller [MBM13].

Location analysis plays a crucial role in the design of networks, e.g., in transport logistics or telecommunication. The cost for opening and operating depots, central offices, and similar facilities constitutes a large share of the overall solution cost. A fundamental combinatorial optimization problem addressing location decisions is the *uncapacitated facility location* problem. It asks for a subset of facilities from a given set to be opened in order to serve a set of clients with minimum cost. In this very basic setting, each client is directly connected to the nearest open facility. In many practical applications, however, such dedicated connections for each client are rarely encountered, as it is often more economical to serve multiple clients by a shared infrastructure. Logistics carriers, e.g., usually serve several clients on a single tour. Likewise, fiber cables in optical access networks are split so they can connect multiple neighboring clients to a central office in a tree-based network topology.

In this chapter, we consider optimization problems that combine facility location with various ways of connecting clients to facilities. We discuss a general framework for obtaining approximation algorithms for such problems by combining different lower bounds and approximation techniques. We apply this framework to two problem classes from logistics planning and telecommunication network design and derive approximation algorithms for several variants of these problems. Besides a thorough theoretical analysis of the approximation factors, we also evaluate the resulting practical performance of our approach on a set of large-scale benchmark instances.

	CLR	MCVR	Section
standard	4.38	4	4.2.2
prize-collecting	6	4	4.2.3
group	$4.38 L$	$4 L$	4.2.4
cross-docking	3.5	3	4.2.5

Table 4.1: Approximation ratios for different variants of capacitated location routing (CLR) and multi-depot capacitated vehicle routing (MCVR). The value L denotes the size of the largest group in group CLR.

Chapter outline

In Section 4.1, we introduce the basic notions of facility location and discuss related literature, including approximation results for the fundamental *uncapacitated facility location problem* (UFL) and the work of Ravi and Sinha [RS06] on the *capacitated-cable facility location problem*, which provides the basis for our algorithms. We also discuss the necessity for addressing a combined optimization problem as compared to solving two separate problems for location and network design. Finally, we introduce a procedure for extracting subtrees with clustered demand from a given input tree. This procedure plays a central role in the algorithms introduced in the following sections.

In Section 4.2, we study the *capacitated location routing* problem (CLR), a combination of UFL and vehicle routing. In this setting, clients are served by tours originating at open facilities using vehicles with uniform capacity. In Section 4.2.1, we give an overview of work related to vehicle routing and location routing, with a focus on approximation results. In Section 4.2.2, we introduce two combinatorial lower bounds for the problem, based on spanning trees and facility location. Combining these lower bounds using the clustering procedure introduced in the preceding section, we obtain the first constant factor approximation for CLR. As a by-product, the algorithm also improves on the previously best-known approximation factor for *multi-depot capacitated vehicle routing*, the special case of CLR where location decisions are already taken. We extend our approximation results to several variants of location routing. In Section 4.2.3, we consider *prize-collecting* CLR, where individual clients can be left unserved by paying a client-dependent penalty. In Section 4.2.4, we investigate *group* CLR, where the set of clients is partitioned into groups, and only one client from each group has to be served. The results in this section are also interesting because they are derived from an LP relaxation that combines the two combinatorial lower bounds mentioned above. In Section 4.2.5, we study a version of CLR where *cross-docking* is allowed, i.e., shipments can be loaded from one vehicle to another at intermediate vertices. While we derive constant factor approximations for the prize-collecting and cross-docking versions, the approximation guarantee for the group version depends on the cardinality of the largest group. In fact, we show that this version of the problem is hard to approximate better than by a factor logarithmic in the number of groups. A concise overview of the respective approximation results is given in Table 4.1. In Section 4.2.6, we present a computational study of our algorithm on benchmark instances and large-scale randomly generated instances. It reveals that the quality of the computed solutions is much closer to optimality than guaranteed by the proven approximation factor.

	length	cost	Section
general	$\mathcal{O}(\log \mathcal{C})$	$\mathcal{O}(\log \mathcal{C})$	4.3.3
general ^{qp}	$3 + \varepsilon$	$\mathcal{O}(\log^2 \mathcal{C})$	4.3.3
length-proportional cost	3α	$(1 + \frac{2}{\alpha-1})\beta_{\text{ST}} + \gamma_\alpha + 1$	4.3.4
hop constraint	$1 + \varepsilon$	$\mathcal{O}(\log \mathcal{C})$	4.3.5

Table 4.2: Approximation ratios for length bound and optimal solution cost for different cases of uncapacitated facility location with capacitated and length-bounded trees (UFL-CLT). The set \mathcal{C} is the set of clients. The superscript *qp* denotes quasi-polynomial running time. The value β_{ST} is the approximation factor of a Steiner tree algorithm. The value $\alpha > 1$ is a parameter of the algorithm. The value γ_α is defined as the unique solution to the equation $(1 + \frac{2}{\alpha-1})(2 + 4e^{-\gamma_\alpha}) = \gamma_\alpha$.

In Section 4.3, we turn our attention to a problem that occurs in the planning of optical access networks in telecommunication, the *uncapacitated facility location problem with capacitated and length-bounded tree connections* (UFL-CLT). In this setting, clients are connected to open facilities via shared access trees that have to obey both a capacity restriction on the total demand served by the tree, as well as a bound on the length of each client-facility-path. In the most general version of the problem, cost and length of an edge are two independent values. We study bicriteria (α, β) -approximation algorithms that relax the length bound by a factor of α and approximate the cost of the optimal solution by a factor of β . In Section 4.3.1, we discuss the related *shallow-light Steiner tree problem* (SLST), which asks for a minimum cost Steiner tree obeying a length bound. We give an overview of approximation results for SLST and related problems. We then use a well-known connection between SLST and directed Steiner trees in the so-called layered graph to obtain an additional pseudo-polynomial approximation algorithm for SLST that relaxes the length bound only by a factor of $(1 + \varepsilon)$. In Section 4.3.2, we show that UFL-CLT remains hard to approximate even in a very restricted special case and we give two lower bounds on the value of an optimal solution. In Section 4.3.3, we show how to adapt our approximation framework to cope with length bounds. By using different SLST-approximations as subroutines, we obtain two different approximation algorithms for the general version of our problem. The first runs in polynomial time and approximates both the length bound and the optimal cost by a logarithmic factor. Our second algorithm, which runs in quasi-polynomial time, approximates the length bound by a constant factor while giving a polylogarithmic guarantee for the cost. In the remaining sections, we investigate two important special cases of the problem, for which we can obtain considerably better results. In Section 4.3.4, we consider the case where lengths and costs are both proportional to a common metric. For this case, combining a greedy covering technique with so-called *light approximate shortest-path trees* yields an approximation algorithm that guarantees constant factors both for length bound and solution cost. By modifying an input parameter of the algorithm, the two factors can be adjusted to obtain different levels of trade-off between length and cost. In Section 4.3.5, we consider UFL-CLT with hop constraints, i.e., the case where the length of a path corresponds to the number of its edges. For the case that the cost function is metric, we can achieve an arbitrarily good approximation ratio for the hop constraint together with logarithmic cost approximation. A concise overview of the respective approximation results for UFL-CLT is given in Table 4.2.

4.1 Introduction to combined facility location and network design

In this section, we introduce some basic notation and discuss results from the literature related to facility location and combinations of facility location and network design. We also motivate the use of such combined problems by showing that the additional cost incurred by addressing location and network design separately can be arbitrarily high. Finally, we describe a procedure for partitioning a tree into several subtrees, which we will use extensively in the remainder of this chapter.

4.1.1 Basic notation

The notation presented in this section will be used throughout this chapter. In particular, we will frequently use the standard notation

$$c(S) := \sum_{e \in S} c(e)$$

for a cost vector $c \in \mathbb{Q}^E$ and a subset $S \subseteq E$ of edges. Also recall our definition of the bidirected graph $B(G)$ and the function ψ mapping edges to their end points introduced in Section 1.1.3. The latter will be used regularly in Section 4.3, where parallel edges with different costs and lengths might occur.

Distances. The distances of clients and facilities in facility location problems are usually specified by a function $c : \mathcal{C} \times \mathcal{F} \rightarrow \mathbb{Q}_+$, where \mathcal{C} is the set of clients and \mathcal{F} is the set of facilities. We define

$$c(v, F) := \min_{w \in F} c(v, w)$$

for all $v \in \mathcal{C}$ and $F \subseteq \mathcal{F}$ to be the distance from v to the closest facility in F .

Trees and subtrees, depth and diameter. Let T be a tree. For $v, w \in V(T)$ we let $T[v, w]$ denote the unique path from v to w in T . Furthermore, for a given *root* vertex $r \in V(T)$, we denote the *subtree* of T rooted at $v \in V(T)$ by $T_r[v]$, i.e., $T_r[v] \subseteq T$ is the subtree that spans all vertices $w \in V(T)$ with $T[v, w] \cap T[v, r] = \emptyset$. We omit the subindex r if the root of the tree is clear from the context.

Let $\ell : E(T) \rightarrow \mathbb{Q}_+$ be a length function on the edges. The ℓ -*depth* of T with respect to the root $r \in V(T)$ is

$$\text{depth}_\ell(T, r) := \max_{v \in V(T)} \ell(T[v, r]).$$

Similarly, the ℓ -*diameter* of T is

$$\text{diam}_\ell(T) := \max_{v, w \in V(T)} \ell(T[v, w]).$$

4.1.2 Related work

Facility location problems are a central topic in combinatorial optimization and approximation algorithms for these problems have been studied extensively. We discuss results regarding approximation for the classic uncapacitated facility location problem and the

capacitated-cable facility location problem, a basic combination of facility location and network design. However, we defer the discussion of results related specifically to either location routing or network design with length bounds to the corresponding Sections 4.2.1 and 4.3.1.

Uncapacitated facility location

Facility location problems ask for opening a subset of facilities from a given set of possible locations and connecting clients to the open facilities. One of the most basic variants of facility location is known as *uncapacitated facility location*.

Problem: Uncapacitated facility location (UFL)

Input: A set of clients \mathcal{C} , a set of facilities \mathcal{F} , opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^{\mathcal{C} \times \mathcal{F}}$, and demands $d \in \mathbb{Q}_+^{\mathcal{C}}$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$ such that the total cost $\sum_{w \in F} f(w) + \sum_{v \in \mathcal{C}} d(v)c(v, F)$ is minimized.

In the *metric* uncapacitated facility location problem, the connection costs c are required to be metric, i.e., $c(v, w) \leq c(v, w') + c(v', w') + c(v', w)$ for all $v, v' \in \mathcal{C}$ and all $w, w' \in \mathcal{F}$.

The non-metric version of UFL is closely related to set cover. On the one hand, it is easy to see that UFL generalizes set cover and is thus hard to be approximated better than by a logarithmic factor—corresponding to the inapproximability lower bound of set cover [Fei98]. On the other hand, the greedy algorithm for set cover can be adapted to UFL to obtain an $\ln(|\mathcal{C}|)$ -approximation, as observed by Hochbaum [Hoc82].

If the connection costs are metric, however, the reduction from set cover breaks down and a wide range of different techniques for achieving constant factor approximation algorithms has been developed, including greedy approaches, LP rounding, primal-dual schemes, and local search—see [MYZ06] for an overview. Shmoys, Tardos, and Aardal [STA97] were the first to achieve a constant approximation factor for metric UFL, using LP-rounding combined with a filtering technique by Lin and Vitter [LV92]. Since then, the approximation factor has been improved from 3.16 to the current value of 1.5, which is accomplished by combining a randomized rounding algorithm of Byrka and Aardal [BA10] with a primal-dual approximation by Jain, Mahdian, and Saberi [JMS02]. The algorithm in [BA10] is special in that it is a bifactor approximation algorithm: Given a parameter $\gamma > 1.68$ and a feasible solution to the LP relaxation of the UFL instance, it returns a solution with opening cost at most γ times the opening cost of the LP solution and connection cost at most $1 + 2e^{-\gamma}$ times the connection cost of the LP solution. We will make use of this parameterized analysis to improve the approximation factors of our algorithms.

Combining facility location and network design

As already indicated at the beginning of this chapter, the assumption of a dedicated connection for each individual client does not reflect the reality of many practical applications of facility location, where trees, tours, or more general network structures are

used to serve multiple clients jointly. The corresponding combinations of facility location and network design have been studied extensively in operations research literature. An early model can be found in the survey by Magnanti and Wong [MW84]. See the surveys by Gourdin, Labbé, and Yaman [GLY04] and Melo, Nickel, and Saldanha-Da-Gama [MNSDG09] for models and applications in telecommunication and logistics, respectively. For the latter, also see the literature on location routing cited in Section 4.2.1.

The above references focus on modeling aspects, applicability, and heuristic solution approaches for combined location and network design problems. In contrast to classic facility location, however, research on approximation algorithms for such combined problems is rather sparse. The first result with respect to approximation algorithms in this context is due to Ravi and Sinha [RS06], who studied the *capacitated-cable facility location* problem, in which clients are connected to facilities via cables of uniform capacity installed on the edges of a graph. Using the definition of the bidirected graph $B(G)$ in Section 1.1.3, the problem can be formally stated as follows.

Problem: Capacitated-cable facility location (CCFL)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, cable costs $c \in \mathbb{Q}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, and a cable capacity $U \in \mathbb{Q}_+$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$ and for each edge $e \in E$, a number of cables $z(e) \in \mathbb{Z}_+$ to be installed such that there is a flow $x \in \mathbb{Q}_+^{B(E)}$ in $B(G)$ with

$$(1) \sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a) = d(v) \text{ for all } v \in \mathcal{C},$$

$$(2) \sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a) = 0 \text{ for all } v \in V \setminus (F \cup \mathcal{C}),$$

$$(3) x(a_e^-) + x(a_e^+) \leq Uz(e) \text{ for all } e \in E,$$

minimizing the cost $\sum_{w \in F} f(w) + \sum_{e \in E} c(e)z(e)$.

Ravi and Sinha [RS06] proposed to combine a β_{ST} -approximation for Steiner tree and a β_{UFL} -approximation for metric UFL to obtain a $(\beta_{\text{ST}} + \beta_{\text{UFL}})$ -approximation algorithm for CCFL. Their algorithm constructs a feasible solution by iteratively relieving the load on the initial Steiner tree using the UFL solution. The algorithms presented in Sections 4.2 and 4.3 for capacitated location routing and facility location with capacitated and length-bounded trees, respectively, are based on the same framework of merging solutions for subproblems corresponding to different lower bounds to the combined problem.

Chen and Chen [CC09a] considered the *soft-capacitated facility location and cable installation problem*. In this generalization of CCFL, the demand served by a facility is bounded by a facility-dependent capacity, but arbitrarily many copies of each facility can be opened. The authors give a 19.84-approximation for this problem, using a combinatorial primal-dual scheme to obtain a forest clustering the clients, which is then combined with a solution for soft-capacitated facility location.

Bley, Hashemi, and Rezapour [BHR13, BR13] investigated the *connected facility location problem with buy at bulk edge costs*, in which various cable types of different cost and

capacity can be installed on the edges and the open facilities have to be connected via a Steiner tree using an additional cable type of infinite capacity. They devise two different constant factor approximation algorithms for the case that the cable costs observe an economies of scale assumption, one using a clustering technique by Talwar [Tal02] for the single-sink version of the problem, and one using a random sampling method.

Maßberg and Vygen [MV08] studied the related *sink clustering problem*, in which clients are served using Steiner trees with a capacity that bounds the sum of the demand and the cost of the tree. Instead of choosing facilities from a given set of possible locations, every tree incurs a uniform facility opening cost. The authors obtain a constant factor approximation for this problem. Their demand clustering technique, which was introduced by Alpert et al. [AKL⁺03], is also similar to the one employed by Ravi and Sinha [RS06].

4.1.3 Integrated planning vs. two-phase planning

We want to point out that fixing location decisions based solely on solving a UFL instance and then optimizing the connecting network based on those facilities can lead to solutions that are arbitrarily far away from the optimum of the combined problem. This can be seen from the following two examples of CCFL instances, which also directly extend to the problems considered in the following sections. Note that the classic UFL is not designed to take cable capacities into account by default. In the first example, the UFL solution is computed with respect to the original connection costs, completely ignoring the cable capacity. In the second example, the capacity is incorporated into the UFL instance by scaling all connection costs by a factor of $1/U$.

Example 4.1 Consider the following sequence of instances of CCFL. Let $\varepsilon > 0$. Instance I_n has n clients v_1, \dots, v_n with unit demands and two facilities w_1 and w_2 with opening costs $f(w_1) = \varepsilon$ and $f(w_2) = n$. Furthermore $U = n$, and there are edges from every client to every other vertex, with costs $c(v_i, w_1) = 1$, $c(v_i, w_2) = 0$, and $c(v_i, v_j) = 0$ for all $i, j \in \{1, \dots, n\}$. An optimal solution to this instance opens facility w_1 and installs a single cable on each of the edges $\{w_1, v_1\}, \{v_1, v_2\}, \dots, \{v_{n-1}, v_n\}$. This solution has cost $1 + \varepsilon$. On the other hand, interpreting I_n as an instance of UFL, the unique optimal solution opens w_2 and connects all clients to this facility. The opening cost of this solution is n , exceeding the cost of the optimal CCFL solution by a factor of $\theta(n)$.

Example 4.2 Consider the following family of instances of CCFL. Let $\varepsilon > 0$. Instance I_U with cable capacity $U \in \mathbb{Q}_+$ has one client v with demand $d(v) = 1$ and two facilities w_1 and w_2 with opening costs $f(w_1) = 0$ and $f(w_2) = 1 + \varepsilon$. The costs of the only two edges present in the graph are $c(v, w_1) = U$ and $c(v, w_2) = 0$. An optimal solution to this instance opens facility w_2 and connects v directly to w_2 . This solution has cost $1 + \varepsilon$. Interpreting I_U as an instance of UFL with connection cost c/U , the unique optimal solution opens facility w_1 . The only option to serve v based on this location decision is to install a cable from v to w_1 . The cost of the resulting CCFL solution is U , exceeding the cost of the optimal solution by a factor of $\theta(U)$.

4.1.4 Relieving overloaded trees

In the following sections, we will investigate approximation algorithms for problems combining facility location and network design similar to the CCFL problem introduced

Algorithm 4.1: Subroutine for relieving overloaded trees

Input: a rooted tree T' , a set of clients \mathcal{C} , demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, a capacity $U \in \mathbb{Q}_+$, a set of edges $\tilde{E} = \{e_v : v \in V(T') \cap \mathcal{C}\}$, costs $c \in \mathbb{Q}_+^{\tilde{E}}$
Output: a set of trees \mathcal{T} , a tree assignment $\phi : V(T') \cap \mathcal{C} \rightarrow \mathcal{T}$

```

procedure relieve ( $T', \tilde{E}, \mathcal{C}, d, U, c$ )
  Initialize  $\mathcal{T} = \emptyset$ .
  while  $d(V(T') \cap \mathcal{C}) > U$  do
    Find  $v' \in V(T')$  with  $d(V(T'[v']) \cap \mathcal{C}) > 0$  and  $d(V(T'[v]) \cap \mathcal{C}) \leq U$  for all
    children  $v$  of  $v'$  in  $T'$ .
    Let  $\mathcal{S} = \{T'[v] \cup T'[v, v'] : v \text{ is a child of } v' \text{ in } T'\}$ .
    Partition  $\mathcal{S}$  into sets  $\mathcal{S}_0, \dots, \mathcal{S}_k$  such that  $\sum_{S \in \mathcal{S}_0} d(V(S) \cap \mathcal{C}) \leq U$  and
     $U/2 \leq \sum_{S \in \mathcal{S}_i} d(V(S) \cap \mathcal{C}) \leq U$  for all  $i \in [k] \setminus \{0\}$ .
    for all  $i \in [k] \setminus \{0\}$  do
      Find  $v_i \in \bigcup_{S \in \mathcal{S}_i} V(S) \cap \mathcal{C}$  such that  $c(e_{v_i})$  is minimum.
      Set  $T_i = \bigcup_{S \in \mathcal{S}_i} S \cup \{e_{v_i}\}$ .
      Set  $\phi(v) = T_i$  for all  $v \in V(T_i) \cap \mathcal{C}$ .
      Set  $T' = T' \setminus T_i$ , and  $\mathcal{T} = \mathcal{T} \cup \{T_i\}$ .
  Set  $\mathcal{T} = \mathcal{T} \cup \{T'\}$  and  $\phi(v) = T'$  for all  $v \in V(T') \cap \mathcal{C}$ .
  return  $(\mathcal{T}, \phi)$ 

```

above. In the solution process of all these algorithms, an initial forest spanning all clients will serve as the backbone of the connecting network. While every tree of this forest is rooted at an open facility, the trees do not obey the capacity restrictions imposed by the respective optimization problems. In order to retrieve a feasible solution, overloaded trees, i.e., those trees who carry more demand than allowed, will be relieved by reconnecting some of their subtrees directly to a set of open facilities.

We describe a procedure that takes as input a tree and a set of additional edges, one for each client, connecting it to its nearest facility. It partitions the tree into several trees such that the demands within every tree obey the capacity restriction. It uses a subset of the additional edges to connect these new trees to open facilities. The total cost of the resulting solution can be bounded against the lower bounds presented later for the corresponding problems. We will make use of the procedure in all algorithms in the following sections.

The idea of relieving overloaded subtrees goes back to Alpert et al. [AKL⁺03], and Ravi and Sinha [RS06], who also combine a tree with a UFL solution using a flow rerouting scheme to obtain their approximation result for CCFL. Different from their approach, our procedure does not reroute individual client demands but partitions the set of children of an overloaded vertex, ensuring that the subtrees remain intact. This is necessitated by the fact that our algorithms will turn those trees to tours and cannot install singular cables on individual links.

The procedure

In all contexts in which we use the procedure, a set of clients \mathcal{C} with demands $d \in \mathbb{Q}_+^{\mathcal{C}}$ and a capacity $U \in \mathbb{Q}_+$ will be present. In addition, the procedure is given a tree T' ,

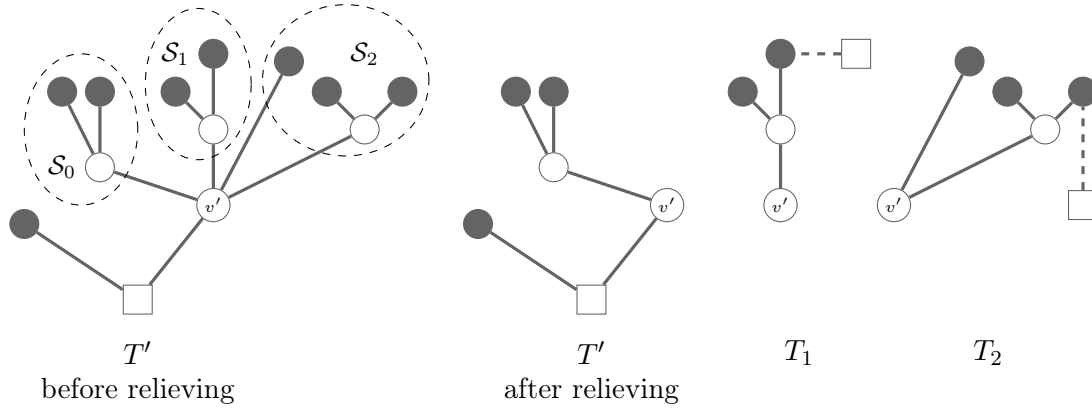


Figure 4.1: Relieving an overloaded tree T' . All clients have unit demands and $U = 3$. Clients are depicted as black dots, non-clients as white circles. The roots of the trees are depicted as squares. Dashed edges are from the set \tilde{E} given in the input. The subtree rooted at node v' is partitioned into groups \mathcal{S}_0 , \mathcal{S}_1 , and \mathcal{S}_2 with the first one remaining in the tree.

which is rooted at a open facility, and a set of edges $\tilde{E} = \{e_v : v \in V(T') \cap \mathcal{C}\}$ with costs c , such that every edge e_v connects client v with an open facility. In order to enable a more unified presentation, we will assume throughout the procedure that clients only occur as leaves of the tree; this can be achieved without loss of generality by introducing dummy vertices and edges at clients that are inner vertices of T' . Note, however, that when reverting this construction for the returned set of trees \mathcal{T} , a client might occur in multiple trees, as the constructed trees are not disjoint and may intersect at inner vertices. We therefore return an additional tree assignment function ϕ that specifies for each client v the tree $\phi(v) \in \mathcal{T}$ serving the client.

The procedure iteratively identifies a vertex $v' \in V(T')$ such that the demand in $T'[v']$ exceeds the capacity U , but does not exceed the capacity in any tree $T'[v]$ for a child v of v' . Such a vertex can be found by following the path from the root to a leaf of T' . Let \mathcal{S} be the set containing all subtrees $T'[v]$ with v being a child of v' . This set is greedily partitioned into groups $\mathcal{S}_0, \dots, \mathcal{S}_k$ such that the sum of demands of all subtrees in group \mathcal{S}_i for $i \neq 0$ is at least $U/2$ and at most U , and the sum of demands of the subtrees in \mathcal{S}_0 is at most U . The trees in \mathcal{S}_0 will remain in T' . For each other group $i \in [k] \setminus \{0\}$, we identify a client v_i with minimum connection cost $c(e_{v_i})$ among all clients in that group. The subtrees in \mathcal{S}_i together with the edges connecting them to v' and the edge e_{v_i} form a new tree T_i , which is extracted from T' and added to the output. We repeat this procedure until the total remaining demand in T' is at most U . Finally, we also add the remainder of T' to the output. The procedure described formally as procedure `relieve` in Algorithm 4.1 and an example is given in Figure 4.1.

Analysis

The `relieve` procedure will play an important role as a subroutine in all algorithms presented in the following sections. It is easy to verify that the trees constructed by the procedure indeed serve all clients of the original tree without violating the capacity—assuming that each edge in \tilde{E} is incident to an open facility.

Observation 4.3 The output (\mathcal{T}, ϕ) of the procedure described in Algorithm 4.1 fulfills the following properties.

- Every client $v \in V(T') \cap \mathcal{C}$ is contained in the tree $\phi(v)$.
- Every tree $T \in \mathcal{T}$ either contains the root of T' , or an edge from \tilde{E} .
- $\sum_{v \in \mathcal{C} : \phi(v)=T} d(v) \leq U$ for all $T \in \mathcal{T}$

The next lemma will be helpful to bound the cost of the solutions produced by algorithms using the procedure.

Lemma 4.4 Let \mathcal{T} be the set of trees returned by the `relieve` procedure described in Algorithm 4.1, and let T' and $\tilde{E} = \{e_v : v \in \mathcal{C}\}$ be the tree and the edge set given as input to the procedure. Then

$$\sum_{T \in \mathcal{T}} c(T) \leq c(T') + 2 \sum_{v \in V(T') \cap \mathcal{C}} d(v) \frac{c(e_v)}{U}.$$

Proof. Let $T \in \mathcal{T}$. Observe that if $e \in T \cap T'$, the edge e does not appear in any other tree of \mathcal{T} , as the edges in T were immediately extracted from T' when T was created. Furthermore, there is at most one edge $e \in T \cap \tilde{E}$. This edge $e = e_{v_i}$ was added in the inner loop of the procedure when connecting a group \mathcal{S}_i of the partition. Observe that no edge e_v for any $v \in C_i := \bigcup_{S \in \mathcal{S}_i} V(S) \cap \mathcal{C}$ can be part of any other tree in \mathcal{T} , as the corresponding clients were removed from T' with the creation of T . The lemma thus follows by observing

$$c(e_{v_i}) \leq \sum_{v \in C_i} \frac{2d(v)}{U} c(e_{v_i}) \leq 2 \sum_{v \in C_i} d(v) \frac{c(e_v)}{U}$$

where the first inequality holds because $d(C_i) \geq U/2$ and the second inequality is implied by the choice of e_{v_i} minimizing the cost. \square

4.2 Capacitated location routing

In many logistics networks, client deliveries are performed by vehicles based at regional depots. The task of optimizing the tours along which these vehicles are operating is known as *vehicle routing* and constitutes an important area of operations research. When planning the location of depots in such a setting, both the operating costs of the depots and the prospective vehicle routing costs have to be taken into account. The integrated problem of jointly making location and routing decisions is known as *location routing* and has received significant attention in the operations research community as well. A basic variant of location routing is the capacitated location routing problem defined as follows.

Problem: Capacitated location routing (CLR)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, and a vehicle capacity $U \in \mathbb{Q}_+$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$ and a set of closed walks \mathcal{T} with a demand assignment $x \in \mathbb{Q}_+^{\mathcal{C} \times \mathcal{T}}$ such that

- (1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathcal{T}$,
- (2) $\sum_{T \in \mathcal{T}: v \in V(T)} x(v, T) = d(v)$ for all $v \in \mathcal{C}$,
- (3) $\sum_{v \in \mathcal{C}} x(v, T) \leq U$ for all $T \in \mathcal{T}$,

minimizing the cost $\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

The special case of CLR where location decisions have already been made (i.e., $f \equiv 0$) is the *multi-depot capacitated vehicle routing problem* (MCVR).

Throughout this section, the closed walks in \mathcal{T} will be referred to as *tours*, and we will use the terms *depot* and *facility* interchangeably.

Splittable demands and single assignments. In the version of the problem described above, the demand of a client may be split up and served by multiple facilities, which is not always desired or even possible in practice. This motivates the following terminology. A solution to CLR fulfills the *single-assignment property* [LNT88, NS07], if the demand of each client is served by exactly one facility. A solution fulfills the *single-tour property*, if each the demand of each client is served by exactly one tour, i.e., if it is a feasible solution to the version of the problem with *unsplittable demand*. Clearly, this latter property can only be fulfilled if $d(v) \leq U$ for all $v \in \mathcal{C}$.

Remark 4.5

- (1) Because the vehicles serve the clients along closed walks—which might traverse the same edge multiple times—the following assumptions are without loss of generality.
 - G is complete and c is a metric.
 - Every tour visits exactly one open facility and no other non-client vertices.
 - The vertex set is partitioned into clients and facilities, i.e., $V = \mathcal{C} \dot{\cup} \mathcal{F}$.
- (2) Given a set of open facilities F and a set of tours \mathcal{T} , a feasible demand assignment x (if one exists) can be found by solving a maximum flow problem in a bipartite graph.
- (3) Note that the above model also implicitly covers depot-dependent fixed costs per tour, i.e., each vehicle sent out from facility $w \in \mathcal{F}$ incurs a cost of $a(w) \in \mathbb{Q}_+$. This can be easily modeled by adding $\frac{1}{2}a(w)$ to the cost of all edges incident to w , as each tour originating at w contains exactly two of these edges.

- (4) In the uncapacitated case ($U = \infty$), CLR and MCVR are equivalent: By the triangle inequality, every optimal solution to either problem can be transformed such that at most one tour originates from each open depot without increasing cost. Hence, facility opening costs can be modeled by adding $\frac{1}{2}f(w)$ to $c(e)$ for all edges e incident to a facility $w \in \mathcal{F}$.

4.2.1 Work related to vehicle routing and location routing

Both vehicle routing and location routing have occupied a central place in operations research literature over the past decades. We will give an overview of works regarding approximation algorithms in this area, with some pointers to text books and survey articles when referring to other main streams of research.

Vehicle routing

Vehicle routing problems have been studied in countless variants from various perspectives. Algorithmic results focus particularly on combinatorial heuristics and exact branch-and-bound algorithms. For an overview of the rich literature in this field, we refer the reader to the books edited by Toth and Vigo [TV02] and Golden and Assad [GA88].

Single depot. Regarding approximation algorithms, there is a large body of work for the classic *capacitated vehicle routing problem* (CVR), in which only a single depot is present. The most fundamental special case is the famous *traveling salesperson problem* (TSP), which asks for a single tour visiting all vertices. The currently best known approximation algorithm for TSP is the $3/2$ -approximation of Christofides [Chr76], which combines a minimum spanning tree with a matching on its odd degree vertices. It is a longstanding open question whether this result can be improved. A seminal result by Arora [Aro96] yields a PTAS for TSP with Euclidean distances. Altinkemer and Gavish [AG87] proposed a tour partitioning technique that transforms a TSP tour into several tours respecting the vehicle capacity. They showed that this algorithm is a $2 + \beta_{\text{TSP}}$ -approximation for CVR with unsplittable demands, where the initial tour is computed by a β_{TSP} -approximation for TSP. For CVR with Euclidean distances and unit demands, Haimovich and Rinnoy Kan [HRK85] presented a $(1 + \varepsilon)$ -approximation whose running time is exponential in U/ε . On the negative side, it is known that, unless $P = NP$, CVR with arbitrary unsplittable demands cannot be approximated better than by a factor of 1.5, even when all clients are located on the same point of the Euclidean plane [GW81].

Multiple depots. Li and Simchi-Levi [LSL90] studied algorithms for multi-depot capacitated vehicle routing. Among other results, they generalized the tour partitioning technique from [AG87] to the multi-depot case, providing a $(2 + 2\beta_{\text{TSP}})$ -approximation algorithm for arbitrary, unsplittable demands. They also showed that their analysis of this algorithm is tight. Previous to the work presented in this chapter, this has been the best known approximation algorithm for the unsplittable demand version of MCVR, giving a 5-approximation using the TSP algorithm from [Chr76]. Additional algorithms for the uncapacitated case of multi-depot vehicle routing are listed below as they also apply to location routing.

Location routing

One of the earliest models of location routing appeared in an article by Webb [Web68]. Laporte [Lap88] gave a comprehensive overview of the literature prior to the late 80s. More recent survey articles summarizing heuristic algorithms and mathematical programming formulations for many variants of location routing have been published by Mina, Jayaraman, and Srivastava [MJS98], and Nagy and Salhi [NS07].

Approximation results for the uncapacitated case. There are only a few works that are concerned with approximation algorithms for location routing problems, all of them restricted to the case of unbounded vehicle capacity. In the uncapacitated case, opening costs can be modeled as connection costs and the problem is equivalent to multi-depot vehicle routing; see Remark 4.5 (4). Goemans and Williamson [GW95] remarked that their primal-dual technique for constrained forest problems also yields a 2-approximation for uncapacitated location routing. Glicksman and Penn [GP08] generalized this result to the case of uncapacitated group location routing, where one is given a system of groups of clients, and only one client from each group needs to be served. They derived a $2L$ -approximation algorithm, with L denoting the cardinality of the largest group. Finally, Chen and Chen [CC09b] provided a 24-approximation for location routing with soft facility capacities, i.e., facilities can be installed multiple times with each copy capable of serving a limited amount of demand, while vehicle loads are unbounded.

Extended models

In later sections, we will study several extensions of the standard CLR model. Here we give an overview of previous work regarding these variants.

Prize-collecting variant. In the *prize-collecting* (PC) version of an optimization problem, a feasible solution does not have to serve all clients. Instead, an individual penalty may be paid for each unserved client. Thereby, PC can precisely model outsourcing decisions and is hence of profound practical interest. For the PC version of metric UFL, Jain et al. [JMM⁺03] presented a 2-approximation, improving on the 3-approximation by Charikar et al. [CKMN01], but omitting a complete proof. We are not aware of any previous approximation results for PC vehicle routing or PC location routing.

Group variant. In the *group* variant, the set of clients is partitioned into disjoint subsets, or groups of clients, and only one client from every group has to be served. The group version of UFL does not allow for a constant factor approximation, even when the costs are metric, as we shall see in Section 4.2.4. For the group version of uncapacitated location routing, the only previous result we know of is the algorithm by Glicksman and Penn [GP08] mentioned above.

Cross-docking. In capacitated location routing and multi-depot capacitated vehicle routing, *cross-docking* may be allowed in certain application scenarios, i.e., shipments may be loaded from one vehicle to another when tours intersect. Cross-docking plays a significant role in numerous logistics applications. Some heuristic approaches have recently been proposed for vehicle routing with cross-docking by Wen et al. [WLC⁺09] and Vahdani and Zandieh [VZ10]. Their models also exhibit strong similarity to the

mixed truck delivery problem, which was studied by Liu, Li, and Chan [LLC03]. Here, clients may be served by tours either from facilities or from hubs, which are in turn served by facilities. The authors developed a heuristic solution approach and presented computational results suggesting that routing cost can be reduced on average by around 10% for random instances when allowing cross-docking.

4.2.2 A constant factor approximation for CLR

In this section, we derive a constant factor approximation for CLR by combining two lower bounds on the value of an optimal solution: a minimum spanning tree in an auxiliary graph and an optimal solution to a UFL instance with scaled costs.

Two combinatorial lower bounds

Assume we are given an instance of CLR and let OPT be the cost of an optimal solution to this instance. The following two lower bounds are adaptations of the UFL and Steiner tree lower bounds used in [RS06] to the tour connections occurring in location routing.

Lemma 4.6 *Let $\tilde{F} \subseteq \mathcal{F}$ be an optimal solution to the UFL instance with facilities \mathcal{F} , clients \mathcal{C} , opening costs f and connection costs $\tilde{c} := \frac{2}{\alpha}c$. Then*

$$\sum_{w \in \tilde{F}} f(w) + \sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F}) \leq OPT.$$

Proof. Let (F, \mathcal{T}, x) be an optimal solution of the CLR instance. Note that $2\tilde{c}(v, F) \leq \tilde{c}(T)$ for all $v \in \mathcal{C}$ and all $T \in \mathcal{T}$ with $v \in V(T)$, as every tour T containing v can be decomposed into two paths from v to an open facility. Thus

$$\sum_{v \in \mathcal{C}} d(v)\tilde{c}(v, F) \leq \frac{1}{2} \sum_{T \in \mathcal{T}} \sum_{v \in \mathcal{C}} x(v, T)\tilde{c}(T) \leq \sum_{T \in \mathcal{T}} c(T),$$

which proves the lemma, when interpreting F as a feasible solution of the UFL instance. \square

Lemma 4.7 *Consider the graph $G' = (V \cup \{r\}, E \cup E')$, where $E' = \{\{r, w\} : w \in \mathcal{F}\}$ and define costs*

$$c'(e) = \begin{cases} 0 & \text{if } e = \{r, w\} \text{ for some } w \in \mathcal{F} \\ c(e) + \frac{1}{2}f(w) & \text{if } e = \{v, w\} \text{ for some } v \in \mathcal{C}, w \in \mathcal{F} \\ c(e) & \text{otherwise} \end{cases}$$

Let T' be a minimum spanning tree in G' with respect to c' . Then $c'(T') \leq OPT$.

Proof. Consider an optimal solution (F, \mathcal{T}, x) to the CLR instance. We will construct a spanning tree in G' that has at most the same cost as this solution. For every open facility $w \in F$, let T_1, \dots, T_k be the tours based at w (in an arbitrary but fixed order) with $T_i = (w, v_1^i, \dots, v_{\ell_i}^i, w)$ where ℓ_i is the number of clients in T_i . For every $i \in \{1, \dots, k-1\}$, replace the last edge $\{v_{\ell_i}^i, w\}$ of T_i and the first edge $\{w, v_1^{i+1}\}$ of T_{i+1} by the edge $\{v_{\ell_i}^i, v_1^{i+1}\}$. Also remove the final edge $\{v_{\ell_k}^k, w\}$ of T_k . As a result, we get a walk P_w from w to $v_{\ell_k}^k$ along all clients that are served by w . Note

Algorithm 4.2: Approximation algorithms for CLR**Step 1:**

Create a UFL instance with edge costs $\tilde{c} = \frac{2}{U}c$ as described in Lemma 4.6.

Apply the bifactor approximation algorithm in [BA10] with $\gamma = 2.38$ on this instance and let \tilde{F} be the resulting set of open facilities.

For $v \in \mathcal{C}$, choose $w(v) \in \tilde{F}$ such that $c(v, w(v))$ is minimal.

Step 2:

Construct the graph G' with edge costs c' as described in Lemma 4.7.

Compute a minimum spanning tree T' in G' with respect to c' .

Let F' be the set of facilities that are incident to an edge in $T' \cap E$.

Step 3:

for all $v \in \mathcal{C}$ **with** $d(v) \geq U$ **do**

Construct $\lceil \frac{d(v)}{U} \rceil$ copies of a tour from v to a closest facility in $\tilde{F} \cup F'$.
Add those tours to \mathcal{T} , set x accordingly, and remove v from \mathcal{C} .

Step 4:

for all $w \in F'$ **do**

Let $\tilde{E}_w = \{\{v, w(v)\} : v \in V(T'[w]) \cap \mathcal{C}\}$.
Call `relieve` ($T'[w], \tilde{E}_w, \mathcal{C}, d, U, c$) and obtain trees \mathcal{T}_w and assignments ϕ .
for all $T \in \mathcal{T}_w$ **do**
 Construct a tour \bar{T} visiting all vertices in $V(T)$ by doubling the edges of T
 and short-cutting. Set $x(v, \bar{T}) = d(v)$ for all $v \in \mathcal{C}$ with $\phi(v) = T$.
 Add \bar{T} to \mathcal{T} .

return $(\tilde{F} \cup F', \mathcal{T}, x)$.

that $c'(P_w) \leq \sum_{i=1}^k c(T_i) + \frac{1}{2}f(w)$ by triangle inequality and the fact that P_w contains only one edge incident to w .

Now let $S = \bigcup_{w \in F'} E(P_w) \cup E'$. As S spans r and all facilities and contains a walk from any client to a facility, it contains a spanning tree of G' with cost at most

$$c'(S) \leq \sum_{w \in F'} c'(P_w) \leq \sum_{T \in \mathcal{T}} c(T) + \sum_{w \in F'} f(w) = OPT. \quad \square$$

Remark 4.8 A different analysis using the undirected cut relaxation of the Steiner tree problem reveals a stronger, LP-based upper bound on the cost of the minimum spanning tree; see Lemma 4.18 and Corollary 4.19 for details. However, as this does not improve the performance guarantee of the approximation algorithm discussed in this section, we restrict ourselves to the simpler, combinatorial proof of the tree lower bound here.

Algorithm

We now use the lower bounds described above to obtain an approximate solution to the CLR instance. Our algorithm computes an approximate solution to the UFL instance described in Lemma 4.6 and a minimum spanning tree as described in Lemma 4.7 and uses the `relieve` procedure described in Algorithm 4.1 to decompose the spanning tree into subtrees obeying the capacity constraints. These are connected to the facilities from the UFL solution and turned into tours by doubling the edges.

In detail, the algorithm works as follows. After obtaining an approximate UFL solution \tilde{F} and computing a minimum spanning tree T' , we open all facilities in \tilde{F} and also the set F' of all facilities w that are incident to any edge other than $\{r, w\}$ in T' . We first serve all clients with large demands, i.e., every $v \in \mathcal{C}$ with $d(v) \geq U$ is assigned to a closest open facility and served by $\lceil d(v)/U \rceil$ tours comprising only the assigned facility and the client. The tours for the remaining clients are constructed from the tree T' . Note that by removing r , the tree T' decomposes into a forest, with each tree rooted at the facilities in F' . Using the edges induced by the facility set \tilde{F} , each of the trees in the forest is partitioned further by the `relieve` procedure described in Algorithm 4.1. The resulting trees have demands between $U/2$ and U . They are turned into tours by doubling edges and short-cutting using the triangle inequality. A formal listing of the approximation algorithm for CLR is given as Algorithm 4.2.

Analysis

We analyze the algorithm presented above and show that it is a 4.38-approximation for CLR. We start by using our previous analysis of the `relieve` procedure to bound the cost of the solution produced by the algorithm against the cost of the spanning tree and the facility location solution.

Lemma 4.9 *Let (F, \mathcal{T}, x) be the solution computed by Algorithm 4.2 and let T' be the spanning tree computed in Step 1 and \tilde{F} be the set of open facilities computed in Step 2. Then*

$$\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T) \leq 2c'(T') + \sum_{w \in \tilde{F}} f(w) + 2 \sum_{v \in \mathcal{C}} d(v) \tilde{c}(v, \tilde{F}).$$

Proof. Every tour T constructed in Step 3 for a client v with large demand has cost at most

$$c(T) \leq 2 \underbrace{\left\lceil \frac{d(v)}{U} \right\rceil}_{\leq 2d(v)/U} c(v, \tilde{F}) \leq 2d(v) \tilde{c}(v, \tilde{F})$$

as $d(v)/U \geq 1$. The remaining tours have cost at most twice the cost of the trees produced by the `relieve` procedure. Thus, by Lemma 4.4 their cost is bounded by

$$2 \sum_{w \in F'} \sum_{T \in \mathcal{T}_w} c(T) \leq \sum_{w \in F'} 2c(T'[w]) + 4 \frac{d(v)}{U} c(\tilde{E}_w) \leq c(T') + 2 \sum_{v \in \mathcal{C} : d(v) < U} d(v) \tilde{c}(v, \tilde{F}).$$

The opening cost of the facilities in F' is furthermore bounded by $2(c'(T') - c(T))$. Summing everything up yields the claim of the lemma. \square

Consequently, if \tilde{F} is a β -approximation to a minimum cost solution to the UFL instance, Algorithm 4.2 constructs a $(2 + 2\beta)$ -approximation to the CLR instance. Note, however, that in this analysis the opening cost for the facilities in \tilde{F} is counted twice, while the actual solution only pays it once. We can improve the approximation factor by using the bifactor approximation algorithm for UFL from [BA10]. Recall that, given a parameter $\gamma > 1.68$, this algorithm returns a solution whose opening cost exceeds the opening cost of an initially computed optimal LP solution by at most a factor of γ , and whose connection cost exceeds the connection cost of the fractional solution by at

most $1 + 2e^{-\gamma}$. Let \tilde{c}_{LP} be the connection cost of the LP solution and let f_{LP} be the opening cost of the LP solution. Then

$$2 \sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F}) + \sum_{w \in \tilde{F}} f(w) \leq 2(1 + 2e^{-\gamma}) \tilde{c}_{\text{LP}} + \gamma f_{\text{LP}},$$

which is bounded by $\gamma(\tilde{c}_{\text{LP}} + f_{\text{LP}}) \leq \gamma \text{OPT}$ for $\gamma \geq 2.38$. Choosing $\gamma = 2.38$, Lemma 4.9 yields our main result.

Theorem 4.10 *Algorithm 4.2 is a 4.38-approximation algorithm for CLR. It fulfills the single-assignment property. If $d(v) \leq U$ for all $v \in \mathcal{C}$, it fulfills the single-tour property.*

Of course, the approximation ratio of our algorithm improves for classes of instances that allow for a better UFL approximation. An example is the case of Euclidean costs. Here, the PTAS by Arora, Raghavan, and Rao [ARR98] can be applied to obtain a $(4 + \epsilon)$ -approximation for CLR.

Multi-depot vehicle routing. Recall that the special case of CLR where opening facilities does not incur cost ($f \equiv 0$) is known as multi-depot capacitated vehicle routing problem (MCVR). In this case, the corresponding UFL instance can be solved optimally, by connecting each client to its nearest facility. Thus, we can replace the factor incurred by the UFL approximation algorithm by 1 and obtain the following result, which improves the previously best known approximation guarantee of 5 for MCVR fulfilling the single-tour property [LSL90].

Theorem 4.11 *Algorithm 4.2 is a 4-approximation algorithm for MCVR. It fulfills the single-assignment property. If $d(v) \leq U$ for all $v \in \mathcal{C}$, it fulfills the single-tour property.*

4.2.3 Prize-collecting location routing

We now apply our algorithmic framework for CLR and MCVR to the *prize-collecting* (PC) variant of these problems. In a prize-collecting setting, we can decide for each client whether to serve it by our solution, or to pay a penalty for not serving it.

Problem: Prize-collecting CLR (PC-CLR)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, a vehicle capacity $U \in \mathbb{Q}_+$, and penalties $p \in \mathbb{Q}_+^{\mathcal{C}}$.

Task: Find a set of clients $C \subseteq \mathcal{C}$, a set of facilities $F \subseteq \mathcal{F}$, and a set of closed walks (called *tours*) \mathcal{T} with a demand assignment $x \in \mathbb{Q}_+^{\mathcal{C} \times \mathcal{T}}$ such that

- (1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathcal{T}$,
- (2) $\sum_{T \in \mathcal{T}: v \in V(T)} x(v, T) = d(v)$ for all $v \in \mathcal{C} \setminus C$,
- (3) $\sum_{v \in \mathcal{C}} x(v, T) \leq U$ for all $T \in \mathcal{T}$,

minimizing the cost $\sum_{v \in \mathcal{C}} p(v) + \sum_{w \in \mathcal{F}} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

Note that prize-collecting can naturally be viewed as a way of incorporating outsourcing decisions into an optimization model: In this case, the penalty for not serving a customer corresponds to the cost of having it served by an outside service provider. As outsourcing is an important option in many logistics applications, the prize-collecting variants of CLR and MCVR are highly relevant in practice.

Remark 4.12 All observations made in Remark 4.5 remain true for the prize-collecting version of the problem. It is also not hard to see that PC-CLR is a generalization of CLR: By setting penalties high enough, we can force any optimal solution to serve all clients.

Algorithm

We solve the prize-collecting variant of CLR by utilizing an approximation algorithm for prize-collecting UFL, and an LP-based approximation algorithm for the prize-collecting Steiner tree problem to determine two respective sets of customers served. We then compute a solution to PC-CLR serving exactly those customers served by both the tree and the facility location solution.

A formal description of the algorithm is given in Algorithm 4.3. We will prove that it is a $(\beta_{\text{PC-ST}} + 2\beta_{\text{PC-UFL}})$ -approximation algorithm for PC-CLR, where $\beta_{\text{PC-ST}}$ and $\beta_{\text{PC-UFL}}$ denote the approximation factors of the approximation algorithms used for prize-collecting Steiner tree with respect to the undirected cut relaxation and PC-UFL, respectively. Currently, the best known approximation algorithm for PC-UFL by Jain et al. [JMM⁺03] achieves an approximation ratio of $\beta_{\text{PC-UFL}} = 2$, while for prize-collecting Steiner tree the primal-dual algorithm of Goemans and Williamson [GW95] achieves an approximation factor of 2, meeting the integrality gap of the LP relaxation. Using these algorithms as subroutines results in an approximation factor of 6 for our algorithm.

First note that both Lemmas 4.6 and 4.7 can directly be transferred to the prize-collecting setting: In the corresponding proofs, we construct a UFL solution or a spanning tree, respectively, from an optimal solution of CLR without increasing the cost. It is easy to see that this construction adapts naturally when transferring the set of clients served from an optimal PC-CLR solution to feasible solutions of prize-collecting UFL or Steiner tree: The penalties for customers not served are exactly the same in both solutions. This immediately gives an approximation guarantee of $2(\beta_{\text{PC-UFL}} + \beta_{\text{PC-ST}})$ for Algorithm 4.3.

However, we can improve our analysis by using a tighter lower bound in case of the tree. To this end, we consider a prize-collecting Steiner tree instance defined as follows. We consider the same graph G' as constructed in Lemma 4.7. We then extend the cost function c to the edges in E' by defining cost $c(r, w) = \frac{1}{2}f(w)$ for each $w \in \mathcal{F}$ and define new penalties by setting $p' := \frac{1}{2}p$. We let $R = \mathcal{C} \cup \{r\}$ be the set of terminals. We will use a primal-dual approximation algorithm for prize-collecting Steiner tree due to Goemans and Williamson [GW95]. It is based on the following LP relaxation.

$$\begin{aligned}
 \text{[PC-ST}_{\text{LP}}] \quad \min \quad & \sum_{e \in E \cup E'} c(e)y(e) + \sum_{N \subseteq \mathcal{C}} p'(N)z(N) \\
 \text{s.t.} \quad & \sum_{e \in \delta_{G'}(S)} y(e) + \sum_{N \subseteq \mathcal{C}: S \cap \mathcal{C} \subseteq N} z(N) \geq 1 \quad \forall S \subseteq V, S \cap \mathcal{C} \neq \emptyset \\
 & y(e) \geq 0 \quad \forall e \in E \cup E'
 \end{aligned}$$

Algorithm 4.3: Approximation algorithms for PC-CLR

Step 1: Create a PC-UFL instance with edge costs $\tilde{c} = \frac{2}{U}c$ as described in Lemma 4.6. Apply a $\beta_{\text{PC-UFL}}$ -approximation algorithm and let \tilde{F} be the resulting set of open facilities and \tilde{C} the set of clients not served.

Step 2: Use the algorithm from [GW95] to obtain a 2-approximate prize-collecting Steiner tree T' with respect to the instance defined by [PC-ST_{LP}] in graph G' . Let C' be the set of clients not spanned by T' and let F' be the set of facilities spanned by T' .
Remove $\tilde{C} \cup C'$ from \mathcal{C} .

Step 3: The same as Step 3 from Algorithm 4.2.

Step 4: The same as Step 4 from Algorithm 4.2.

return $(\tilde{F} \cup F', \mathcal{T}, x, \tilde{C} \cup C')$.

The intuition for the LP relaxation is the following: Given a feasible solution to the prize-collecting Steiner tree problem, define $z(C) = 1$ for the set C of clients that are not connected to the Steiner tree, and $z(N) = 0$ for all other sets of clients. Moreover, set $y(e) = 1$ if edge e is in the Steiner tree, $y(e) = 0$ otherwise. The inequalities follow from the fact that any cut that separates a served client from the root $r \notin V$ has to be crossed by at least one edge of the tree.

Lemma 4.13 *Let OPT be the cost of an optimal solution to a PC-CLR instance and let (y, z) be an optimal solution to [PC-ST_{LP}]. Then*

$$\sum_{e \in E \cup E'} c(e)y(e) + \sum_{N \subseteq \mathcal{C}} \sum_{v \in N} p'(v)z(N) \leq \frac{1}{2}OPT.$$

Proof. Let (C, F, \mathcal{T}, x) be an optimal solution to the PC-CLR instance. Construct a solution (\tilde{y}, \tilde{z}) to [PC-ST_{LP}] by setting $\tilde{z}(C) = 1$ and $\tilde{z}(N) = 0$ for all $N \subseteq \mathcal{C}$ with $N \neq C$, and $\tilde{y}(\{r, w\}) = 1$ for all $w \in F$, $\tilde{y}(\{r, w\}) = 0$ for all $w \in \mathcal{F} \setminus F$, and $\tilde{y}(e) = \frac{1}{2}|\{T \in \mathcal{T} : e \in T\}|$. It is easy to observe that the constructed solution (\tilde{y}, \tilde{z}) has cost $\frac{1}{2} \sum_{w \in F} f(w) + \frac{1}{2} \sum_{v \in \mathcal{C}} p(v) + \frac{1}{2} \sum_{T \in \mathcal{T}} c(T)$.

It remains to show that (\tilde{y}, \tilde{z}) is feasible for [PC-ST_{LP}]. So let $S \subseteq V$ with $S \cap \mathcal{C} \neq \emptyset$. If S contains an open facility w , then $\{r, w\} \in \delta_{G'}(S)$, and by definition of \tilde{y} , the constraint for S is fulfilled. Otherwise, if $S \cap \mathcal{C} \subseteq C$, then by definition of \tilde{z} , the constraint for S is satisfied as well. Finally, if S does not contain an open facility and $(S \cap \mathcal{C}) \setminus C \neq \emptyset$, then there is a client $v \in S \setminus C$ connected to an open facility outside of S by a tour. At least two edges of this tour cross the cut $\delta_{G'}(S)$, hence the constraint for S is again satisfied by definition of \tilde{y} . \square

Theorem 4.14 *Algorithm 4.3 is a $(2+2\beta_{\text{PC-UFL}})$ -approximation algorithm for PC-CLR, and it is a 4-approximation algorithm for PC-MCVR, i.e., for instances of PC-CLR with $f \equiv 0$. It fulfills the single-assignment property. If $d(v) \leq U$ for all $v \in \mathcal{C}$, it fulfills the single-tour property.*

Proof. By Lemma 4.9, the cost of the solution constructed by the algorithm is bounded

by

$$\begin{aligned}
& \sum_{v \in \mathcal{C}} p(v) + 2c(T') + \sum_{w \in \tilde{F}} f(w) + 2 \sum_{v \in \mathcal{C} \setminus \mathcal{C}} \tilde{c}(v, \tilde{F}) \\
& \leq 2 \left(c(T') + \sum_{v \in \mathcal{C}'} p'(v) \right) + \sum_{w \in \tilde{F}} f(w) + 2 \sum_{v \in \mathcal{C} \setminus \tilde{\mathcal{C}}} \tilde{c}(v, \tilde{F}) + \sum_{v \in \tilde{\mathcal{C}}} p(v) \\
& \leq (2 + 2\beta_{\text{PC-UFL}}) \text{OPT},
\end{aligned}$$

where the last inequality uses Lemma 4.13 and the fact that the algorithm in [GW95] computes a 2-approximation with respect to the LP relaxation [PC-STLP]. \square

4.2.4 Group location routing and LP relaxation

In this section, we consider a group version of location routing (G-CLR) where the set of clients is partitioned into groups and only one client from each group needs to be served.

Problem: Group capacitated location routing (G-CLR)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a partition $\mathcal{C}_0, \dots, \mathcal{C}_k$ of \mathcal{C} , a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, and a vehicle capacity $U \in \mathbb{Q}_+$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$ and a set of closed walks (called *tours*) \mathcal{T} with a demand assignment $x \in \mathbb{Q}_+^{\mathcal{C} \times \mathcal{T}}$ such that

- (1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathcal{T}$,
- (2) for every $i \in [k]$ there is a client $v \in \mathcal{C}_i$ in the corresponding group with $\sum_{T \in \mathcal{T}: v \in V(T)} x(v, T) = d(v)$,
- (3) $\sum_{v \in \mathcal{C}} x(v, T) \leq U$ for all $T \in \mathcal{T}$,

minimizing the cost $\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

The uncapacitated version of the problem was studied by Glicksman and Penn [GP08], who give a $(2 - \frac{1}{|V|-1})L$ -approximation algorithm with L being the cardinality of the largest group. Their idea is to solve an LP relaxation of the problem and use the resulting fractional solution to decide which client is to be served from each group. We extend this approach to the capacitated case which is significantly more complex: In the absence of vehicle capacities, facility opening costs can be transferred to edges of the graph, i.e., location routing is equivalent to multi-depot vehicle routing in this case. In contrast to the result in [GP08], our LP relaxation has to explicitly incorporate the facility location aspect of the problem.

The dependence of our approximation factor on the parameter L gives rise to the question whether there is a constant factor approximation algorithm for G-CLR that is independent of any parameters in the input. At the end of this section, we answer this question in the negative by showing hardness of approximation for G-CLR for any factor better than $\ln(k)$.

Remark 4.15 Again, the assertions made in Remark 4.5 remain valid for the group version of the problem. However, although we can assume the connection costs to be metric, the groups can be used to emulate certain aspects of non-metric connection costs, rendering the problem much harder; see Theorem 4.21 for a non-approximability result making use of this fact.

LP relaxation

In order to obtain an approximation for G-CLR, we describe how to transform a solution of G-CLR into a multi-commodity flow. We then prove a set of valid inequalities fulfilled by all variable assignments obtained from feasible G-CLR solutions. The LP relaxation resulting from these inequalities can be used to decide on a set of representatives, one for each client group. Replacing each group by its representative, we obtain an instance of (non-group) CLR, which can be approximated by an adaption of Algorithm 4.2 with the spanning tree replaced by a Steiner tree. We will show that the resulting solution to G-CLR is a $4.38L$ -approximation.

While the problem remains based on an undirected graph, it is more convenient to consider the corresponding bidirected digraph $B(G) = (V, B(E))$ as defined in Section 1.1.3. We start constructing a multi-commodity flow in $B(G)$ from a given (undirected) solution of G-CLR by fixing an arbitrary orientation for every tour. Let $y(a)$ be the number of directed tours using arc $a \in B(E)$. Let $\mathcal{T}_{v \leftarrow w}(a)$ and $\mathcal{T}_{v \rightarrow w}(a)$ be the set of all tours that serve client $v \in \mathcal{C}$ from facility $w \in \mathcal{F}$ with an occurrence of arc $a \in B(E)$ on the path from w to v or, respectively, from v to w . Accordingly, define variables $x_{v \leftarrow w}(a) = \sum_{T \in \mathcal{T}_{v \leftarrow w}(a)} x(v, T)$ and $x_{v \rightarrow w}(a) = \sum_{T \in \mathcal{T}_{v \rightarrow w}(a)} x(v, T)$. Finally, for each facility $w \in \mathcal{F}$, let $z(w) = 1$ if w is open and $z(w) = 0$ otherwise.

The values $x_{v \leftarrow w}(a)$ and $x_{v \rightarrow w}(a)$ for $a \in B(E)$ can be interpreted as multi-commodity flow with two commodities $v \leftarrow w$ and $v \rightarrow w$ for each pair $v \in \mathcal{C}$ and $w \in \mathcal{F}$, respectively. The first commodity $v \leftarrow w$ corresponds to goods transported from facility w to client v , the second commodity $v \rightarrow w$ emulates the empty truck capacity on the tour returning from v to w . We will now establish several inequalities fulfilled by the triple (x, y, z) .

First observe that the total amount of flow on any arc can at most be the capacity U times the number of tours using the arc, i.e.,

$$\sum_{v \in \mathcal{C}} \sum_{w \in \mathcal{F}} (x_{v \leftarrow w}(a) + x_{v \rightarrow w}(a)) \leq U y(a) \quad \forall a \in B(E). \quad (4.1)$$

Furthermore, we obtain

$$\sum_{v \in \mathcal{C}_i} \sum_{w \in \mathcal{F}} \frac{x_{v \leftarrow w}(a) + x_{v \rightarrow w}(a)}{d(v)} \leq y(a) \quad \forall a \in B(E), i \in [k] \quad (4.2)$$

by observing that the left hand side of the equation is at most 1 per tour that is using the arc: Only one client v in a group is served, only $d(v)$ units are transported to this client in total, and in any tour, each arc occurs either before or after v but never both.

Recall the definition of the excess at node $v \in V$ with respect to the flow x_h of commodity $h \in \{v \leftarrow w, v \rightarrow w : v \in \mathcal{C}, w \in \mathcal{F}\}$ as

$$\text{ex}(x_h, v) := \sum_{a \in \delta^-(v)} x_h(a) - \sum_{a \in \delta^+(v)} x_h(a).$$

By construction of x , flow conservation holds for each commodity at all nodes that neither correspond to the facility nor to the client of the commodity. At every client $v \in \mathcal{C}$, the value of any commodity $v \rightarrow w$ for some $w \in \mathcal{F}$ leaving the client equals the value of $v \leftarrow w$ entering it:

$$\text{ex}(x_{v \rightarrow w}, u) = 0 = \text{ex}(x_{v \leftarrow w}, u) \quad \forall v \in \mathcal{C}, w \in \mathcal{F}, u \in V \setminus \{v, w\} \quad (4.3)$$

$$\text{ex}(x_{v \leftarrow w}, v) = -\text{ex}(x_{v \rightarrow w}, v) \quad \forall v \in \mathcal{C}, w \in \mathcal{F} \quad (4.4)$$

Moreover, as one client from every group needs to be served, the variables fulfill

$$\sum_{v \in \mathcal{C}_i} \sum_{w \in \mathcal{F}} \frac{\text{ex}(x_{v \leftarrow w}, v)}{d(v)} = 1 \quad \forall i \in [k]. \quad (4.5)$$

Finally, for every $i \in [k]$ at most $d(v)$ units of flow are sent from an open facility to one of the clients $v \in \mathcal{C}_i$, and no flow is sent if the facility is not open. This implies

$$\sum_{v \in \mathcal{C}_i} \frac{\text{ex}(x_{v \rightarrow w}, w)}{d(v)} \leq z(w) \quad \forall w \in \mathcal{F}, i \in [k]. \quad (4.6)$$

We conclude that the value of an optimal solution to the group location routing problem is at least the value of an optimal solution of the following LP.

$$\begin{aligned} [\text{G-CLR}_{\text{LP}}] \quad & \min \sum_{a \in B(E)} c(a)y(a) + \sum_{w \in \mathcal{F}} f(w)z(w) \\ \text{s.t.} \quad & x, y, z \text{ fulfill (4.1) -- (4.6)} \\ & x, y, z \geq 0 \end{aligned}$$

Group representatives

Let (x^*, y^*, z^*) be an optimal solution to $[\text{G-CLR}_{\text{LP}}]$ with cost OPT . For $i \in [k]$, let $r_i \in \mathcal{C}_i$ be a client maximizing $\sum_{w \in \mathcal{F}} \text{ex}(x_{v \leftarrow w}^*, v)/d(v)$ over all $v \in \mathcal{C}_i$. We now define the set of *representatives* as $R := \{r_0, \dots, r_k\}$.

Let $L := \max\{|\mathcal{C}_i| : i \in [k]\}$ be the maximum group size. The following inequality will be useful for deriving lower bounds on the optimum value of $[\text{G-CLR}_{\text{LP}}]$.

Lemma 4.16 $L \cdot \sum_{w \in \mathcal{F}} \text{ex}(x_{r_i \leftarrow w}^*, r_i) \geq d(r_i)$ for all $i \in [k]$.

Proof. By applying the pigeon hole principle to (4.5), for every $i \in [k]$ there must be at least one client $v \in \mathcal{C}_i$ with $\sum_{w \in \mathcal{F}} \text{ex}(x_{v \leftarrow w}^*, v)/d(v) \geq \frac{1}{L}$, and thus this inequality holds for r_i in particular. \square

Lower bounds

Now denote the instance of (non-group) CLR defined by replacing \mathcal{C} with the set of representatives R by $\text{CLR}(R)$. Consider the following LP relaxation for the uncapacitated facility location problem arising from $\text{CLR}(R)$ as described in Lemma 4.6. We will use

it to derive a lower bound on the value of an optimal solution to [G-CLR_{LP}].

$$\begin{aligned}
[\text{UFL}_{\text{LP}}(R)] \quad & \min \quad \sum_{v \in R} \sum_{w \in \mathcal{F}} \tilde{c}(v, w) x(v, w) + \sum_{w \in \mathcal{F}} f(w) z(w) \\
\text{s.t.} \quad & \sum_{w \in \mathcal{F}} x(v, w) \geq d(v) && \forall v \in R \\
& x(v, w) \leq d(v) z(w) && \forall v \in R, w \in \mathcal{F} \\
& x, z \geq 0
\end{aligned}$$

Lemma 4.17 $OPT([\text{UFL}_{\text{LP}}(R)]) \leq L \cdot OPT([\text{G-CLR}_{\text{LP}}])$

Proof. Consider the solution (\tilde{x}, \tilde{z}) to $[\text{UFL}_{\text{LP}}(R)]$ obtained by setting $\tilde{z}(w) = L \cdot z^*(w)$ and $\tilde{x}(v, w) = L \cdot \text{ex}(x_{v \leftarrow w}^*, w)$ for all $v \in R, w \in \mathcal{F}$. Observe that by Lemma 4.16

$$\sum_{w \in \mathcal{F}} \tilde{x}(r_i, w) = L \cdot \sum_{w \in \mathcal{F}} \text{ex}(x_{r_i \leftarrow w}^*, r_i) \geq d(r_i)$$

for every $i \in [k]$. Together with (4.6), this immediately implies that (\tilde{x}, \tilde{z}) is a feasible solution to $[\text{UFL}_{\text{LP}}(R)]$.

The flow of each commodity $v \leftarrow w$ ($v \rightarrow w$, respectively) can be decomposed into flow on v - w -paths (w - v -paths, respectively), each of which has length at least $c(v, w)$ by triangle inequality. Combining this with (4.1), we obtain

$$\begin{aligned}
\sum_{a \in B(E)} c(a) y^*(a) & \geq \sum_{a \in B(E)} \sum_{v \in \mathcal{C}} \sum_{w \in \mathcal{F}} c(a) \frac{x_{v \leftarrow w}^*(a) + x_{v \rightarrow w}^*(a)}{U} \\
& \geq \sum_{v \in \mathcal{C}} \sum_{w \in \mathcal{F}} \frac{2}{U} c(v, w) \text{ex}(x_{v \leftarrow w}^*, v) \\
& \geq \frac{1}{L} \sum_{v \in R} \sum_{w \in \mathcal{F}} \tilde{c}(v, w) \tilde{x}(v, w).
\end{aligned}$$

Furthermore, $L \cdot \sum_{w \in \mathcal{F}} f(w) z^*(w) = \sum_{w \in \mathcal{F}} f(w) \tilde{z}(w)$ by construction, which implies $OPT([\text{UFL}_{\text{LP}}(R)]) \leq L \cdot OPT([\text{G-CLR}_{\text{LP}}])$. \square

A second lower bound can be obtained from the LP relaxation of a Steiner tree instance similar to that in Section 4.2.3. Again, consider the graph $G' = (V \cup \{r\}, E \cup E')$ with $E' = \{\{r, w\} : w \in \mathcal{F}\}$ as constructed in Lemma 4.7. We extend the cost function c to E' by defining cost $c_{rw} = \frac{1}{2} f(w)$ for each $w \in \mathcal{F}$. We now consider the undirected cut relaxation of the Steiner tree instance on G' with terminals $R \cup \{r\}$.

$$\begin{aligned}
[\text{ST}_{\text{LP}}(R)] \quad & \min \quad \sum_{e \in E \cup E'} c(e) y(e) \\
\text{s.t.} \quad & \sum_{e \in \delta_{G'}(S)} y(e) \geq 1 && \forall S \subseteq V, S \cap R \neq \emptyset \\
& y(e) \geq 0 && \forall e \in E \cup E'
\end{aligned}$$

Lemma 4.18 $OPT([\text{ST}_{\text{LP}}(R)]) \leq \frac{1}{2} L \cdot OPT([\text{G-CLR}_{\text{LP}}])$

Proof. Consider the solution \tilde{y} of $[\text{ST}_{\text{LP}}(R)]$ obtained by setting

$$\tilde{y}(e) = \frac{1}{2}L \cdot (y^*(a_e^+) + y^*(a_e^-))$$

for all $e \in E$ and setting $\tilde{y}(r, w) = L \cdot z^*(w)$ for all $w \in \mathcal{F}$. Let $S \subseteq V$ with $r_i \in S$ for some $i \in [k]$. By flow conservation and (4.6) we obtain

$$\begin{aligned} \sum_{w \in \mathcal{F} \setminus S} \sum_{a \in \delta_{B(G)}^+(S)} x_{r_i \rightarrow w}^*(a) + \sum_{w \in S} d(r_i)z^*(w) &\geq \sum_{w \in \mathcal{F}} \text{ex}(x_{r_i \leftarrow w}^*, r_i) \quad \text{and} \\ \sum_{w \in \mathcal{F} \setminus S} \sum_{a \in \delta_{B(G)}^-(S)} x_{r_i \leftarrow w}^*(a) + \sum_{w \in S} d(r_i)z^*(w) &\geq \sum_{w \in \mathcal{F}} \text{ex}(x_{r_i \leftarrow w}^*, r_i). \end{aligned}$$

By construction of \tilde{y} and (4.2), the above inequalities yield

$$\begin{aligned} \sum_{e \in \delta_{G'}(S)} \tilde{y}(e) &= \frac{1}{2}L \left(\sum_{a \in \delta_{B(G)}^+(S)} y^*(a) + \sum_{a \in \delta_{B(G)}^-(S)} y^*(a) \right) + \sum_{w \in S} z^*(w) \\ &\geq \frac{L}{2d(r_i)} \left(\sum_{w \in \mathcal{F}} \left(\sum_{a \in \delta_{B(G)}^+(S)} x_{r_i \rightarrow w}^*(a) + \sum_{a \in \delta_{B(G)}^-(S)} x_{r_i \leftarrow w}^*(a) \right) + 2 \cdot \sum_{w \in S} d(r_i)z^*(w) \right) \\ &\geq \frac{L}{d(r_i)} \cdot \sum_{w \in \mathcal{F}} \text{ex}(x_{r_i \leftarrow w}^*, r_i). \end{aligned}$$

This last expression is at least 1 by Lemma 4.16. Thus, \tilde{y} is a feasible solution to $[\text{ST}_{\text{LP}}(R)]$ with

$$\sum_{e \in E \cup E'} c(e)\tilde{y}(e) = \frac{1}{2}L \left(\sum_{a \in B(E)} c(a)y^*(a) + \sum_{w \in \mathcal{F}} f(w) \right) = \frac{1}{2}L \cdot \text{OPT}([\text{G-CLR}_{\text{LP}}]). \quad \square$$

Corollary 4.19 *Let T' be a minimum cost spanning tree in $G'[R \cup \mathcal{F} \cup \{r\}]$ with respect to c' as defined in Lemma 4.7. Then $c'(T') \leq L \cdot \text{OPT}([\text{G-CLR}_{\text{LP}}])$.*

Proof. Let \hat{T} be a minimum cost tree spanning $R \cup \{r\}$ in the metric closure of (G', c) . It is known that a minimum terminal spanning tree is a 2-approximation for the undirected cut formulation of a Steiner tree instance [Vaz01]. Thus, Lemma 4.18 implies that the cost of \hat{T} in the metric closure is bounded by $L \cdot \text{OPT}([\text{G-CLR}_{\text{LP}}])$. For $v \in R$, choose $w(v) \in \mathcal{F}$ minimizing the cost $c(v, w(v)) + \frac{1}{2}f(w)$. Observe that

$$T' = (\hat{T} \cup E' \cup \{\{v, w(v)\} : \{r, v\} \in \hat{T}\}) \setminus \{\{r, v\} : v \in R\}$$

is a spanning tree in $G'[R \cup \mathcal{F} \cup \{r\}]$ and $c'(T')$ is bounded by the cost of \hat{T} in the metric closure of (G', c) . \square

Algorithm 4.4: Approximation algorithm for G-CLR

Compute an optimal solution (x^*, y^*, z^*) to $[\text{G-CLR}_{\text{LP}}]$.
for all $i \in [k]$ **do**
 Let $r_i \in \mathcal{C}_i$ be a client with $\sum_{w \in \mathcal{F}} \text{ex}(x_{v \leftarrow w}^*, v)/d(v)$ maximum over all $v \in \mathcal{C}_i$.
 Set $R = R \cup \{r_i\}$.
Apply Algorithm 4.2 on the CLR instance with clients R .

Algorithm

Lemma 4.17 and Corollary 4.19 immediately yield a 4.38 L -approximation algorithm for G-CLR: Compute an optimal solution to $[\text{G-CLR}_{\text{LP}}]$, obtain a set of representatives R from this solution, and compute an approximation to the resulting instance of CLR using Algorithm 4.2; see Algorithm 4.4 for a formal listing.

Theorem 4.20 *Algorithm 4.4 is a 4.38 L -approximation for G-CLR and it is a 4 L -approximation for G-MCVR, i.e., the case of G-CLR with $f \equiv 0$. It fulfills the single-assignment property. It fulfills the single-tour property if $d(v) \leq U$ for all $v \in \mathcal{C}$.*

Lower bound on approximability

Observing that the approximation guarantee of Algorithm 4.4 depends on the cardinality of the largest group, it is natural to ask whether the group version of CLR is indeed considerably harder than the standard version or whether there is a constant factor approximation whose performance is independent of any instance parameters. We answer this question in the negative by showing that there is no approximation algorithm for G-CLR with a factor better than $\ln(k)$. In fact, the inapproximability result already holds for the special case of G-CLR with unit demands and unit capacity, which corresponds to the group version of metric uncapacitated facility location, as well as for the uncapacitated case considered in [GP08]. It is derived by a straightforward reduction from set cover.

Theorem 4.21 *For any $\gamma < 1$ there is no $\gamma \ln k$ -approximation for G-CLR, unless $\text{NP} \subseteq \text{DTIME}(n^{\mathcal{O}(\log \log n)})$, even when restricted to instances with $d \equiv 1$ and $U = 1$, or instances with $U = \infty$.*

Proof. We give an approximation preserving reduction from set cover to G-CLR. The theorem then follows from the inapproximability result by Feige [Fei98].

Given an instance of set cover with set of ground elements H , set system $\mathcal{S} \subseteq 2^H$, and weights $w \in \mathbb{Q}_+^{\mathcal{S}}$, we construct an instance of G-CLR as follows. For every $S \in \mathcal{S}$, we introduce a facility u_S with opening cost $f(u_S) = w(S)$. For every $h \in H$ and every $S \in \mathcal{S}$ with $h \in S$ we introduce a client v_{hS} with unit demand. We also introduce a client group \mathcal{C}_h for each element $h \in H$ of the ground set and let it contain all clients v_{hS} . Finally, we set $c(v_{hS}, w_{S'}) = 0$, whenever $S = S'$, and to ∞ otherwise.¹

Note that any feasible solution of this G-CLR instance with finite costs corresponds to a feasible solution of the set cover instance with the same costs and vice versa, as there is a one-to-one correspondence between facilities that can serve a client from group \mathcal{C}_h

¹Note that the resulting client-facility distances are actually metric, and the G-CLR instance thus corresponds to an instance of the group version of metric uncapacitated facility location.

with connection cost 0 and sets that contain element h . This is true irrespective of whether $d \equiv 1$ and $U = 1$ or $U \equiv \infty$. Thus, any β -approximation for G-CLR immediately implies a β -approximation for set cover. Further note that $|H|$ is the number of groups in the constructed G-CLR instance. Thus, by [Fei98], we conclude that there is no $\gamma \ln(k)$ -approximation for G-CLR unless $NP \subseteq DTIME(n^{\mathcal{O}(\log \log n)})$. \square

4.2.5 Location routing with cross-docking

In the location routing models discussed in the previous sections, vehicles are loaded exclusively at the depots. While this is a realistic assumption in many applications, in other contexts freight may be shifted from one vehicle to another at intersection points of their tours. This operation of reloading goods between vehicles is known as cross-docking in the transportation literature. In this section, we study a basic cross-docking model, in which tours may start not only at depots but also at clients, and cross-docking operations may be performed at all nodes of the network.

In order to incorporate cross-docking operations into the CLR model, we once again replace the underlying undirected graph by a bidirected one and require the tours of the vehicles to provide sufficient capacity for a flow from the open depots to the clients.

Problem: Capacitated location routing with cross-docking (CLR-CD)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, and a vehicle capacity $U \in \mathbb{Q}_+$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$, a set of directed closed walks (called *tours*) \mathcal{T} in $B(G) = (V, B(E))$ and a flow $x \in \mathbb{Q}_+^{B(E)}$ such that

$$(1) \sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a) = d(v) \text{ for all } v \in \mathcal{C},$$

$$(2) \sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a) = 0 \text{ for all } v \in V \setminus (F \cup \mathcal{C}),$$

$$(3) x(a) \leq U \cdot |\{T \in \mathcal{T} : a \in T\}| \text{ for all } a \in B(E),$$

minimizing the cost $\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

Remark 4.22

- (1) Because the vehicles serve the clients along closed walks, we can assume G to be complete and c to be a metric without loss of generality. Note, however, that we can no longer assume $V = \mathcal{C} \dot{\cup} \mathcal{F}$, as nodes that are neither clients nor facilities can still be used as cross-docks for intersecting tours.
- (2) Given a set of open facilities F and a set of tours \mathcal{T} , a feasible flow x (if one exists) can be found by solving a maximum flow problem.
- (3) The special case of CLR-CD with $U = \infty$ is equivalent to MCVR without cross-docking.

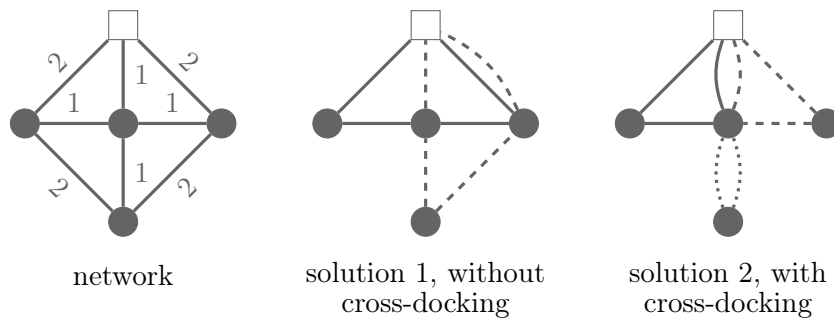


Figure 4.2: A CLR instance and its optimal solutions with and without cross-docking. The numbers on the edges indicate the edge costs. The demand at the central client is 1, the demand at the other clients is 3, the vehicle capacity is 5. The optimal routing scheme in solution 1 without cross-docking has total cost 12. The routing scheme in solution 2 uses cross-docking to consolidate two tours at the central vertex, starting a third tour. Its total cost is 10.

Remark 4.23 Note that the lower bounds established in Section 4.2 for standard CLR remain valid when allowing cross-docking. To see this, observe that the any solution to a CLR-CD instance induces a flow that is valid for the LP relaxation introduced in Section 4.2.4 for G-CLR. Thus, Lemma 4.17 and Corollary 4.19 for $L = 1$ yield the desired UFL and spanning tree lower bounds for the cross-docking variant.

Cost-savings by cross-docking. Cross-docking can reduce transportation cost by improving the utilization of vehicle capacities. The example depicted in Figure 4.2 shows that the ratio of an optimal solution without cross-docking to an optimal solution with cross-docking can be at least 1.2 for some instances. On the other hand, as a result of Remark 4.23, the lower bounds used by Algorithm 4.2 are still valid, and thus the algorithm is a constant factor approximation on CLR-CD that does not use cross-docking. This implies an upper bound on the aforementioned cross-docking ratio.

Theorem 4.24 *Algorithm 4.2 is a 4.38-approximation for CLR-CD.*

Corollary 4.25 *For any instance of CLR-CD, there is a feasible solution without cross-docking with cost at most 4.38 times the cost of an optimal solution with cross-docking.*

Algorithm with improved approximation guarantee

The original algorithm for location routing without cross-docking described in Algorithm 4.2 can be adapted to achieve an improved approximation factor for cross-docking. While every step of the algorithm is modified slightly, the most extensive modification applies to the `relieve` procedure employed in Step 4. It will create two types of tours, *facility tours*, which include a facility directly serving the demand of the clients in the tour, and *cross-dock tours*, which do not contain a facility but intersect with facility tours. See Algorithm 4.5 for a formal listing of the algorithm.

As explained in Remark 4.22 (1), we cannot remove vertices that are neither clients nor facilities from the graph without increasing the cost. However, those vertices will not appear in our algorithm: Step 2 computes a spanning tree only on the clients, the facilities and the artificial root r . As usual, we will denote the closest open facility to a client

v by $w(v)$. Throughout the algorithm, every client v is assigned a residual demand \bar{d} , a representative $\bar{r}(v)$ and a corresponding cost $\bar{c}(v)$. The interpretation of the values is the following. The residual demand $\bar{d}(v)$ keeps track of the remaining demand in the tree $T'[v]$, after some of the subtrees of $T'[v]$ have already been connected to facilities. Step 3 of the algorithm ensures that $\bar{d}(v) < U$ for all clients $v \in \mathcal{C}$ by constructing direct client-facility-tours serving all demand exceeding U . The representative $\bar{r}(v)$ is a client in those vertices of $T'[v]$ that are not yet in a subtree connected to a facility. It is chosen in such a way that $\bar{c}(v) := c(\bar{r}(v), w(\bar{r}(v)))$ is minimized. Initially $\bar{r}(v) = v$.

Relieving overloaded subtrees with cross-docking. As in the original `relieve` procedure described in Algorithm 4.1, we consider a vertex v' such that the tree $T'[v']$ exceeds the capacity but all subtrees rooted at its children obey the capacity. Without loss of generality, we assume v' to be a client as otherwise we can introduce a dummy client vertex between the facility and the remainder of the tree. We define \mathcal{S} to be the set containing all subtrees rooted at children of v' , including the edge incident to v' . We order the sets $S \in \mathcal{S}$ non-decreasingly by the connection cost $\min_{v \in V(S)} \bar{c}(v)$ and define the set of *source trees* \mathcal{S}_+ as the first $\lfloor d(V(T'[v'] \cap \mathcal{C}))/U \rfloor$ elements of \mathcal{S} . The remaining elements $\mathcal{S} \setminus \mathcal{S}_+$ comprise the set of *sink trees* \mathcal{S}_- . Among the vertices in the sink trees, we identify a new representative for v' minimizing \bar{c} . We extract all trees in \mathcal{S} from T' and add them to a collection $\bar{\mathcal{T}}$ of trees that are turned into tours at the end of the algorithm. For each source tree $S \in \mathcal{S}_+$ we furthermore find a client $v_S \in V(S)$ minimizing $\bar{c}(v_S)$ and connect the corresponding representative $\bar{r}(v_S)$ to its closest facility. Note that $\bar{r}(v_S)$ is not necessarily contained in $V(S)$ but can be contained in a previously extracted subtree of $T'[v_S]$. However, we will argue below that connecting $\bar{r}(v_S)$ to a facility induces an additional excess of U units at v' .

Analysis

We first show that the tours constructed by Algorithm 4.5 indeed suffice to serve the demand of every client.

Lemma 4.26 *Algorithm 4.5 constructs a feasible solution*

Proof. We show by induction that the following is true after the algorithm has processed the subtree $T[v']$ in the inner while loop in Step 4: The remaining demand of all clients in $T[v']$ can be either served by sending a flow of $\bar{d}(v')$ on an additional tour to v' or by connecting $\bar{r}(v')$ to its closest facility, in which case an excessive demand of $U - \bar{d}(v')$ can be sent along an additional tour containing v' . This is also true initially for any vertex v in the subtree $T[v']$, which either can provide an excess of $U - \bar{d}(v)$ if it is connected to its facility, or has to be served by $\bar{d}(v)$ units of flow.

To apply the induction step, note that all tours constructed from \mathcal{S} visit v' . Hence, any spare capacity on a facility tour from a source tree can be used to satisfy demands ensuing at v' . Using the induction hypothesis, in total a demand of $\lfloor d(T'[v'])/U \rfloor \cdot U$ can be covered by sending flow along these facility tours and distributing the excess at v' further to sink trees. Now, if the remaining demand $\bar{d}(v')$ is provided by an additional tour visiting v' , also this flow can be forwarded into the sink tours, satisfying all demands in the subtrees by induction hypothesis. If, alternatively, the representative $\bar{r}(v')$ is connected to its closest facility, an additional sink tree can be turned into a source

Algorithm 4.5: Algorithm for CLR-CD.

Step 1: Compute a β_{UFL} -approximate solution \tilde{F} to the UFL instance described in Lemma 4.6. For $v \in \mathcal{C}$, choose $w(v) \in \tilde{F}$ such that $c(v, w(v))$ is minimal.

Step 2: Construct the graph G' with edge costs c' as described in Lemma 4.7. Compute a minimum spanning tree T' in $G'[\mathcal{F} \cup \mathcal{C} \cup \{r\}]$ with respect to c' . Let F' be the set of facilities that are incident to an edge in $T' \cap E$.

Step 3:

Initialize $\bar{d}(v) = d(v)$, $\bar{r}(v) = v$, and $\bar{c}(v) = c(v, w(v))$ for all $v \in \mathcal{C}$.

for all $v \in \mathcal{C}$ **do**

Construct $\lfloor \frac{d(v)}{U} \rfloor$ copies of the tree $\{\{v, w(v)\}\}$ and add them to $\bar{\mathcal{T}}$.
 Set $\bar{d}(v) = d(v) - \lfloor \frac{d(v)}{U} \rfloor \cdot U$.

Step 4:

for all $w \in F'$ **do**

while $\bar{d}(V(T'[w])) > U$ **do**

Let $v' \in V(T'[w])$ such that $\bar{d}(V(T'[v'])) > U$ but $\bar{d}(V(T'[v])) \leq U$ for all children v of v' . Let $\mathcal{S} = \{T'[v] \cup \{v', v\} : v \text{ is a child of } v'\}$.

Order the sets in \mathcal{S} non-decreasingly by $\min_{v \in V(S)} \bar{c}(v)$ and include the first $\lfloor \frac{\bar{d}(V(T'[v']))}{U} \rfloor$ trees in \mathcal{S}_+ . Let $\mathcal{S}_- = \mathcal{S} \setminus \mathcal{S}_+$.

Choose $\bar{v} \in \bigcup_{S \in \mathcal{S}_-} V(S)$ minimizing $\bar{c}(\bar{v})$.

Set $\bar{r}(v') = \bar{r}(\bar{v})$ and $\bar{c}(v') = \bar{c}(\bar{v})$.

Set $\bar{d}(v') = \bar{d}(V(T'[v'])) - \lfloor \frac{\bar{d}(V(T'[v']))}{U} \rfloor \cdot U$.

Set $\bar{d}(v) = 0$ for all $v \in V(T'[v']) \setminus \{v'\}$.

for all $S \in \mathcal{S}$ **do**

Add S to $\bar{\mathcal{T}}$. Remove S from T' .

Let $\phi(v) = S$ for all $v \in V(S)$.

if $S \in \mathcal{S}_+$ **then**

Choose $v_S \in V(S)$ minimizing $\bar{c}(v_S)$.

Add the edge $\{\bar{r}(v_S), w(\bar{r}(v_S))\}$ to $\phi(\bar{r}(v_S)) \in \bar{\mathcal{T}}$.

Add the remainder of $T'[w]$ to $\bar{\mathcal{T}}$.

for all $T \in \bar{\mathcal{T}}$ **do**

Construct a tour visiting $V(T)$ by doubling the edges of T and shortcutting,
 and add it to \mathcal{T} .

Compute a feasible flow x corresponding to $\tilde{F} \cup F'$ and \mathcal{T} .

return $(\tilde{F} \cup F', \mathcal{T}, x)$.

tree. The flow that was sent from v' into this sink tree cancels out and the excess at v' increases by U . This suffices to serving all remaining sink trees, still leaving an excess of $U - \bar{d}(v')$ at v' .

Note that for every vertex v' processed in the while loop, either there is a later iteration in which v' is contained in a source or sink tree of an ancestor v'' of v' or it is contained in the remainder of T' , when the remaining demand is at most U . In either case, the remaining demand of v' is served with an additional tour or the representative of v' is connected to a facility. \square

The modified method of partitioning the tree in Algorithm 4.5 ensures that every edge introduced from the UFL solution is used up to its full capacity. This yields an improved approximation guarantee for the algorithm.

Lemma 4.27 *Let (F, \mathcal{T}, x) be the solution computed by Algorithm 4.5 and let T' be the spanning tree computed in the tree phase and \tilde{F} be the set of open facilities computed in the UFL phase. Then*

$$\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T) \leq 2c'(T') + \sum_{w \in \tilde{F}} f(w) + \sum_{v \in \mathcal{C}} d(v)\tilde{c}(v, \tilde{F}).$$

Proof. At some point in the algorithm, let \bar{E} be the multiset of edges $\{v, w(v)\}$ added to trees so far, including multiple copies for occurrences in multiple trees. We will show that the algorithm maintains the invariant

$$\sum_{e \in \bar{E}} c(e) + \sum_{v \in \mathcal{C}} \frac{\bar{d}(v)}{U} \bar{c}(v) \leq \sum_{v \in \mathcal{C}} \frac{d(v)}{U} c(v, \tilde{F}). \quad (4.7)$$

This in particular implies $c(\bar{E}) \leq \sum_{v \in \mathcal{C}} \frac{d(v)}{U} c(v, \tilde{F})$ at the end of the algorithm, which in turn yields

$$\sum_{T \in \mathcal{T}} c(T) \leq 2c(T') + 2c(\bar{E}) \leq 2c(T') + \sum_{v \in \mathcal{C}} d(v)\tilde{c}(v, \tilde{F})$$

proving the lemma.

Obviously, the invariant is true at the beginning of Step 3, when \bar{E} is empty, $\bar{d} = d$, and $\bar{c}(v) = c(v, \tilde{F})$ for all $v \in \mathcal{C}$. The invariant is maintained throughout Step 3, as $\bar{d}(v)$ is decreased by U for every tree $\{\{v, w(v)\}\}$ introduced to serve v and $c(v, w(v)) = \bar{c}(v)$. To verify that also (4.7) is preserved by any iteration of the inner while loop in the algorithm, let \bar{d}^{old} and \bar{c}^{old} be the residual demands and costs before the iteration, and \bar{d} and \bar{c} be the respective values at the end of the iteration. Observe that $\bar{c}^{\text{old}}(v) = \bar{c}(v)$ for all $v \in \mathcal{C} \setminus \{v'\}$ and $\bar{c}^{\text{old}}(v') \leq \bar{c}(v')$. The iteration changes the left-hand side of (4.7) by

$$\Delta := \sum_{S \in \mathcal{S}_+} \min_{v \in V(S)} \bar{c}(v) + \bar{c}(v') \frac{\bar{d}(v')}{U} - \sum_{v \in V(T'[v'])} \frac{\bar{d}^{\text{old}}(v)}{U} \bar{c}^{\text{old}}(v).$$

We claim that

$$\begin{aligned} & \sum_{v \in V(T'[v'])} \frac{\bar{d}^{\text{old}}(v)}{U} \bar{c}^{\text{old}}(v) \\ & \geq \sum_{S \in \mathcal{S}_+} \sum_{v \in V(S) \setminus \{v'\}} \bar{d}^{\text{old}}(v) \frac{\bar{c}(v)}{U} + \left(\bar{d}^{\text{old}}(v') + \sum_{S \in \mathcal{S}_-} \bar{d}^{\text{old}}(V(S) \setminus \{v'\}) \right) \frac{\bar{c}(v')}{U} \\ & \geq \sum_{S \in \mathcal{S}_+} \min_{v \in V(S)} \bar{c}(v) + \underbrace{(\bar{d}^{\text{old}}(V(T'[v'])) - |\mathcal{S}_+|U)}_{=\bar{d}(v')} \frac{\bar{c}(v')}{U} \end{aligned}$$

which implies $\Delta \leq 0$. The first inequality follows from the fact that $\bar{c}(v') \leq \bar{c}^{\text{old}}(v)$ for every sink tree $S \in \mathcal{S}_-$ and every vertex $v \in V(S)$, which is true by definition of $\bar{c}(v')$. The second inequality follows from $\min_{v \in V(S)} \bar{c}(v) \leq \bar{c}(v')$ for every source tree $S \in \mathcal{S}_+$, which is true by construction of the source trees. Thus, $\Delta \leq 0$ and invariant (4.7) is maintained. \square

Using the 1.5-approximation for UFL given in [BA10] yields the following result.

Theorem 4.28 *Algorithm 4.5 is a 3.5-approximation algorithm for CLR-CD and a 3-approximation for MCVR with cross-docking, i.e., the special case of CLR-CD with $f \equiv 0$.*

4.2.6 Computational study

In Section 4.2.2, we have proven that our polynomial time algorithm for CLR is guaranteed to compute solutions which are at most 4.38 times as expensive as the optimum. In this section, we shall see that the algorithm’s performance in practice exceeds this theoretical worst-case estimate by far. We would like to emphasize that we do not expect our algorithm to compete with (meta-)heuristic approaches without approximation guarantee and polynomial running time. Rather, the question addressed in this computational study is how much solution quality on typical instances needs to be sacrificed in exchange for polynomial running time and a worst case performance guarantee across all instances.

Implementation details. For our experiments, we implemented Algorithm 4.2 with the following minor modifications: First, instead of using the bifactor approximation algorithm from [BA10] in the first step, we implemented the greedy approximation algorithm from [JMM⁺03]. While the latter has a slightly worse approximation guarantee of 1.861, it is purely combinatorial, avoiding randomization and linear programming, and far easier to implement. In this context, note that although the instances in our study are equipped with Euclidian distances, we do not apply the PTAS from [ARR98], which is not tailored for practical use in regards of running time. Moreover, before applying Prim’s algorithm [Pri57] in the second step of the algorithm, we set the opening costs of all facilities opened in the first step to zero. We also close open facilities not used in the final solution. Doing so yields slightly improved results, while it does not interfere with our theoretical analysis of the algorithm. Finally, once the algorithm has computed all tours, we added an option to improve each single tour by solving the corresponding TSP instance using LKH, an implementation of the Lin-Kernighan heuristic by Helsgaun [Hel00].

Observation 4.29 Our implementation of Algorithm 4.2 has an approximation guarantee of 5.722. Its running time is $\mathcal{O}(|\mathcal{C}|^2|\mathcal{F}|)$.

The approximation guarantee results directly from Lemma 4.9 and the approximation factor of the greedy algorithm used for the UFL computation. The running time of the implementation is dominated by that of the UFL algorithm; see [JMM⁺03]. Moreover, experiments in [Hel00] indicate that the practical running time of LKH is quite low (close to quadratic). Our study supports this observation, as the additional running time incurred by a-posteriori tour optimization using LKH turns out to be small—immeasurable on moderately sized instances. Our implementation was done in C++ using GCC 4.5 under SUSE Linux 11.3 and all computations were conducted on an Intel Core2 Duo E8400 processor at 3 GHz with 4 GB RAM.

Instance sets. We report results for two different sets of instances: The first, referred to as the *benchmark set*, comprises 45 instances appearing frequently in the location routing literature [TB99, BFPS07]; see the next section for details. We compare our results for the benchmark instances with those obtained by recent (meta-)heuristic algorithms as

well as best known solutions (BKS) from the literature. While the benchmark instances are moderate in size (20–200 clients, 5–20 facilities), our second test set consists of 27 randomly generated instances which are considerably larger (up to 10000 clients and 1000 facilities). We refer to these latter instances as the *randomized set*.

4.2.7 Benchmark instances

Key properties of the benchmark instances used are listed in Table 4.3. The first 36 instances were introduced by Tuzun and Burke [TB99], the last nine by Barreto et al. [BFPS07]; we will refer to them as sets *TB* and *BFPS*, respectively. While set *TB* is adopted as-is, set *BFPS* contains only those instances introduced in [BFPS07] which have no capacity limits on facilities, as only those mirror the location routing problem addressed here. The best known solutions reported for *TB* were obtained by Prins et al. [PPR⁺07]. For *BFPS*, some proven optima were already reported in [BFPS07], while the remaining instances were solved to optimality by Baldacci, Mingozzi, and Wolfler Calvo [BMW09], as reported in [CCG11].

Table 4.4 contains gaps to BKS and CPU times for our implementation of Algorithm 4.2, with and without a-posteriori optimization of tours using LKH, compared to those of four other algorithms for CLR: a greedy randomized adaptive search procedure (GRASP) proposed by Prins, Prodhon, and Wolfler Calvo [PPW06]; a Lagrangean relaxation granular tabu search (LRGTS) developed by Prins et al. [PPR⁺07]; a two-phase tabu search (TS) studied by Tuzun and Burke [TB99]; and finally an exact branch-and-cut-and-price approach (BCP) proposed by Baldacci, Mingozzi, and Wolfler Calvo [BMW09]. Results for algorithms GRASP and LRGTS are stated in [PPR⁺07] for all 45 benchmark instances, while results for TS and BPS are only available for the instances in *TB* and *BFPS*, respectively.

Note that GRASP, LRGTS, TS, BCP, and our algorithm each were tested on different machines, so the CPU times stated here cannot be compared directly. Since all tests were performed on modern desktop computers, however, we do believe that a comparison of the order of magnitudes of the running times remains feasible.

On average, our approximation algorithm delivers solutions with cost about 19% above the BKS value. This figure improves to 10% when LKH is used to optimize tours a-posteriori. Moreover, the running time of our algorithm is negligible on these instances, regardless of whether LKH is used or not. In comparison, the (meta-)heuristic algorithms GRASP, LRGTS, and TS compute solutions with objective 1–4% above that of BKS on average, while their running times vary strongly from 2–26 seconds (TS) to up to 7 minutes (GRASP). The exact approach BCP is able to find optimal solutions for all instances in *BFPS*, while its running time is naturally very high, needing several hours on some of the instances.

Since gaps to BKS for GRASP and LRGTS are no greater for the instances in *TB* than for those in *BFPS*, where optimality has been proven, it seems reasonable to assume that the gap between BKS and an optimum solution is generally small. In this case, our algorithm vastly outperforms its theoretical approximation guarantee of 5.722. When employing a simple post-optimization step using LKH, it yields solutions within a factor of 1.25 of BKS on all instances, within 1.1 on average. Moreover, its polynomial running time is reflected in very small CPU times on these benchmark instances. When compared to heuristic algorithms, solution quality suffers only by a single-digit percentage on average, while computation times are improved by several magnitudes. Moreover, recall

name	#facilities	#clients	\varnothing demand	vehicle capacity	BKS value
111112	10	100	15.17	150	1468.40
111122	20	100	15.00	150	1449.20
111212	10	100	14.39	150	1396.46
111222	20	100	15.19	150	1432.29
112112	10	100	15.28	150	1167.53
112122	20	100	14.32	150	1102.70
112212	10	100	15.06	150	793.97
112222	20	100	14.73	150	728.30
113112	10	100	14.81	150	1238.49
113122	20	100	15.10	150	1246.34
113212	10	100	14.73	150	902.38
113222	20	100	14.78	150	1021.31
121112	10	200	14.95	150	2281.78
121122	20	200	15.15	150	2185.55
121212	10	200	14.81	150	2234.78
121222	20	200	14.94	150	2259.52
122112	10	200	15.24	150	2101.90
122122	20	200	14.47	150	1709.56
122212	10	200	14.69	150	1467.54
122222	20	200	15.21	150	1084.78
123112	10	200	15.13	150	1973.28
123122	20	200	14.66	150	1957.23
123212	10	200	15.09	150	1771.06
123222	20	200	15.29	150	1393.62
131112	10	150	14.79	150	1866.75
131122	20	150	14.93	150	1841.86
131212	10	150	15.02	150	1981.37
131222	20	150	14.71	150	1809.25
132112	10	150	14.95	150	1448.27
132122	20	150	14.75	150	1444.25
132212	10	150	14.91	150	1206.73
132222	20	150	15.15	150	931.94
133112	10	150	14.95	150	1699.92
133122	20	150	14.93	150	1401.82
133212	10	150	15.18	150	1199.51
133222	20	150	14.91	150	1152.86
Chr69-100x10	10	100	14.58	200	842.90*
Chr69-50x5	5	50	15.54	160	565.60*
Chr69-75x10	10	75	18.19	160	861.60*
Gas67-22x5	5	22	463.14	4500	585.11*
Gas67-29x5	5	29	439.66	4500	512.10*
Gas67-32x5	5	32	917.81	8000	562.20*
Gas67-32x5-2	5	32	917.81	11000	504.30*
Gas67-36x5	5	36	25.00	250	460.40*
Min92-27x5	5	27	311.48	2500	3062.00*

Table 4.3: Properties of benchmark instances and cost of a best known solution (BKS, * denotes proven optimality). The BKS values for the first 36 instances are from [PPR⁺07], those for the last nine from a series of papers [TB99, BFPS07, BMW09].

instance	approx		approx+tsp		GRASP		LRGTS		TS/BCP	
	gap	time	gap	time	gap	time	gap	time	gap	time
111112	0.207	0.00	0.079	0.00	0.039	32.40	0.015	3.30	0.060	5.00
111122	0.235	0.00	0.117	0.00	0.054	40.70	0.016	6.50	0.057	3.00
111212	0.133	0.00	0.043	0.00	0.019	27.60	0.011	4.20	0.034	3.00
111222	0.342	0.00	0.246	0.00	0.035	36.20	0.008	7.40	0.055	4.00
112112	0.164	0.00	0.076	0.00	0.028	27.70	0.017	6.90	0.054	4.00
112122	0.133	0.00	0.095	0.01	0.019	34.30	0.012	6.80	0.027	2.00
112212	0.086	0.00	0.041	0.00	0.025	22.50	0.024	5.20	0.039	3.00
112222	0.119	0.00	0.070	0.00	0.027	37.30	0.020	5.90	0.017	3.00
113112	0.183	0.00	0.090	0.00	0.028	21.50	0.024	4.30	0.063	3.00
113122	0.201	0.00	0.131	0.00	0.021	36.00	0.008	6.30	0.023	4.00
113212	0.140	0.00	0.082	0.00	0.011	20.30	0.012	4.00	0.020	4.00
113222	0.166	0.00	0.126	0.00	0.004	38.40	0.007	4.90	0.023	3.00
131112	0.253	0.01	0.142	0.01	0.075	113.00	0.042	12.50	0.072	12.00
131122	0.230	0.01	0.110	0.01	0.026	161.40	0.018	18.50	0.028	12.00
131212	0.153	0.00	0.067	0.01	0.027	100.00	0.015	11.10	0.021	14.00
131222	0.206	0.01	0.102	0.01	0.026	132.40	0.006	15.80	0.025	13.00
132112	0.163	0.01	0.081	0.01	0.041	117.70	0.000	22.00	0.074	9.00
132122	0.301	0.01	0.230	0.02	0.009	166.10	0.034	28.00	0.024	12.00
132212	0.101	0.01	0.050	0.00	0.028	106.70	0.004	14.60	0.020	9.00
132222	0.170	0.00	0.123	0.01	0.010	142.40	0.005	13.70	0.018	9.00
133112	0.155	0.01	0.098	0.00	0.022	92.80	0.017	17.90	0.037	9.00
133122	0.127	0.01	0.075	0.01	0.017	128.40	0.016	18.50	0.062	9.00
133212	0.128	0.00	0.068	0.01	0.020	88.50	0.014	14.50	0.054	10.00
133222	0.081	0.00	0.029	0.01	0.068	134.90	0.008	14.30	0.026	9.00
121112	0.217	0.01	0.145	0.01	0.055	308.00	0.016	32.60	0.053	22.00
121122	0.139	0.01	0.050	0.02	0.047	410.00	0.010	39.60	0.012	22.00
121212	0.191	0.01	0.105	0.02	0.017	311.40	0.012	32.80	0.024	23.00
121222	0.225	0.02	0.122	0.02	0.042	418.90	0.004	40.20	0.047	26.00
122112	0.145	0.02	0.088	0.02	0.017	338.00	0.009	47.20	0.027	20.00
122122	0.179	0.02	0.125	0.02	0.057	370.00	0.017	59.30	0.045	18.00
122212	0.107	0.01	0.050	0.01	0.020	242.70	0.014	36.70	0.056	18.00
122222	0.119	0.01	0.049	0.00	0.010	308.50	0.005	38.70	0.026	18.00
123112	0.170	0.01	0.081	0.01	0.036	282.80	0.005	41.60	0.042	23.00
123122	0.126	0.01	0.050	0.02	0.068	399.20	0.015	51.80	0.023	20.00
123212	0.183	0.02	0.146	0.02	0.010	199.00	0.009	34.00	0.060	20.00
123222	0.182	0.01	0.134	0.01	0.011	296.30	0.005	43.20	0.015	17.00
Chr69-100x10	0.283	0.00	0.108	0.00	0.022	25.50	0.000	28.20	0.000	13074.7
Chr69-50x5	0.220	0.00	0.079	0.00	0.059	2.30	0.037	2.40	0.000	112.9
Chr69-75x10	0.177	0.00	0.104	0.00	0.000	9.80	0.002	10.10	0.000	3413.5
Gas67-22x5	0.244	0.00	0.021	0.00	0.000	0.20	0.004	0.20	0.000	6.0
Gas67-29x5	0.279	0.00	0.165	0.00	0.006	0.40	0.000	0.40	0.000	178.2
Gas67-32x5	0.245	0.00	0.179	0.00	0.017	0.60	0.040	0.60	0.000	63.4
Gas67-32x5-2	0.205	0.00	0.123	0.00	0.000	0.50	0.001	0.50	0.000	117.9
Gas67-36x5	0.448	0.00	0.094	0.00	0.000	0.80	0.035	0.70	0.000	2.9
Min92-27x5	0.181	0.00	0.115	0.00	0.000	0.40	0.001	0.30	0.000	47.0
∅	0.188	0.01	0.100	0.01	0.026	128.54	0.013	17.96	0.038	11.5
									0.000	1890.69

Table 4.4: Gaps to best known solution and CPU times for various algorithms on benchmark instances. Results for algorithm TS are only available for the first 36 instances, those for BCP only for the last nine; hence they share a column. The last row contains average values, with those for TS (first 36 instances) and BCP (last nine) one above the other.

that this improvement in running time comes in addition to the advantage of having a guarantee on solution quality across all possible instances, including malicious examples where heuristics might perform very poorly. In light of its high efficiency in terms of computation time, our algorithm can also be used to compute feasible start solutions for other search heuristics.

4.2.8 Larger, randomly generated instances

The extremely short running time of our algorithm on benchmark instances, which are all of moderate size, suggests that our algorithm is suitable for larger instances as well. To the best of our knowledge, no instances of CLR which are significantly larger than those in the benchmark set have been solved in the literature; hence, we generated a random test set from three input parameters: size, facility opening cost, and vehicle capacity.

Instances were generated on three base networks of different sizes: M (1000 clients, 100 facilities), L (5000, 500), and XL (10000, 1000). Facility opening costs were drawn uniformly at random from three different ranges: $[0; 100]$, $[100; 200]$, and $[200; 500]$. Vehicle capacities were set to either 9, 100, or 1000, while client demands were drawn uniformly at random from $[0; 10]$ in all cases. Finally, x - and y -coordinates for clients and facilities were drawn uniformly at random from $[0; 100]$, and Euclidean distances $d(i, j) := \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ are used in all instances. Our approach of generating the random instances is similar to the approach of [TB99], except that we did not use clustering. The experimental design, using the same base network with different parameters, allows us to compare the effects of these parameters on solution structure, performance of the algorithm, and quality of the lower bounds derived from the minimum spanning tree and UFL subproblems, respectively.

All possible combinations of the three input parameters yield 27 different instances, which we name by their size, indexed with their choice of facility opening cost and vehicle capacity. E.g., $M_{2,2}$ is an instance with 1000 clients, 100 facilities, facility opening costs in $[100; 200]$, and vehicle capacity 100.

Key properties of the solutions computed by our algorithm, again with and without LKH, together with CPU times are depicted in Table 4.5. The column “lower bound” denotes the better of the two lower bounds arising from the UFL and MST instances as described in Lemmas 4.6 and 4.7. While the minimum spanning tree computed within the algorithm is optimal and can thus be directly used as lower bound, deriving a reasonable UFL lower bound requires more care, as the UFL solutions used in the algorithm are only approximations: For smaller instances (size M), we computed the optimal solution value of the corresponding UFL instances using the mixed integer programming solver CPLEX 12.1 [IBM]. For the instances of size L and XL, where using a MIP solver was not possible, we derived a lower bound by constructing a feasible dual solution from the client bids occurring in the UFL greedy algorithm by [JMM⁺03].

CPU time for the largest instances is at most about twenty minutes. On average, using LKH to optimize tours a-posteriori reduces total cost by about 5%, while increasing CPU time by roughly 10%. Naturally, the effect of using LKH on both solution quality and CPU time is more significant when vehicle capacity is large, i.e., when tours are long. Regarding the lower bounds, we observe that the MST yields stronger bounds for larger vehicle capacities, while the UFL bound is stronger when vehicle capacities are small. On average, our algorithm shows a gap of 61.6% to the corresponding lower bounds when LKH post-optimization is enabled.

name	lower bound	#open fac.	fac. cost	#tours	approx			approx+tsp		
					cost	gap	time	cost	gap	time
M _{1,1}	8800.8 ^O	33	779.4	757	13563.4	0.541	0.47	13478.9	0.532	0.55
M _{1,2}	41249.5 ^O	13	82	58	4111.6	0.961	0.95	3499.05	0.669	1.00
M _{1,3}	2096.3 ^T	8	31.8	10	3343.6	0.595	1.59	2478.9	0.183	1.65
M _{2,1}	12166.6 ^O	18	2157.9	756	18086.2	0.487	0.53	17997	0.479	0.62
M _{2,2}	2288.8 ^O	5	528.2	55	5098.9	1.228	0.92	4468.74	0.952	0.98
M _{2,3}	2151.6 ^T	1	205.2	6	3520.2	0.636	1.62	2620.84	0.218	1.67
M _{3,1}	15432.2 ^O	10	2370.3	756	23008.8	0.491	0.68	22926.8	0.486	0.76
M _{3,2}	2938.7 ^O	3	869.1	55	6012	1.046	1.27	5345.92	0.819	1.32
M _{3,3}	2203.4 ^T	1	414.3	6	3656.8	0.66	0.83	2779.25	0.261	0.88
L _{1,1}	17502 ^D	128	2426.4	3695	32473	0.855	23.39	32325.9	0.847	35.90
L _{1,2}	4607 ^T	47	337.3	272	9433.5	1.048	50.84	8106.1	0.76	59.29
L _{1,3}	4607 ^T	16	42.1	31	7344.7	0.594	120.12	5463.71	0.186	121.95
L _{2,1}	29519.6 ^D	50	6000.5	3694	50380.6	0.707	28.89	50229.7	0.702	41.35
L _{2,2}	5946.5 ^D	10	1163.8	271	13435.5	1.259	65.79	12059.5	1.028	73.81
L _{2,3}	4659.4 ^T	2	306	29	8477	0.819	162.99	6624.78	0.422	165.08
L _{3,1}	38728.3 ^D	31	7293.4	3694	64058.9	0.654	38.3	63905.1	0.65	50.94
L _{3,2}	7515.9 ^D	6	1473	271	15694.9	1.088	89.8	14372.4	0.912	97.81
L _{3,3}	4709.4 ^T	1	409.3	29	8835.3	0.876	210.71	6966.54	0.479	213.44
XL _{1,1}	25449.7 ^D	229	3394.8	7480	48879.5	0.921	136.48	48677.1	0.913	214.23
XL _{1,2}	6494.6 ^T	78	405.1	554	13752.3	1.117	314.81	11872	0.828	369.47
XL _{1,3}	6494.6 ^T	33	52.7	69	10400	0.601	741.08	7754.66	0.194	749.91
XL _{2,1}	46601.8 ^D	82	9264.9	7473	77796.3	0.669	165.57	77580.6	0.665	243.40
XL _{2,2}	9253.7 ^D	17	1752.7	547	20133.7	1.176	383.13	18159.1	0.962	434.58
XL _{2,3}	6550.3 ^T	4	507.8	57	12018.2	0.835	879.48	9296.1	0.419	886.83
XL _{3,1}	60461.3 ^D	48	10593.1	7473	101676	0.682	228.14	101454	0.678	307.12
XL _{3,2}	11838.1 ^D	11	2255.2	547	23304.5	0.969	518.15	21341.5	0.803	570.46
XL _{3,3}	6600.5 ^T	2	610.2	57	13091.1	0.983	1314.86	10389.9	0.574	1322.69
∅	14328.43	32.85	2063.94	1433.41	22651.35	0.833	203.01	21562	0.616	221.03

Table 4.5: Best known lower bounds (T: MST, O: optimal UFL solution, D: dual UFL solution), solution properties, costs, gaps and CPU times of the approximation algorithm with and without TSP post-optimization for randomly generated instances.

While we do not expect the lower bounds to be very close to the optimum solution values, we do not have any other primal solutions to compare with our results on instances of similar size. However, we encourage the authors of other algorithms for CLR to perform experiments on our random test set, which are publically available for download [CLR], and compare their results to ours.

4.3 Facility location with capacitated and length-bounded tree connections

In this section, we study a combination of facility location and network design in which the length of each connection from a client to its closest facility is bounded. Such length bounds play an important role in the design of telecommunication networks. They are used, e.g., to ensure reliability of the connection in case of limited signal strength, or to minimize the transmission delay in real-time applications such as video conferencing. We will show how to incorporate length bounds into an approximation framework for facility location problems where clients are connected to open facilities via capacitated trees.

The particular application motivating our research stems from the planning of opti-

cal access networks. In these so-called fiber-to-the-home or fiber-to-the-curb networks, optical splitters are used to split a single fiber emanating from the central office into a fiber tree serving multiple clients. The main advantage of this technology, compared to using an individual fiber for each client, is a considerable reduction in the total length of fibers in use and the amount of active fiber termination equipment at the central office location. On the other hand, all clients in the same fiber tree share the limited transmission capacity of a single fiber. Therefore, not too many clients may be aggregated into such a tree. Moreover, the optical signal emitted at the central office must fulfill several power and quality requirements when reaching a client, in order to guarantee a reliable connection. These technical requirements imply an upper bound on the path length between the central office and any client within the fiber tree or, in other words, on the depth of the tree.

When planning the deployment of an optical access network, one generally has to decide where to set up central offices and how to connect the clients to these offices via fiber trees. One of the most important objectives of the planning process is to minimize the total network cost, which is comprised by the cost for setting up the offices and the cost for laying out the fibers. Together with the technological constraints discussed above, this results in the following optimization problem.

Problem: UFL with capacitated and length-bounded trees (UFL-CLT)

Input: A graph $G = (V, E)$, a set of clients $\mathcal{C} \subseteq V$, a set of facilities $\mathcal{F} \subseteq V$, opening costs $f \in \mathbb{Q}_+^{\mathcal{F}}$, connection costs $c \in \mathbb{Q}_+^E$, lengths $\ell \in \mathbb{Z}_+^E$, demands $d \in \mathbb{Q}_+^{\mathcal{C}}$, a tree capacity $U \in \mathbb{Q}_+$, and a length bound $L \in \mathbb{Z}_+$.

Task: Find a set of facilities $F \subseteq \mathcal{F}$ and a set of trees \mathcal{T} together with a tree assignment $\phi : \mathcal{C} \rightarrow \mathcal{T}$ such that

- (1) every tree $T \in \mathcal{T}$ is rooted at a facility $w_T \in F$,
- (2) $\sum_{v \in \mathcal{C} : \phi(v)=T} d(v) \leq U$ for all $T \in \mathcal{T}$,
- (3) $\text{depth}_\ell(T, w_T) \leq L$ for all $T \in \mathcal{T}$,

minimizing the cost $\sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

Note that the above problem definition does not require trees to be disjoint. In fact, several trees can share the same open facility.

Bicriteria approximation algorithms. We will study several special cases of UFL-CLT. In order to obtain the most insightful results for each of them, we will devise *bicriteria* approximation algorithms, which approximate length bound and optimal cost at the same time with separate approximation factors. An (α, β) -*approximation algorithm* for UFL-CLT is an algorithm that computes in polynomial time a solution fulfilling (1), (2), and the approximate length bound $\text{depth}_\ell(T, w_T) \leq \alpha L$ for all $T \in \mathcal{T}$, with cost at most βOPT , where OPT denotes the minimum cost of a solution fulfilling (1), (2), and (3).

Before we turn our attention to the approximability of UFL-CLT in Sections 4.3.2 to 4.3.5,

we take a short detour to discuss the closely related subject of shallow-light trees in the next section.

4.3.1 Shallow-light trees

UFL-CLT is closely related to the *shallow-light Steiner tree* problem (SLST), which asks for a Steiner tree with the property that the length of a path from each terminal to the given root obeys the length bound while minimizing the cost. This corresponds to the special case of UFL-CLT where $U = \infty$ and $|\mathcal{F}| = 1$.

Problem: Shallow-light Steiner tree problem (SLST)

Input: A graph $G = (V, E)$, a set of terminals $S \subseteq V$, a root $r \in V$, edge costs $c \in \mathbb{Q}_+^E$, lengths $\ell \in \mathbb{Z}_+^E$, and a length bound $L \in \mathbb{Z}_+$.

Task: Find a tree $T \subseteq E$ spanning $S \cup \{r\}$ such that $\text{depth}_\ell(T, r) \leq L$, minimizing $c(T)$.

In this section, we will give an overview of work related to approximation algorithms for shallow-light trees and discuss the connections of SLST to bounded diameter tree problems and directed Steiner trees via so-called layered graphs.

Related work

The concept of (α, β) -approximation algorithms naturally extends to SLST. Although the problem has been studied extensively by the combinatorial optimization community, its approximability is not well-understood at this point. A significant gap remains between the best known approximation algorithms and results establishing hardness of approximation for SLST. We give an overview of these results and discuss important special cases for which better approximation guarantees are known.

Shallow-light Steiner trees. Marathe et al. [MRS⁺98] studied approximation algorithms for different variants of bicriteria tree problems. Using a matching augmentation technique, they achieved a $(\mathcal{O}(\log |S|), \mathcal{O}(\log |S|))$ -approximation for SLST. On the negative side, a hardness result for the *group Steiner tree problem* by Halperin and Krauthgamer [HK03] implies that SLST does not allow for a $(3 - \varepsilon, \log^{2-\varepsilon} |S|)$ -approximation for any $\varepsilon > 0$, unless $NP \subseteq ZTIME(n^{\text{polylog}(n)})$. While this lower bound does not rule out the possibility of an $(\mathcal{O}(1), \mathcal{O}(1))$ -approximation or a $(1, \log^2 |S|)$ -approximation for the problem, the only improvement on the approximation factor in [MRS⁺98] so far is a parameterized $(\mathcal{O}(p \frac{\log |S|}{\log p}), \mathcal{O}(\frac{\log |S|}{\log p}))$ -approximation for an input parameter p , with $1 \leq p < |S|$, by Kapoor and Sarwat [KS07].²

²Naor and Schieber [NS97] presented what promised to be a $(2, \mathcal{O}(\log |V|))$ -approximation for SLST, using a flow-based LP relaxation. However, their rounding scheme, which employs a decomposition technique by Edmonds, was flawed: the branchings resulting from the decomposition do not fulfill the length bound given in the LP.

Directed Steiner trees. When relaxing the requirement of polynomial running time, an improved approximation guarantee can be achieved using directed Steiner trees. Charikar et al. [CCC⁺99] provided an algorithm for the directed Steiner tree problem that returns an $\mathcal{O}(p^2|S|^{1/p})$ -approximation in time $\mathcal{O}(|S|^{3p})$ for a parameter p . Setting $p = \log |S|$ yields an $\mathcal{O}(\log^2 |S|)$ -approximation in time $\mathcal{O}(|S|^{3 \log |S|})$. Using a condensed layered graph, this result immediately translates into a quasi-polynomial time $(1+\varepsilon, \mathcal{O}(\log^2 |S|))$ -approximation for SLST; see Theorem 4.32 for details.

Restricted shortest paths. If $|S| = 1$, SLST asks for a path of bounded ℓ -length connecting the root with the single terminal. For this special case, known as *restricted shortest path problem*, Hassin [Has92] gave both a $(1, 1+\varepsilon)$ -approximation and a $(1+\varepsilon, 1)$ -approximation. Both approximation schemes run in time polynomial in $1/\varepsilon$ and the input size.

Hop-constrained trees. The special case of SLST with $\ell \equiv 1$ is known as *hop-constrained Steiner tree problem*. For this problem, Kortsarz and Peleg [KP99b] gave a $(1, \mathcal{O}(\log n))$ -approximation, whose running time is polynomial when L is bounded by a constant. For the hop-constrained spanning tree problem with metric costs, Althaus et al. [AFHP⁺05] provided a $(1, \mathcal{O}(\log n))$ -approximation algorithm using tree metrics.

Light approximate shortest-path trees. Finally, for SLST instances with $c = \ell$, Khuller, Raghavachari, and Young [KRY95] devised an algorithm that transforms a given tree T into a tree T' with cost $c(T') \leq (1 + 2/(1 - \alpha))c(T)$ such that the path length of any vertex v to the specified root r in T' is at most α times the length of a shortest v - r -path in the graph. They called such trees *light approximate shortest-path trees* (LAST).

Condensed lengths

Using a standard scaling and rounding technique for constructing approximation schemes, we can ensure the values of ℓ and L to be polynomially bounded by the size of the input, losing only a factor of $1 + \varepsilon$ in the precision of the length bound. We can make use of this compression technique to model SLST as a directed Steiner tree problem in a polynomially sized layered graph, and to establish a relation between shallow light trees and bounded diameter trees.

Lemma 4.30 *Let $G = (V, E)$ be a graph, $\ell \in \mathbb{Z}_+^E$, $L \in \mathbb{Z}_+$, and $\varepsilon > 0$. For all $e \in E$, define $\tilde{\ell}(e) = \left\lfloor \frac{|V|}{\varepsilon L} \ell(e) \right\rfloor$ and let $\tilde{L} = \left\lfloor \frac{|V|}{\varepsilon} \right\rfloor$.*

- If P is a simple path in G with $\ell(P) \leq L$, then $\tilde{\ell}(P) \leq \tilde{L}$.
- If P is a simple path in G with $\tilde{\ell}(P) \leq \tilde{L}$, then $\ell(P) \leq (1 + \varepsilon)L$.

Proof. Let P be a simple path in G .

- If $\ell(P) \leq L$, then $\tilde{\ell}(P) = \sum_{e \in P} \left\lfloor \frac{|V|}{\varepsilon L} \ell(e) \right\rfloor \leq \left\lfloor \frac{|V|}{\varepsilon L} L \right\rfloor \leq \tilde{L}$.
- If $\tilde{\ell}(P) \leq \tilde{L}$, then $\ell(P) \leq \sum_{e \in P} \frac{\varepsilon L}{|V|} (\tilde{\ell}(P) + 1) \leq \frac{\varepsilon L}{|V|} \tilde{L} + \frac{\varepsilon L}{|V|} |P| \leq (1 + \varepsilon)L$. □

The layered graph and directed Steiner trees

Similar to the time-expanded networks for flows over time, we can construct a directed *layered graph* from a given graph with edge lengths and length bound. The shallow-light Steiner tree problem is closely related to the directed Steiner tree problem in the corresponding layered graph; see, e.g., the recent study by Gouveia, Simonetti, and Uchoa [GSU11], who use this connection to obtain stronger integer programming formulations for hop-constrained and diameter-constrained tree problems. We will explore this connection in the context of approximation algorithms.

Layered graph. Given an instance $(G = (V, E), S, r, c, \ell, L)$ of SLST, the corresponding *layered graph* is a directed graph $G_L = (V_L, A_L)$ with vertices V_L , arcs A_L , and costs c_L defined as follows. For every vertex $v \in V$ and every $t \in [L]$, there is a node v_t in V_L . For every edge $e \in E$ with $\psi(e) = \{v, w\}$, and every $t \in [L - \ell(e)]$, there is an arc $(v_t, w_{t+\ell(e)})$ and an arc $(w_t, v_{t+\ell(e)})$, both with cost $c(e)$. Finally, there is also an arc (v_t, v_L) for every terminal $v \in S$ and every $t \in [L - 1]$ with cost 0.

Lemma 4.31 *Given an instance $(G = (V, E), S, c, \ell, L)$ of SLST, the minimum cost of a directed Steiner tree in G_L rooted at r_0 and spanning all terminals v_L for $v \in S$ is equal to the minimum cost of a shallow-light Steiner tree in G . Given a directed Steiner tree T_L in G_L , one can compute in time polynomial in $|T_L|$ a shallow-light Steiner tree T in G with $c(T) \leq c_L(T_L)$.*

Proof. Let T be a shallow light Steiner tree in G . For every edge $e \in T$, let v^e be the one of its end vertices that closer to the root and let w^e be the other end vertex. Let $t_e := \ell(T[r, v_e])$. Note that for every $w \in V(T) \setminus \{r\}$, there is exactly one $e \in T$ with $w^e = w$. Thus, the set $T_L := \{(v_{t_e}^e, w_{t_e+\ell(e)}^e) : e \in T\}$ is an arborescence in G_L . It is rooted at r_0 and spans exactly one copy s_{t_s} of every terminal $s \in S$ with $t_s \leq L$. If $t_s < L$ for a terminal $s \in S$, add $\{s_{t_s}, s_L\}$ to T_L . Then T_L is a directed Steiner tree in G_L spanning all terminals s_L and having cost $c(T)$.

Conversely, let T_L be a directed Steiner tree in G_L . Let $A := \{(v_t, w_{t'}) \in T_L : v \neq w\}$. For an arc $a \in A$ let e_a be the corresponding edge in G and let $B := \{e_a \in E : a \in A\}$. As there is an r_0 - s_L -path in T_L for every terminal $s \in S$, there is an r - s -path of ℓ -length at most L in B . Computing a shortest path tree from r to every $s \in S$ in B with respect to ℓ yields a shallow-light Steiner tree in G spanning S of depth at most L and cost at most $c(A) = c(T_L)$. \square

Combining Lemmas 4.30 and 4.31, we can use the quasi-polynomial time $\mathcal{O}(\log^2 |S|)$ -approximation algorithm for directed Steiner tree in [CCC⁺99] to obtain a corresponding approximation algorithm for SLST.

Theorem 4.32 *For every $\varepsilon > 0$, there is an $\mathcal{O}(|S|^{3 \log |S|})$ -time algorithm that computes a $(1 + \varepsilon, \mathcal{O}(\log^2 |S|))$ -approximate solution for SLST.*

Bounded diameter vs. bounded length

The results in [MRS⁺98], [KS07], and [KP99b] in fact refer to the *bounded diameter Steiner tree problem* (BDST) instead of SLST. In this setting, the bound L affects the ℓ -diameter of the tree rather than its ℓ -depth. However, up to a constant factor for the length bound, the two problems are equivalent with respect approximation.

Lemma 4.33 *If there is an (α, β) -approximation for bounded diameter Steiner tree, then there is a $(2\alpha - 1, \beta)$ -approximation for SLST. This is also true if both problems are restricted to the case $\ell \equiv 1$.*

Proof. Let $(G = (V, E), S, r, c, \ell, L)$ be an instance of SLST and let T be an optimal solution to this instance. Let $G' = (V \cup \{r'\}, E \cup \{e'\})$ for some new vertex r' and new edge e' with $\psi(e') = \{r', r\}$, $\ell(e') = L$, and $c(e') = 0$. Then $T \cup \{\{r', r\}\}$ is a tree feasible solution to the BDST instance defined by $(G', S \cup \{r'\}, c, \ell, 2L)$. Let T' be an (α, β) -approximate solution to this instance. Then $c(T') \leq \beta c(T)$ and

$$\text{depth}_\ell(T' \setminus \{e'\}, r) \leq \text{diam}_\ell(T') - L \leq (2\alpha - 1)L.$$

Thus $T' \setminus \{e'\}$ is a $(2\alpha - 1, \beta)$ -approximate shallow-light Steiner tree. The same reduction also works for the case $\ell \equiv 1$ by splitting the edge e' into L edges. \square

Lemma 4.34 *If there is an (α, β) -approximation for SLST, then for every $\varepsilon > 0$ there is an $((1 + \varepsilon)\alpha, \beta)$ -approximation for bounded diameter Steiner tree. When restricting to instances with L being polynomial in the input size, ε can be chosen to be 0.*

Proof. Consider an instance $(G = (V, E), S, c, \ell, L)$ of BDST and let T^* be an optimal solution. Choose $s, t \in S$ such that $\ell(T^*[s, t])$ is maximum and choose $e^* \in T^*[s, t]$ with $\psi(e^*) = \{v, w\}$ such that $\ell(T^*[s, v]) \leq L/2$ and $\ell(T^*[w, t]) \leq L/2$. Furthermore define $\lambda := L/2 - \ell(T^*[s, v])$. Construct the graph $G^* = (V \cup \{u\}, E \setminus \{e^*\} \cup \{e_v, e_w\})$ by splitting e^* into two edges e_v and e_w with endpoints $\psi(e_v) = \{v, u\}$ and $\psi(e_w) = \{u, w\}$, costs $c(e_v) = c(e)$ and $c(e_w) = 0$, and lengths $\ell(e_v) = \lambda$ and $\ell(e_w) = \ell(e^*) - \lambda$. Let T' be an (α, β) -approximate shallow-light Steiner tree in G^* with respect to costs c , lengths 2ℓ , and length bound L , spanning the terminals S with root u . Observe that $T^{**} = T^* \setminus \{e^*\} \cup \{e_v, e_w\}$ is a feasible solution to the same SLST instance, as $\ell(T^{**}[u, u']) \leq \max\{\ell(T^{**}[u, s]), \ell(T^{**}[u, t])\} \leq L/2$ by maximality of $T^*[s, t]$. This implies $c(T') \leq \beta c(T^*)$. Therefore, $T' \setminus \{e_v, e_w\} \cup \{e\}$ is an (α, β) -approximate solution to the initial BDST instance.

If L is polynomial in the input size, we can compute an (α, β) -approximate SLST for every possible choice of $e^* \in E$ and $0 \leq \lambda \leq \ell(e^*) \leq L$ in polynomial time and choose the cheapest among those trees. If L is not polynomially bounded, we can condense the lengths at the cost of a factor of $1 + \varepsilon$ in the length bound as described in Lemma 4.30. \square

4.3.2 Inapproximability and lower bounds for UFL-CLT

In this section, we give a hardness result on the approximability of UFL-CLT and provide two lower bounds used in the following sections.

Inapproximability

UFL-CLT generalizes several problems that are known to be hard to approximate better than by a polylogarithmic factor, e.g., non-metric UFL or SLST. In fact, the problem remains hard to approximate even in the special case of c and ℓ being proportional to a common metric.

Theorem 4.35 *For any $\varepsilon > 0$, there is no $(3 - \varepsilon, (1 - \varepsilon) \ln |\mathcal{C}|)$ -approximation for UFL-CLT, unless $NP \subseteq DTIME(n^{\mathcal{O}(\log \log n)})$, even when restricting to instances with $c = \ell$ is a metric.*

Proof. The result follows from a simple reduction from (unweighted) set cover. Let (H, \mathcal{S}) be an instance of set cover with ground set H and set system $\mathcal{S} \subseteq 2^H$. We construct an instance of UFL-CLT as follows. For each element $h \in H$, we introduce a client v_h and for every set $S \in \mathcal{S}$, we introduce a facility w_S with opening cost $f(w_S) = M := |H| \ln |H|$. Set $\ell(v_h, w_S) = c(v_h, w_S) = 1$ if $e \in S$ and 3 otherwise. We set $U = \infty$ and $L = 1$. Let OPT be the cost of the resulting UFL-CLT instance. Let $\mathcal{S}^* \subseteq \mathcal{S}$ be a set cover of minimum cardinality. This set cover induces a solution to this UFL-CLT instance by opening all facilities corresponding to sets in \mathcal{S}^* and connecting each client directly to its closest open facility. Hence $OPT \leq M|\mathcal{S}^*| + |H|$.

Now assume by contradiction there is a $(3 - \varepsilon, (1 - \varepsilon) \ln |\mathcal{C}|)$ -approximation algorithm for UFL-CLT and apply it on the constructed instance. As $(3 - \varepsilon)L < 3$, the resulting solution only contains edges $\{v_h, w_S\}$ for which $h \in S$. Accordingly, the set of open facilities F induces a set cover. The cost of the solution is $M|F| + |H| \leq (1 - \varepsilon) \ln |H| OPT$. Hence, the number of sets in this cover is

$$|F| \leq \frac{(1 - \varepsilon) \ln |H| OPT - |H|}{M} \leq (1 - \varepsilon) |\mathcal{S}^*| \ln |H| + \frac{(\ln |H| - 1) |H|}{M}$$

which is bounded by $(1 - \varepsilon) \ln |H| |\mathcal{S}^*|$ for all sufficiently large values of $|H|$. Hence there is a $(1 - \varepsilon) \ln |H|$ -approximation for set cover, implying $NP \subseteq DTIME(n^{\mathcal{O}(\log \log n)})$. \square

Lower bounds

We assume we are given an instance of UFL-CLT. Let OPT be the cost of an optimal solution to this instance. The following two lower bounds are again generalizations of the lower bounds used in [RS06].

Lemma 4.36 For $v \in \mathcal{C}$ and $w \in \mathcal{F}$ define

$$\tilde{c}(v, w) := \frac{1}{U} \min\{c(P) : P \text{ is a } v\text{-}w\text{-path in } G \text{ with } \ell(P) \leq L\}.$$

Let $\tilde{F} \subseteq \mathcal{F}$ be an optimal solution to the UFL instance with facilities \mathcal{F} , clients \mathcal{C} , opening costs f and connection costs \tilde{c} . Then $\sum_{w \in \tilde{F}} f(w) + \sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F}) \leq OPT$.

Proof. Let (F, \mathcal{T}, ϕ) be an optimal solution of the UFL-CLT instance. Interpret F as a solution to the UFL instance with connection cost \tilde{c} . Clearly, $\tilde{c}(v, F) \leq c(\phi(v))/U$ as the tree $\phi(v)$ contains a path from a facility to v of ℓ -length at most L . Thus

$$\sum_{v \in \mathcal{C}} d(v) \tilde{c}(v, F) \leq \sum_{v \in \mathcal{C}} d(v) \frac{c(\phi(v))}{U} = \sum_{T \in \mathcal{T}} \sum_{v \in \mathcal{C} : \phi(v)=T} d(v) \frac{c(T)}{U} \leq \sum_{T \in \mathcal{T}} c(T)$$

where the last inequality follows from the capacity constraint (2). Note that the last term equals the connection cost of the optimal UFL-CLT solution and thus the total cost of the UFL solution F with respect to f and \tilde{c} is at most the total cost of (F, \mathcal{T}, ϕ) with respect to f and c . \square

Lemma 4.37 Let $G' := (V', E')$ with $V' = V \cup \{r\}$ and $E' = E \cup \{e_w : w \in \mathcal{F}\}$ where $\psi(e_w) = \{r, w\}$, $c(e_w) = f(w)$, and $\ell(e_w) = 0$. Let $T \subseteq E'$ be a tree of minimal cost among all trees spanning $\mathcal{C} \cup \{r\}$ with $\ell(T[r, v]) \leq L$ for all $v \in \mathcal{C}$, i.e., T is an optimal shallow-light Steiner tree on G' . Then $c(T) \leq OPT$.

Proof. Let (F, \mathcal{T}, ϕ) be an optimal solution of the UFL-CLT instance. Define the set of edges $S = \bigcup_{T \in \mathcal{T}} T \cup \{e_w : w \in F\}$. This set spans $\mathcal{C} \cup \{r\}$ and contains a path of length L from each client to r . Computing a shortest path tree from r to each $v \in \mathcal{C}$ with respect to ℓ in the graph induced by S yields a tree T' with $\text{depth}_\ell(T', r) \leq L$ and $c(T') \leq c(S) \leq \sum_{w \in F} f(w) + \sum_{T \in \mathcal{T}} c(T)$. \square

4.3.3 Approximation algorithms for general UFL-CLT

In this section, we introduce an algorithmic framework for approximating UFL-CLT similar to the one used in Section 4.2 for location routing. We apply it to the general version of the problem and derive two approximation algorithms. The first runs in polynomial time and approximates both length bound and optimal cost by a logarithmic factor. The second runs in quasi-polynomial time, violating the length bound only by a constant while approximating the cost by a polylogarithmic factor.

Algorithmic framework

The algorithm consists of three main steps. In the first two steps, the facility location and shallow-light tree instances introduced in Lemmas 4.36 and 4.37 are constructed and approximately solved. The two solutions are merged in the third step, using the `relieve` procedure described in Algorithm 4.1. The UFL instance is computed with respect to $(1, 1 + \varepsilon)$ -approximate shortest paths, which are also passed to the `relieve` procedure, each path compressed to a single edge. In a clean up step, compressed edges that are contained in trees returned by the `relieve` procedure are expanded again to paths. The complete algorithm is listed as Algorithm 4.6.

Remark 4.38 Note that the replacement of an edge e in a tree T with the corresponding path P might create cycles. However, these cycles can be removed by replacing the tree T with a shortest path tree with respect to costs ℓ in the graph induced by T from the root of T to any vertex spanned by T . This neither increases the cost of the tree nor the length of any path in the tree.

Lemma 4.39 *Let \mathcal{T} be the set of trees computed by Algorithm 4.6 with $w_T \in V(T)$ being the root of tree $T \in \mathcal{T}$. Then*

$$\text{depth}_\ell(T, w_T) \leq (2\alpha_{\text{SLST}} + 1)L$$

for all $T \in \mathcal{T}$.

Proof. Let T' be the tree computed in Step 1 of the algorithm. Let $T \in \mathcal{T}_w$ for some $w \in \mathcal{F}$. Note that expanding the paths does not increase the ℓ -depth of T . We thus consider T as it is returned from the `relieve` procedure, before expanding the paths. If $T \subseteq T'[w]$, i.e., the `relieve` procedure did not insert an edge from \tilde{E}_w into T , then $w_T = w$ and $\text{depth}_\ell(T, w) \leq \text{depth}_\ell(T'[w], w) \leq \alpha_{\text{SLST}}L$. Otherwise, the `relieve` procedure inserted an edge from $e_v = \tilde{E}_w$ with $\psi(e_v) = \{v, w(v)\}$ into T . Note that $w_T = w(v)$ and let $u \in V(T)$. Then

$$\ell(T[u, w_T]) \leq \ell(T'[u, w]) + \ell(T'[w, v]) + \ell(e_v) \leq (2\alpha_{\text{SLST}} + 1)L$$

where the last inequality follows from the fact that T' is an approximate shallow-light tree and from the construction of e_v . \square

Algorithm 4.6: Algorithm for UFL-CLT**Step 1:**

Construct the UFL instance described in Lemma 4.36 using the $(1, 1 + \varepsilon)$ -approximation for restricted shortest path [Has92]. For $v \in \mathcal{C}$ and $w \in \mathcal{F}$, let P_{vw} be the resulting restricted approximately shortest v - w -path. Compute a β_{UFL} -approximate solution $\tilde{F} \subseteq \mathcal{F}$ to this instance. For $v \in \mathcal{C}$, choose $w(v) \in \tilde{F}$ with $c(P_{vw(v)})$ minimal. Let e_v be an edge with $\psi(e_v) = \{v, w(v)\}$, $c(e_v) = c(P_{vw(v)})$, and $\ell(e_v) = \ell(P_{vw(v)})$.

Step 2:

Construct the graph G' as described in Lemma 4.37 and compute an $(\alpha_{\text{SLST}}, \beta_{\text{SLST}})$ -approximate shallow-light Steiner tree T' in G' spanning $\mathcal{C} \cup \{r\}$. Let $F' = \{w \in \mathcal{F} : e_w \in T'\}$.

Step 3:

for all $w \in F'$ **do**

Let $\tilde{E}_w = \{e_v : v \in V(T'[w]) \cap \mathcal{C}\}$.
 Call `relieve` ($T'[w], \tilde{E}_w, \mathcal{C}, d, U, c$) and obtain trees \mathcal{T}_w and assignments ϕ_w .
 For every $T \in \mathcal{T}_w$, expand the edge $e_v \in T$ to the corresponding path.
 Set $\phi(v) = \phi_w(v)$ for all $v \in V(T[w]) \cap \mathcal{C}$.

return $(\tilde{F} \cup F', \bigcup_{w \in F'} \mathcal{T}_w, \phi)$

The algorithmic framework introduced above can be implemented using different variants of approximation algorithms for the UFL and SLST instances in Steps 1 and 2, resulting in different overall approximation factors and running times.

A polynomial-time $(\mathcal{O}(\log |\mathcal{C}|), \mathcal{O}(\log |\mathcal{C}|))$ -approximation

A natural choice for the UFL approximation of Step 1 is the greedy $\mathcal{O}(\log |\mathcal{C}|)$ -approximation by Hochbaum [Hoc82]. In Step 2, the $(\mathcal{O}(\log |\mathcal{C}|), \mathcal{O}(\log |\mathcal{C}|))$ -approximation for diameter-constrained Steiner trees by Marathe et al. [MRS⁺98] can be used to approximate the SLST instance. Thus, Lemmas 4.4 and 4.39 yield the following theorem.

Theorem 4.40 *Using the greedy UFL approximation [Hoc82] in Step 1 and the (diameter, cost)-algorithm of [MRS⁺98] in Step 2, Algorithm 4.6 computes in polynomial time an $(\mathcal{O}(\log |\mathcal{C}|), \mathcal{O}(\log |\mathcal{C}|))$ -approximate solution for UFL-CLT.*

A quasi-polynomial $(3 + \varepsilon, \mathcal{O}(\log^2 |\mathcal{C}|))$ -approximation

In order to improve the approximation guarantee for the length bound to a constant factor, we employ the quasi-polynomial directed Steiner tree algorithm from [CCC⁺99] in the layered graph in Step 2, as described in Theorem 4.32. By Lemmas 4.4 and 4.39, we get the following approximation ratios.

Theorem 4.41 *For every $\varepsilon > 0$, there is an algorithm computing a $(3 + \varepsilon, \mathcal{O}(\log^2 |\mathcal{C}|))$ -approximate solution for UFL-CLT in time $\mathcal{O}(|\mathcal{C}|^{3 \log |\mathcal{C}|})$.*

Algorithm 4.7: Greedy 3-stretched covering algorithm

```

Initialize  $\mathcal{C}' = \mathcal{C}$ ,  $F = \emptyset$ .
while  $\mathcal{C}' \neq \emptyset$  do
    Let  $v' \in \mathcal{C}'$  and let  $\mathcal{F}_{v'} = \{w \in \mathcal{F} : \ell(v', w) \leq L\}$ .
    Let  $w' \in \mathcal{F}_{v'}$  minimizing  $f(w')$ .
    Set  $F = F \cup \{w'\}$ .
    Set  $\mathcal{C}' = \mathcal{C}' \cap \{v \in \mathcal{C} : \ell(v, w') > 3L\}$ .
return  $F$ 

```

4.3.4 UFL-CLT with length-proportional costs

The approximation algorithms presented in the previous section are designed to work well even when cost and length are mutually independent. In practice, it is often realistic to assume a connection between the two values—the cost of installing a cable usually increases with the distance it covers. Throughout this section, we assume G to be complete and both c and ℓ to be proportional to a common metric on G . Note that we can thus assume $c = \ell$ without loss of generality. For this special case, a modified version of Algorithm 4.6, using the LAST algorithm of Khuller et al. [KRY95] introduced in Section 4.3.1 and a greedy covering, yields a solution that approximates both the length bound and the optimal cost by constant factors. We can adjust the two constants by modifying the parametric approximation factor of the LAST.

μ -stretched covers. Before applying the LAST algorithm, we need to ensure that for each client we have opened a facility whose distance to the client does not exceed the length bound by more than a constant factor. A μ -stretched cover is a set of facilities $F \subseteq \mathcal{F}$ such that for every client $v \in \mathcal{C}$ there is a facility $w \in F$ with $\ell(v, w) \leq \mu L$. When ℓ is metric, a 3-stretched cover whose opening cost is bounded by the opening cost of the optimal solution can be computed by a simple greedy procedure; see Algorithm 4.7.

Lemma 4.42 *A 3-stretched cover F with $\sum_{w \in F} f(w) \leq OPT$ can be computed in time $\mathcal{O}(|\mathcal{C}||\mathcal{F}|)$ using Algorithm 4.7.*

Proof. The running time of $\mathcal{O}(|\mathcal{C}||\mathcal{F}|)$ is obvious. During the course of Algorithm 4.7, a client v is only removed from \mathcal{C}' if $\ell(v, F) \leq 3L$. Thus, when the algorithm terminates, F is a 3-stretched cover. For any facility $w \in F$, let $v(w) \in \mathcal{C}$ be the client that was chosen in the first line of the while loop in the iteration where w was added to F , and let $o(w) \in \mathcal{F}$ be the facility that serves $v(w)$ in a fixed optimal solution. Clearly, $f(w) \leq f(o(w))$ as $o(w) \in \mathcal{F}_{v(w)}$. We now show $o(w) \neq o(w')$ for $w \neq w'$, which implies the claim of the lemma. By contradiction assume $o(w) = o(w')$ for $w \neq w'$. Without loss of generality, w was added to F before w' . Then $o(w) = o(w')$ implies $\ell(v(w'), w) \leq \ell(v(w'), o(w')) + \ell(v(w), o(w)) + \ell(v(w), w) \leq 3L$. Thus $v(w')$ was erased from \mathcal{C}' when w was added to F , contradicting the fact that $v(w')$ was chosen in a later iteration. \square

Remark 4.43 By a simple reduction from set cover, it is also easy to see that there is no polynomial time algorithm computing a $(3 - \varepsilon)$ -stretched cover with cost at most OPT , unless $P = NP$.

Algorithm 4.8: Algorithm for UFL-CLT with length-proportional cost**Step 1:**

Construct a UFL instance with clients \mathcal{C} , facilities \mathcal{F} , demands d , opening cost f and connection cost c/U . Compute an approximate solution $\tilde{F} \subseteq \mathcal{F}$ to this instance using the algorithm in [BA10] with $\gamma = \gamma_\alpha$.

For $v \in \mathcal{C}$, choose $w(v) \in \tilde{F}$ such that $c(v, w(v))$ is minimal.

Step 2:

Construct the graph G' as described in Lemma 4.37 and a β_{ST} -approximate Steiner tree T' in G' spanning $\mathcal{C} \cup \{r\}$.

Let $F' = \{w \in \mathcal{F} : e_w \in T'\}$.

Step 3:

for all $w \in F'$ **do**

Let $\tilde{E}_w = \{\{v, w(v)\} : v \in V(T[w]) \cap \mathcal{C}\}$.
 Call `relieve` ($T[w], \tilde{E}_w, \mathcal{C}, d, U, c$) and obtain trees \mathcal{T}_w and assignments ϕ_w .
 Set $\phi(v) = \phi_w(v)$ for all $\mathcal{C} \in V(T[w]) \cap \mathcal{C}$.

Step 4:

Compute a 3-stretched cover \bar{F} using Algorithm 4.7.

Let $F = \tilde{F} \cup F' \cup \bar{F}$ and $\mathcal{T} = \bigcup_{w \in F'} \mathcal{T}_w$.

Contract F to a single vertex r' .

For $v \in V$, choose $e_v \in E$ with $\psi(e_v) = \{r', v\}$ and $c(e_v)$ minimal.

for all $T \in \mathcal{T}$ **do**

Apply the algorithm of Khuller et al. [KRY95] using T as initial tree and the star $\{e_v : v \in V(T)\}$ as shortest path tree; obtain an $(\alpha, 1 + 2/(\alpha - 1))$ -LAST T^* with respect to c and root r' . Replace T by T^* .

Expand r' to F .

return (F, \mathcal{T}, ϕ)

The algorithm

The first three steps of the algorithm for length-proportional cost resemble those of Algorithm 4.6 with the only difference being the absence of the length bound. We therefore can apply constant factor approximations for the resulting metric UFL and Steiner tree instances in Steps 1 and 2, respectively. After applying the `relieve` procedure, the algorithm computes a 3-stretched cover of facilities that are opened in addition to those stemming from the previous steps. Finally, each tree in the solution computed thus far is processed using the LAST algorithm, ensuring that the distance of each client to the root of its tree is at most α times the distance to the closest open facility (which is at most $3L$), while increasing the cost of the tree by a factor of at most $1 + 2/(\alpha - 1)$. The algorithm is formally described in Algorithm 4.8.

In order to improve the approximation guarantee of the algorithm, the connection and opening costs of the UFL approximation are balanced carefully. As in Section 4.2, we use the bifactor approximation algorithm of Byrka and Aardal [BA10], which takes as additional input a parameter γ , returning a solution whose opening cost is at most γ times the opening cost of an initial LP solution and whose connection cost is at most $1 + 2e^{-\gamma}$ times the connection cost of that LP solution. We denote the optimal choice of γ for a given α by γ_α . It is the unique solution to $(1 + 2/(\alpha - 1))(2 + 4e^{-\gamma}) = \gamma$, as can be derived from the proof of Theorem 4.44.

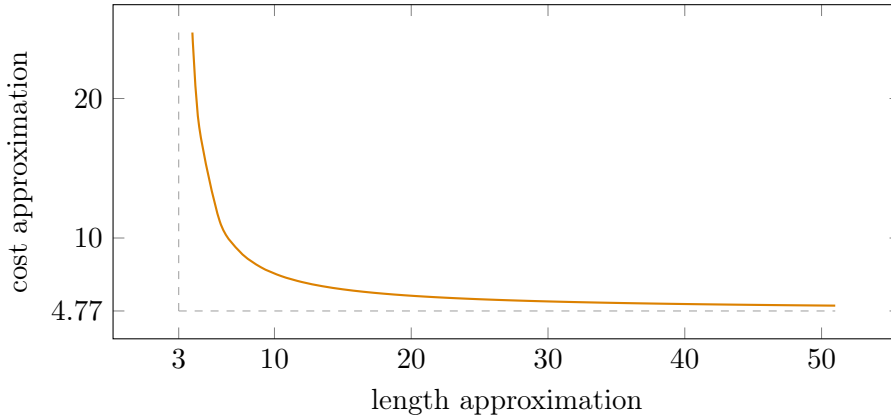


Figure 4.3: Approximation factors achieved by Algorithm 4.8 for UFL-CLT with length-proportional cost, using the Steiner tree approximation by Byrka et al. [BGRS10] with $\beta_{ST} = 1.39$.

Theorem 4.44 *For every $\alpha > 1$, there is a $(3\alpha, (1 + \frac{2}{\alpha-1})\beta_{ST} + \gamma_\alpha + 1)$ -approximation algorithm for UFL-CLT restricted to instances with $\ell = c$ being a metric, where γ_α is the unique value fulfilling the equation $(1 + \frac{2}{\alpha-1})(2 + 4e^{-\gamma_\alpha}) = \gamma_\alpha$ and β_{ST} is the approximation factor of an approximation algorithm for Steiner tree.*

Proof. The approximation ratio of the length bound immediately follows from the fact that \tilde{F} is a 3-stretched cover and thus the shortest path distance from any client to r' in Step 4 is at most $3L$.

By Lemma 4.4, the cost of the solution constructed at the end of Step 3 is at most $\sum_{w \in \tilde{F}} f(w) + c(T') + 2 \sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F})$. Thus, the cost of the final solution is bounded by

$$\underbrace{\sum_{w \in \tilde{F}} f(w)}_{\leq OPT} + \underbrace{\sum_{w \in \tilde{F}} f(w)}_{\leq \gamma_\alpha f_{LP}} + (1 + \frac{2}{\alpha-1}) \left(\underbrace{c(T')}_{\leq \beta_{ST} OPT} + 2 \underbrace{\sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F})}_{\leq (1+2e^{-\gamma_\alpha}) \tilde{c}_{LP}} \right)$$

where f_{LP} and \tilde{c}_{LP} denote the opening cost and connection cost of the solution to the LP relaxation used in the algorithm from [BA10]. By definition of γ_α the cost sums up to the optimal cost times the ratio claimed in the theorem. \square

4.3.5 UFL-CLT with hop constraints

In many applications, the quality of a connection within the network depends on the number of *hops*, i.e., the number of intermediate nodes between the sender and the receiver. In this section, we consider the corresponding special case of UFL-CLT where $\ell \equiv 1$, the graph is complete and c is a metric. We will show how to adapt Algorithm 4.6 so as to approximate the length bound—also known *hop constraint* in this case—with arbitrary precision while still achieving a logarithmic cost approximation and polynomial running time.

The improved guarantee on the hop constraint is achieved by applying two different tree algorithms, depending on the number of hops allowed in the instance. If L is large, we will use a $(1, \mathcal{O}(\log |V|))$ -approximation for the minimum hop-constrained spanning tree

problem with metric costs by Althaus et al. [AFHP⁺05]. However, the transformation of the corresponding lower bound from Steiner tree to spanning tree incurs an increase in the number of hops by an additive constant. We will compensate for this by using a different algorithm for instances where L is small: The $(1, \mathcal{O}(\log |S|))$ -approximation for bounded diameter Steiner trees by Kortsarz and Peleg [KP99b] runs in polynomial time when the number of hops is constant. It also does not require the costs to be metric. The final ingredient is a slight modification of the relieve subprocedure, ensuring that the depth of newly created trees does not exceed that of the original tree.

The spanning tree lower bound. We start by showing that the cost of an optimal UFL-CLT solution can be bounded against twice the minimum cost of a hop-constrained tree spanning exactly the clients and an artificial root node derived from merging all facilities—note that for general length functions, the minimum cost of a length-bounded spanning tree can exceed that of an length-bounded Steiner tree by a factor of $\theta(|V|)$.

Lemma 4.45 *Let $\hat{G} = (\mathcal{C} \cup \{r\}, \hat{E})$ be the complete graph on $\mathcal{C} \cup \{r\}$. Extend the metric c by defining $c(r, v) = \min_{w \in \mathcal{F}} c(v, w) + f(w)$ for all $v \in \mathcal{C}$. Let \hat{T} be an $(L+1)$ -hop spanning tree in \hat{G} with root r of minimum cost. Then $c(\hat{T}) \leq 2OPT$.*

Proof. Let (F, \mathcal{T}, ϕ) be an optimal solution to the UFL-CLT instance. We will modify every tree in \mathcal{T} to ensure it only contains clients and is rooted at r . Without loss of generality, we can assume all leaves of the tree to be clients. Furthermore, if $w_T = w_{T'}$ for the roots of $T, T' \in \mathcal{T}$, then merge T and T' to obtain a single tree rooted at w_T by computing a shortest path tree in $T \cup T'$ with respect to ℓ . Thus, for every $w \in F$, there is only a single tree $T \in \mathcal{T}$ with $w_T = w$.

For $T \in \mathcal{T}$ and a vertex $w \in V(T)$, let $v_{w,T}^{\min}$ be a child of w in T that minimizes $c(v, w)$ among all children v of w . We modify each tree $T \in \mathcal{T}$ by iteratively applying the following change to a vertex $w \in V(T) \setminus \mathcal{C}$. We choose w so as to maximize its distance $|T[w, w_T]|$ from the root; note that this implies $v \in \mathcal{C}$ for all children v of w . If $w = w_T$, then let $u = r$. Otherwise, let u be the parent of w in T . Remove the edges of $\{v_{w,T}^{\min}, w\}$ and $\{w, u\}$ from T and insert $\{v_{w,T}^{\min}, u\}$. Then, for each child $v \in \mathcal{C}$ of w with $v \neq v_{w,T}^{\min}$, replace $\{v, w\}$ by $\{v, v_{w,T}^{\min}\}$ in T . The vertex w is thus removed from the tree. Repeat the procedure until r is the only non-client vertex in T .

Observe that by triangle inequality, every replacement of an edges $\{v, w\}$ by $\{v, v_{w,T}^{\min}\}$ increases the cost of the corresponding tree by at most $c(v_{w,T}^{\min}, w) \leq c(v, w)$. Note that $v_{w,T}^{\min}$ is always a client in such a situation and thus v is involved in such a replacement at most once. A replacement of the edge $\{v_{w,T}^{\min}, w\}$ and $\{w, u\}$ by $\{v_{w,T}^{\min}, u\}$ does not increase the cost of the tree unless $u = r$. If $u = r$, it increases the cost of the tree by at most $f(w_T)$. Note that this happens at most once for every $w \in F$, as there is only one $T \in \mathcal{T}$ with $w_T = w$. Hence, the total increase in cost by all edge replacements is bounded by $\sum_{T \in \mathcal{T}} c(T) + \sum_{w \in F} f(w)$. Furthermore, observe that if $u \neq r$, then the modification does not change the length of any client-root-path in the tree (the subpath $v - w - u$ is replaced by $v - v_{w,T}^{\min} - u$). If $u = r$, then the modification may increase the length of some client-root-paths by 1 (the subpath $v - w_T$ is replaced by $v - v_{w,T}^{\min} - r$).

Thus, for each client $v \in \mathcal{C}$ there is a tree $T \in \mathcal{T}$ with a v - r -path of length at most $L + 1$. Therefore, the set of edges $\bigcup_{T \in \mathcal{T}} T$ contains an $L + 1$ -hop spanning tree T^* of \hat{G} with $c(T^*) \leq 2 \sum_{T \in \mathcal{T}} c(T) + \sum_{w \in F} f(w) \leq 2OPT$. \square

Algorithm 4.9: Algorithm for UFL-CLT with hop constraints.

Step 1:

Construct a metric UFL instance with clients \mathcal{C} , facilities \mathcal{F} , demands d , opening cost f and connection cost c/U . Compute an $\mathcal{O}(\log |\mathcal{C}|)$ -approximate solution $\tilde{F} \subseteq \mathcal{F}$ to this instance using the algorithm from [Hoc82].

For $v \in \mathcal{C}$, choose $w(v) \in \tilde{F}$ such that $c(v, w(v))$ is minimal.

Step 2:

if $L \leq \lceil 1/\varepsilon \rceil$ **then**

Construct the graph G' as described in Lemma 4.37 and compute an $\mathcal{O}(\log |\mathcal{C}|)$ -approximate $(L+1)$ -hop Steiner tree T' in G' spanning $\mathcal{C} \cup \{r\}$ using the algorithm in [KP99b].

else

Construct the graph \hat{G} as described in Lemma 4.45 and compute an $\mathcal{O}(\log |\mathcal{C}|)$ -approximate $(L+1)$ -hop spanning tree T' rooted at r in \hat{G} using the algorithm in [AFHP⁺05]. Replace every edge $\{r, v\}$ in T' by edges $\{r, w\}$ and $\{w, v\}$ for some $w \in \mathcal{F}$ with $c(v, w) + f(w)$ minimal.

Let $F' = \{w \in \mathcal{F} : \{r, w\} \in T'\}$.

Step 3:

for all $w \in F'$ **do**

Let $\tilde{E}_w = \{\{v, w(v)\} : v \in V(T[w]) \cap \mathcal{C}\}$.
 Call `mod_relieve`($T'[w], \tilde{E}_w, \mathcal{C}, d, U, c$) and obtain trees \mathcal{T}_w and assignments ϕ_w .
 Set $\phi(v) = \phi_w(v)$ for all $v \in V(T[w]) \cap \mathcal{C}$.

return $(\tilde{F} \cup F', \bigcup_{w \in F'} \mathcal{T}_w, \phi)$

Modified relieve procedure. Our modification to the original `relieve` procedure as described in Algorithm 4.1 is very subtle. When processing a group of subtrees \mathcal{S}_i rooted at children of a vertex v' , we do not add the edge $e_{v_i} = \{v_i, w(v_i)\}$ to the tree, but instead insert $\{v', w(v_i)\}$. We denote this modified procedure by `mod_relieve`.

Lemma 4.46 *Let \mathcal{T} be the set of trees that is returned by `mod_relieve` when given the tree T' rooted at $w_{T'}$ and edges $\tilde{E} = \{e_v : v \in \mathcal{C}\}$ as input. Then*

$$\text{depth}(T, w_T) \leq \text{depth}(T', w_{T'})$$

for all $T \in \mathcal{T}$. Furthermore,

$$\sum_{T \in \mathcal{T}} c(T) \leq 2c(T') + 2 \sum_{v \in V(T') \cap \mathcal{C}} d(v) \frac{c(e_v)}{U}.$$

Proof. Let $T \in \mathcal{T}$. If $T \subseteq T'[w]$, i.e., none of the edges from \tilde{E} were added to T , the depth of T is at most the depth of T' . In any other case, `mod_relieve` added an edge $\{v', w(v_i)\}$ to T when connecting the subtrees in some group \mathcal{S}_i of subtrees rooted at children of vertex v' . Thus,

$$|T[v, w(v_i)]| = |T[v, v']| + 1 \leq |T'[v, v']| + |T'[v', w_{T'}]| = |T'[v, w_{T'}]|$$

for every $v \in V(T) \cap \mathcal{C}$.

It remains to show the cost bound. Compared to the original procedure described in Algorithm 4.1, the procedure `mod_relieve` inserts edge $\{v', w(v_i)\}$ instead of the edge $\{v_i, w(v_i)\}$ for each tree T_i it creates for group \mathcal{S}_i . Observe that $c(v', w(v_i)) \leq c(v_i, w(v_i)) + c(T'[v_i, v'])$ by triangle inequality. As the paths $T[v_i, v']$ are pairwise disjoint for different groups \mathcal{S}_i , the total cost of the trees in \mathcal{T} increases at most by the cost of the original tree T' compared to the analysis given in Lemma 4.4. \square

Theorem 4.47 *Algorithm 4.9 is a $(1 + \varepsilon, \mathcal{O}(\log |\mathcal{C}|))$ -approximation for UFL-CLT restricted to instances with $\ell \equiv 1$ and c being a metric.*

Proof. By Lemma 4.46, the cost of the solution produced by the algorithm is bounded by

$$\sum_{w \in \tilde{F} \cup F'} f(w) + \sum_{T \in \mathcal{T}} c(T) \leq \sum_{w \in \tilde{F}} f(w) + 2c(T') + 2 \sum_{v \in \mathcal{C}} \tilde{c}(v, \tilde{F}).$$

Using the approximation guarantees of the UFL and tree algorithms, and the lower bounds from Lemmas 4.36, 4.37 and 4.45, we deduce that this is within a factor of $\mathcal{O}(\log |\mathcal{C}|)$ of the optimal solution cost.

If $L \leq \lceil 1/\varepsilon \rceil$, then every tree $T'[w]$ has depth at most L , as T' is an $(L+1)$ -hop tree. If $L > \lceil 1/\varepsilon \rceil$, then every tree $T'[w]$ has depth at most $L+1$, as T' is an $(L+2)$ -hop tree after insertion of the edges $\{r, w\}$ and $\{w, v\}$ for $\{r, v\}$. However, note that in the latter case, $L+1 \leq (1+\varepsilon)L$. By Lemma 4.46, applying `mod_relieve` to these trees does not increase their cost, and thus the hop-constraint is approximated by a factor of $1 + \varepsilon$. \square

4.4 Conclusion

In this chapter, we studied a framework for approximating combined facility location and network design problems. We applied the framework to different variants of capacitated location routing (CLR) and to facility location with capacitated and length-bounded trees (UFL-CLT). For the latter problem, we achieved individual approximation guarantees for maximum connection length and cost, with improved factors in two important special cases. In addition to our theoretical results, a computational study revealed that the actual solution quality achieved by our algorithm for CLR is much closer to optimality than suggested by the theoretical worst-case bounds.

Open problems and future research

We want to close by pointing out several interesting open questions related to our research.

Capacitated facilities. In many practical settings, the amount of demand that can be served from an individual facility is bounded. Such facility capacities are particularly relevant in the case of location routing. Most heuristic algorithms for CLR have been shown to cope well with facility capacities, and also our algorithm has been adapted to work heuristically in this setting [Bod12]. Unfortunately, the situation is much less clear in the context of approximation. Even for the basic *capacitated facility location problem*, many of the established approximation techniques fail; see also Problem 5 in the list of open problems in [WS11]. Still, there exist constant factor approximation

algorithms for capacitated facility location based on local search [KPR98]. Generalizing these results to CLR with facility capacities would be of profound interest for the vehicle routing community. So far, the only known results in this direction are due to Chen and Chen [CC09a, CC09b] for the case of soft-capacitated facilities.

Lower bounds on the approximability of location routing. Our study does not address the issue of lower bounds on the possible approximation factor for basic capacitated location routing. Of course, hardness results for metric UFL or capacitated vehicle routing carry over to CLR. This means that, unless $P = NP$, no approximation factor better than 1.43 in case of splittable demands or 1.5 in case of unsplittable demands can be achieved; see [GK98] and [GW81], respectively. However, a significant gap between these lower bounds and the upper bound of 4.38 established by our results remains. It would be interesting to see a hardness result that combines the complexity of the two problems in order to achieve a stronger inapproximability result.

Uncertainty of demands. In the practical application motivating our work on UFL-CLT, precise client demands are unknown during the early planning phase but can only be estimated roughly. However, fixing location decisions with sufficient lead time can reduce installment costs considerably. Developing approximation algorithms for a generalization of UFL-CLT that incorporates this uncertainty in a two-stage optimization problem is an interesting subject for future research.

Approximability of shallow-light trees. As already indicated in Section 4.3.1, there is a considerable gap between known approximation algorithms for the shallow-light Steiner tree problem (SLST) and the corresponding non-approximability results. The existence of an $(\mathcal{O}(1), \mathcal{O}(1))$ -approximation algorithm or a $(1, \mathcal{O}(\log^2 n))$ -approximation algorithm for SLST remains one of the most important open question in this area.

Chapter 5

Degree-constrained orientations of embedded graphs

In this chapter, we investigate the problem of orienting the edges of an embedded graph in such a way that the in-degrees of both the vertices and faces in the resulting digraph and its induced dual meet given values. We show that the number of feasible solutions is bounded by 2^{2g} , where g is the genus of the embedding, and that all solutions can be determined within time $\mathcal{O}(2^{2g}|E|^2 + |E|^3)$. In particular, in the case of planar embeddings, the solution is unique if it exists, and in general, the problem of finding a feasible orientation is fixed-parameter tractable in g . In sharp contrast to these results, we show that the problem becomes *NP*-complete even for a fixed genus if only upper and lower bounds on the in-degrees are specified instead of exact values.

Publication remark: The results presented in this chapter are joint work with Yann Disser [DM12].

Graph orientation is a special variant of network design that deals with the assignment of directions to the edges of an undirected graph, subject to certain problem-specific requirements. Besides yielding useful structural insights, e.g., with respect to connectivity of graphs [Rob39, NW60] and hypergraphs [FKK03], research in graph orientation is motivated by applications in areas such as evacuation planning [Wol01, RAEP10], graph drawing [EW90, BCG⁺05], or efficient data structures for planar graphs [CE91].

A particularly well-studied class of orientation problems are degree-constrained problems, where the in-degree of each vertex in the resulting digraph has to lie within certain bounds. Hakimi [Hak65] and Frank and Gyárfás [FG76] provided good characterizations¹ for the existence of such orientations. In this chapter, we answer a question raised by András Frank [Fra10], asking for a good characterization for the following problem: Given an embedding of a graph in the plane, is there an orientation of the edges that meets prescribed in-degrees both in the primal and the dual graph at the same time? We show that if such an orientation exists, it is unique and can be computed by combining a feasible orientation for the primal graph with a feasible orientation for the dual graph. Our result generalizes to graph embeddings of higher genus, showing that the number of feasible orientations is bounded by a function of the genus, and the set of all solutions can be computed efficiently as long as the genus is fixed. We also show that the problem becomes *NP*-complete as soon as upper and lower bounds on the in-degrees are specified instead of exact values.

¹A *good characterization* of a decision problem in the sense of Edmonds [Edm65] is a description of polynomially verifiable certificates for both yes- and no-instances of the problem.

Chapter outline. In Section 5.1, we give a short introduction to orientations and embedded graphs. Section 5.2 then deals with the *fixed-degree primal-dual orientation problem*, which asks for an orientation of a given embedded graph, such that exact in-degree prescriptions are met not only for every vertex but also for every face of the embedding. The section contains two different proofs that yield the answer to Frank’s question for a good characterization: Section 5.2.1 comprises a combinatorial proof for the uniqueness of the solution in plane graphs, also reducing the problem to solving the original degree-constrained orientation problem once in the primal and once in the dual graph. In Section 5.2.2, an alternative proof based on a simple linear algebra argument also yields a bound on the number of feasible orientations in embeddings of higher genus. In Section 5.3, we show that if we accept upper and lower bounds on the in-degrees instead of exact values, the problem becomes *NP*-complete.

5.1 Introduction to graph orientation

In this section, we introduce the basic notions of graph orientation and embedded graphs, and give an overview of results in literature related to graph orientation.

5.1.1 Orientations and embedded graphs

Throughout this chapter, we will assume all graphs to be connected but not necessarily simple, i.e., loops and parallel edges are allowed. While the connectedness assumption is very common in the context of graph embeddings, all results presented here can be extended to non-connected graphs by temporarily introducing additional edges—adjusting the in-degree specifications accordingly—so as to render the graph connected.

Orientations

An *orientation* of a graph $G = (V, E)$ is a digraph $D = (V, A)$ such that $A \subseteq B(E)$ contains for every edge $e \in E$ exactly one of the two corresponding arcs $a_e^+, a_e^- \in B(E)$ of the bidirected graph $B(G) = (V, B(E))$ as defined in Section 1.1.3. Given an orientation D , note that $\delta_D^-(v)$ denotes the set of edges that are oriented towards vertex v and that $\delta_D^+(v)$ denotes those edges that are oriented away from v . The *degree-constrained orientation problem* asks for an orientation fulfilling given degree bounds.

Problem: Degree-constrained orientation problem

Input: A graph $G = (V, E)$, and in-degree bounds $\alpha, \beta \in \mathbb{Z}_+^V$.

Task: Find an orientation D of G such that $\alpha(v) \leq |\delta_D^-(v)| \leq \beta(v)$ for all $v \in V$, or prove that there is no such orientation.

The special case of the problem with $\alpha = \beta$ is called *fixed-degree orientation problem*.

Remark 5.1 The fixed-degree orientation problem is equivalent to checking the existence of a b -flow in the digraph $D = (V, A)$ corresponding to an arbitrary orientation of G when

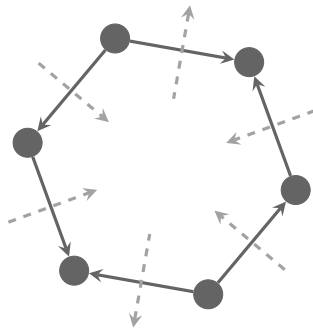


Figure 5.1: Induced orientations of the edges in the dual graph. Primal edges are drawn as solid lines, dual edges are drawn as dashed lines. An edge in the dual graph crosses its corresponding edge in the primal graph from right to left.

defining $b(v) := \alpha(v) - \delta_D^-(v)$ for all $v \in V$. Observe that $x \in \{0, 1\}^A$ is a feasible b -flow if and only if reversing all arcs a with $x(a) = 1$ results in a feasible orientation. Checking the existence of such a b -flow is, in turn, equivalent to the maximum flow problem with unit capacities.

Embedded graphs

An *embedding* of a graph is a mapping of the vertices and edges of the graph onto a closed surface—e.g., a sphere or a torus—such that edges meet only at common vertices. This mapping partitions the surface into several regions, called *faces*. The *dual* of an embedded graph is the graph that is obtained by the following procedure: For every face in the embedding, introduce a vertex in the dual graph. For every edge of primal graph, introduce an edge in the dual graph that connects the faces that are adjacent to the original edge. The genus g of the embedding is determined by Euler’s formula: If E is the set of edges, V is the set of vertices and V^* is the set of faces, then $|V| + |V^*| - |E| = 2 - 2g$.

Planar embeddings. If $g = 0$, i.e., the graph is embedded in a sphere, the embedding is called *planar*—note that embeddings in spheres and planes are combinatorially equivalent. Planar embeddings have several features that make them particularly interesting. In particular, we will make use of the following fact, called *cycle/cut duality*, which was first discovered by Whitney [Whi32] and holds exclusively in planar embeddings: A set of edges is a simple cycle in the primal if and only if it is a simple cut in the dual and vice versa.

Embeddings of digraphs. The concepts of embeddings and graph duality naturally transfer to directed graphs using the convention that arcs in the dual graph are oriented in such a way that they cross their primal “alter egos” from right to left; see Figure 5.1 for an example. In particular, cycle/cut duality extends to planar embeddings of digraphs in the sense that a directed simple cycle in the primal is a directed simple cut in the dual and vice versa.

For a comprehensive introduction to combinatorial embeddings see, e.g., the lecture notes in by Klein [Kle09], which also contain interesting algorithmic techniques exploiting planarity.

Orientations of primal graph and dual graph. By the convention for embeddings of digraphs introduced above, every orientation of the primal graph induces an orientation of the dual graph and vice versa. In accordance with our notation for the primal orientation, we let $\delta_D^-(f)$ be the set of edges whose left face is f , and $\delta_D^+(f)$ be the set of edges whose right face is f .

5.1.2 Related work

Research in graph orientation has a long history that revealed many interesting structural insights and applications. Classic results include the orientation theorem by Robbins [Rob39] stating that an undirected graph is 2-edge-connected if and only if it has an orientation that is strongly connected; see also the generalizations by Nash-Williams [NW60] and Frank, Király, and Király [FKK03]. Graph orientation is also closely related to both graph drawing and network flows. We will discuss the latter connection in more detail below. Before, we will give an overview of results related to degree-constrained orientations and orientations of planar graphs.

Degree-constrained orientations. Hakimi [Hak65] considered the fixed-degree orientation problem. He showed that a feasible orientation exists if and only if $\sum_{v \in V} \alpha(v) = |E|$ and $\sum_{v \in S} \alpha(v) \geq |E[S]|$ for all $S \subseteq V$. He gave similar characterizations for the existence of orientations that fulfill either lower or upper bounds on the in-degrees, i.e., the special cases $\alpha = 0$ or $\beta = 0$ of the degree-constrained orientation problem. Frank and Gyárfás [FG76] observed that the results for lower and upper bounds can easily be combined in a constructive way to find orientations that fulfill upper and lower bounds at the same time. Also optimization versions of the degree-constrained orientation problem have been studied. Gabow [Gab06] considered the problem of finding a subset of edges with maximum cardinality that can be oriented without violating any degree constraints, leaving the other edges unoriented. He derives a $\frac{3}{4}$ -approximation algorithm for this problem, which he also shows to be MAXSNP-hard. Asahiro et al. [AJMO12] investigated a version where a penalty function on the violated degree-bounds is to be minimized. They found that the problem is solvable in polynomial time if the penalty function is convex, but APX-hard in case of concave penalty functions.

Orientations of planar graphs. Orientations of planar graphs received special attention by the research community because they revealed several interesting properties. Based on the insight that every planar graph allows for an orientation with maximum in-degree 3, Chrobak and Eppstein [CE91] designed a highly efficient data structure for adjacency queries in planar graphs. In a distinct line of research, Felsner [Fel04] showed that the set of orientations fulfilling a prescribed in-degree in a planar graph carries the structure of a distributive lattice.

Graph orientation and network flows. Graph orientation is connected to network flows in various ways. Important applications combining the two topics arise in evacuation planning and traffic management, where certain arcs of the network may be reversed in order to enable faster evacuation or to resolve traffic jams; studies in this direction have, e.g., been conducted by Wolshon [Wol01] and Hausknecht et al. [HAS⁺11], respectively. Rebennack et al. [RAEP10] discuss the complexity of the related *contraflow problem*, which asks for which arcs to reverse in order to maximize the flow value. Recently,

Arulselvan, Groß, and Skutella [AGS13] investigated the *price of orientation* for flows over time, i.e., the impact of orienting the edges of the graph on the value of the maximum transshipment or the time needed for satisfying all demands. Finally, as already pointed out in Remark 5.1, the fixed-degree orientation problem, which is also the basis for the problem studied in the next section, is equivalent to the maximum flow problem with unit capacities. This in particular implies that the problem can be solved in time $\mathcal{O}(|E|^{\frac{3}{2}})$ using Dinic's algorithm [Din70]. In the case of planar graphs, this further improves to a time of $\mathcal{O}(|E| \log^3 |E|)$ using the recent multiple-sources multiple-sinks maximum flow algorithm by Borradaile et al. [BKM⁺11].

5.2 Orientations with fixed in-degrees

We consider the problem of finding an orientation that meets given fixed in-degrees for both the vertices and faces of the embedded graph, called the fixed-degree primal-dual orientation problem.

Problem: Fixed-degree primal-dual orientation problem

Input: An embedded graph $G = (V, E)$, in-degree specifications $\alpha \in \mathbb{Z}_+^V$ and $\alpha^* \in \mathbb{Z}_+^{V^*}$.

Task: Find an orientation D of G such that $|\delta_D^-(v)| = \alpha(v)$ for all $v \in V$ and $|\delta_D^-(f)| = \alpha^*(f)$ for all $f \in V^*$, or prove that there is no such orientation.

Primal and dual feasibility. The following notation will be useful throughout the proofs in this section. Given an instance of the fixed-degree primal-dual orientation problem, we say an orientation D is

- *primally feasible* if $|\delta_D^-(v)| = \alpha(v)$ for all $v \in V$.
- *dually feasible* if $|\delta_D^-(f)| = \alpha^*(f)$ for all $f \in V^*$.
- *totally feasible* if it is primally and dually feasible.

The fixed-degree primal-dual orientation problem thus asks for a totally feasible orientation. It is clear that the existence of both a primally feasible orientation and a dually feasible orientation is necessary for the existence of a totally feasible orientation. However, it can easily be checked that this is not sufficient.

Example 5.2 Consider a planar graph with two vertices and two parallel edges connecting them, and let $\alpha(v) = 1$ and $\alpha^*(f) = 1$ for all $v \in V$ and $f \in V^*$. While orienting both edges in opposite directions in the primal graph is primally feasible, orienting them in the same direction—which corresponds to orienting them in opposite directions in the dual graph—is dually feasible. However, neither of the orientations is totally feasible.

In this section, we will present two approaches for obtaining necessary and sufficient conditions for the existence of a totally feasible orientation.

5.2.1 A combinatorial approach for planar embeddings

Using the duality of cycles and cuts in planar graphs yields a combinatorial proof for the uniqueness of a feasible solution to the fixed-degree primal-dual orientation problem in the case of planar embeddings. We will also show how to construct a totally feasible solution from an orientation that is feasible in the primal graph and an orientation that is feasible in the dual graph.

Rigid edges. Consider a subset $S \subseteq V$ with $\sum_{v \in S} \alpha(v) = |E[S]|$. Observe that each edge in $E[S]$ contributes 1 to the in-degree of a vertex in S , no matter how it is oriented, and thus all edges $\delta(S)$ must be oriented from S to $V \setminus S$ in all primally feasible orientations. We call edges whose orientation is fixed in this way *primally rigid*² and denote the set of all primally rigid edges by R . Analogously, we define the set of *dually rigid* edges R^* as those that are fixed for all dually feasible orientations due to a tight set $S^* \subseteq V^*$ of faces with $\sum_{f \in S^*} \alpha^*(f) = |E[S^*]|$. It is easy to check that an edge is primally rigid if and only if it is on a directed cut in the primal graph with respect to any primally feasible orientation.³ Likewise, an edge is dually rigid if it is on a directed cut in the dual graph with respect to any dually feasible orientation. Note that this also implies that the set of edges on directed cuts is invariant for all feasible orientations.

Our main result in this section follows from this characterization of rigid edges and the duality of cycles and cuts in planar graphs.

Theorem 5.3 *In case of a planar embedding, there exists a totally feasible orientation if and only if the following three conditions are fulfilled.*

- (1) *There exists both a primally feasible orientation D and a dually feasible orientation D^* .*
- (2) *The edge set can be partitioned into primally and dually rigid edges, i.e., $E = R \dot{\cup} R^*$.*
- (3) *The orientation obtained by orienting all primally rigid edges in the same direction as they are oriented in D and all dually rigid edges in the same orientation as they are oriented in D^* is totally feasible.*

If it exists, the solution is unique.

Proof. The sufficiency of the conditions is trivial, as the third condition requires the existence of a totally feasible orientation.

In order to show necessity, assume there exists a totally feasible orientation D_0 . As D_0 is both primally and dually feasible, it fulfills condition (1) of the theorem. An edge is primally rigid if and only if it is on a directed cut with respect to D_0 in the primal graph. It is dually rigid if and only if it is on a directed cut in the dual graph. Thus, by cycle/cut duality of planar graphs, an edge is dually rigid if and only if it is on a directed cycle in the primal graph. As every edge in the primal graph is either on a directed cut or on a directed cycle, the sets of primally and dually rigid edges comprise a partition of E , proving condition (2). Now, let D be a primally feasible orientation and D^* be a dually

²The term “rigid” for edges that are oriented in an identical way in all feasible orientations was introduced by Felsner [Fel04].

³Recall that a cut $\delta(S)$ in a digraph is directed if $\delta_D^+(S) = \emptyset$ or $\delta_D^-(S) = \emptyset$.

feasible orientation. As D_0 equals D on all primally rigid edges and equals D^* on all dually rigid edges, the construction described in condition (3) yields D_0 and is feasible.

As all edges are either primally or dually rigid, they must have the same orientation in all totally feasible solutions, and D_0 is unique. \square

Note that the totally feasible solution constructed in the third condition does not depend on the choice of D and D^* . As primally and dually feasible solutions can be found in polynomial time, and rigid edges can be identified by determining the strongly connected components with respect to D and D^* , respectively, Theorem 5.3 yields a polynomial time algorithm for solving the problem for planar embeddings.

Corollary 5.4 *The fixed-degree primal-dual orientation problem for planar embeddings can be solved in time $\mathcal{O}(|E| \log^3 |E|)$.*

Proof. By Theorem 5.3, the problem can be solved by computing a primally feasible solution and a dually feasible solution and identifying the corresponding rigid edges. Both a primally feasible orientation and a dually feasible orientation can be found by solving the corresponding maximum flow problems in the primal graph and in the dual graph using the multiple-sources multiple-sink planar maximum flow algorithm by Borradaile et al. [BKM⁺11], which runs in time $\mathcal{O}(|E| \log^3 |E|)$. Note that identifying directed cuts is equivalent to identifying strongly connected components, which can be done in time $\mathcal{O}(|E|)$. \square

5.2.2 A linear algebra analysis for general embeddings

The fixed-degree primal-dual orientation problem can be formulated as a system of linear equalities over binary variables. To this end, we fix an arbitrary orientation $D = (V, A)$ of the graph and introduce for every arc $a \in A$ a decision variable $x(a)$ that determines whether the orientation of the arc should be reversed (if it is 1) or not (if it is 0) in order to become totally feasible. The vector $x \in \{0, 1\}^A$ yields a feasible orientation if and only if it satisfies the following system of equalities:

$$\sum_{a \in \delta_D^+(v)} x(a) - \sum_{a \in \delta_D^-(v)} x(a) = \alpha(v) - |\delta_D^-(v)| \quad \forall v \in V \quad (5.1)$$

$$\sum_{a \in \delta_D^+(f)} x(a) - \sum_{a \in \delta_D^-(f)} x(a) = \alpha^*(f) - |\delta_D^-(f)| \quad \forall f \in V^* \quad (5.2)$$

The matrix corresponding to the equalities (5.1) is the incidence matrix of the primal graph, while the matrix corresponding to the equalities (5.2) is the incidence matrix of the dual graph. As we assume the graph to be connected, we know that the rank of the former matrix is $|V| - 1$, while the rank of the latter matrix is $|V^*| - 1$. Using the fact that the boundary of a face is a closed walk in the primal graph, it is easy to see that the rows of the first matrix are orthogonal to the rows of the second matrix. This implies that all feasible solutions are contained in a subspace of \mathbb{R}^A of dimension $|E| - |V| - |V^*| + 2 = 2g$.

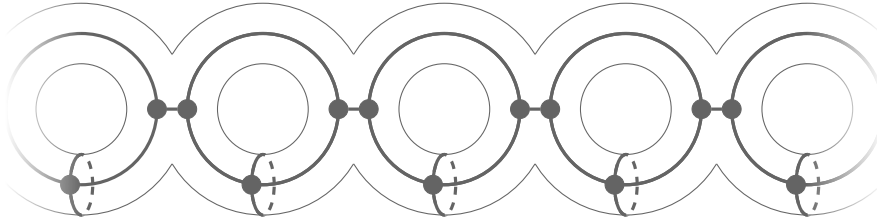


Figure 5.2: Construction of an instance with 2^{2g} feasible orientations, showing the tightness of the bound in Theorem 5.5. The base graph consists of two cycles of length 3 intersecting in a common vertex and is embedded in a torus. Examples of genus g are obtained by introducing g copies of the base graph.

Theorem 5.5 *There are at most 2^{2g} distinct solutions to the fixed-degree primal-dual orientation problem. The set of all totally feasible orientations can be determined in time $\mathcal{O}(2^{2g}|E|^2 + |E|^3)$. The bound on the number of orientations is tight, i.e., there are embedded graphs of genus g that allow for 2^{2g} distinct orientations.*

Proof. By basis augmentation, there is a set $A' \subseteq A$ of $2g$ arcs such that adding the equalities $x(a) = b(a)$ with $b(a) \in \{0, 1\}$ for all $a \in A'$ to the system (5.1) and (5.2) results in a system with full rank, i.e., it has at most one solution. If for some $b \in \{0, 1\}^{A'}$ the unique solution exists and is a 0-1-vector, it corresponds to the unique totally feasible orientation that orients the edges of A' according to the values $b(a)$. Otherwise, there is no such totally feasible orientation. Thus, solving the equality system for all $|\{0, 1\}^{A'}| = 2^{2g}$ possible values of b yields all possible solutions to the fixed-degree primal-dual orientation problem. This takes time $\mathcal{O}(|E|^3)$ for inverting the $|E| \times |E|$ -matrix and $\mathcal{O}(2^{2g}|E|^2)$ for multiplying the 2^{2g} distinct right hand side vectors.

To see that the bound on the number of orientations is tight, consider the example depicted in Figure 5.2. The example is constructed from a base graph consisting of a cycle of length 3 with vertices a, b, c and an additional loop at vertex c . The base graph is embedded in a torus, thus featuring only a single face f . When setting $\alpha^*(f) = |E| = 4$, any orientation is dually feasible as all dual edges are loops. We set the in-degree specifications to $\alpha(a) = \alpha(b) = 1$ and $\alpha(c) = 2$. Now, an orientation of the base graph is primarily feasible if and only if the edges of the cycle are all oriented in the same direction. As the cycle and the loop can be oriented independently, the base graph has 4 feasible orientations.

Examples of higher genus can be obtained by introducing g copies of the embedding described above. The graphs are joined via an edge from vertex b_i to a_{i+1} for every $i \in \{1, \dots, g-1\}$. The resulting embedding has $3g$ vertices and $5g-1$ edges, and it still has only a single face. We increase the in-degree specifications of each base graph by setting $\alpha(a_{i+1}) = 2$ for $i \in \{1, \dots, g-1\}$, so that the new edges joining the copies have to be oriented from copy i to copy $i+1$. The in-degree specification of the face is set to $|E| = 5g-1$. Now each copy of the base graph still has its 4 feasible orientations, so in total there are 4^g feasible orientations.⁴ \square

⁴Note that while the primal graph in the construction described in the proof could also be embedded in a plane, an example where g is the actual genus of the graph can be constructed by introducing additional vertices and edges.

5.3 Orientations with upper and lower bounds

A generalization of the fixed-degree primal-dual orientation problem asks for an orientation that fulfills upper and lower bounds on the in-degrees of vertices and faces instead of attaining fixed values. We show that this problem is *NP*-complete, even when restricted to instances with embeddings of a fixed genus as, e.g., planar embeddings.

Problem: Degree-constrained primal-dual orientation problem

Input: An embedded graph $G = (V, E)$, in-degree bounds $\alpha, \beta \in \mathbb{Z}_+^V$ and $\alpha^*, \beta^* \in \mathbb{Z}_+^{V^*}$.

Task: Find an orientation D of G such that $\alpha(v) \leq |\delta_D^-(v)| \leq \beta(v)$ for all $v \in V$ and $\alpha^*(f) \leq |\delta_D^-(f)| \leq \beta^*(f)$ for all $f \in V^*$, or prove that there is no such orientation.

In order to show the *NP*-completeness of the problem, we use a reduction from *planar 3-SAT*, which was shown to be *NP*-complete by Lichtenstein [Lic82].

Problem: Planar 3-SAT

Input: A set of n variables $V = \{v_1, \dots, v_n\}$ and a set of m clauses $C = \{C_1, \dots, C_m\}$, each containing exactly three literals over V , such that the bipartite graph $G_{3\text{SAT}} = (V \cup C, E)$ with edges $E = \{\{v_i, C_j\} : C_j \text{ contains a literal of } v_i\}$ is planar.

Task: Find a truth assignment for the variables in V such that all clauses in C are satisfied.

Theorem 5.6 *The degree-constrained primal-dual orientation problem is NP-complete, even when restricted to embeddings with a fixed genus.*

Proof. An orientation that solves the degree-constrained primal-dual orientation problem can easily be verified in polynomial time. Hence, it remains to show that the problem is *NP*-hard. It is sufficient to do this for planar graphs. We use a reduction from planar 3-SAT. In the following, we let $G_{3\text{SAT}}$ denote a fixed embedding of the planar graph corresponding to a given instance of planar 3-SAT. We proceed to construct an instance $(G, \alpha, \beta, \alpha^*, \beta^*)$ of the degree-constrained primal-dual orientation problem that has a solution if and only if the instance of planar 3-SAT has a solution. The construction consists of three parts: a *variable gadget* for each variable in $G_{3\text{SAT}}$, a *clause gadget* for each clause in $G_{3\text{SAT}}$, and an *edge gadget* for each edge in $G_{3\text{SAT}}$, connecting a clause and a variable gadget.

For each variable v_i of degree $d_i = |\delta_{G_{3\text{SAT}}}(v_i)|$ in $G_{3\text{SAT}}$, we introduce a variable gadget; see Figure 5.3 for a depiction. The gadget consists of a cycle of length $2d_i$, and we refer to the vertices in this cycle as $v_i^{1,\text{T}}, v_i^{1,\text{F}}, v_i^{2,\text{T}}, \dots, v_i^{d_i,\text{T}}, v_i^{d_i,\text{F}}$. The cycle induces a single face which we call f_i . We set $\alpha^*(f_i) = 0$ and $\beta^*(f_i) = 2d_i$. For now, in order to understand the idea behind the variable gadget, we set $\alpha(v) = \beta(v) = 1$ for every $v \in \{v_i^{1,\text{T}}, \dots, v_i^{d_i,\text{F}}\}$, but we will change this when extending the construction

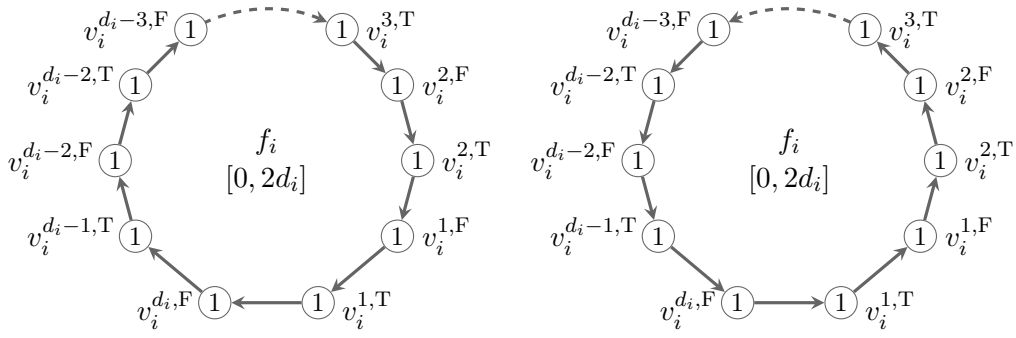


Figure 5.3: Illustration of the variable gadget for a variable v_i , which has d_i occurrences in clauses. The gadget admits only the depicted orientations, the one on the left is interpreted as v_i being ‘true’ and the other as v_i being ‘false’.

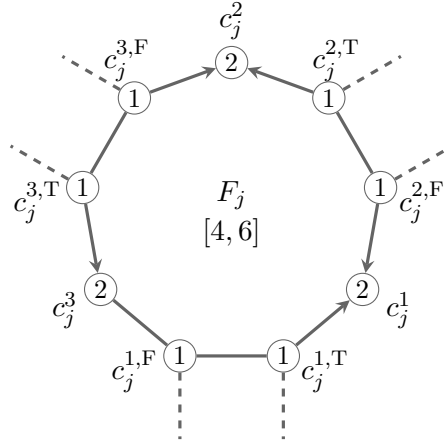


Figure 5.4: Illustration of the clause gadget for a clause C_j . All directed edges are rigid, and the orientation of each of the three remaining edges represents a truth assignment to a literal of the clause. At least one of these three edges needs to be oriented counter-clockwise with respect to F_j . The dashed edges belong to edge gadgets connected to the clause gadget.

later. Let us analyze the construction so far. Since every vertex requires an in-degree of exactly 1, all edges of the cycle need be oriented the same with respect to f_i , i.e., only two orientations of the gadget are permitted. We interpret each of the two possible orientations as a truth assignment for the variable v_i , depending on the direction of the edges between $v_i^{k,T}$ and $v_i^{k,F}$ for $k \in \{1, \dots, d_i\}$. Directing the edge towards $v_i^{k,T}$ is interpreted as setting v_i to ‘true’, and directing it towards $v_i^{k,F}$ is interpreted as setting v_i to ‘false’.

For each clause C_j in G_{3SAT} we introduce a clause gadget that is a cycle with nine vertices $c_j^1, c_j^{1,F}, c_j^{1,T}, c_j^2, c_j^{2,F}, c_j^{2,T}, c_j^3, c_j^{3,F}, c_j^{3,T}$ enclosing a face F_j ; see Figure 5.4 for a depiction. We set $\alpha(c_j^\ell) = \beta(c_j^\ell) = 2$ for $\ell \in \{1, \dots, 3\}$ and $\alpha^*(F_j) = 4$ and $\beta^*(F_j) = 6$. We set the lower and upper bounds for the remaining vertices to 1, and remark that there will be one additional edge incident to each of these vertices in the final construction. For now, observe that any valid orientation has to direct the edges incident to c_j^1, c_j^2, c_j^3 towards these vertices. Each of the three remaining edges can be oriented either way, provided that at least one is in counter-clockwise orientation relative to the face F_j . For each $\ell \in \{1, \dots, 3\}$, the edge between $c_j^{\ell,F}$ and $c_j^{\ell,T}$ will determine whether

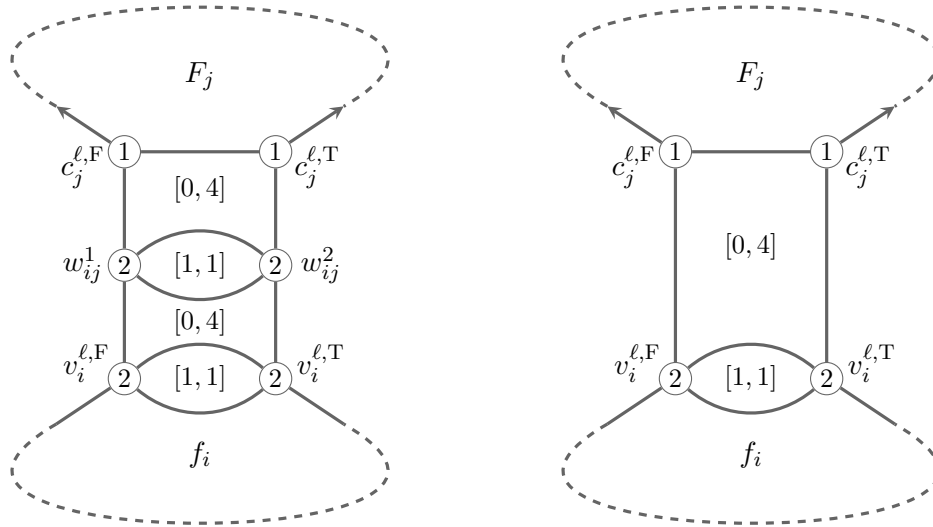


Figure 5.5: Illustration of the edge gadget for an edge connecting variable v_i with clause C_j . The gadget on the left is used when v_i appears in a positive literal in C_j , and the one on the right is used when v_i appears in a negative literal. In both cases, the orientation of the edges in the variable gadgets induces an orientation in of the edge in the clause gadget that corresponds to the value of the literal.

the corresponding literal of C_j is false or true. If it is directed from $c_j^{\ell,T}$ to $c_j^{\ell,F}$, the corresponding literal is considered ‘false’, otherwise it is considered ‘true’. In these terms, our construction enforces that at least one literal of C_j has to be ‘true’.

So far, we have provided a construction for each variable that can be oriented in two ways only, and we have given an interpretation of this orientation as a truth assignment to the variable. Also, we have provided a construction for each clause together with an interpretation of each valid orientation as a truth assignment to the literals of the clause. What remains is to show how to connect the two constructions in a way that guarantees consistency of the truth assignments to variables and literals. To this end, we introduce an edge gadget for each edge $e_{ij} = \{v_i, C_j\}$ in G_{3SAT} between variable v_i and clause C_j as follows; see Figure 5.5 for an illustration. We assume a fixed counter-clockwise ordering of the edges at each vertex in the embedding of G_{3SAT} . Suppose that e_{ij} is the k -th edge at v_i and the ℓ -th edge at C_j with respect to this ordering. We introduce an additional edge between $v_i^{k,T}$ and $v_i^{k,F}$ and set $\alpha^*(f) = \beta^*(f) = 1$ for the new face f enclosed by the two parallel edges. We reassign $\alpha(v_i^{k,T}) = \beta(v_i^{k,T}) = \alpha(v_i^{k,F}) = \beta(v_i^{k,F}) = 2$. The remaining construction depends on whether v_i appears in a positive or negative literal in C_j . If v_i appears in a positive literal, we add two vertices w_{ij}^1, w_{ij}^2 connected by two parallel edges with $\alpha^*(f) = \beta^*(f) = 1$ for the induced face f . We add the edges $\{v_i^{k,F}, w_{ij}^1\}, \{w_{ij}^1, c_j^{\ell,F}\}, \{v_i^{k,T}, w_{ij}^2\}$, and $\{w_{ij}^2, c_j^{\ell,T}\}$, which yields two additional faces f_1, f_2 . We set $\alpha(w_{ij}^1) = \beta(w_{ij}^1) = \alpha(w_{ij}^2) = \beta(w_{ij}^2) = 2$, $\alpha^*(f_1) = \alpha^*(f_2) = 0$, and $\beta^*(f_1) = \beta^*(f_2) = 4$. Observe that in any valid orientation, the edge $\{c_j^{\ell,F}, c_j^{\ell,T}\}$ is directed towards $c_j^{\ell,T}$ (i.e., the corresponding literal is ‘true’) if and only if v_i is ‘true’. Now, if v_i appears in a negative literal, we instead simply add the two edges $\{v_i^{k,F}, c_j^{\ell,F}\}, \{v_i^{k,T}, c_j^{\ell,T}\}$. This yields an additional face f , for which we set $\alpha^*(f) = 0, \beta^*(f) = 4$. Observe that in any valid orientation, the edge $\{c_j^{\ell,F}, c_j^{\ell,T}\}$ is

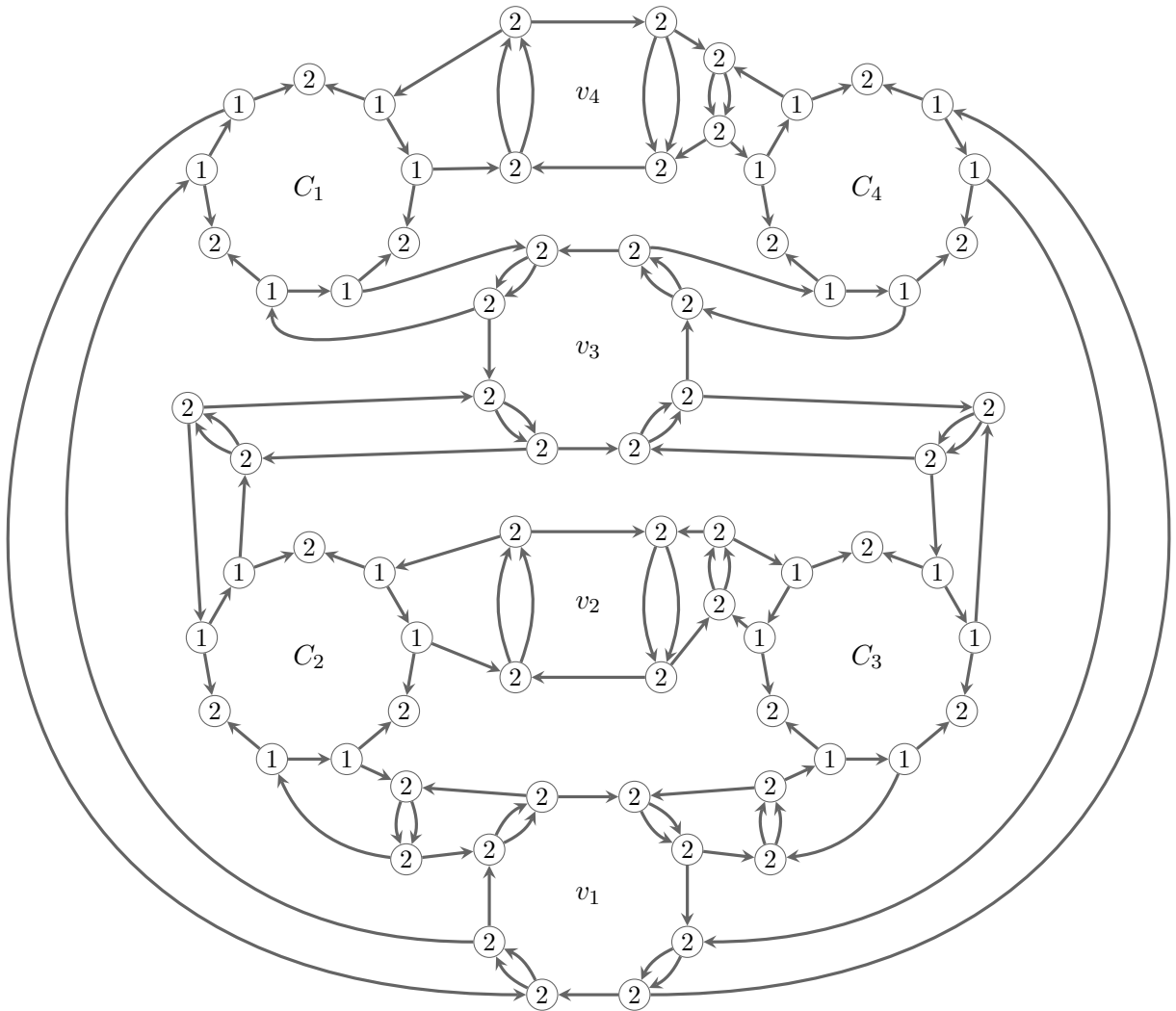


Figure 5.6: Example for the reduction of a planar 3-SAT instance with four clauses and four variables. The clauses are $C_1 = \neg v_1 \vee \neg v_3 \vee \neg v_4$, $C_2 = v_1 \vee \neg v_2 \vee v_3$, $C_3 = v_1 \vee v_2 \vee v_3$, and $C_4 = \neg v_1 \vee \neg v_3 \vee v_4$. The depicted orientation corresponds to the assignment setting v_1, v_2 , and v_4 to ‘true’ and setting v_3 to ‘false’.

directed towards $c_j^{\ell,T}$ (i.e., the corresponding literal is ‘true’) if and only if v_i is ‘false’. Figure 5.6 shows an example of the complete construction for a 3-SAT instance.

The above construction admits an orientation if and only if the corresponding instance of planar 3-SAT admits a satisfying truth assignment. If it exists, the truth assignment can easily be inferred from the orientation by the interpretation given above. Finally, the construction can be made in polynomial time, which concludes our reduction. \square

Corollary 5.7 *The degree-constrained primal-dual orientation problem is NP-complete even when restricted to instances with either $\alpha = \beta$ or $\alpha^* = \beta^*$.*

Proof. The corollary follows from the fact that the construction in the proof of Theorem 5.6 fulfills $\alpha = \beta$. By duality, the reduction can also be achieved by an instance with $\alpha^* = \beta^*$. \square

5.4 Conclusion

In this chapter, we have studied orientation problems in embedded graphs with constraints on the in-degrees both in the primal graph and in the dual graph. Using the orthogonality of cycles and cuts, we have shown that the fixed-degree primal-dual orientation problem for embedded graphs of genus g has at most 2^{2g} feasible solutions and that the set of all solutions can be computed in time $\mathcal{O}(2^{2g}|E|^2 + |E|^3)$. In particular, the solution is unique if the embedding is planar, which answers the question by Frank [Fra10] that motivated our study. However, the problem becomes *NP*-complete immediately, even in the planar case, if only upper and lower bounds on the in-degrees are specified.

Open problems and future research

While our results already give an almost complete characterization of the complexity of the primal-dual orientation problem, several interesting open questions remain.

Complexity of the fixed-degree primal-dual orientation problem. The running time of our algorithm for the fixed-degree primal-dual orientation problem is exponential in the genus of the embedding. Is it possible to devise an algorithm that finds a totally feasible orientation in time polynomial in the genus of the embedding?

Optimization variants. Gabow [Gab06] and Asahiro et al. [AJMO12] studied optimization variants of the degree-constrained orientation problem, aiming for minimizing the number of oriented edges in a partial orientation, or minimizing a penalty function for violated degree bounds. These concepts can be transferred directly to the primal-dual orientation problem, and it is natural to ask for approximation algorithms for these optimization variants. Note, however, that our hardness result in Theorem 5.6 already implies the non-existence of approximation algorithms in the case of general upper and lower bounds (unless $P = NP$). It thus seems to be advisable to restrict to the fixed-degree version of the problem.

One-sided bounds. While we showed the degree-constrained primal-dual orientation problem to be *NP*-complete, even when restricted to instances where all vertices require a fixed in-degree and only the faces allow for intervals of different degrees, the complexity of the following special case, suggested by Woeginger [Woe12], remains unclear: Consider only instances where for each vertex $v \in V$ either $\alpha(v) = 0$ or $\beta(v) = |\delta(v)|$, and for each face $f \in V^*$ either $\alpha^*(f) = 0$ or $\beta^*(f) = |\delta(f)|$. Both constructing a reduction from an *NP*-hard problem or devising an exact polynomial algorithm appears to be challenging in this case, as the one-sidedness of the bounds leaves very large degrees of freedom.

Notation index

General notation

symbol	description	page
\subset	proper subset	
\subseteq	subset or equal	
2^E	the power set $\{S : S \subseteq E\}$ of set E	
$S \dot{\cup} T$	disjoint union; implies $S \cap T = \emptyset$	
A^B	set of functions/vectors $x : B \rightarrow A$	
$(x)^+$	positive part of $x \in \mathbb{R}$, i.e., $(x)^+ = \max\{x, 0\}$	
$\mathbb{R}_+, \mathbb{Q}_+, \mathbb{Z}_+$	set of non-negative reals, rationals, integers, respectively	
$[k]$	set of integers $\{0, 1, \dots, k\}$	
$x \leq y$	componentwise less or equal, $x(e) \leq y(e)$ for all $e \in E$	
$c(S)$	$\sum_{e \in S} c(e)$ for $c \in \mathbb{Q}^E$ and $S \subseteq E$	
$\text{span}(S)$	the linear hull $\{\sum_{v \in S} \lambda(v)v : \lambda \in \mathbb{Q}^S\}$ of the vectors in S	
$\ln(x)$	logarithm of x with basis e	
$\log(x)$	logarithm of x with basis 2	
$\langle I \rangle$	encoding size of the instance I	4
$f = \mathcal{O}(g)$	\mathcal{O} -notation for “ f is asymptotically bounded by g ”	4
$\psi(e)$	the end points of the edge e	7
$V(F)$	set of vertices incident to the edges in F	8
$\text{tail}(a)$	the tail of the arc a	8
$\text{head}(a)$	the head of the arc a	8
$E[S]$	edges with both end points in $S \subseteq V$	8
$G[S]$	induced subgraph $(S, E[S])$ of the vertices $S \subseteq V$	8
$U(D)$	the underlying undirected graph of a digraph D	8
$U(A)$	the set of edges $\{e_a : a \in A\}$ corresponding to the arcs in A	8
$B(G)$	the bidirected digraph corresponding to a graph G	8
$B(E)$	the arcs $\{a_e^+, a_e^- : e \in E\}$ of the bidirected graph	8
$\delta(S)$	set of edges $e \in E$ with $\psi(e) \cap S \neq \emptyset$ and $\psi(e) \cap V \setminus S \neq \emptyset$	8
$\delta^+(S)$	set of arcs $a \in A$ with $\text{tail}(a) \in S$ and $\text{head}(a) \in V \setminus S$	8
$\delta^-(S)$	set of arcs $a \in A$ with $\text{tail}(a) \in V \setminus S$ and $\text{head}(a) \in S$	9
$\text{ex}(x, v)$	excess at node v with respect to flow $x \in \mathbb{Q}_+^A$;	9

Abstract flows

symbol	description	page
$[P, e]$	$\{p \in P : p \leq_P e\}$ for an abstract path P	21
$P \times_e Q$	abstract path in $[P, e] \cup [e, Q]$ guaranteed by the switching axiom	21
\mathbb{O}_{path}	path oracle for accessing the abstract network	22
$\tau(e)$	transit time of element e	23
\mathcal{T}	points in time $\mathcal{T} = \{0, \dots, T - 1\}$	23
E_T	elements of the time expansion of an abstract network	23
\mathcal{P}_T	set of temporal paths	23
$\gamma(P_t, e)$	time $t + \sum_{p \in (P, e)} \tau(p)$ at which flow arrives at e when sent along P_t	23
x_T	temporally repeated abstract flow derived from the abstract flow x	26
\mathbb{O}_{sep}	separation oracle for accessing the abstract network	28

Transportation planning

symbol	description	page
B	base network	43
$\mathcal{D} = (\mathcal{V}, \mathcal{R})$	pattern-expanded network	43
K	set of commodities	43
P	set of properties	43
α_{ij}	per-unit extent of commodity i for property j	43
$\alpha(x)$	aggregated properties of the vector $x \in \mathbb{Q}_+^K$; $\alpha_j(x) := \sum_{i \in K} \alpha_{ij} x_i$	43
$b_i(v)$	supply/demand of node v for commodity i	43
$T(R)$	set of available tariffs for transport relation R	44
$D = (V, A)$	tariff-expanded network	46
$\beta_j(a)$	capacity of container type a for property j	46
$u(a)$	upper bound on the number of containers installed on arc a	46

Combined network design and facility location problems

symbol	description	page
$c(v, F)$	distance of client v to the nearest facility in F	78
$T[v, w]$	the unique v - w -path in tree T	78
$T_r[v]$	subtree rooted at vertex v of the tree T with respect to root r	78
$\text{depth}_\ell(T, r)$	depth of the tree T rooted at r with respect to lengths ℓ	78
$\text{diam}_\ell(T)$	diameter of the tree T with respect to lengths ℓ	78

Graph orientation

symbol	description	page
R	set of primally rigid edges	132
R^*	set of dually rigid edges	132

Subject index

- abstract flows, 19–33
 - abstract cut over time, 24
 - abstract flow over time, 23
 - abstract network, 20
- abstract max flow/min cut over time, 25
- aggregated MIP formulation, 61, 63
- algorithm, 3–6
- all-unit discount cost, 45, 48
- ALP, *see* aggregated MIP formulation
- AMIP, *see* aggregated MIP formulation
- anti-parallel, 8
- approximation algorithm, 5
- arc, 8
- arrival time, 23
- augmenting path algorithm, 11

- base network, 42
- bounded diameter Steiner tree problem, 114

- capacitated location routing problem, 85
 - with cross-docking, 100
- capacitated-cable facility
 - location problem, 80
- CCFL, *see* capacitated-cable facility
 - location problem
- CLR, *see* capacitated location
 - routing problem
- CLR-CD, *see* capacitated location
 - routing problem
- commodities, 12, 43
- complexity class, 4
- connected component, 8
- consolidation, 35, 38
- container, 46
- cross-docking, 100
- cut, 8
 - directed, 8
 - induced, 8
 - simple, 8
- cycle, 8

- cycle/cut duality, 129

- decomposition, *see* network flows
- degree, 8
- degree-constrained orientation problem, 128
- degree-constrained primal-dual
 - orientation problem, 135
- depth, 78
- diameter, 78
- digraph, *see* graph, directed graph
- directed Steiner forest problem, 17
- directed Steiner tree problem, 17, 113
- DTIME*, 4
- dual feasibility, 131
- dual graph, 129
- duality
 - linear programming, 7
 - of embedded graphs, 129

- edge, 7
 - rigid, 132
- embedded digraph, 129
- embedded graph, 129
 - orientation, 130
- embedding, 129
- encoding size, 4
- end points, 7
- Euler’s formula, 129
- excess, 9

- face, 129
- fixed-charge network flow problem, 18
- fixed-degree primal-dual orientation
 - problem, 131
- fixed-parameter tractability, 5
- flow, *see* network flows

- G-CLR, *see* group capacitated
 - location routing problem
- good characterization, 127

- graph, 7–9
 - bidirected, 8
 - connected, 8
 - directed, 8
 - embedded, 129
 - planar, 129
 - undirected, 7
- group capacitated location
 - routing problem, 94
- head, *see* arc
- heuristic, 5
- holdover arc, 14
- hop constraint, 113
- in-degree, 9
- incremental discount cost, 45, 48
- integer programming, 6–7
- IP, *see* integer programming
- LAST algorithm, 113
- layered graph, 114
- linear programming, 6–7
 - duality, 7
 - equivalence of optimization
 - and separation, 7
 - relaxation, 7
- local search, 56, 59–60
- location routing, 84–110
 - prize-collecting, 91
 - with cross-docking, 100
 - with groups, 94
- loop, 7
- LP, *see* linear programming
- max flow/min cut theorem, 10
- maximum abstract flow
 - over time problem, 24
- maximum flow over time problem, 14
- maximum flow problem, 10–11
- MCVR, *see* multi-depot capacitated
 - vehicle routing problem
- minimum cost flow problem, 11
- minimum cost multi-commodity
 - flow problem, 13
- minimum spanning tree problem, 16
- MIP, *see* mixed integer programming
- mixed integer programming, 6
- multi-commodity flows, 12–13
 - multi-commodity b -flow, 12
- multi-depot capacitated vehicle
 - routing problem, 85
- network design, 15–18
- network flows, 9–15
 - b -flow, 9
 - flow conservation, 9
 - flow decomposition, 9, 59
 - flows over time, 13–15
 - s - t -flow, 9
- node, *see* vertex
- NP, 4
- oracle, 21, 28
- orientation
 - dually feasible, 131
 - of a graph, 128
 - primally feasible, 131
 - totally feasible, 131
- out-degree, 9
- out-tree, 9
- P , 4
- parallel, *see* edge, 8
- path, 8
 - abstract path, 20
 - path decomposition, *see* network flows
- pattern optimization, 63
- pattern-expanded network, 42
- PC-CLR, *see* prize-collecting capacitated
 - location routing problem
- planar 3-SAT problem, 135
- planar graph, 129
- primal feasibility, 131
- prize-collecting capacitated
 - location routing problem, 91
- properties, 43
 - aggregated, 43
- relieve procedure, 81–84
- residual network, 11
- restricted shortest paths, 113
- rigid, 132
- running time, 4
- separation, *see* linear programming
- set cover, 4
- shallow-light Steiner tree problem, 112

- shallow-light trees, 112–116
- shortest path with linearized costs, 56–57
- shortest path with tariff selection, 58–59
- single-assignment property, 85
- single-tour property, 85
- sink, 9
- SLST, *see* shallow-light Steiner tree
 - problem
- source, 9
- spanning tree, 9
- SPLC, *see* shortest path with linearized costs
- SPTS, *see* shortest path with tariff selection
- Steiner tree problem, 16
- storage at intermediate elements, 14, 25, 29
- strengthened container inequalities, 62
- subgraph, 8
- subtree, 78
- successive shortest path algorithm, 12
- supermodularity, 22
- switching axiom, 20

- tactical transportation planning, 37–38
 - capacitated network design
 - formulation, 46
 - complexity, 49
 - problem formulation, 44
- tail, *see* arc
- tariff, 44
- tariff selection, 50–55
 - cost estimation, 55
 - greedy algorithm, 53–55
 - hardness, 52
- tariff selection problem, 50
- tariff-expanded network, 46
- TDI, *see* total dual integrality
- temporal path, 23
- temporally repeated abstract flow, 26
- time expansion of abstract networks, 23
- time horizon, 13
- time-expanded ground set, 23
- time-expanded network, 14, 24
- total dual integrality, 7
- total feasibility, 131
- transit time, 14
- tree, 9
 - depth, 78
 - diameter, 78
- TS, *see* tariff selection problem

- UFL, *see* uncapacitated facility location
- unbounded knapsack problem, 52
- uncapacitated facility location problem, 79

- vehicle routing, 84, 86
- vertex, 7
 - degree, 8

- walk, 8
- weighted abstract cut problem, 21
- weighted abstract flow problem, 21

- ZTIME*, 4

References

- [ABW12] D. Adjiashvili, S. Bosio, and R. Weismantel. Dynamic combinatorial optimization: A complexity and approximability study. Unpublished manuscript, 2012.
- [Ach09] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
- [AFHP⁺05] E. Althaus, S. Funke, S. Har-Peled, J. Könemann, E. A. Ramos, and M. Skutella. Approximating k -hop minimum-spanning trees. *Operations Research Letters*, 33(2):115–120, 2005.
- [AG86] P. Afentakis and B. Gavish. Optimal lot-sizing algorithms for complex product structures. *Operations Research*, 34(2):237–249, 1986.
- [AG87] K. Altinkemer and B. Gavish. Heuristics for unequal weight delivery problems with a fixed error guarantee. *Operations Research Letters*, 6(4):149–158, 1987.
- [AGK84] P. Afentakis, B. Gavish, and U. Karmarkar. Computationally efficient optimal solutions to the lot-sizing problem in multistage assembly systems. *Management Science*, 30(2):222–239, 1984.
- [AGS13] A. Arulselvan, M. Groß, and M. Skutella. Graph orientation and network flows over time. Unpublished manuscript, 2013.
- [AJMO12] Y. Asahiro, J. Jansson, E. Miyano, and H. Ono. Upper and lower degree bounded graph orientation with minimum penalty. In *Proceedings of the 18th Computing: The Australasian Theory Symposium*, volume 128, pages 139–146, 2012.
- [AKL⁺03] C. J. Alpert, A. B. Kahng, B. Liu, I. I. Mandoiu, and A. Z. Zelikovsky. Minimum buffered routing with bounded capacitive load for slew rate and reliability control. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(3):241–253, 2003.
- [AL97] E. E. H. Aarts and J. K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 1997.
- [AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice Hall, 1993.

- [Aro96] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 2–11, 1996.
- [ARR98] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k -medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106–113, 1998.
- [BA10] J. Byrka and K. Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.
- [BBD⁺87] D. Blumenfeld, L. Burns, C. Daganzo, M. Frick, and R. Hall. Reducing logistics costs at general motors. *Interfaces*, 17(1):26–47, 1987.
- [BCG⁺05] T. Biedl, T. Chan, Y. Ganjali, M. T. Hajiaghayi, and D. R. Wood. Balanced vertex-orderings of graphs. *Discrete Applied Mathematics*, 148(1):27–48, 2005.
- [Ben62] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.
- [BFPS07] S. Barreto, C. Ferreira, J. Paixão, and B. S. Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3):968 – 977, 2007.
- [BG61] R. Busaker and P. Gowen. A procedure for determining minimal-cost flow network patterns. Technical report, ORO-15, Operational Research Office, Johns Hopkins University, 1961.
- [BGRS10] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 583–592, 2010.
- [BHBD85] L. Burns, R. Hall, D. Blumenfeld, and C. Daganzo. Distribution strategies that minimize transportation and inventory costs. *Operations Research*, 33(3):469–490, 1985.
- [BHR13] A. Bley, S. M. Hashemi, and M. Rezapour. Approximation algorithms for a combined facility location buy-at-bulk network design problem. In *Theory and Applications of Models of Computation*, pages 72–83, 2013.
- [Bix12] R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012.
- [BK09] G. Borradaile and P. Klein. An $O(n \log n)$ algorithm for maximum st -flow in a directed planar graph. *Journal of the ACM*, 56(2):9, 2009.
- [BKM⁺11] G. Borradaile, P. N. Klein, S. Mozes, Y. Nussbaum, and C. Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science*, pages 170–179, 2011.

- [BKM⁺12] S. Bosio, J.-P. W. Kappmeier, J. Matuschke, B. Peis, and M. Skutella. Flows over time with negative transit times and arc release dates. In *Proceedings of the 11th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, pages 30–33, 2012.
- [BMW09] R. Baldacci, A. Mingozzi, and R. Wolfler Calvo. The capacitated location routing problem. Presented at ROUTE 2009, Rolighed, Denmark, 2009.
- [Bod12] K. Bodack. Kapazitierte Standortoptimierung mit integrierter Routenplanung: Algorithmen und Komplexität. Diploma thesis at TU Berlin, 2012.
- [Bor26] O. Borůvka. Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí. *Elektrotechnický obzor*, 15:153–154, 1926.
- [BR13] A. Bley and M. Rezapour. Approximating connected facility location with buy-at-bulk edge costs via random sampling. *Electronic Notes in Discrete Mathematics*, 44(0):313–319, 2013.
- [CC09a] X. Chen and B. Chen. Approximation algorithms for soft-capacitated facility location in capacitated network design. *Algorithmica*, 53(3):263–297, 2009.
- [CC09b] X. Chen and B. Chen. Cost-effective designs of fault-tolerant access networks in communication systems. *Networks*, 53(4):382–391, 2009.
- [CCC⁺99] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.
- [CCG09a] M. Chouman, T. G. Crainic, and B. Gendron. A cutting-plane algorithm for multicommodity capacitated fixed-charge network design. Technical Report 2009-20, Université de Montréal, Centre de recherche sur les transports, 2009.
- [CCG09b] A. Costa, J.-F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009.
- [CCG11] C. Contardo, J.-F. Cordeau, and B. Gendron. A branch-and-cut algorithm for the capacitated location-routing problem. Technical Report 2011-44, Université de Montréal, Centre de recherche sur les transports, 2011.
- [CCKK11] D. Chakrabarty, C. Chekuri, S. Khanna, and N. Korula. Approximability of capacitated network design. In *Integer Programming and Combinatorial Optimization*, pages 78–91, 2011.
- [CE91] M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theoretical Computer Science*, 86(2):243–266, 1991.
- [Çet05] S. Çetinkaya. Coordination of inventory and shipment consolidation decisions: A review of premises, models, and justification. In *Applications of*

- Supply Chain Management and E-Commerce Research*, volume 92 of *Applied Optimization*, pages 3–51. Springer, 2005.
- [CG02] T. G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627, 2002.
- [CGF00] T. G. Crainic, M. Gendreau, and J. Farvolden. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, 12(3):223, 2000.
- [CGH04] T. G. Crainic, B. Gendron, and G. Henu. A slope scaling/lagrangian perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10(5):525–545, 2004.
- [CGNS08] J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of some network design problems. *ACM Transactions on Algorithms*, 4(2):23, 2008.
- [Chr76] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Carnegie Mellon University, Graduate School of Industrial Administration, 1976.
- [CKMN01] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.
- [CLR] CLRLib. <http://www.coga.tu-berlin.de/clrlib>, 2012.
- [CM07] S. Chopra and P. Meindl. *Supply chain management: strategy, planning, and operations*. Prentice Hall, 2007.
- [CMS⁺02] L. Chan, A. Muriel, Z.-J. Shen, D. Simchi-Levi, and C.-P. Teo. Effective zero-inventory-ordering policies for the single-warehouse multiretailer problem with piecewise linear cost structures. *Management Science*, 48(11):1446–1460, 2002.
- [Cos05] A. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32:1429–1450, 2005.
- [Cra00] T. G. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.
- [CS60] A. Clark and H. Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490, 1960.
- [Dan51a] G. B. Dantzig. Application of the simplex method to a transportation problem. *Activity analysis of production and allocation*, 13:359–373, 1951.
- [Dan51b] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In *Activity Analysis of Production and Allocation*. Wiley, 1951.

- [DGK⁺10] D. Dressler, M. Groß, J.-P. Kappmeier, T. Kelter, J. Kulbatzki, D. Plümpe, G. Schlechter, M. Schmidt, M. Skutella, and S. Temme. On the use of network flow techniques for assigning evacuees to exits. *Procedia Engineering*, 3:205–215, 2010.
- [Din70] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Mathematics Doklady*, volume 11, pages 1277–1280, 1970.
- [DK99] Y. Dodis and S. Khanna. Designing networks with bounded pairwise distance. In *Proceedings of the 31st annual ACM Symposium on Theory of Computing*, pages 750–759. ACM, 1999.
- [DM12] Y. Disser and J. Matuschke. Degree-constrained orientations of embedded graphs. In *Algorithms and Computation*, volume 7676 of *Lecture Notes in Computer Science*, pages 506–516. Springer, 2012.
- [Dob82] G. Dobson. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Mathematics of Operations Research*, 7(4):pp. 515–531, 1982.
- [DS69] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Series B*, 73(2):91–118, 1969.
- [DW60] G. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):pp. 101–111, 1960.
- [DW71] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.
- [Edm65] J. Edmonds. Minimum partition of a matroid into independent subsets. *Journal of Research of the National Bureau of Standards, Series B*, 69:67–72, 1965.
- [Edm67] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.
- [EG77] J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. In *Studies in Integer Programming*, volume 1, pages 185–204. North-Holland, 1977.
- [EK72] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.
- [ET75] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, 1975.
- [Eva77] J. R. Evans. Some network flow models and heuristics for multiproduct production and inventory planning. *AIIE Transactions*, 9(1):75–81, 1977.

- [EW90] P. Eades and N. C. Wormald. Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28(2):111–134, 1990.
- [Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
- [Fel04] S. Felsner. Lattice structures from planar graphs. *Journal of Combinatorics*, 11(1):R15, 2004.
- [FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.
- [FF57] L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canadian Journal of Mathematics*, 9(210-218), 1957.
- [FF58a] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.
- [FF58b] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.
- [FF62] L. R. Ford and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.
- [FG76] A. Frank and A. Gyárfás. How to orient the edges of a graph? *Colloquia mathematica societatis Janos Bolyai*, 18:353–364, 1976.
- [FHI89] M. Frigge, D. Hoaglin, and B. Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, 1989.
- [FKK03] A. Frank, T. Király, and Z. Király. On the orientation of graphs and hypergraphs. *Discrete Applied Mathematics*, 131(2):385–400, 2003.
- [FR99] J. Feldman and M. Ruhl. The directed steiner network problem is tractable for a constant number of terminals. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 299–308, 1999.
- [Fra10] A. Frank. Seminar of the Egerváry research group on combinatorial optimization. Budapest, February 2010.
- [FS03] L. Fleischer and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 66–75, 2003.
- [FS07] L. Fleischer and M. Skutella. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.
- [FSZ10] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders’ cuts. *Mathematical Programming*, 124:175–182, July 2010.
- [FT98] L. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23(3):71–80, 1998.

- [Fuj03] S. Fujishige. A maximum flow algorithm using MA ordering. *Operations Research Letters*, 31(3):176–178, 2003.
- [GA88] B. L. Golden and A. A. Assad, editors. *Vehicle Routing: Methods and Studies*. North-Holland, 1988.
- [Gab06] H. N. Gabow. Upper degree-constrained partial orientations. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm*, pages 554–563, 2006.
- [Gai97] A. Gaillard. Switchdec polyhedra. *Discrete Applied Mathematics*, 76(1):141–163, 1997.
- [GCF99] B. Gendron, T. G. Crainic, and A. Frangioni. Multicommodity capacitated network design. In *Telecommunications Network Planning*, pages 1–19. Kluwer Academic Publishers, 1999.
- [GCG03] I. Ghamlouche, T. G. Crainic, and M. Gendreau. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research*, 51(4):655–667, 2003.
- [GCG04] I. Ghamlouche, T. G. Crainic, and M. Gendreau. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations research*, 131(1):109–133, 2004.
- [GG74] A. Geoffrion and G. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844, 1974.
- [GH82] H. Gröflin and A. J. Hoffman. Lattice polyhedra II: Generalization, constructions and examples. *North-Holland Mathematics Studies*, 65:189–203, 1982.
- [GJ79] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979.
- [GK98] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.
- [GK07] N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.
- [GKSS12] M. Groß, J.-P. W. Kappmeier, D. R. Schmidt, and M. Schmidt. Approximating earliest arrival flows in arbitrary networks. In *Algorithms—ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 551–562. Springer, 2012.
- [GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.
- [GLY04] E. Gourdin, M. Labbé, and H. Yaman. Telecommunication and location. In *Facility Location: Applications and Theory*, pages 275–306. Springer, 2004.

- [GMS95] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. *Handbooks in operations research and management science*, 7:617–672, 1995.
- [GNS11] J. Guo, R. Niedermeier, and O. Suchý. Parameterized complexity of arc-weighted directed steiner problems. *SIAM Journal on Discrete Mathematics*, 25(2):583–599, 2011.
- [GP68] E. Gilbert and H. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.
- [GP90] G. Guisewite and P. Pardalos. Minimum concave-cost network flow problems: Applications, complexity, and algorithms. *Annals of Operations Research*, 25:75–99, 1990.
- [GP03] J. Geunes and P. Pardalos. Network optimization in supply chain management and financial engineering: An annotated bibliography. *Networks*, 42(2):66–84, 2003.
- [GP08] H. Glicksman and M. Penn. Approximation algorithms for group prize-collecting and location-routing problems. *Discrete Applied Mathematics*, 156(17):3238–3247, 2008.
- [GS12] M. Groß and M. Skutella. Maximum multicommodity flows over time without intermediate storage. In *Algorithms—ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 539–550. Springer, 2012.
- [GSU11] L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Mathematical Programming*, 128(1-2):123–148, 2011.
- [GT88] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, 1988.
- [GT89] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.
- [Gur] Gurobi Optimization, Inc. Gurobi Optimizer. <http://www.gurobi.com>.
- [GW81] B. Golden and R. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.
- [GW95] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
- [Hak65] S. Hakimi. On the degrees of the vertices of a directed graph. *Journal of the Franklin Institute*, 279(4):290–308, 1965.
- [Has92] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

- [HAS⁺11] M. Hausknecht, T. Au, P. Stone, D. Fajardo, and T. Waller. Dynamic lane reversal in traffic management. In *14th International IEEE Conference on Intelligent Transportation Systems*, pages 1929–1934, 2011.
- [Hel00] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operations Research*, 126(1):106–130, 2000.
- [HHS07] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379(3):387–404, 2007.
- [Hit41] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematical Physics*, 20(2):224–230, 1941.
- [HK03] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th annual ACM Symposium on Theory of Computing*, pages 585–594, 2003.
- [HKM⁺] T. Harks, F. G. König, J. Matuschke, A. Richter, and J. Schulz. An integrated approach to tactical transportation planning in logistics networks. *Transportation Science*, to appear.
- [HKM13] T. Harks, F. G. König, and J. Matuschke. Approximation algorithms for capacitated location routing. *Transportation Science*, 47(1):3–22, 2013.
- [Hoc82] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.
- [Hoc96] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1996.
- [Hof74] A. J. Hoffman. A generalization of max flow—min cut. *Mathematical Programming*, 6(1):352–359, 1974.
- [Hof78] A. J. Hoffman. On lattice polyhedra III: Blockers and anti-blockers of lattice clutters. In *Polyhedral Combinatorics*, pages 197–207. Springer, 1978.
- [HRK85] M. Haimovich and A. H. G. Rinnoy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.
- [HS78] A. J. Hoffman and D. E. Schwartz. On lattice polyhedra. In *Proceedings of the 5th Hungarian Colloquium on Combinatorics*, pages 593–598, 1978.
- [HT00] B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000.
- [IBM] IBM Corporation. IBM ILOG CPLEX Optimizer. <http://www.ibm.com/software/integration/optimization/cplex-optimizer/>.
- [Iri60] M. Iri. A new method of solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3(1):2, 1960.

- [Jai01] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
- [Jar30] V. Jarník. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 6:57–63, 1930.
- [Jay98] V. Jayaraman. Transportation, facility location and inventory issues in distribution network design: An investigation. *International Journal of Operations and Production Management*, 18(5):471–494, 1998.
- [JD07] R. Jans and Z. Degraeve. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3):1855–1875, 2007.
- [Jew58] W. S. Jewell. *Optimal flow through networks*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering, 1958.
- [JMM⁺03] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
- [JMS02] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, pages 731–740, 2002.
- [Kar72] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.
- [Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th annual ACM symposium on Theory of computing*, pages 302–311, 1984.
- [Kha80] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.
- [KKS10] J. Kempkes, A. Koberstein, and L. Suhl. A resource based mixed integer modelling approach for integrated operational logistics planning. In *Advanced Manufacturing and Sustainable Logistics*, volume 46 of *Lecture Notes in Business Information Processing*, pages 281–294. Springer, 2010.
- [Kle09] P. Klein. Planar graph algorithms. Lecture at Brown University, Providence, available from <http://www.cs.brown.edu/courses/cs250/>, 2009.
- [KMB81] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta informatica*, 15(2):141–145, 1981.
- [KMP12] J.-P. W. Kappmeier, J. Matuschke, and B. Peis. Abstract flows over time: A first step towards solving dynamic packing problems. In *Algorithms and Computation*, volume 7676 of *Lecture Notes in Computer Science*, pages 433–443. Springer, 2012.

- [KMR12] F. G. König, J. Matuschke, and A. Richter. Multi-Dimensional Commodity Covering for Tariff Selection in Transportation. In *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 25 of *OpenAccess Series in Informatics*, pages 58–70. Dagstuhl Publishing, 2012.
- [KN07] G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems. In *Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007.
- [Kön31] D. König. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.
- [KP99a] D. Kim and P. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24(4):195–203, 1999.
- [KP99b] G. Kortsarz and D. Peleg. Approximating the weight of shallow Steiner trees. *Discrete Applied Mathematics*, 93(2):265–285, 1999.
- [KPR98] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1998.
- [KRY95] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.
- [KS07] S. Kapoor and M. Sarwat. Bounded-diameter minimum-cost graph problems. *Theory of Computing Systems*, 41(4):779–794, 2007.
- [KV08] B. Korte and J. Vygen. *Combinatorial problems in chip design*. Springer, 2008.
- [KV12] B. Korte and J. Vygen. *Combinatorial optimization: Theory and algorithms*. Springer, 2012.
- [KW04] B. Klinz and G. J. Woeginger. Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43(3):153–162, 2004.
- [Lap88] G. Laporte. Location-routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 163–198. North-Holland, 1988.
- [LD05] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.
- [Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.
- [LLC03] J. Liu, C.-L. Li, and C.-Y. Chan. Mixed truck delivery systems with both hub-and-spoke and direct shipment. *Transportation Research Part E: Logistics and Transportation Review*, 39(4):325–339, 2003.

- [LNT88] G. Laporte, Y. Nobert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22(3):161–172, 1988.
- [LSL90] C. Li and D. Simchi-Levi. Worst-case analysis of heuristics for multidepot capacitated vehicle routing problems. *ORSA Journal on Computing*, 2(1):64–73, 1990.
- [Lue75] G. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report 178, Princeton University, Computer Science Laboratory, 1975.
- [LV92] J.-H. Lin and J. S. Vitter. ϵ -approximations with minimum packing constraint violation. In *Proceedings of the 24th annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.
- [Mag84] T. L. Magnanti. Models and algorithms for predicting urban traffic equilibria. In *Transportation Planning Models*, pages 153–186. North-Holland, 1984.
- [MBM13] J. Matuschke, A. Bley, and B. Müller. Approximation algorithms for facility location with capacitated and length-bounded tree connections. In *Algorithms—ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 707–718. Springer, 2013.
- [McC96] S. T. McCormick. A polynomial algorithm for abstract maximum flow. In *Proceedings of the 7th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 490–497, 1996.
- [McC13] S. T. McCormick. Personal communication. Vancouver, December 2013.
- [Men27] K. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.
- [MJS98] H. Mina, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108(1):1–15, 1998.
- [MM08] M. Martens and S. T. McCormick. A polynomial algorithm for weighted abstract flow. In *Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2008.
- [MNSDG09] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama. Facility location and supply chain management—a review. *European Journal of Operational Research*, 196(2):401–412, 2009.
- [MRS⁺98] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998.

- [MV08] J. Maßberg and J. Vygen. Approximation algorithms for a facility location problem with service capacities. *ACM Transactions on Algorithms*, 4(4):50, 2008.
- [MW84] T. Magnanti and R. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.
- [MYZ06] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.
- [Nie06] R. Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.
- [NN12] J. Nešetřil and H. Nešetřilová. The origins of minimal spanning tree algorithms – Borůvka and Jarník. *Documenta Mathematica*, pages 127–141, 2012.
- [NS97] J. Naor and B. Schieber. Improved approximations for shallow-light spanning trees. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 536–541, 1997.
- [NS07] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.
- [NW60] C. S. J. Nash-Williams. On orientations, connectivity and odd vertex pairings in finite graphs. *Canadian Journal of Mathematics*, 12(555-567):8, 1960.
- [Orl13] J. B. Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing*, pages 765–774, 2013.
- [PPR⁺07] C. Prins, C. Prodhon, A. B. Ruiz, P. Soriano, and R. Wolfer Calvo. Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483, 2007.
- [PPW06] C. Prins, C. Prodhon, and R. Wolfer Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4(3):221–238, 2006.
- [Pri57] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.
- [PS02] H. J. Prömel and A. Steger. *The Steiner tree problem: A tour through graphs, algorithms, and complexity*. Vieweg, 2002.
- [Raa12] C. Raack. *Capacitated network design: Multi-commodity flow formulations, cutting planes, and demand uncertainty*. PhD thesis, TU Berlin, 2012.
- [RAEP10] S. Rebenmack, A. Arulselvan, L. Elefteriadou, and P. M. Pardalos. Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, 19(2):200–216, 2010.

- [Rob39] H. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939.
- [RS06] R. Ravi and A. Sinha. Approximation algorithms for problems combining facility location and network design. *Operations Research*, 54(1):73–81, 2006.
- [Sch84] A. Schrijver. Total dual integrality from directed graphs, crossing families and sub- and supermodular functions. *Progress in Combinatorial Optimization*, pages 315–361, 1984.
- [Sch98] A. Schrijver. *Theory of linear and integer programming*. Wiley, 1998.
- [Sch02] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.
- [Sch03] A. Schrijver. *Combinatorial optimization: Polyhedra and efficiency*. Springer, 2003.
- [Sha93] J. F. Shapiro. Mathematical programming models and methods for production planning and scheduling. *Handbooks in Operations Research and Management Science*, 4:371–443, 1993.
- [SKS10] T. Schöneberg, A. Koberstein, and L. Suhl. An optimization model for automated selection of economic and ecologic delivery profiles in area forwarding based inbound logistics networks. *Flexible Services and Manufacturing Journal*, 22(3-4):214–235, 2010.
- [Sku09] M. Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization*, pages 451–482. Springer, 2009.
- [SLKSL03] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. McGraw Hill, 2003.
- [SS98] B. Sanso and P. Soriano. *Telecommunications network planning*. Kluwer Academic Publishers, 1998.
- [STA97] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.
- [Sta03] H. Stadtler. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, 51(3):487–502, 2003.
- [Tal02] K. Talwar. The single-sink buy-at-bulk lp has constant integrality gap. In *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 475–486. Springer, 2002.
- [TB99] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

- [Tol30] A. Tolstoï. Metody nakhozhdeniya naimen'shego summovogo kilometrazha pri planirovanii perezozok v prostranstve. *Planirovanie Perevozok, Sbornik pervyï, Transpechat' NKPS*, pages 23–55, 1930.
- [TV02] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.
- [Vaz01] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.
- [vN47] J. von Neumann. Discussion of a maximum problem. *John von Neumann. Collected Works*, 6:89–95, 1947.
- [Voß92] S. Voß. Steiner's problem in graphs: Heuristic methods. *Discrete Applied Mathematics*, 40(1):45–72, 1992.
- [Vyg04] J. Vygen. Near-optimum global routing with coupling, delay bounds, and power consumption. In *Integer Programming and Combinatorial Optimization*, volume 3064 of *Lecture Notes in Computer Science*, pages 308–324. Springer, 2004.
- [VZ10] B. Vahdani and M. Zandieh. Scheduling trucks in cross-docking systems: Robust meta-heuristics. *Computers & Industrial Engineering*, 58(1):12–24, 2010.
- [Web68] M. H. J. Webb. Cost functions in the location of depots for multiple-delivery journeys. *Operations Research Quarterly*, 19:311–320, 1968.
- [Weg05] I. Wegener. *Complexity theory*. Springer, 2005.
- [Whi32] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(2):339–362, 1932.
- [Win87] P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.
- [WLC⁺09] M. Wen, J. Larsen, J. Clausen, J.-F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60(11):1708–1718, 2009.
- [Woe12] G. J. Woeginger. Personal communication. Berlin, November 2012.
- [Wol01] B. Wolshon. “one-way-out”: Contraflow freeway operation for hurricane evacuation. *Natural Hazards Review*, 2:105, 2001.
- [Won84] R. T. Wong. A dual ascent approach for steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984.
- [WS11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
- [WW58] H. Wagner and T. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

