Solving Two-Stage Stochastic Network Design Problems to Optimality

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Bernd Thomas Zey

Dortmund
2017

Tag der mündlichen Prüfung:
14.11.2017

Dekan:
Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr. Petra Mutzel, TU Dortmund, Fakultät für Informatik
Prof. Dr. Christoph Buchheim, TU Dortmund, Fakultät für Mathematik

Contents

Acknowledgements

I thank my advisor Prof. Petra Mutzel for supporting me from the very beginning when I started working on my diploma thesis, for introducing me to the fields of network design and stochastic programming, for proposing this nice topic for my thesis (and for the freedom to select my own research directions), and for the scientific guidance over the last years. I am really thankful for the opportunity to work as a researcher at the TU Dortmund. Moreover, I am deeply grateful for the support of my parental leaves; this was very helpful for my family and me.

I thank Prof. Christoph Buchheim for being the second referee of this thesis and Prof. Johannes Fischer and Prof. Peter Buchholz for serving on my defense commission.

My colleagues and co-authors Markus Chimani and Ivana Ljubić deserve special thanks for all the support and help, in particular at the beginning of our SSTP research, for the countless discussions, many ideas, and a lot of good advice.

I thank my co-authors Michael Jünger, Immanuel Bomze, Fritz Bökler, Denis Kurz, and Daniel Schmidt for the helpful and inspiring discussions and the great collaboration. Moreover, I thank my friends, colleagues, roommates, and (former) members of Chair XI and the Algorithm Engineering group for the joint work and the nice and relaxing coffee breaks: Andreas Thom, Carsten Gutwenger, Karsten Klein, Hoi-Ming Wong, Maria Kandyba-Chimani, Nils Kriege, Fritz Bökler, Denis Kurz, Christopher Morris, Andre Droschinsky, Till Schäfer, Christiane Spisla, and Adalat Jabrayilov. Furthermore, I am grateful to Andre, Nils, Christopher, and Christiane for proof-reading parts of this thesis. Last but not least, I thank Gundel Jankord for so much help on the organizational work and the nice off-topic chats.

My family and friends always supported, helped, and encouraged me and for that I am truly grateful to all of them. Most of all, I thank my parents for the unconstrained support, encouragement, and for the opportunities they have given me.

Edyta, Lukas, and Lena, you are my motivation and you are the most important parts in my life – I love you – Ich liebe euch – Kocham was.

Bernd Zey
Dortmund, 2017

Abstract

The Steiner tree problem (STP) is a central and well-studied graph-theoretical combinatorial optimization problem which plays an important role in various applications. It can be stated as follows: Given a weighted graph and a set of terminal vertices, find a subset of edges which connects the terminals at minimum cost. However, in real-world applications the input data might not be given with certainty or it might depend on future decisions. For the STP, for example, edge costs representing the costs of establishing links may be subject to inflations and price deviations. In this thesis we tackle data uncertainty by using the concept of stochastic programming and we study the two-stage stochastic version of the Steiner tree problem (SSTP). Thereby, a set of scenarios defines the possible outcomes of a random variable; each scenario is given by its realization probability and defines a set of terminals and edge costs. A feasible solution consists of a subset of edges in the first stage and edge subsets for all scenarios (second stage) such that each terminal set is connected. The objective is to find a solution that minimizes the expected cost. We consider two approaches for solving the SSTP to optimality: combinatorial algorithms, in particular fixed-parameter tractable (FPT) algorithms, and methods from mathematical programming.

Regarding the combinatorial algorithms we develop a linear-time algorithm for trees, an FPT algorithm parameterized by the number of terminals, and we consider treewidth-bounded graphs where we give the first FPT algorithm parameterized by the combination of treewidth and number of scenarios.

The second approach is based on deriving strong integer programming (IP) formulations for the SSTP. By using orientation properties we introduce new semi-directed cut- and flow-based IP formulations which are shown to be stronger than the undirected models from the literature. To solve these models to optimality we use a decomposition-based two-stage branch&cut algorithm, which is improved by a fast and efficient method for strengthening the optimality cuts. Moreover, we develop new and stronger integer optimality cuts. The computational performance is evaluated in a comprehensive computational study, which shows the superiority of the new formulations, the benefit of the decomposition, and the advantage of using the strengthened optimality cuts.

The Steiner forest problem (SFP) is a related problem where sets of terminals need to be connected. On the one hand, the SFP is a generalization of the STP and on the other hand, we show that the SFP is a special case of the SSTP. Therefore, our results are transferable to the SFP and we present the first FPT algorithm for treewidth-bounded graphs and we model new and stronger (semi-)directed cut- and flow-based IP formulations for the SFP.

In the second part of this thesis we consider the two-stage stochastic survivable network design problem, an extension of the SSTP where pairs of vertices may demand a higher connectivity. Similarly to the first part we introduce new and stronger semi-directed cut-based models, apply the same decomposition along with the cut strengthening technique, and argue the validity of the newly introduced integer optimality cuts. A computational study shows the benefit, robustness, and good performance of the decomposition and the cut strengthening method.

Zusammenfassung

Das Steinerbaumproblem (STP) ist ein zentrales und gut untersuchtes graphentheoretisches kombinatorisches Optimierungsproblem. Hierbei ist ein gewichteter Graph und eine Menge von Terminalknoten gegeben und es soll eine Teilmenge der Kanten gefunden werden, welche die Terminale kostenminimal verbindet. In realen Anwendungen ist die Eingabe allerdings oft nicht mit Sicherheit gegeben oder die Eingabe hängt von zukünftigen Entscheidungen ab. Für das STP könnten z. B. die Kantenkosten, welche die Kosten für die Installation von Verbindungen darstellen, von Inflationen und Preisschwankungen abhängen. In dieser Arbeit gehen wir die Datenunsicherheit mit dem Konzept der stochastischen Programmierung an und studieren die zweistufige stochastische Variante des Steinerbaumproblems (SSTP). Bei diesem Problem definiert eine Menge an Szenarien die möglichen Ereignisse einer Zufallsvariablen; jedes Szenario hat eine Eintrittswahrscheinlichkeit und definiert Kantenkosten und eine Menge von Terminalen. Eine zulässige Lösung besteht aus einer Teilmenge der Kanten in der ersten Phase und Kanten-Teilmengen für jedes Szenario (zweite Phase), so dass jede Terminalmenge verbunden wird. Das Ziel ist die Minimierung der erwarteten Kosten. Wir betrachten zwei Ansätze um das SSTP optimal zu lösen: kombinatorische Algorithmen, insbesondere Fixed-Parameter Tractable (FPT)-Algorithmen, und Methoden aus dem Bereich der mathematischen Programmierung.

Bezüglich der kombinatorischen Algorithmen entwickeln wir einen Linearzeit-Algorithmus für Bäume, einen FPT-Algorithmus mit Parameter Anzahl Terminale und wir betrachten Graphen mit beschränkter Baumweite, für die wir den ersten FPT-Algorithmus mit kombiniertem Parameter Baumweite und Anzahl Szenarien beschreiben.

Der zweite Ansatz basiert auf der Formulierung des SSTP als ganzzahliges lineares Programm (IP). Mit Hilfe von Orientierungen führen wir neue semi-gerichtete Schnitt- und Fluss-basierte IP-Formulierungen ein, von denen wir zeigen, dass sie stärker sind als die ungerichteten Modelle aus der Literatur. Um die Modelle optimal zu lösen wenden wir einen Dekompositions-basierten zweistufigen Branch&Cut-Algorithmus an, welcher durch eine effiziente Methode zur Verstärkung der Optimalitätsungleichungen verbessert wird. Außerdem entwickeln wir neue und stärkere Ganzzahligkeits-Optimalitätsungleichungen. Die Performanz wird in einer experimentellen Studie ausgewertet, welche die Überlegenheit der neuen Formulierungen, den Vorteil der Dekomposition und die Verbesserung durch die verstärkten Optimalitätsungleichungen zeigt.

Das Steinerwaldproblem (SFP) ist ein verwandtes Problem, bei dem Mengen von Terminalknoten verbunden werden müssen. Einerseits ist das SFP eine Verallgemeinerung des STP, andererseits zeigen wir, dass es ein Spezialfall des SSTP ist. Dadurch können unsere Ergebnisse auf das SFP übertragen werden und wir präsentieren den ersten FPT-Algorithmus für Baumweiten-beschränkte Graphen und wir modellieren neue und stärkere (semi-)gerichtete Schnitt- und Fluss-basierte IP-Formulierungen für das SFP.

Im zweiten Teil dieser Arbeit betrachten wir das zweistufige stochastische Survivable Network Design Problem, eine Erweiterung des SSTP, bei dem Knotenpaare einen höheren Zusammenhang fordern können. Ähnlich zum ersten Teil führen wir neue und stärkere semi-gerichtete Schnitt-basierte Modelle ein, wenden die gleiche Dekomposition mit der Technik zur Verstärkung der Optimalitätsgleichungen an und wir zeigen die Zulässigkeit der neu eingeführten Ganzzahligkeits-Optimalitätsungleichungen. Eine experimentelle Studie zeigt die Vorteile, Robustheit und gute Performanz der Dekomposition und der Methode zur Verstärkung der Optimalitätsungleichungen.

Notations

In the following we summarize some important notations which are used in this thesis more frequently. Rarely used notations are not listed here and we refer to the related chapters. The notations are grouped by topics.

Basics:

Notation	Definition
$M^\top$, $v^\top$	transpose of matrix M, transpose of vector v
2^S	power set of a set S
$\mathbb{N}$	positive integers including 0
$\mathbb{N}^{\geq i}$	positive integers greater than or equal to i
$\mathbb{R}^{(\geq i)}$	real numbers (greater than or equal to i)
$\mathcal{P}$	polytope
$\mathcal{P}^{\mathrm{P}}_{\mathrm{f}}$	polytope of formulation f for problem P
$\mathrm{Proj}_x(\mathcal{P})$	projection of $\mathcal{P}$ onto the space of x variables

Complexity: Most problems considered in this thesis are optimization problems. To simplify and shorten the description we call an optimization problem NP-hard if the related decision problem is NP-complete.

Graphs, cuts, flows, and treewidth: The notations are introduced in Chapter 2.1. Most importantly, we use G as identifier for undirected or directed graphs, V are the vertices, E the undirected edges, and A the directed arcs. Mostly, A is used as the bidirection of E.

Parameterized algorithms: This part is self-contained and we refer to Chapter 5 for the notations. Since the algorithm on treewidth-bounded graphs plays an important role we highlight B_ℓ, the ℓth Bell-number, and tw, the treewidth of a graph.

Linear, integer, and stochastic programs: Mainly see Section 2.2 and 2.3, respectively. All vectors are column vectors. Moreover, vectors and matrices are typeset in normal letters for better readability. The only exceptions are the **0**- and **1**-vector where the dimensions are not given explicitly.

A subscript e or a for an edge or arc, e.g., x_e or z_a, gives the element of the vector x or z corresponding to edge e or arc a. We write z_a or z_{ij} for an arc $a = (i, j)$; for an edge $e = \{i, j\}$ the edge variable is x_e or $x_{\{i,j\}}$.

A superscript 0 indicates the first stage and a superscript k denotes the kth scenario, e.g., x^0 are first-stage edge variables and x^k are edge variables for scenario k. The purpose of this notation is to avoid double subscripts with scenarios and edges: an edge e will be assigned variables x^0_e and x^k_e, respectively.

Throughout this thesis we let K denote the number of scenarios and use $\mathcal{K}$ as the index set, i.e., $\mathcal{K} := \{1, \ldots, K\}$.

To shorten the notation we use the superscript "$1 \dots K$" to abbreviate K scenario vectors. For example, the vector $x^{1\dots K}$ is the transposed concatenation of the vectors $x^1, \dots, x^K$, i.e., $x^{1\dots K} = ((x^1)^\top, \dots, (x^K)^\top)^\top$. We use $0 \dots K$ analogously. Moreover, if, e.g., x^0 and $y^{1\dots K}$ are variable vectors for the first and second stage we abbreviate the vector $((x^0)^\top, (y^{1\dots K})^\top)^\top$ by $(x^0, y^{1\dots K})$.

More notations are listed in the following table.

Notation	Definition
$\tilde{x}$, $\bar{x}$, $\hat{x}$, $\check{x}$	solution vectors, i.e., assignments to the variable vector x
k	index of a scenario
p^k	probability of scenario k
(LP)	primal linear program
(D:LP)	dual of program (LP)
($\mathrm{P}^{\mathrm{rel}}$)	relaxed program of P (all integer variables are relaxed)
($\mathrm{P}^{\mathrm{rel}:x}$)	program P with relaxed x variables
$L^{(k)}$	lower bound for the second-stage cost (of scenario k)
θ	variable for second-stage cost
θ^k	variable for second-stage cost of scenario k
$\tilde{q}^{(k)}$	value of the second-stage cost (of scenario k) w.r.t. the first-stage solution $\tilde{x}^0$
$Q^{(k)}(\tilde{x}^0)$	second-stage cost function (for scenario k) w.r.t. the first-stage solution $\tilde{x}^0$, i.e., $Q^{(k)}(\tilde{x}^0) = \tilde{q}^{(k)}$
$R^{(k)}(\tilde{x}^0)$	relaxed second-stage cost function (for scenario k) w.r.t. $\tilde{x}^0$

IP formulations for (stochastic) network design problems: Mainly see Chapter 3, Chapter 6, and Section 9.2. We use the following variable identifiers: f is used for flow variables, x always denotes undirected edge variables, z is an identifier for directed arc variables, and y is used for undirected or directed edge variables.

For an edge set E with edge variables x and x' we use the notations $x(E) := \sum_{e\in E} x_e$ and $(x + x')(E) := \sum_{e\in E} x_e + x'_e$. The analogue notation is used for directed arcs.

Let S be a valid cut set in a graph. With undirected edge variables x and directed arc variables y we use $(x + y)(\delta^-(S)) := x(\delta(S)) + y(\delta^-(S)) = \sum_{(i,j)\in\delta^-(S)} x_{\{i,j\}} + y_{ij}$. The analogue notation is used for the outgoing semi-directed cut $\delta^+(S)$.

Some important notations are given in the following table.

Notation	Definition
$\bar{G}$	bidirection of an undirected graph G
$c^{(k)}$	edge/arc costs (of scenario k)
c^*	expected edge/arc costs
r	designated root node
r^k	designated root node in scenario k *or* for terminal set k
$\rho^{(k)}$	connectivity requirement matrix *or* vector (for scenario k)
$\mathcal{R}_i^{(k)}$	set of vertices with connectivity requirement i (in scenario k)
t	a terminal

Notation	Definition
T	set of terminals
T_r	$T \backslash \{r\}$
T^k	terminal set of scenario k *or* kth terminal set
t^*	$\sum_{k \in \mathcal{K}} \lvert T^k \rvert$
T_r^k	$T^k \backslash \{r^k\}$ *or* $T^k \backslash \{r\}$
t_r^*	$\sum_{k \in \mathcal{K}} \lvert T_r^k \rvert$
V_r	$V \backslash \{r\}$
V_r^k	$V \backslash \{r^k\}$
$\mathcal{S}_1^{(k)}, \mathcal{S}_{\geq 2}^{(k)}$	set of valid (directed) cuts (for scenario k) with right-hand side 1 and ≥ 2, respectively
$f^{(k)}(S)$	connectivity requirement function for a cut S (in scenario k)

Two-stage branch&cut: See Chapter 7 and Section 9.3.

Notation	Definition
((R)MP$_\text{f}$)	(relaxed) master problem of model f
((R)SP$_\text{f}$)	(relaxed) subproblem of model f
(D:RSP$_\text{f}$)	dual subproblem of model f
S	directed cut *or* 1-index set of a solution
$\mathcal{S}^{(k)}$	set of valid (directed) cuts (for scenario k)
α_S^k	dual variable for a directed cut S and scenario k
β_e^k, β_a^k	dual variable for a capacity constraint and scenario k and edge e or arc a, respectively
γ^k, τ^k	dual variables for scenario k

Computational studies: Notations are given in Section 8.3 and Section 10.1.

Abbreviations: The following table lists the used abbreviations.

Abbreviation	Definition
2BC	two-stage branch&cut
AGL	aggregated L-shaped optimality cut
avg	average
b&b	branch&bound (algorithm)
b&c	branch&cut (algorithm)
BIP	binary program
DA	direct approach
dc	directed cut
DE	deterministic equivalent
df	directed flow
dSTP	directed Steiner tree problem

Abbreviation	Definition
EEV	expected result of EV solution
EV	expected value (problem)
FPT	complexity class fixed parameter tractable
(G)SEC	(generalized) subtour elimination constraint
IP	integer linear program
LP	linear program
med	median
MIP	mixed integer linear program
MP	master problem
MST	minimum spanning tree (problem)
MW	Magnanti & Wong
ncSNDP	node-connectivity survivable network design problem
NDP	network design problem
NoCS	no cut strengthening
NoNG	no no-good cuts
NP	complexity class nondeterministic polynomial time
opt	optimum
P	complexity class polynomial time
PCSTP	prize-collecting Steiner tree problem
rel	relaxation
rev	reverse
RMP	relaxed master problem
RSP	relaxed subproblem
rSSTP	rooted stochastic Steiner tree problem
sdc	semi-directed cut
sdf	semi-directed flow
SFP	Steiner forest problem
SNDP	survivable network design problem
sol	solution
SP	subproblem
SSNDP	stochastic survivable network design problem
SSTP	stochastic Steiner tree problem
std. dev.	standard deviation
STP	Steiner tree problem
sUp	speedup
TSP	traveling salesman problem
uc	undirected cut
uf	undirected flow
VSS	value of stochastic solution

Part I

Introduction & preliminaries

Chapter 1

Introduction

1.1 Motivational example

Although farmers and news vendors are mostly the first choice working class for introducing the field of stochastic optimization we like to consider a young manager working at a telecommunication company. Our manager is assigned a new task concerning an important customer whose company wants to relocate to a new headquarter building. Both companies already signed a contract about a high-speed internet connection using fiberglass. Since all pre-existing cables are copper cables they need proper upgrades. Unfortunately, it is unsure which one of two possible new headquarter sites will be chosen. However, the two possible locations are known and moreover, it is known when the decision will be made.

To fulfill the contract it has to be decided which cables to upgrade—and when. Since the new headquarter location is uncertain there are four possible strategies:

(S1) Upgrade the cheapest connections to both possible headquarter locations now.

(S2) Wait until the customer makes his decision and upgrade the cheapest connection afterwards.

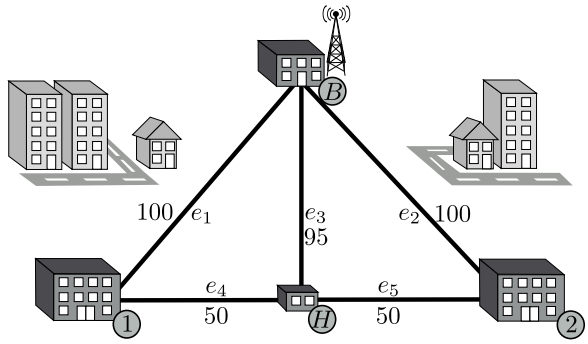
(S3) Combine both ideas by upgrading some connections now and complete the remaining parts after the decision is made.

(S4) "Gamble" and install an upgrade to one of the two possible sites now.

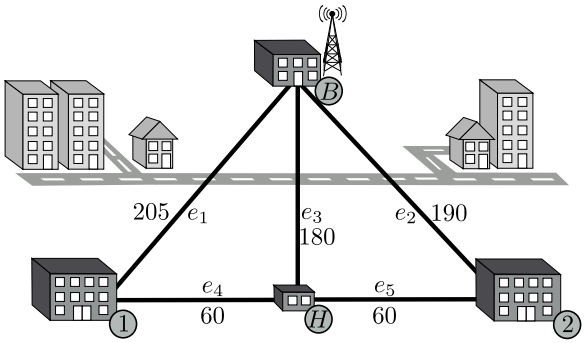
Since the chances for both possible sites are 50% strategy (S4) is too risky and ruled out pre-emptively.

To find the best solution our manager analyzed the situation carefully and identified the important parts of the current network topology, as depicted in Figure 1.1(a). He calculated the current upgrading costs and estimated costs for the future; these costs are given in Figure 1.1 and in the following table.

	connection				
upgrading costs	e_1	e_2	e_3	e_4	e_5
today	100	100	95	50	50
future	205	190	180	60	60



(a)



(b)

Figure 1.1: (a) The current situation and the underlying topology for the network. The two buildings with captions "1" and "2", respectively, depict the possible sites for the new headquarter. The building labeled "B" is the backbone and "H" an additional hub. Upgrading costs for the five connections $e_1, \ldots, e_5$ are given for the current situation as well as for the future, as depicted in (b). Here, a new street highly influences the costs for connections e_1, e_2, and e_3.

The cost for connecting sites 1 or 2 directly by connections e_1 and e_2, respectively, is 100 each. The detour over some additional hub costs 95 (e_3) plus $2 \cdot 50$ for the connection to the sites via e_4 and e_5. Therefore, the optimum solution for strategy (S1) has cost 195.

Since our manager is familiar with the basics of *integer linear programming* he formulated the mathematical problem for strategy (S1) to prove the optimality of this solution. By using decision variables $x_1, \ldots, x_5$ for the connections $e_1, \ldots, e_5$, respectively, he came up with the following model. For $i \in \{1, \ldots, 5\}$ variable $x_i = 1$ if and only if connection e_i is upgraded, otherwise $x_i = 0$. The restrictions (1.1) to (1.5) enforce the connections from site 1 and 2 to the backbone and the objective is to minimize the overall cost:

$$\begin{aligned} \min\ & 100x_1 + 100x_2 + 95x_3 + 50x_4 + 50x_5 \\ \text{s.t. } & x_1 + x_4 \geq 1 && (1.1)\\ & x_2 + x_5 \geq 1 && (1.2)\\ & x_1 + x_2 + x_3 \geq 1 && (1.3)\\ & x_1 + x_3 + x_5 \geq 1 && (1.4)\\ & x_2 + x_3 + x_4 \geq 1 && (1.5)\\ & x_1, \ldots, x_5 \in \{0, 1\} && (1.6) \end{aligned}$$

The optimum solution for this integer program is $x^\top = (x_1, \ldots, x_5)^\top = (0, 0, 1, 1, 1)^\top$ with cost 195 which proves his preceding arguments concerning strategy (S1).

However, this solution seems inefficient and too conservative since one upgraded connection (e_4 or e_5) always will be redundant. Therefore, our manager tries to find optimal solutions for strategies (S2) and (S3), respectively.

Due to new streets in the area—influencing the construction costs—and higher expected prizes for fiberglass all future costs are increasing: the estimated upgrading costs for all connections are given in Figure 1.1 (b) and the previous table.

Hence, applying strategy (S2) obviously induces cost 190 or 205, respectively, by using one of the direct connections and depending on the new location. On the one hand, the worst-case cost of 205 for the second strategy is worse than connecting both sites now as in strategy (S1). On the other hand, in the best case the cost is only 190 and this solution would be preferable. But which strategy is better? Our manager gave some thought to the interpretation of best- and worst-case and decided that this point of view is not adequate since the actual location is currently not known.

Instead, by using the mathematical notion of expectation and the assumption that both locations are equally probable he calculated the *expected cost* of this solution as $0.5 \cdot 205 + 0.5 \cdot 190 = 197.5$ (connections e_1 or e_2, respectively). Therefore, our manager concludes that the first approach—with (expected) cost 195—is preferable over the second one.

However, this is still not satisfying because both of the first two strategies concentrate only on exactly one point in time: Either all connections are upgraded now (strategy (S1)) or only the necessary connection is upgraded in the future (strategy (S2)). The purpose of the third strategy is to find a compromise and use the most profitable connections—now and in the future. Notice that both previously described solutions already are extreme solutions for this third strategy.

The (reasonable) solutions for strategy (S3) differing from the two previously described solutions are as follows:

(i) Upgrade connection e_3 now at cost 95 and, after the customer makes his decision connections, e_4 or e_5, respectively, can be used and upgraded at cost 60: The expected cost for this solution is $95 + 0.5 \cdot 60 + 0.5 \cdot 60 = 155$.

(ii) To cover site 1 upgrade e_1 now (cost 100). Later on, if site 1 is chosen everything is fine. On the other hand, if site 2 is chosen use e_4 and e_5 at cost 120 (expected cost 60). Hence, the expected cost is $100 + 0.5 \cdot (60 + 60) = 160$.

(iii) Analogue to solution (ii) one can upgrade e_2 now and use e_4 and e_5 in the future if site 1 is chosen; the expected cost is 160, too.

Hence, the best solution for strategy (S3) is option (i) with expected cost 155. Notice that due to identical costs of e_4 and e_5 this cost is not only the expected cost but this cost is certain.

Again, to prove optimality, our network planner formulated the corresponding *integer program* by using decision variables $x_1, \ldots, x_5$ for the connections $e_1, \ldots, e_5$ if upgraded now and $y_1^1, \ldots, y_5^1, y_1^2, \ldots, y_5^2$ for the connections in the future, respectively. The latter set of variables has an additional upper index indicating which site has to be connected, i.e., y^1 models decisions if location 1 is chosen and analogously y^2 for the second location. The

objective is to minimize the expected cost (future costs are weighted by probability) and the constraints ensure that the necessary connections are upgraded.

$$
\begin{aligned}
\min\ & 100x_1 + 100x_2 + 95x_3 + 50x_4 + 50x_5 + \\
& 0.5(205y_1^1 + 190y_2^1 + 180y_3^1 + 60y_4^1 + 60y_5^1) + \\
& 0.5(205y_1^2 + 190y_2^2 + 180y_3^2 + 60y_4^2 + 60y_5^2) \\
\text{s.t. } & x_1 + x_4 + y_1^1 + y_4^1 \geq 1 && (1.7)\\
& x_1 + x_2 + x_3 + y_1^1 + y_2^1 + y_3^1 \geq 1 && (1.8)\\
& x_1 + x_3 + x_5 + y_1^1 + y_3^1 + y_5^1 \geq 1 && (1.9)\\
& x_2 + x_5 + y_2^2 + y_5^2 \geq 1 && (1.10)\\
& x_1 + x_2 + x_3 + y_1^2 + y_2^2 + y_3^2 \geq 1 && (1.11)\\
& x_2 + x_3 + x_4 + y_2^2 + y_3^2 + y_4^2 \geq 1 && (1.12)\\
& x_1, \ldots, x_5, y_1^1, \ldots, y_5^1, y_1^2, \ldots, y_5^2 \in \{0, 1\} && (1.13)
\end{aligned}
$$

Solving this *integer program* gives an optimum solution with $x_3 = 1$, $y_4^1 = 1$, and $y_5^2 = 1$ (all other variables are 0) with cost 155 and which induces solution (i). Compared to the previously mentioned solutions this one offers the minimum expected cost and this solution is considerably cheaper than the solutions of the straight-forward strategies (S1) and (S2).

Now, imagine our manager would have *perfect information* and would be able to obtain all the uncertain information beforehand. Then, he obviously would connect either site 1 or 2 directly with cost 100. The difference between the optimal expected cost and the cost under perfect information, i.e., $155 - 100 = 55$, is called *value of perfect information*. This value measures the upper bound our manager would invest for this perfect information while still being able of making a good and profitable decision.

The described optimization problem is an instance of the *two-stage stochastic Steiner tree problem*, which plays a central role in this thesis. The term *two-stage* refers to the two points in time with possible decisions. In the *first stage*—also said to be today, or "on Monday"—profitable connections are installed and in the *second stage*—in the future, "on Tuesday"—it is possible to take corrective actions, also called *recourse (actions)*. Thereby, all connectivity requirements of all possible *scenarios*—the possible outcomes in the future—have to be satisfied.

The adjective *stochastic* implies that some input values are "uncertain" and depend on the realization of random variables. In this thesis, we assume all random variables having finite support. Hence, the possible outcomes can always be modeled by a finite number of scenarios.

In our example, the first stage defines the current upgrading decisions under todays costs. The second stage is in the future after the customer has announced his decision. Since future edge costs do not depend on the new location it consists of two scenarios defined by the two possible headquarter sites. Notice that if upgrading costs would depend on another random variable, e.g., new streets in the area may be built or not, there would be at least four scenarios.

1.2 Introduction

Solving stochastic optimization problems. Assuming all values of an optimization problem being given is the classical view on algorithmic problems. In fact, it is a requirement for most algorithms and without complete and valid input data algorithms cannot work. However, when trying to apply theoretically developed algorithms in practice this premise is often impossible—or at least difficult—to satisfy.

But, what if necessary input is missing or uncertain? For example, consider a truck driver who is working for a cargo company and delivering and picking up shipments at customers. What if the truck driver has to plan his tour for tomorrow without precise knowledge of the customers to visit? Or what if the size or weight of the cargo that needs to be picked up is unknown such that the truck weight limit might be exceeded? Or, maybe even more unpredictable, what is when unexpected traffic jams influence the driving times and time limit restrictions need to be kept?

For planning the tours of several trucks, maybe by using a formulation of the vehicle routing problem, demands, customers, and driving times are necessary input values. If these values are missing, any classical algorithm is inapplicable.

On the other hand, if the cargo company has a long standing experience and can estimate tomorrows customers, demands, and traffic jams during rush hours is it possible to use algorithms to compute "better" solutions by using this information?—Like in our motivational example if the company can predict a set of possible scenarios which can occur.

The intuitive and direct approach would be to adopt a standard algorithm by simply using expected or average values. Unfortunately, this solution method does mostly not give an actual optimum solution. Moreover, this method is not easily adaptable for many problems. E.g., consider the telecommunication manager where the expected value would yield expected connectivity requirements of 0.5 for each possible headquarter site: Despite the fact that upgrading a fraction of a cable is not reasonable classical algorithms require definite, e.g., binary information.

During the last decades several approaches have been proposed how to deal with uncertainty. The two most prominent ones are *stochastic programming* and *robust optimization*. Comparing both ideas is difficult because they have diverging viewpoints and follow different objectives. Moreover, robust optimization developed a lot over the years and has spawn many differing robustness concepts; with some of them even converging to *stochastic optimization*.

In this thesis, we use and focus on the concept of *two-stage stochastic optimization problems*. Informally, the main aspects, assumptions, and requirements are:

- Two points in time are considered: the *first stage* happens today and some decisions can be made "here-and-now". In the *second stage*, which is a predefined point of time in the future, corrective "wait-and-see" actions, also called recourse actions, can be taken.
- The data of the first stage is known and defined entirely.
- The future data is "uncertain"; however, it can be modeled by a finite number of *scenarios*. Each scenario defines a possible setting in the future and has a certain probability of being realized. Moreover, all possible outcomes are known, i.e., the probabilities of the scenarios sum up to one.

- The goal is to optimize the sum of the first-stage objective plus the objective of the second stage with the latter one being an expected value over all scenarios.

- A feasible solution implies feasibility for all scenarios.

In general, the optimization of *two-stage stochastic programs* is more challenging than the optimization of classical *deterministic* programs. The classical problems have (only) one decision horizon and imply the viewpoint of all the input data being known in advance.

In this thesis, we focus on the exact solving of *two-stage stochastic network design problems* by applying the following three methods:

- Find strong formulations for the problems as *stochastic (mixed) integer programs* and use techniques from *mathematical programming*.

- Develop a *parameterized algorithm*.

- Identify easier special cases and design efficient *combinatorial algorithms*.

One promising way of solving large-scale *linear programs* exactly is by using *Benders' decomposition*. For *stochastic integer programs* the approach utilizing this decomposition is called *integer L-shaped algorithm*. We adapt this method for solving the *two-stage stochastic Steiner tree* and *stochastic survivability network design problem*.

Another efficient approach for finding exact solutions are *parameterized* algorithms for *fixed parameter tractable* problems. In this thesis we present algorithms based on the *tree decomposition* for *stochastic Steiner tree problems* and we discuss approaches for other parameters like the overall number of terminals.

(Stochastic) network design problems. Studying *network design problems* can be motivated easily because these problems arise in various important areas, e.g.:

- In *telecommunications* there are many types of *network design problems*. Like in our introductory example telecommunication companies want to install/upgrade profitable connections between the customers. Sometimes the resulting network topology needs to be resistant to failures which leads to *survivability problems*. Another related type of problems are *augmentation problems* where given networks are modified to reach a certain survivability. All these problems aim for cost-minimum networks. If, in addition, connected costumers yield profits these problem variants are called *prize-collecting*.

- The previously described problems arise in several fields like *computer networks*, *street* or *railway* planning, or *transportations*.

- In *VLSI chip design* the problem of computing the best wiring can be modeled as a *network design problem*.

- Another area for *network design problems* can be found in biology, in particular in the reconstruction of *phylogenetic trees*, see e.g., Catanzaro, Labbé, Pesenti, and Salazar González [34].

Besides the numerous applications in practice *network design problems* belong to the set of classical combinatorial optimization problems and have been studied intensively over the past decades. Some of the most famous *network design problems* are the *minimum spanning tree problem*, the *shortest path problem*, the *Steiner tree problem*, or the *traveling salesman problem*, see Section 3.1.

Remarks. Optimization under uncertainty was firstly considered by Dantzig [56] for *linear programs* in 1955. Since then this field developed rapidly and several concepts of uncertainty, theoretical results, and algorithms were published. In this thesis we focus on *stochastic programming*—for an overview we refer the reader to the various textbooks, e.g., Birge and Louveaux [20], Shapiro, Dentcheva, and Ruszczyński [161], or Kall and Wallace [104]. Further references are given in Section 2.3.3.

Most of the classical textbooks also discuss other concepts and, in particular, introduce the approach of *robust optimization*. Robust optimization and stochastic programming are both approaches for optimization under uncertainty but attack the problem at different angles. Moreover, both concepts have their advantages and drawbacks.

In general, stochastic programming depends on knowing the probability distribution of a random variable. In particular, the possible outcomes—including their realization probabilities—have to be available and there exist two decision horizons. Here, the goal is to find a solution which is feasible for all outcomes and which optimizes the expected value.

Contrary to stochastic programming, robust optimization does not require such a probability distribution. Moreover, in its basic setting there is only one decision horizon. It therefore concentrates on the worst case and the decisions have to be feasible for all realizations of the uncertain data. However, there are several robustness concepts with some of them even converging to stochastic programming. For example, the book by Ben-Tal, El Ghaoui, and Nemirovski [14] and the PhD thesis by Goerigk [76] provide a good overview.

1.3 Overview and relevant publications

Overview. This thesis is divided into four parts. Part I contains the introduction and the required preliminaries introducing graphs as well as linear, integer, and stochastic programming (Chapter 1 and 2). Moreover, we recall the definitions and IP formulations of relevant network design problems in Chapter 3.

Part II deals with the stochastic Steiner tree problem (SSTP). We start in Chapter 4 with the definitions of the SSTP and the rooted SSTP (rSSTP), some observations, and the description of new linear-time algorithms for the stochastic problems on trees.

Chapter 5 is dedicated to fixed parameter tractable (FPT) algorithms. Section 5.1 contains a reduction from the rSSTP to the directed STP leading to an FPT algorithm for the rSSTP which is parameterized by the overall number of terminals. The results are contained in the diploma thesis by Kurz [115] but have been developed independently by the author during the preparation of the mentioned diploma thesis; moreover, they are published in the conference paper [116]. In Section 5.2 we mention an FPT algorithm for partial 2-trees which is due to Bökler [26] and which is published in [25]. Section 5.3 contains an NP-hardness proof for graphs with treewidth 3 by reducing the SSTP to the Steiner forest

problem—also from [25, 26]. Afterwards, we investigate the general case: Section 5.4 describes an FPT algorithm for the deterministic STP on treewidth-bounded graphs. This algorithm is published in the journal article [41] and the related conference paper [40]. In Section 5.5 and 5.6 we present new and unpublished extensions of this algorithm for the SSTP and rSSTP, respectively. Both algorithms are FPT algorithms parameterized by the combination of treewidth and number of scenarios.

Chapter 6 is dedicated to the IP formulations for the stochastic STPs and the Steiner forest problem (SFP). After recalling the undirected models in Section 6.1, the new semi-directed models for the SSTP and the directed models for the rSSTP are described in Section 6.2 and 6.3, respectively. In Section 6.4 we compare the strength of all introduced models. The models and their comparisons are contained in the technical report [176] and the stronger semi-directed model is described in the conference paper [27]. In Section 6.5 we adopt the ideas of the stochastic models and introduce new and stronger semi-directed and directed models for the SFP; these results are contained in the manuscript [133].

The decomposition of the stochastic models is described in Chapter 7. The basic parts, i.e., the master problem and subproblem, and the L-shaped optimality cuts are described in Section 7.1 and 7.2 and also (briefly) in [27]. Moreover, we introduce a method for strengthening the L-shaped cuts which is described (for the stochastic survivable network design problem) in the journal article [127] and the related conference paper [126]. Section 7.2 contains unpublished results. Here, we develop another heuristic for improving the optimality cuts, describe new and stronger integer optimality cuts, give new cut-based constraints, and consider the disaggregation of all constraints.

Chapter 8 describes details of our implementation, the generated instances, and as main part of this chapter, the results of a new and comprehensive computational study. The experiments from [27] cover only a small portion of our new study.

Part III of this thesis deals with the stochastic survivable network design problem (SSNDP). Chapter 9 introduces the problem, presents new and stronger IP formulations, and describes the decomposition approach. This chapter is mainly based on the two publications [126, 127]. The unpublished parts are some results adopted from the SSTP, i.e., relaxing the first-stage variables does not influence overall integrality and the newly introduced integer optimality cuts, respectively.

In Chapter 10 we present the results of a computational study for the SSNDP. Again, some results are published in [127]. However, this thesis contains more details and some more experiments on, e.g., no-good cuts, the new integer optimality cuts, or larger instances.

Part IV contains Chapter 11 with the conclusion, discussion, and outlook, and a list of open problems.

Relevant publications. The mentioned publications containing results presented in this thesis are listed in the following (sorted by publication date):

[133] François Margot, Daniel Schmidt, and Bernd Zey. MIP formulations for the Steiner forest problem. *Manuscript*, in preparation, 2017.

[127] Ivana Ljubić, Petra Mutzel, and Bernd Zey. Stochastic survivable network design problems: Theory and practice. In *European Journal on Operations Research (EJOR)*, 256(2):333–348, 2017

[176] Bernd Zey. ILP formulations for the two-stage stochastic Steiner tree problem. *CoRR*, arXiv:1611.04324, 2016

[126] Ivana Ljubić, Petra Mutzel, and Bernd Zey. Stochastic survivable network design problems. In *International Network Optimization Conference (INOC)*, volume 41 of *Electronic Notes in Discrete Mathematics*, 2013, pages 245–252

[116] Denis Kurz, Petra Mutzel, and Bernd Zey. Parameterized algorithms for stochastic Steiner tree problems. In *Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, volume 7721 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 143–154, 2013

[41] Markus Chimani, Petra Mutzel, and Bernd Zey. Improved Steiner tree algorithms for bounded treewidth. In *Journal of Discrete Algorithms (JDA)*, 16:67–78, 2012

[25] Fritz Bökler, Petra Mutzel, and Bernd Zey. The stochastic Steiner tree problem on partial k-trees. In *Proceedings of the Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, NOVPRESS Brno, 2012

[40] Markus Chimani, Petra Mutzel, and Bernd Zey. Improved Steiner tree algorithms for bounded treewidth. In *International Workshop on Combinatorial Algorithms (IWOCA)*, volume 7056 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 374–386, 2011

[27] Immanuel Bomze, Markus Chimani, Michael Jünger, Ivana Ljubić, Petra Mutzel, and Bernd Zey. Solving two-stage stochastic Steiner tree problems by two-stage branch-and-cut. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 6506 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 427–439, 2010

Chapter 2

Preliminaries

This chapter introduces the required preliminaries and notations used in this thesis; relevant references are given at the beginning of each subtopic. In Section 2.1 we start with *graphs, networks, cuts, flows*, and *tree decompositions*. We cover *linear and integer programming* in Section 2.2. Here, we focus on *integer programs* and describe the *branch&cut* algorithm. The main part is Section 2.3 which introduces the important concepts of *stochastic programming*. We define *two-stage stochastic linear* and *integer programs* and describe the *L-shaped*, *integer L-shaped*, and *two-stage branch&cut algorithm*.

2.1 Graphs and networks

The definitions concerning *graphs* are based on the existing literature, e.g., Diestel [61], Jünger and Mutzel [102], and Tollis, Di Battista, Eades, and Tamassia [167].

Undirected Graphs. An *undirected graph* $G = (V, E)$ consists of the finite set V of *vertices*—also called *nodes*—and the finite multiset E of *(undirected) edges*. Each edge $e \in E$ is an unordered pair of vertices $e = \{u, v\}, u, v \in V$. We use $n := |V|$ and $m := |E|$.

For an edge $e = \{u, v\}$ the vertices u and v are its *endpoints*, u and v are *adjacent*, and both u and v are *incident* to e. The set of incident edges of a vertex v is denoted by $\delta(\{v\})$, or short $\delta(v)$. The *neighbors* of a vertex are the adjacent vertices and the *degree* of a vertex is its number of incident edges.

An edge $e = \{v, v\}$ is a *self-loop*, edges e_1 and e_2 are *adjacent* iff $|e_1 \cap e_2| = 1$, and two edges with the same endpoints are called *multiedges*. A graph without self-loops and multiedges is a *simple (undirected) graph*.

The *density* of a non-empty graph $G = (V, E)$ is defined as $|E|/|V|$.

The vertices of a graph are assumed to be uniquely numbered from 1 to n by a bijective function $id\colon V \to \{1, \dots, n\}$. We often use this identifier as lower index such that the ith vertex in the list sorted by vertex identifiers is vertex v_i with $i = id(v_i)$. Moreover, vertices and integers are used interchangeably, i.e., i and v_i are references for the same vertex.

Subgraphs. A graph $G' = (W, F)$ is a *subgraph* of $G = (V, E)$ if $W \subseteq V$ and $F \subseteq E$. The *vertex-induced subgraph* of a vertex set $V' \subseteq V$ is the subgraph $G' = (V', E')$ with $E' := \{\{u, v\} \in E \mid u, v \in V'\}$. Analogously, the *edge-induced subgraph* of an edge set $E'' \subseteq E$ is the subgraph $G'' = (V'', E'')$ with $V'' := \{v \in V \mid \exists e \in E''\colon v \in e\}$. The vertex-

and edge-induced subgraphs of V' and E'' are denoted by $G[V']$ and $G[E'']$, respectively. If a *connected* subgraph H of G contains all vertices of G then H is a *spanning subgraph*.

Paths and connectivity. A *path* $P = \langle v_1, v_2, \ldots, v_k \rangle$ in a graph $G = (V, E)$ is a sequence of $k \geq 2$ adjacent vertices, i.e., $\{v_i, v_{i+1}\} \in E, \forall i \in \{1, \ldots, k-1\}$. Hereby, vertices v_1 and v_k are the designated *endpoints* of the path P. The *length* of a path is its number of edges. A *cycle* $C = \langle v_1, \ldots, v_{k-1}, v_1 \rangle$ of length k is a path $P = \langle v_1, \ldots, v_{k-1} \rangle$ with an additional edge connecting v_{k-1} and v_1. We analogously allow a *path* and a *cycle* to be defined by a valid edge set, i.e., a *path* is a sequence of $k \geq 1$ adjacent edges $P = \langle e_1, e_2, \ldots, e_k \rangle$ if $\langle v_1, \ldots, v_k, v_{k+1} \rangle$ is a path as defined before with $\{v_i, v_{i+1}\} = e_i \in E, \forall i \in \{1, \ldots, k\}$.

Two vertices $u, w \in V$ are *connected* if there exists a path with endpoints u and w. A graph, or a set of vertices, is *connected* if it consists of one vertex or if all pairs of vertices are connected; otherwise it is called *disconnected*. The maximal connected subgraphs of a graph are called *connected components*.

For an undirected graph $G = (V, E)$ the removal of a vertex set $\emptyset \neq S \subseteq V$, denoted by $G - S$, results in an undirected graph $G' = (V', E')$ with $V' := V \backslash S$ and $E' := E \backslash \{\{u, v\} \in E \mid u \in S \vee v \in S\}$. Notice that G' is a subgraph of G; in fact it is the vertex-induced subgraph of $V \backslash S$.

An undirected graph is *k-(node)-connected*, for some $k \in \mathbb{N}, k > 0$, if $G - S$ is connected for any vertex set $S \subseteq V$ with $|S| < k$. Hence, for a 2-(node)-connected graph, also called *biconnected graph*, there exists no *cutvertex*; a *cutvertex* is a vertex whose removal increases the number of connected components.

If the number of connected components increases after the removal of an edge then this edge is called a *bridge*. If no such edge exists the graph is said to be *edge-biconnected* or *bridge-connected*. A graph G is k-edge-connected if the removal of at most $k - 1$ edges does not disconnect G.

Directed graphs. *Directed graphs* are defined similarly to undirected graphs and differ in the edge set: Instead of an unordered pair of vertices a *directed edge* $a \in E$ is an ordered pair of vertices $a = (u, v), u, v \in V$. To distinguish directed from undirected graphs we call the directed edges *arcs* and denote the *arc set* by A.

If any following definition concerns only one type of graph this is explicitly mentioned. Definitions on graphs without further specifier are valid for both types.

By ignoring the ordering of each arc for a directed graph $G = (V, A)$ we obtain the *underlying (undirected) graph* $G' = (V, E)$ consisting of the same vertex set and the edge set $E := \{\{u, v\} \mid (u, v) \in A \vee (v, u) \in A\}$. Conversely, a directed graph $G = (V, A)$ is an *orientation* of an undirected graph $G = (V, E)$ if it contains exactly one directed version of each undirected edge, i.e., $\forall \{u, v\} \in E \colon |\{(u, v), (v, u)\} \cap A| = 1$. Last but not least, the *bidirection* of an undirected graph $G = (V, E)$ is the directed graph $\bar{G} = (V, A)$ which contains both orientations for each edge, i.e., $A = \{(u, v), (v, u) \mid \{u, v\} \in E\}$, cf. Figure 2.1.

Since undirected and directed graphs are closely related most of the introduced notations are also used for directed graphs. In the following we recall the most important—differing, new, and similar—ones.

The *endpoints* u, v of an arc $a = (u, v)$ are called *source* and *target*, respectively. Since arcs are ordered pairs of vertices a is *outgoing from* or *leaves* u, and a is *incident to* or *ingoing into* v.

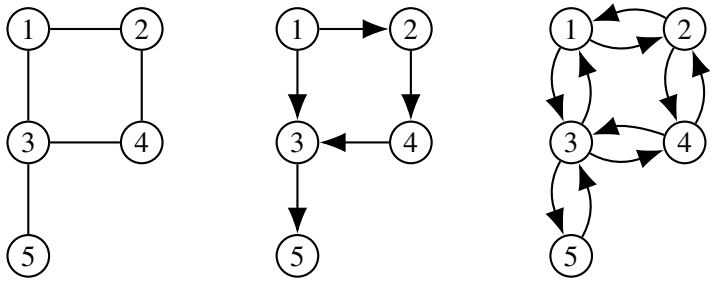


Figure 2.1: (a) An undirected graph, (b) a possible orientation, and (c) its bidirection. Conversely, the graph depicted in (a) is the underlying graph of the directed graph depicted in (b) and (c), respectively.

The set of ingoing arcs of a vertex v is denoted by $\delta^-(\{v\})$, or short $\delta^-(v)$, and analogously, the set of outgoing arcs is denoted by $\delta^+(\{v\})$, or short $\delta^+(v)$. The *indegree* of a vertex v is its number of ingoing arcs, i.e., $|\delta^-(v)|$, and the *outdegree* the number of outgoing arcs, i.e., $|\delta^+(v)|$. Vertices with indegree 0 are called *sources* and vertices with outdegree 0 are called *sinks*. *Self-loops*, *multiedges*, and *simple graphs* are defined analogously for directed graphs.

A *directed path* $P = \langle v_1, v_2, \ldots, v_k \rangle$ in a directed graph $G = (V, A)$ is a sequence of $k \geq 2$ vertices with $(v_i, v_{i+1}) \in A, \forall i \in \{1, \ldots, k-1\}$. Again, vertices v_1 and v_k are the *endpoints* of the path P, the *length* is its number of arcs, and a path can also be given by its arc sequence. A *directed cycle* is a directed path with identical endpoints.

Vertex u is *connected* to vertex w in G if there exists a directed path $P = \langle v_1, \ldots, v_k \rangle$ with $u = v_1$ and $w = v_k$. A directed graph is *strongly connected* if all ordered pairs of vertices are connected.

Weighted graphs. A *weighted graph* $G = (V, E)$ has each edge $e \in E$ assigned a real number $c_e \in \mathbb{R}$ and is denoted by $G = (V, E, c)$; the vector of *edge weights*—a column vector of size $|E|$—is denoted by c. In this thesis we use the vector notation $c^\top = (c_1, \ldots, c_{|E|})$ with edge identifiers as indices. Depending on the considered problem the weights are also called *costs* or *capacities*. The sum over the edge costs of a subset $E' \subseteq E$ is denoted by $c(E') = \sum_{e \in E'} c_e$.

For an undirected weighted graph $G = (V, E, c)$ the *weighted bidirection* is the bidirected graph $\bar{G} = (V, A)$ with each arc assigned the same cost as the corresponding undirected edge, i.e., $c_{ij} := c_{ji} := c_e, \forall e = \{i, j\} \in E$. We use the same identifier c for edges and arcs and denote the *weighted bidirection* by $\bar{G} = (V, A, c)$.

Forests and trees. An undirected graph is *acyclic*, and called a *forest*, if it contains no cycles. A *tree* is a forest consisting of exactly one connected component. We use the naming convention of calling the vertices of a tree *nodes*. A node of a tree with degree one is called a *leaf*.

Sometimes it is convenient to consider a tree being hierarchical by defining a special *root node* which is mostly denoted by r. Then, this *rooted* and *directed tree* is obtained by orienting the underlying tree. Edges are directed from the root *outwards* such that the root

has indegree 0 and all other nodes have indegree 1. Notice that the path connecting the root node with any other node is unique. For each node $u \neq r$ the predecessor on the path from the root node is called *parent* and conversely, u is a *child* of its parent. A directed tree rooted at r is also called *(r-)arborescence*.

Cuts. A *cut* S in a graph $G = (V, E)$ is a non-empty subset of vertices $\emptyset \neq S \subset V$. The cut is called *undirected* or *directed*, respectively, depending on the type of graph.

We denote by $\delta(S)$ the edges *lying*, *contained in*, or *crossing* an undirected cut S which are the very edges having one endpoint in the set S and the other endpoint in the complementary set $V \backslash S$, i.e., $\delta(S) := \{e \in E \mid |e \cap S| = |e \cap V \backslash S| = 1\}$.

A vertex set $\emptyset \neq S \subset V$ in a directed graph $G = (V, A)$ defines two *directed cuts*: The arc set of the *ingoing cut*—or *entering cut*—of set S is denoted by $\delta^-(S)$ with $\delta^-(S) := \{(u, v) \in A \mid v \in S, u \in V \backslash S\}$. The arcs of the *outgoing cut*—or *leaving cut*—are defined analogously as $\delta^+(S) := \{(u, v) \in A \mid u \in S, v \in V \backslash S\}$.

The *weight* of a cut S is the sum over all weights of the contained edges and is defined as $c(\delta(S))$ in the undirected case, and $c(\delta^-(S))$ or $c(\delta^+(S))$ in the directed case. A *minimum cut* in a graph is a cut with minimum weight.

Given two vertices $s, t \in V, s \neq t$, a (minimum) cut S *separating* s and t, i.e., $s \notin S, t \in S$, is called *(minimum) s-t-cut*.

Flows and flow networks. A *flow network* $N = (V, A, c)$ is a directed weighted graph $G = (V, A)$ with non-negative arc capacities $c_a \geq 0, \forall a \in A$. Moreover, a flow network contains two designated vertices: the *source* s with indegree 0 and the *sink* t with outdegree 0.

A *flow* f in a network N is a function $f \colon A \to \mathbb{R}^{\geq 0}$ satisfying the following two properties:

- *capacity constraints:* $\forall a \in A \colon 0 \leq f(a) \leq c_a$
- *flow conservation:* $\forall v \in V \backslash \{s, t\} \colon \sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$

Since a flow is "going from s to t", it is also called *s-t-flow*. The *value* of a flow $v(f)$ is the sum of flow leaving the source s: $v(f) := \sum_{a \in \delta^+(s)} f(a)$, and a *maximum flow* f^* is a flow with maximum value.

The famous "max flow = min cut" theorem by Ford and Fulkerson [67] determines the relationship between cuts and flows in networks. Let S^* be a minimum s-t-cut and let f^* be a maximum s-t-flow in a network N.

Theorem 2.1 ([67]). *The value of a maximum flow equals the value of a minimum s-t-cut, i.e.,* $c(\delta^-(S^*)) = v(f^*)$.

Semi-directed paths and cuts. Let $G = (V, E)$ be an undirected graph and $\bar{G} = (V, A)$ be the bidirection of G. A sequence $P = \langle f_1, \ldots, f_\ell \rangle$ is a *semi-directed path* if (a) $\ell \geq 1$, (b) $f_i \in E$ or $f_i \in A, \forall i \in \{1, \ldots, \ell\}$, and (c) if there exists a directed path $P' = \langle a_1, \ldots, a_\ell \rangle$ in $\bar{G}$ such that for all $j \in \{1, \ldots, \ell\}$: $a_j = f_j \in A$ or a_j is a valid orientation of $f_j \in E$. Hence, undirected and directed paths are also semi-directed paths, but the opposite is not always true.

A *semi-directed cut* $\emptyset \neq S \subset V$ contains all edges $e \in \delta(S)$ and all arcs $a \in \delta^-(S)$, if it is ingoing, and $\delta^+(S)$ if it is outgoing. We use the same notation for semi-directed as for directed cuts: $\delta^-(S) := \{(i, j) \mid i \notin S, j \in S\} \cup \delta(S)$; δ^+ is defined analogously.

Special graph classes. A *complete (undirected) graph* $G = (V, E)$ on n vertices contains one edge between each pair of vertices, i.e., $|E| = n \cdot (n-1)/2$, and is denoted by K_n. A *clique* is a complete (sub)graph and the size of a clique is its number of vertices.

(Partial) k-trees are defined recursively as follows. The complete graph on k vertices is a *k-tree*. Inserting a new vertex into a *k-tree* and connecting this vertex to a clique of size k gives a *k-tree*. A spanning subgraph of a *k-tree* is called *partial k-tree*. Partial 2-trees are also called *series-parallel graphs*. Moreover, partial 2-trees are graphs with *treewidth* 2, cf. next paragraph.

Planar graphs are graphs that can be drawn into the plane without crossing edges. A special case of planar graphs and partial 2-trees are *outerplanar graphs* which allow planar drawings with all vertices lying on the boundary.

Tree decomposition and treewidth. *Tree decompositions* of graphs have the purpose of giving a better insight into its structure. Informally, the *tree decomposition* of a graph is a tree with each node representing a subset of vertices of the original graph. The topology of such a tree reflects the tree structure of the underlying graph. From an algorithmic point of view, such a decomposition can be utilized to compute solutions for several graph problems by applying dynamic programming.

The concept of *treewidth* was introduced by Robertson and Seymour [152] by the term *tree decomposition*. Since then it has been used intensively for many (optimization) problems. For in-depth discussions and comprehensive overviews see, e.g., Bodlaender [22, 24], Niedermeier [136], or Kloks [111].

Definition 2.2 (Tree decomposition). Let $G = (V, E)$ be a given undirected graph. A *tree decomposition* $(\mathcal{T}, \mathcal{X})$ is a pair consisting of a tree $\mathcal{T} = (I, F)$ and a collection $\mathcal{X} = \{X_i\}_{i \in I}$ of vertex subsets $X_i \subseteq V$ (called *bags*) with the following properties:

1. Every vertex $v \in V$ is contained in at least one bag X_i, $i \in I$.
2. For every edge $\{u, v\} \in E$ there is at least one bag X_i, $i \in I$, containing both u and v.
3. For every vertex $v \in V$, the nodes i with $v \in X_i$ form a subtree of $\mathcal{T}$.

Again, to avoid confusion, we speak of vertices V in the graph G, and of nodes I in the tree $\mathcal{T}$. The size of a bag is its number of vertices and the *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of the largest bag in $\mathcal{X}$ minus 1. The *treewidth tw* of a graph is the smallest width over all feasible tree decompositions. An example is given by Figure 2.2.

The treewidth measures how similar the decomposed graph is to a tree: trees have treewidth 1, series-parallel graphs have treewidth ≤ 2, (partial) k-trees have treewidth $\leq k$, etc. On the other side of the spectrum are complete graphs which have treewidth $|V| - 1$, by putting all vertices in one bag.

Most importantly, we note that there always exists a tree decomposition with linear size with respect to $|V|$ and $|E|$ (cf. Kloks [111]), even when considering *nice tree decompositions* [111]. Such tree decompositions always exist even for the optimal treewidth and are defined as follows.

Definition 2.3 (Nice tree decomposition). Let $G = (V, E)$ be an undirected graph. A tree decomposition $(\mathcal{T}, \mathcal{X})$ is a *nice tree decomposition* if the following properties are satisfied.

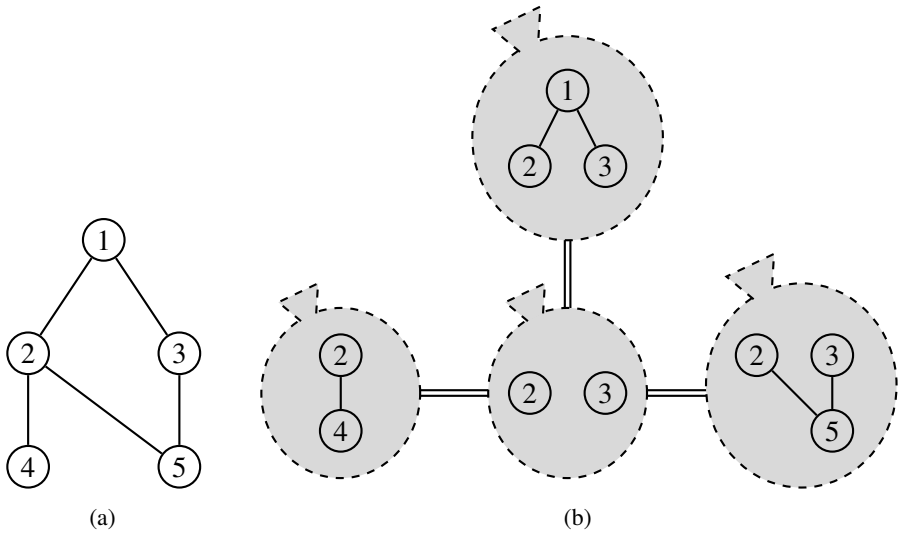


Figure 2.2: (a) An undirected graph with treewidth 2 and (b) a tree decomposition with minimum width. The bags are depicted by the grey shades and tree edges by double edges.

1. The tree $\mathcal{T}$ is rooted at some node $r \in I$.
2. Each node $i \in I$ is one of four possible types:
   (a) A *leaf* node i has 0 children and $|X_i| = 1$.
   (b) An *introduce* node i has 1 child $j \in I$ and X_j contains all vertices of X_i except for one: $X_j \subset X_i$, $|X_j| + 1 = |X_i|$.
   (c) A *forget* node i has 1 child $j \in I$ and X_j contains all vertices of X_i plus one additional one: $X_i \subset X_j$, $|X_i| + 1 = |X_j|$.
   (d) A *join* node i has two children $j \in I$ and $j' \in I$ and all three corresponding bags are identical: $X_j = X_{j'} = X_i$.

Tree decompositions, in particular nice tree decompositions, can be used to find clear and quite simple algorithms based on dynamic programming, cf. e.g., Bodlaender [24] for a comprehensive overview. However, most tree decomposition based algorithms have an exponential running time in the treewidth. Hence, for practical purposes it is important to find a tree decomposition with a small width. Arnborg, Corneil, and Proskurowski [8] showed that determining whether a graph has treewidth *tw*, for a given integer *tw*, is NP-complete. On the other hand, the decision problem is solvable in polynomial time for any constant *tw*, i.e., it is in FPT, cf. Bodlaender [23].

2.2 Linear and integer programming

The main purpose of this section is to recall the definitions and to introduce the used notations for *linear* and *integer programs*. We briefly describe the *branch&cut* algorithm and introduce a concept for comparing the strength of formulations. All definitions and descriptions are based on the following textbooks and mainly follow the first one: Nemhauser and Wolsey [135], Schrijver [155], Bertsimas and Tsitsiklis [18].

Linear programming. *Linear programs* are optimization problems with a linear objective function and a set of linear constraints. The task is to find a solution that minimizes the objective function while all constraints are satisfied. Let $x = (x_1, \ldots, x_n)^\top$ be the decision vector of size n, $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. A *linear program* (*LP*) reads as follows:

$$\begin{aligned} \text{(LP)} \qquad & \min c^\top x \\ & \text{s.t. } Ax \geq b \\ & x \geq \mathbf{0} \end{aligned}$$

For a linear program (LP) there exists a *dual program* (D:LP) defined as follows.

$$\begin{aligned} \text{(D:LP)} \qquad & \max b^\top y \\ & \text{s.t. } A^\top y \leq c \\ & y \geq \mathbf{0} \end{aligned}$$

To emphasize the difference, (LP) is also called the *primal* program. With x^*, y^* being optimum solutions to (LP) and (D:LP) with objective values z^*_{LP} and $z^*_{\text{D:LP}}$, respectively, the theorem of the *strong duality* says that if z^*_{LP} or $z^*_{\text{D:LP}}$ is finite, then both (LP) and (D:LP) have finite solutions and $z^*_{LP} = z^*_{\text{D:LP}}$.

Another important property is given by the theorem of *complementary slackness*. Let x^*, y^* be optimal solutions to (LP) and (D:LP), respectively. Intuitively, the theorem states that a (primal/dual) constraint is satisfied with equality or the associated (dual/primal) variable is 0. More formally, let $s_j := b_j - A_{[j,\cdot]}x^*$, $\forall j \in \{1, \ldots m\}$, be the primal *slack* and $t_i := c_i - y^* A_{[\cdot,i]}$, $\forall i \in \{1, \ldots n\}$, the dual slack. The *complementary slackness* states that for optimal x^*, y^* it holds for all j: $s_j y^*_j = 0$, and for all i: $t_i x^*_i = 0$.

Integer programming. Contrary to linear programs *integer programs (IP)* have a discrete variable space.

$$\begin{aligned} \text{(IP)} \qquad & \min c^\top x \\ & \text{s.t. } Ax \geq b \\ & x \geq \mathbf{0} \\ & x \in \mathbb{Z}^n \end{aligned}$$

One important special case of integer programs are *binary programs (BIP)* where variables have two possible values 0 or 1, i.e., $x \in \{0, 1\}^n$. These 0-1-decisions allow formulations of all combinatorial optimization problems. In case there are integer and continuous variables allowed, the program is called *mixed integer program (MIP)*.

The following descriptions focus on binary programs.

2.2.1 Branch&cut algorithm

Since integer programs can be used to model NP-hard problems it is unlikely, under the hypothesis NP ≠ P, that they can be solved in polynomial time. However, there exist successful approaches to solve binary programs in practice. One of these successful algorithms is *branch and cut (b&c)* which is mainly built on the four pillars: (1) *LP relaxation*, (2) *branch and bound*, (3) *cutting-plane method* and *separation*, (4) *primal heuristics*. We briefly discuss the main ideas in the following.

LP relaxation. By dropping the integrality restrictions from an IP one obtains the *LP relaxation* (or short *relaxation*). For an IP $z_{\text{IP}} = \min\{c^\top x \mid Ax \geq b, x \in \{0,1\}^n\}$ the LP relaxation reads $z_{\text{LP}} = \min\{c^\top x \mid Ax \geq b, \mathbf{0} \leq x \leq \mathbf{1}\}$.

The LP relaxation has some useful properties. First, if the relaxation is infeasible the corresponding IP is infeasible, too. Second, if the optimum solution to the LP relaxation is integer this solution is optimal for the corresponding IP. Third, the LP relaxation can be solved in polynomial time. Fourth, the value of the LP relaxation is a lower bound for the optimal value of the integer program, i.e., $z_{\text{LP}} \leq z_{\text{IP}}$.

Branch and bound (b&b). The b&b algorithm maintains a list of subproblems and starts by solving the LP relaxation. If the optimal solution is *fractional*, i.e., at least one variable is not integer, a *branching* step is performed. Thereby, two new subproblems are generated by setting one fractional variable to 1 and 0, respectively. Both subproblems are added to a list of open problems, the algorithm continues by solving another subproblem, and stops when all subproblems are processed. If an LP solution is integer, the current subproblem is solved to optimality and can be discarded. This also happens if a linear program is infeasible. In the end, the best found solution is the overall optimum solution.

The b&b algorithm naturally leads to the rooted *branch and bound tree*. Each subproblem corresponds to a node in this tree. The root node corresponds to the linear program of the initial LP relaxation and one branching step creates two children b&b nodes in this tree.

Obviously, the worst-case running time of this algorithm is exponential in the number of variables. By storing the best integer solution found so far—also called *primal bound*—a subproblem and its descendants in the b&b-tree can be discarded if its LP value—the *dual bound*—is not better than the primal bound.

Cutting-plane method. Some formulations have a large—many times even exponential—number of constraints. The *cutting plane method* offers a way to solve such LP formulations: Start with a smaller and proper subset of all constraints and solve the corresponding linear program. If the problem is infeasible, the original problem is infeasible. Otherwise, perform a *separation* step by searching for violated constraints. Those violated constraints are added and the procedure continues until no violated constraints are found or the problem turns out to be infeasible.

Primal heuristics. The b&b algorithm relies on good integer solutions to cut off as many b&b nodes as soon as possible. Since the LP relaxation solves a related problem of the original IP it might be easier to find solutions by using the LP solution as a starting point. The idea of *primal heuristics* is to exploit the fractional solutions and derive integer solutions thereof.

Branch and cut (b&c). The *b&c* algorithm mainly is the combination of the b&b approach with the other techniques. In each subproblem, the LP relaxation is solved by using the cutting plane method to obtain a dual bound. Non-optimal subproblems, i.e., if the dual bound is worse than the primal bound, can be discarded as well as infeasible subproblems. If an LP turns out having an optimum integer solution the corresponding subproblem is solved.

To ensure optimality, subproblems with fractional solutions are split into two subproblems by performing a branching step. Moreover, to obtain good integer solutions and eliminate unnecessary subproblems primal heuristics can be applied.

The b&c approach turned out being very efficient in practice and many problems can be solved exactly for large real-world instances. One of the most famous examples is the *traveling salesman problem* with a solved instance consisting of 85,900 cities by Applegate, Bixby, Chvátal, Cook, Espinoza, Goycoolea, and Helsgaun [7], cf. also [51].

2.2.2 Strength of formulations, polyhedral theory

Strength of formulations. Finding a "good" formulation for a problem is very important since the solutions of the LP relaxation highly influence the performance of a b&b based algorithm. If the solutions of LP relaxation and IP are close the number of b&b nodes is likely to be small. Moreover, if the dual bounds are strong more subproblems can be discarded.

By considering two IP formulations for the same problem it is not obvious which formulation yields a better running time when solved with a b&b algorithm. One indication is the *strength* of the LP relaxation, as defined by, e.g., Polzin and Daneshmand [140].

Let (A) and (B) denote two IP formulations for the same minimization problem and let (A^{rel}) and (B^{rel}) be the corresponding LP relaxations. Moreover, let $v_{\mathcal{I}}(A^{rel})$ and $v_{\mathcal{I}}(B^{rel})$ denote the value of the optimum solutions of (A^{rel}) and (B^{rel}), respectively, on instance $\mathcal{I}$.

Definition 2.4 (Strength of LP relaxations). The LP relaxation (A^{rel}) is *weakly stronger* than the LP relaxation (B^{rel}) if and only if $v_{\mathcal{I}}(A^{rel}) \geq v_{\mathcal{I}}(B^{rel})$ for all valid instances $\mathcal{I}$ of the problem. If (A^{rel}) is weakly stronger than (B^{rel}) and (B^{rel}) is weakly stronger than (A^{rel}), then both relaxations are *equivalent*.

(A^{rel}) is *(strictly) stronger* than (B^{rel}) if it is weakly stronger and they are not equivalent. In other words, if for all instances $\mathcal{I}$ it holds $v_{\mathcal{I}}(A^{rel}) \geq v_{\mathcal{I}}(B^{rel})$ and there exists an instance $\mathcal{I}^*$ with $v_{\mathcal{I}^*}(A^{rel}) > v_{\mathcal{I}^*}(B^{rel})$.

If both formulations are not equivalent and neither formulation is stronger they are *incomparable*.

This comparison of LP relaxations is used to compare (mixed) integer programming formulations.

Definition 2.5 (Strength of IP formulations). The IP formulation (A) is *(weakly) stronger* than the formulation (B) if (A^{rel}) is (weakly) stronger than (B^{rel}). (A) and (B) are *equivalent* if (A) is weakly stronger than (B) and (B) weakly stronger than (A). Otherwise, both formulations are *incomparable*.

Polyhedral theory. Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A *polyhedron* $\mathcal{P} \subseteq \mathbb{R}^n$ is the set of points in $\mathbb{R}^n$ that satisfy a finite number of linear inequalities: $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. A polyhedron $\mathcal{P}$ is *bounded* and called *polytope* if there exists a $w \in \mathbb{R}, w > 0$, with $\mathcal{P} \subseteq \{x \in \mathbb{R}^n \mid -w \leq x_i \leq w, \forall i = 1, \ldots, n\}$.

Let (F) be an LP formulation (or relaxation, respectively). Throughout this thesis we will denote the polyhedron defined by (F) by $\mathcal{P}_F$. Moreover, the projection of $\mathcal{P}_F$ into the space of a variable space x is denoted by $\text{Proj}_x(\mathcal{P}_F)$. For an overview of extended formulations and projections we refer the reader to, e.g., Balas [10]. The following observation is adopted from Kandyba-Chimani [105].

Observation 2.6. *Let (A^{rel}) and (B^{rel}) be LP relaxations of two formulations (A) and (B) for the same optimization problem. Moreover, let x be a suitable common variable space of both formulations.*

- *(A^{rel}) is weakly stronger than (B^{rel}) if for all problem instances it holds $Proj_x(\mathcal{P}_{A^{rel}}) \subseteq Proj_x(\mathcal{P}_{B^{rel}})$.*
- *(A^{rel}) and (B^{rel}) are equivalent if for all problem instances it holds $Proj_x(\mathcal{P}_{A^{rel}}) = Proj_x(\mathcal{P}_{B^{rel}})$.*
- *(A^{rel}) is (strictly) stronger than (B^{rel}) if (A^{rel}) is weakly stronger than (B^{rel}) and there exists an instance $\mathcal{I}$ with $Proj_x(\mathcal{P}_{A^{rel}}) \subsetneq Proj_x(\mathcal{P}_{B^{rel}})$; In this case it holds $v_{\mathcal{I}}(A^{rel}) > v_{\mathcal{I}}(B^{rel})$.*

2.3 Stochastic programming

We introduce *two-stage stochastic linear* and *integer programs* in Section 2.3.1, and discuss an exact approach based on *the integer L-shaped method* (Sections 2.3.2 and 2.3.3), namely the *two-stage branch&cut algorithm*. The definitions and algorithms in Sections 2.3.1–2.3.3 are mainly based on the textbooks by Birge and Louveaux [20] and Shapiro, Dentcheva, and Ruszczyński [161].

2.3.1 Two-stage stochastic linear programs

A *two-stage stochastic linear program* can bee seen as a generalization of a linear program where the objective function coefficients c, entries of the matrix A and the right-hand side b depend on the realization of a random vector ξ.

Let x and y be decision vectors of size n_1 and n_2, respectively. Variables x are the first-stage variables and y are the second-stage variables. Let vectors c and b and matrix A be defined as follows: $c \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{m_1}$, and $A \in \mathbb{R}^{m_1 \times n_1}$.

A *two-stage stochastic linear program* reads as follows:

$$\begin{aligned} \text{(2LP)} \quad & \min\ c^\top x + \mathbb{E}_\xi[\min\ q(\omega)^\top y] \\ & \text{s.t. } Ax \geq b \\ & T(\omega)x + Wy \geq h(\omega) \\ & x \geq \mathbf{0}, y \geq \mathbf{0} \end{aligned}$$

Thereby, $\mathbb{E}_\xi$ denotes the expected value w.r.t. ξ and Ω denotes the possible outcomes of ξ. In general, matrices T and vectors q and h depend on the random vector and are denoted by a functional style indicating this dependence, i.e., $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$, $q(\omega) \in \mathbb{R}^{n_2}$, and $h(\omega) \in \mathbb{R}^{m_2}$, $\omega \in \Omega$. Together they form the random vector ξ.

Mostly, the *recourse matrix* $W \in \mathbb{R}^{m_2 \times n_2}$ is assumed to be fixed, which is called *fixed recourse*, see, e.g., Birge and Louveaux [20]. All considered stochastic problems in this thesis have fixed recourse. Moreover, we assume that the isolated first-stage problem itself is feasible, i.e., $\{Ax \geq b, x \geq \mathbf{0}\} \neq \emptyset$.

For a realization $\omega \in \Omega$ and a first-stage solution $\tilde{x}$, $Q(\tilde{x}, \omega)$ is the optimal value of the second-stage problem:

$$\begin{aligned} \text{(2LP:2nd)} \quad Q(\tilde{x}, \omega) := \min\ & q(\omega)^\top y \\ \text{s.t. } & Wy \geq h(\omega) - T(\omega)\tilde{x} \\ & y \geq \mathbf{0} \end{aligned}$$

By assigning dual variables π_j, $j \in \{1, \ldots, m_2\}$, the dual of (2LP:2nd) for a first-stage solution $\tilde{x}$ reads as follows:

$$\begin{aligned} \text{(D:2LP:2nd)} \quad \max\ & \pi^\top (h(\omega) - T(\omega)\tilde{x}) \\ \text{s.t. } & W^\top \pi \leq q(\omega) \\ & \pi \geq \mathbf{0} \end{aligned}$$

Deterministic equivalent. Throughout this thesis we assume that the probability distribution is known and that ξ has a finite support[1], i.e., $\Omega = \{\omega_1, \ldots, \omega_K\}$, $K \in \mathbb{N}$, $K > 0$. Hence, we can formulate a two-stage stochastic program by using K scenarios; let $\mathcal{K} := \{1, \ldots, K\}$. The kth scenario is defined by its probability $p^k > 0$, its objective function coefficients $c^k \in \mathbb{R}^{n_2}$, the constraint matrix $T^k \in \mathbb{R}^{m_2 \times n_1}$, and the right-hand side $h^k \in \mathbb{R}^{m_2}$.

Here and in the following we use upper indices 0 and k to denote the first stage and second stage under scenario k, respectively, see the notations on page vii. Hence, x^0 and x^k are the decision vectors for the first stage and for scenario k, $\forall k \in \mathcal{K}$, with x^0 being n_1- and x^k n_2-dimensional, respectively. Moreover, $b \in \mathbb{R}^{m_1}$ is the right-hand side of the first-stage constraints with constraint matrix $A \in \mathbb{R}^{m_1 \times n_1}$. As before, $W \in \mathbb{R}^{m_2 \times n_2}$ is the fixed recourse matrix.

By expanding the formula for the expected value the *deterministic equivalent* of the two-stage stochastic program in *extensive form* reads as follows:

$$\begin{aligned} \text{(DE)} \quad \min\ & (c^0)^\top x^0 + \sum_{k \in \mathcal{K}} p^k (c^k)^\top x^k \\ \text{s.t. } & Ax^0 \geq b \\ & T^k x^0 + W x^k \geq h^k \qquad \forall k \in \mathcal{K} \\ & x^{0 \ldots K} \geq \mathbf{0} \end{aligned}$$

Notice that we use the vector $x^{0 \ldots K}$ as placeholder for the concatenation of the vectors $x^0, x^1, \ldots, x^K$, cf. the notations on page vii.

2.3.2 Benders' decomposition, L-shaped algorithm

Obviously, the number of variables and constraints of the deterministic equivalent depends on the number of scenarios and with an increasing number it becomes more difficult to solve. One promising approach for solving a large-scale linear program is to use *Benders' decomposition*. This method was originally proposed by Benders [15] for mixed integer programs. The general idea is to partition the program into the *master problem* and the *subproblem*. To obtain the *master problem* the difficult part is projected out into

[1] Assuming finite support is referred to as the *finite scenario model*. We discuss other models in Section 4.3 when considering the *stochastic Steiner tree problem*.

the *subproblem* and replaced by new lower bounding variables and *optimality cuts*. The corresponding algorithm iteratively solves the master problem and generates *optimality cuts* by solving the (dual of the) *subproblem*.

Van Slyke and Wets [168] focussed on two-stage stochastic linear programs and presented a similar approach "which is essentially the same as the algorithm developed by Benders" [168]. The main idea of this algorithm is to exploit the special structure of the constraint matrices A and W: Since the whole constraint matrix contains a lot of zeros and big parts are identical for each scenario the stochastic program can be decomposed with respect to the two stages. This special L-shaped like structure of the non-zeros in the constraint matrix led to the name *L-shaped algorithm*.

The overall idea is to project the second-stage variables and the related constraints out to decompose problem (DE) into a *master problem* (MP) and a set of K *subproblems* (SP$(k, \tilde{x}^0)$), one for each scenario. The main advantage of this approach is that instead of solving one huge LP many smaller and similar LPs have to be solved.

For two-stage stochastic programs the master problem contains all first-stage variables and constraints, and the subproblems are defined for each scenario and a first-stage solution $\tilde{x}^0$. In the multicut version, the second-stage cost is replaced by $K+1$ variables $\theta, \theta^1, \ldots, \theta^K$ with $\theta \geq \sum_{k \in \mathcal{K}} p^k \theta^k$. Furthermore, two types of new cuts are introduced: *feasibility cuts* (MP:1) and *L-shaped optimality cuts* (MP:2). A detailed discussion of these cuts and, in particular, the exact multipliers F_ℓ and D_ℓ and right-hand sides f_ℓ and d_ℓ for the ℓth cut can be found in the following paragraphs.

By using our notations and adopting the formulation from Birge and Louveaux [20] the *master problem* (MP) reads as follows:

$$
\begin{aligned}
\text{(MP)} \quad \min\ & (c^0)^\top x^0 + \theta \\
\text{s.t. } & Ax^0 \geq b \\
& \theta \geq \sum_{k \in \mathcal{K}} p^k \theta^k && \text{(MP:0)} \\
& F_\ell x^0 \geq f_\ell && \text{(MP:1)} \\
& \theta^k + D_\ell x^0 \geq d_\ell && \text{(MP:2)} \\
& x^0 \geq \mathbf{0} \\
& (\theta, \theta^1, \ldots, \theta^K) \geq \mathbf{0}
\end{aligned}
$$

In general, θ variables are free, cf. [20]. Here, we assume $\theta, \theta^1, \ldots, \theta^K \geq \mathbf{0}$ since in this thesis the second-stage costs are always non-negative.

Notice that we use the scenario probabilities as a multiplier in constraint (MP:0) such that the scenario probabilities are not incorporated into the variables $\theta^1, \ldots, \theta^K$. In the formulation by [20] the probabilities were already multiplied into the θ^k variables such that $\theta \geq \sum_{k \in \mathcal{K}} \theta^k$.

The basic idea of the *L-shaped algorithm* is to solve the master problem iteratively and separate cuts of type (MP:1) and (MP:2) by solving the (dual) subproblems, i.e., the K scenarios. For a first-stage solution $\tilde{x}^0$ and a scenario $k \in \mathcal{K}$ the kth subproblem (SP$(k, \tilde{x}^0)$)

is given as follows:

$$\begin{aligned}(\mathrm{SP}(k, \tilde{x}^0)) \quad & \min (c^k)^\top x^k \\ & \text{s.t. } W x^k \geq h^k - T^k \tilde{x}^0 \\ & x^k \geq \mathbf{0}\end{aligned}$$

By defining dual variables $(\pi^k)^\top := (\pi^k_1, \ldots, \pi^k_{m_2})$ for all constraints $j \in \{1, \ldots, m_2\}$, the dual (D:SP$(k, \tilde{x}^0)$) of the kth subproblem is the following:

$$\begin{aligned}(\mathrm{D{:}SP}(k, \tilde{x}^0)) \quad & \max \pi^k (h^k - T^k \tilde{x}^0) \\ & \text{s.t. } W^\top \pi^k \leq c^k \\ & \pi^k \geq \mathbf{0}\end{aligned}$$

Relatively and simple complete recourse. A stochastic program has *relatively complete recourse* if the second-stage subproblem (D:SP$(k, \tilde{x}^0)$) has a feasible solution for each feasible first-stage solution $\tilde{x}^0$, i.e., $\{x^0 \mid Ax^0 \geq b, x^0 \geq \mathbf{0}\} \subseteq \{x^0 \mid \forall k \in \mathcal{K} : (\mathrm{SP}(k, x^0)) \text{ is feasible}\}$.

Moreover, a stochastic program has *complete recourse* if $\{Wy \geq z, y \geq \mathbf{0}\} \neq \emptyset$ for all possible $z \in \mathbb{R}^{m_2}$, i.e., if the subproblem is feasible for all possible first-stage decisions. Notice that this property only depends on the recourse matrix W. By duality, complete recourse is given if problem (D:SP$(k, \tilde{x}^0)$) is bounded for every choice of $\tilde{x}^0, h^k$, and T^k (set $z = h^k - T^k \tilde{x}^0$). In this thesis all considered two-stage stochastic problems have the complete recourse property.

Another type of recourse is *simple recourse* where the constraint matrix W is the identity matrix and matrix T and vector h are deterministic. Since we do not deal with this case we refer the reader to, e.g., Louveaux and van der Vlerk [128].

Feasibility cuts. In case the primal subproblem is infeasible for the current first-stage solution $\tilde{x}^0$ and a scenario $k \in \mathcal{K}$ a *feasibility cut* (MP:1) is introduced into the master program to cut off the current first-stage solution. In this thesis, all considered stochastic problems—and their formulations—do not require feasibility cuts because they have complete recourse. Therefore, we skip the details on feasibility cuts and refer the reader to Birge and Louveaux [20].

L-shaped optimality cuts. Feasibility cuts have the purpose of cutting off the current infeasible first-stage solution $\tilde{x}^0$ without having any direct implications on the θ variables. This is overcome by the so-called *(L-shaped) optimality cuts* (MP:2).

Let $(\tilde{x}^0, \tilde{\theta}, \tilde{\theta}^1, \ldots, \tilde{\theta}^K)$ be the current optimum solution of the master problem. Furthermore, let $\tilde{x}^k$ be the optimal primal and $\tilde{\pi}^k$ the optimal dual solution for the kth scenario subproblem, $k \in \mathcal{K}$.

If the current $\tilde{\theta}^k$ value—which represents a lower bound on the second-stage cost of scenario k—is too low, an *L-shaped optimality cut* is inserted to increase this value or to enforce the selection of additional variables in the first stage: if for any scenario $k \in \mathcal{K}$, it holds $\tilde{\theta}^k < (c^k)^\top \tilde{x}^k = (\tilde{\pi}^k)^\top (h^k - T^k \tilde{x}^0)$, then the following *(disaggregated) L-shaped*

optimality cut is generated:

$$\begin{aligned} & \theta^k \geq (\tilde{\pi}^k)^\top (h^k - T^k x^0) \\ \Leftrightarrow \quad & \theta^k + (\tilde{\pi}^k)^\top T^k x^0 \geq (\tilde{\pi}^k)^\top h^k \end{aligned} \tag{Lc}$$

By adapting these values to the notation of the master program, we have $D_\ell := (\tilde{\pi}^k)^\top T^k$ and $d_\ell := (\tilde{\pi}^k)^\top h^k$ for the ℓth optimality cut, with ℓ being the current index.

Remark 2.7. The described decomposition introduces variables $\theta, \theta^1, \ldots, \theta^K$ which replace the second-stage part in the master program. This approach is already the disaggregated (multicut) version. It is also possible to use a single cut approach with one θ variable as originally proposed by Van Slyke and Wets [168].

Then, (MP) contains only one variable θ and constraint (MP:0) is removed. The feasibility cuts (MP:1) remain the same but the L-shaped optimality cuts are generated slightly differently. In case $\tilde{\theta} < \sum_{k=1}^K p^k (c^k)^\top \tilde{x}^k = \sum_{k=1}^K p^k (\tilde{\pi}^k)^\top (h^k - T^k \tilde{x}^0)$ the following L-shaped optimality cut is added:

$$\begin{aligned} & \theta \geq \sum_{k \in \mathcal{K}} p^k (\tilde{\pi}^k)^\top (h^k - T^k x^0) \\ \Leftrightarrow \quad & \theta + \sum_{k \in \mathcal{K}} p^k (\tilde{\pi}^k)^\top T^k x^0 \geq \sum_{k \in \mathcal{K}} p^k (\tilde{\pi}^k)^\top h^k \end{aligned} \tag{Lc1}$$

A disaggregated L-shaped cut affects exactly one θ^k variable and hence, a collection of K disaggregated cuts (Lc) is not weaker than the related single aggregated L-shaped cut (Lc1). On the other hand, the multicut approach contains K additional variables and inserting multiple L-shaped cuts leads to a larger linear program. However, in practice, using disaggregated cuts in the *L-shaped algorithm* is mostly the preferred approach, cf. e.g., Birge and Louveaux [19, 20].

L-shaped algorithm. The *L-shaped algorithm* by Van Slyke and Wets [168] for stochastic linear programs basically consists of the previously mentioned ingredients. With an empty set of feasibility and L-shaped optimality cuts in the beginning the master problem is solved iteratively in a cutting-plane manner. Violated cuts are generated by solving the K subproblems (SP($k, \tilde{x}^0$)) for each scenario and the currently optimal first stage $\tilde{x}^0$.

Theorem 2.8 (Birge and Louveaux [20]). *When ξ is a finite random variable, the L-shaped algorithm finitely converges to an optimal solution when it exists or proves the infeasibility of problems (2LP) and (DE), respectively.*

2.3.3 Integer L-shaped method, two-stage branch&cut

A *(mixed) integer two-stage stochastic program*, in particular its deterministic equivalent in extensive form, is defined as follows. Again, let $\mathcal{K} = \{1, \ldots K\}$ denote the set of scenarios with p^k being the realization probability of scenario $k \in \mathcal{K}$. The vectors c^0, b, c^k, and h^k and the matrices A, W, and T^k are defined as before: $c^0 \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{m_1}$, $c^k \in \mathbb{R}^{n_2}$, $h^k \in \mathbb{R}^{m_2}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $W \in \mathbb{R}^{m_2 \times n_2}$, and $T^k \in \mathbb{R}^{m_2 \times n_1}$. Notice that we use the same

identifier (DE) for this program; it will be clear from context which one is referred to.

$$\begin{aligned}
\text{(DE)} \quad \min\ & (c^0)^\top x^0 + \sum_{k\in\mathcal{K}} p^k (c^k)^\top x^k \\
\text{s.t. } & Ax^0 \geq b \\
& T^k x^0 + W x^k \geq h^k && \forall k \in \mathcal{K} \\
& x^{0\ldots K} \geq \mathbf{0} \\
& x^0 \in \mathcal{X}^0 \\
& x^k \in \mathcal{X}^k && \forall k \in \mathcal{K}
\end{aligned}$$

Thereby, $\mathcal{X}^0$ and/or $\mathcal{X}^k$ might contain integrality or binary restrictions on the variables x^0 and x^k, respectively. In this thesis, the classical formulations based on undirected cuts of all considered *stochastic network design problems* are binary stochastic programs, i.e., $\mathcal{X}^0 = \{0,1\}^{n_1}$ and $\mathcal{X}^k = \{0,1\}^{n_2}, \forall k \in \mathcal{K}$; mostly with $n_1 = n_2$, too. However, we will strengthen some formulations, in particular by introducing more variables in the second stage, such that some variables in the second stage are binary and some are fractional.

For the remaining part of this section we restrict the description to stochastic programs with binary first and second stage and mainly follow Birge and Louveaux [20]. Analogously to the preceding description of the L-shaped algorithm (DE) can be decomposed with Benders' decomposition. The *master problem* (MP) using the multicut approach is almost identical to the continuous case and only expanded by binary restrictions:

$$\begin{aligned}
\text{(MP)} \quad \min\ & (c^0)^\top x^0 + \theta \\
\text{s.t. } & Ax^0 \geq b \\
& \theta \geq \sum_{k\in\mathcal{K}} p^k \theta^k && \text{(MP:0)} \\
& F_\ell x^0 \geq f_\ell && \text{(MP:1)} \\
& \theta^k + D_\ell x^0 \geq d_\ell && \text{(MP:2)} \\
& x^0 \in \{0,1\}^{n_1} \\
& (\theta, \theta^1, \ldots, \theta^K) \geq \mathbf{0}
\end{aligned}$$

Type (MP:2) constraints denote the same (continuous) L-shaped optimality cuts from the L-shaped algorithm and further types of cuts, the so-called *integer optimality cuts*. This second type of cuts is important to close the integrality gap. Constraints of type (MP:1) are feasibility cuts and *(integer) optimality cuts* which are discussed in the next paragraph.

The *relaxed master problem*, denoted by (RMP), is obtained by relaxing the integrality on x^0. Then, for each (fractional) first-stage solution $\tilde{x}^0$ and each scenario $k \in \mathcal{K}$, the kth subproblem reads as follows:

$$\begin{aligned}
\text{(SP}(k, \tilde{x}^0)\text{)} \quad \min\ & (c^k)^\top x^k \\
\text{s.t. } & W x^k \geq h^k - T^k \tilde{x}^0 \\
& x^k \in \{0,1\}^{n_2}
\end{aligned}$$

Analogously, the *relaxed subproblem* (RSP$(k, \tilde{x}^0)$) is obtained by relaxing the integrality on x^k. The dual program of (RSP$(k, \tilde{x}^0)$) is already given by the dual program (D:SP$(k, \tilde{x}^0)$).

Optimality cuts. Like already mentioned, the (continuous) L-shaped optimality cuts are also valid for binary stochastic programs:

Theorem 2.9 (Birge and Louveaux [20]). *Any continuous L-shaped optimality cut is a valid lower bound on the second-stage cost.*

In the following exact approaches additional cuts are required to close the integrality gap. Let L be a valid lower bound for the expected second-stage cost which is assumed to exist. Moreover, let $\tilde{x}^0$ be a feasible binary first-stage solution with its corresponding 1-index set $\tilde{S}$, i.e., $\tilde{S} = \{i \mid \tilde{x}_i^0 = 1\}$. Let $\tilde{q}$ be the value of the second-stage recourse function: $\tilde{q} := \sum_{k \in \mathcal{K}} p^k \tilde{q}^k$ with $\tilde{q}^k$ being the optimum solution value of $(\mathrm{SP}(k, \tilde{x}^0))$.

Laporte and Louveaux [117] introduced the following *integer optimality cuts*:

$$\begin{aligned} \theta &\geq (\tilde{q} - L)\left(\sum_{i \in \tilde{S}} x_i^0 - \sum_{i \notin \tilde{S}} x_i^0 - |\tilde{S}| + 1\right) + L \\ \Leftrightarrow \theta &+ \sum_{i \in \tilde{S}} (L - \tilde{q}) x_i^0 + \sum_{i \notin \tilde{S}} (\tilde{q} - L) x_i^0 \geq (L - \tilde{q})(|\tilde{S}| - 1) + L \end{aligned} \tag{Ic}$$

The proof of validity can be found in [20] or [117]. These integer optimality cuts are quite weak since they mainly affect only the corresponding solution $\tilde{x}^0$ by giving a lower bound for θ: For $\tilde{x}^0$ the cut states that $\theta \geq \tilde{q}$ which is obviously valid since $\tilde{q}$ is the current second-stage cost. When considering any first-stage integer solution $\bar{x}^0$ different from $\tilde{x}^0$, i.e., $\exists i : \bar{x}_i^0 \neq \tilde{x}_i^0$, the cut states that $\theta \geq L$, if $\tilde{x}^0$ and $\bar{x}^0$ differ in exactly one variable, and otherwise, the obtained bound on θ gets even smaller.

Integer L-shaped method. One approach for solving a binary two-stage stochastic program is the *integer L-shaped method* developed by Laporte and Louveaux [117]. Like the name suggests it is based on the L-shaped method for linear stochastic programs. Basically it is a branch&cut algorithm on the first-stage master program (MP) with separations of feasibility, integer, and L-shaped optimality cuts.

A brief description of the algorithm—its multicut version—is given as follows and a simplified flow chart of the method is presented by Figure 2.3. We use UB as a global upper bound on the optimum solution. Moreover, ν denotes the current node of the b&b tree.

Step 0: *Initialization.* $UB := +\infty$. Create the first pendant node ν; in the initial relaxed master problem (RMP) the set of L-shaped and integer optimality cuts is empty.

Step 1: *Selection.* Select a pendant node ν from the branch and bound tree. If no such node exists STOP.

Step 2: *Solving.* Solve (RMP) at the current node ν. Let $(\tilde{x}^0, \tilde{\theta}, \tilde{\theta}^1, \ldots, \tilde{\theta}^K)$ be the optimal solution with value $z^\nu := (c^0)^\top \tilde{x}^0 + \tilde{\theta}$. If $z^\nu \geq UB$ or (RMP) is infeasible: Fathom the current node. Goto Step 1.

Step 3: *Separation.*

- *3.1*: *Separation of feasibility cuts.*
- *3.2*: *Separation of L-shaped optimality cuts:*

3.2.1: For all $k \in \mathcal{K}$ solve the LP relaxation (RSP$(k, \tilde{x}^0)$). Let $\tilde{x}^k$ denote the optimum solution with value $R^k(\tilde{x}^0)$.
If $R^k(\tilde{x}^0) > \tilde{\theta}^k$: insert L-shaped optimality cut (Lc) into (MP).

3.2.2: If at least one L-shaped cut was inserted: Goto Step 2.

3.3: If $\tilde{x}^0$ is binary: *Separation of integer optimality cuts*:

3.3.1: For all $k \in \mathcal{K}$ where $\tilde{x}^k$ was not binary in the previously computed LP relaxation: solve (SP$(k, \tilde{x}^0)$) to integer optimality. Let $Q^k(\tilde{x}^0)$ be the optimal value, for each $k \in \mathcal{K}$.

3.3.2: $UB := \min\{UB, (c^0)^\top \tilde{x}^0 + \sum_{k\in\mathcal{K}} p^k Q^k(\tilde{x}^0)\}$.

3.3.3: If $\sum_{k\in\mathcal{K}} p^k Q^k(\tilde{x}^0) > \tilde{\theta}$ insert integer optimality cut (Ic) into (MP). Goto Step 2.

3.3.4: Fathom the current node. Goto Step 1.

Step 4: *Branching.* Use a branching criterion on a fractional x^0 variable and create two new b&b nodes. Goto Step 1.

We leave out *Step 3.1*: *Separation of feasibility cuts* from the detailed description since we are dealing with complete recourse. For a complete description of the algorithm we refer the reader to [20].

If the scenarios can be solved in a finite number of steps for any fixed first-stage solution $\tilde{x}^0$ (Assumption 3 in Birge and Louveaux [20]) then the integer L-shaped method is finite and optimal:

Theorem 2.10 ([20, 117]). *Under Assumption 3, for any problem for which a valid set of feasibility cuts and a valid set of optimality cuts exist, the integer L-shaped method yields an optimal solution (when one exists) in a finite number of steps.*

The integer L-shaped method and Benders' decomposition are powerful tools for solving two-stage stochastic programs. Both methods have been applied and adopted successfully for several stochastic and non-stochastic problems like

- *Stochastic vehicle routing problem*: Laporte, Louveaux, and van Hamme [119]
- *Hub location problem*: Contreras, Cordeau, and Laporte [50]
- *Stochastic location problem*: Laporte, Louveaux, and van Hamme [118]
- *Stochastic dial-a-ride*: Heilporn, Cordeau, and Laporte [92]
- *Hop-constrained Survivable network design*: Botton, Fortz, Gouveia, and Poss [30]
- *Stochastic supply chain network design*: Santoso, Ahmed, Goetschalckx, and Shapiro [154]
- *Stochastic assignment problems:* Albareda-Sambola, van der Vlerk, and Fernández [3]
- *Robust facility location problem:* Álvarez-Miranda, Fernández, and Ljubić [4]
- *Capacitated facility location problem:* Fischetti, Ljubić, and Sinnl [66]

A recent literature review on Benders' decomposition can be found in Rahmaniani, Crainic, Gendreau, and Rei [148].

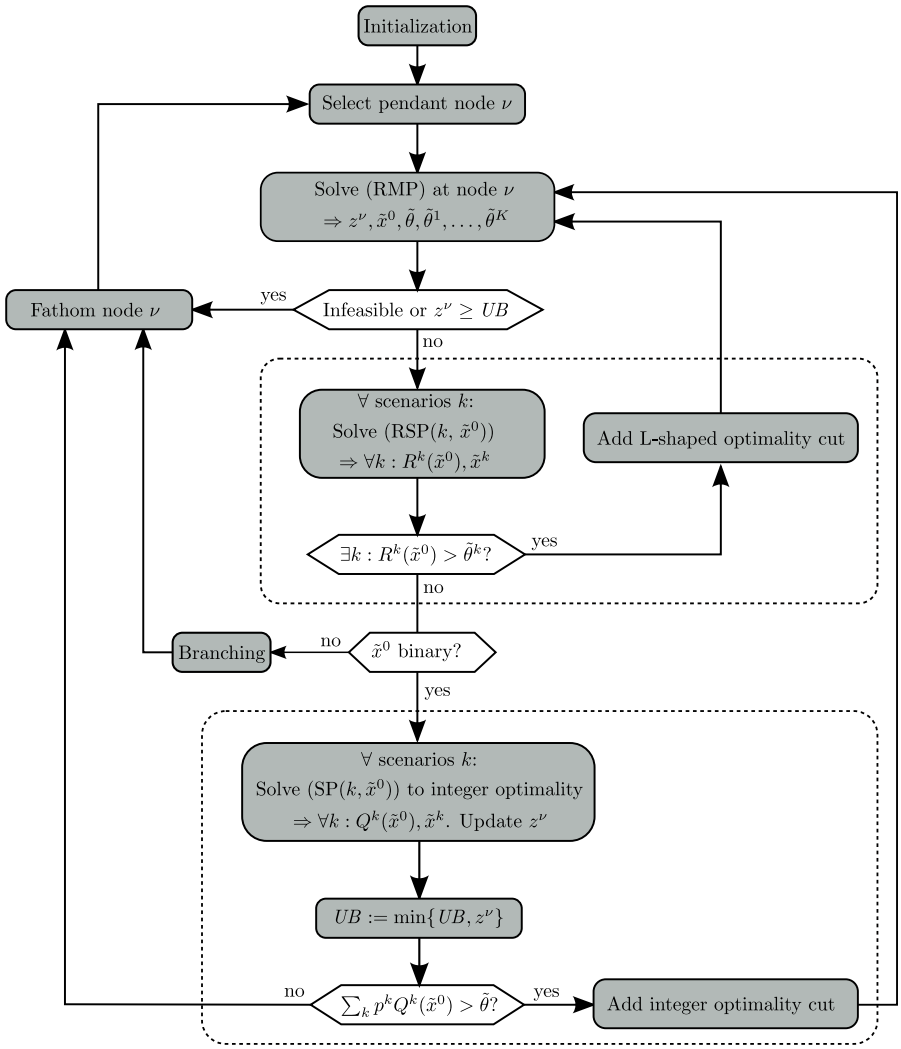


Figure 2.3: Basic steps of the integer L-shaped algorithm as flow chart.

Two-stage branch&cut algorithm. The integer L-shaped algorithm plays an important role in this thesis. We adopt this method for solving *two-stage stochastic network design problems*. When applying the integer L-shaped algorithm/Benders' decomposition concept to the proposed models for the *two-stage stochastic network design problems*, attention should be given to the following two non-standard aspects: a) First of all, one has to deal with the integer recourse. For that purpose, we integrate a separation of the integer optimality cuts within a branch&cut framework. b) The second main difficulty arises with the fact that the associated subproblems contain an exponential number of constraints, and can therefore be solved only by means of a cutting plane approach (for finding optimal LP solutions), or branch&cut (for finding optimal integer solutions). Therefore, in order to apply a Benders-like decomposition, one needs to nest two branch&cut algorithms: a branch&cut is employed for solving the master problem where violated integer and L-shaped optimality cuts are generated by solving the K subproblems with a dedicated branch&cut algorithm. Hence, there are two nested branch&cut algorithms which led to the naming

two-stage branch&cut algorithm, cf. [27].

Stochastic programming references. Although we mainly cite the book by Birge and Louveaux [20] there exist some nice overview articles on stochastic programming, and in particular on stochastic integer programming. We give some references in chronological order of their publication date: Schultz, Stougie, and van der Vlerk [157], Klein Haneveld and van der Vlerk [110], Schultz [156], Higle [93], and Shapiro and Philpott [160]. We also like to mention a homepage dedicated to stochastic programming [144] which offers many links to lecture notes, slides, and further references.

Solution methods for stochastic integer programs. In this thesis we focus on and apply the two-stage branch&cut algorithm and the single-stage branch&cut approach, where the deterministic equivalent in extensive form is solved directly. In literature there exist other approaches for solving two-stage stochastic (integer) programs. Each method mostly covers a different type of stochastic program, i.e., integer, mixed-integer, or binary variables, respectively. Here, we like to mention the generalized version of the integer L-shaped algorithm by Carøe and Tind [33], the dual decomposition approach based on Lagrangian relaxation by Carøe and Schultz [32], and a method based on disjunctive programming by Sen and Higle [158].

One challenge of the (integer) L-shaped method is the repeated solving of each scenario in every step. With an increasing complexity of the subproblems the stochastic program itself becomes more difficult to solve. A possible way to reduce the problem complexity is to apply the so-called *sample average approximation method*. The basic idea of this approach is to solve the problem only for a sampled subset of scenarios. We refer the reader to Shapiro [159] and references therein. Higle and Sen [94] discussed *stochastic decomposition* which is another approach aiming in a similar direction. Here, an adaptive sample size is used which is increased during the algorithm.

Chapter 3

Network design problems

The first section of this chapter contains the definitions of the relevant combinatorial optimization problems. Then, Section 3.2 introduces basic IP formulations for the *network design problems*. Afterwards, we concentrate on the *Steiner tree problem* and the *survivable network design problem* and summarize important results concerning both problems (Section 3.3 and 3.4).

3.1 Definitions

Although the literature on *network design problems (NDP)* is vast there is no standard definition covering all NDPs. In general and informally, NDPs are combinatorial optimization problems on graphs with values assigned to vertices, edges, or certain substructures. The objective is to find a subgraph satisfying some specified properties such that the overall value of the selected graph elements is minimized or maximized.

In the following we give formal definitions of relevant NDPs. We define edge weights to be positive for all problems. Although this is a restriction it is a reasonable assumption for practical purposes since using edges corresponds to, e.g., driving times or upgrading or building costs which are mostly positive. Moreover, to make the distinction to the stochastic problems clearer, we refer to these problems as *deterministic* problems.

Problem 3.1 (Minimum spanning tree problem (MST)):
Given: *undirected graph* $G = (V, E, c)$*, edge cost* $c_e \in \mathbb{R}^{>0}, \forall e \in E$
Solution: *connected subgraph* $G' = (V, E')$ *of* G
Objective: *minimize overall cost* $\sum_{e \in E'} c_e$

The minimum spanning tree problem is solvable in polynomial time, e.g., with the algorithms by Kruskal or Prim, cf. e.g., Cormen, Leiserson, Rivest, and Stein [52]. Contrarily, the following network design problems are NP-hard.

Problem 3.2 (Steiner tree problem (STP)):
Given: *undirected graph* $G = (V, E, c)$*, edge cost* $c_e \in \mathbb{R}^{>0}, \forall e \in E$*, set of* terminal *vertices* $\emptyset \neq T \subseteq V$
Solution: *edge set* $E' \subseteq E$ *such that* $G[E']$ *connects* T
Objective: *minimize overall cost* $\sum_{e \in E'} c_e$

The vertices in $V \backslash T$ are called *non-terminals* and the non-terminals which are contained in a Steiner tree are called *Steiner vertices*.

A solution to the STP does not have to be connected or cycle-free. However, it can be easily seen that the optimum solution satisfies these properties when edge costs are positive. Hence, the optimum solution is in fact a tree, the *Steiner tree*.

A generalization of the STP is the *Steiner forest problem* where several sets of terminals have to be connected.

Problem 3.3 (Steiner forest problem (SFP)):
Given: *undirected graph* $G = (V, E, c)$*, edge cost* $c_e \in \mathbb{R}^{>0}, \forall e \in E$*, set of* $K \geq 1$ *terminal sets* $T = \{T^1, \ldots, T^K\}$ *with* $\emptyset \neq T^k \subseteq V, \forall k \in \{1, \ldots, K\}$
Solution: *edge set* $E' \subseteq E$ *such that* T^k *is connected in* $G[E'], \forall k \in \{1, \ldots, K\}$
Objective: *minimize overall cost* $\sum_{e \in E'} c_e$

Another common definition of the SFP consists of defining a set of p vertex pairs that need to be connected: $(s_1, t_1), \ldots, (s_p, t_p)$ with $s_i, t_i \in V, \forall i \in \{1, \ldots, p\}$. Since the connectivity requirement is transitive both problem definitions are equivalent and can be transformed into each other easily.

Obviously, the STP is a special case of the SFP with $K = 1$. On the other hand, the Steiner forest problem is a special case of the more general *survivable network design problem (SNDP)*—more precisely the *non-unitary SNDP*, see below. The term "survivable" indicates that feasible solutions are immune to a certain failure. In the first considered problem, vertices need to be connected with respect to a certain edge-connectivity such that the failure of edges does not disconnect the vertices.

Problem 3.4 (Survivable network design problem (SNDP)):
Given: *undirected graph* $G = (V, E, c)$*, edge cost* $c_e \in \mathbb{R}^{>0}, \forall e \in E$*, symmetric and* unitary connectivity requirement matrix $\rho \in \mathbb{N}^{|V| \times |V|}$
Solution: *edge set* $E' \subseteq E$ *such that* $G[E']$ *contains* ρ_{uv} *edge-disjoint paths between vertices* $u, v \in V, u \neq v$
Objective: *minimize overall cost* $\sum_{e \in E'} c_e$

While the definition of the SNDP is as general as possible we have one commonly used restriction and assume that matrix ρ implies that all vertices with connectivity requirements need to be connected. In other words, each optimal solution comprises a single connected component. In this case the problem and the connectivity requirement matrix is called *unitary*, cf. Magnanti and Raghavan [130]. As mentioned before, one example for a *non-unitary* SNDP is the Steiner forest problem where optimal solutions may be disconnected.

A closely related problem is the node-connectivity version of the SNDP where failures of vertices need to be covered. To distinguish both problems we call this problem the *node-connectivity SNDP*.

Problem 3.5 (node-connectivity SNDP (ncSNDP)):
Given: *undirected graph* $G = (V, E, c)$*, edge cost* $c_e \in \mathbb{R}^{>0}, \forall e \in E$*, symmetric and unitary connectivity requirement matrix* $\rho \in \mathbb{N}^{|V| \times |V|}$
Solution: *edge set* $E' \subseteq E$ *such that* $G[E']$ *contains* ρ_{uv} *node-disjoint paths between* $u, v \in V, u \neq v$
Objective: *minimize overall cost* $\sum_{e \in E'} c_e$

It is common to consider special cases of the (nc)SNDP. One famous example from literature is the $\{0, 1, 2\}$-ncSNDP, see, e.g., Chimani, Kandyba, Ljubić, and Mutzel [38], where vertices are classified into three categories: each vertex $v \in V$ is assigned a value

$\rho_v \in \{0, 1, 2\}$ (notice that we use the same identifier ρ, but this time ρ is a vector instead of a matrix). Vertices of type 2 require two node-disjoint paths, vertices of type 1 simple connectivity, and all other vertices can be used to establish cost-efficient solutions. Then, the vertex types are given by sets $\mathcal{R}_1, \mathcal{R}_2$, and $\mathcal{R}$ defined as $\mathcal{R}_i := \{v \in V \mid \rho_v = i\}$, for $i = 1, 2$, and $\mathcal{R} := \mathcal{R}_1 \cup \mathcal{R}_2$. The required connectivity between a pair of distinct vertices $u, v \in V$ is defined as $\rho_{uv} = \min\{\rho_u, \rho_v\}$. Hence, $\rho_{uv} = 2$ if both vertices u and v are of type 2, $\rho_{uv} = 1$ if $1 \leq \rho_u \cdot \rho_v \leq 2$, and otherwise $\rho_{uv} = 0$.

Further Problems. The existing literature on Steiner tree problems and network design problems is immense. An overview of problems related to the STP can be found at a dedicated homepage [91]. In the preceding section we covered only the problems which are considered in this thesis. Because of their relevance and the similarities to our topics we like to mention the *traveling salesman problem (TSP)* and the *prize-collecting Steiner tree problem (PCSTP)*. The TSP consists of finding a minimum cost hamiltonian circuit. It is is one of the best studied NP-hard combinatorial optimization problems, cf. e.g., the book by Applegate, Bixby, Chvátal, and Cook [6] or a TSP homepage [51]. The PCSTP is an extension of the STP with no predefined set of required vertices. Instead, vertices reward a profit if they are contained in the solution. The objective is to maximize this profit minus the cost of connecting the vertices. For the PCSTP we refer the reader to Ljubić, Weiskircher, Pferschy, Klau, Mutzel, and Fischetti [125].

However, the introduced problems often do not suffice to cover problems that arise in practice where it is necessary to satisfy additional constraints of different types. For example, such constraints might require a maximum degree of connected vertices (*degree-constrained*), a maximum number of edges or vertices between two connected vertices (so-called *hops*), or an additional resource that has to be satisfied (*resource constrained*). Moreover, there might be different types of vertices, like, e.g., *demand* or *supply* vertices, or different types of edges, e.g., edges representing different technologies, which influence the set of feasible solutions. Some problems are also considered in a stochastic or robust setting or, e.g., with multiple objectives.

Here, we mention some (recent) publications: Gouveia, Simonetti, and Uchoa [78], Gouveia, Leitner, and Ljubić [79], Climaco and Pascoal [47], Di Puglia Pugliese and Guerriero [60], Chimani, Kandyba, Ljubić, and Mutzel [39], Gollowitzer [77], Goerigk [76], and Kandyba-Chimani [105], and refer the reader to references therein.

3.2 Basic IP formulations

Most of the previously described network design problems with predefined connectivity requirements can be formulated as integer programs easily by using undirected cuts. The following formulation was introduced by Williamson, Goemans, Mihail, and Vazirani [172] and Goemans and Williamson [75]. It can be used to define a broad set of network design problems.

Let $G = (V, E)$ be the undirected input graph. Selecting an edge in the solution is indicated by the binary decision variable $x_e, \forall e \in E\colon x_e = 1$ if and only if the edge is contained in the solution. With the classical notation $x(E') := \sum_{e \in E'} x_e, \forall \emptyset \neq E' \subseteq E$, the

cut-based formulation for network design problems reads as follows:

$$
\begin{aligned}
(\mathrm{NDP}_{\mathrm{uc}}) \quad & \min \sum_{e \in E} c_e x_e \\
& \text{s.t. } x(\delta(S)) \geq f(S) \qquad \forall \emptyset \neq S \subseteq V && (3.1) \\
& x \in \{0,1\}^{|E|} && (3.2)
\end{aligned}
$$

The objective function is simply the minimization of the cost of the selected edges. Constraints (3.1) enforce the selection of at least $f(S)$ incident edges crossing a cut S. Thereby, function $f\colon 2^V \to \mathbb{N}$ is used to model the connectivity requirements of the network design problem.

For example, the Steiner forest problem can be modeled by defining f as follows:

$$
f(S) := \begin{cases} 1 & \text{if } \exists i\colon \emptyset \neq T^i \cap S \neq T^i \\ 0 & \text{otherwise} \end{cases} \tag{3.3}
$$

For the edge-connectivity survivable network design problem function f is set to

$$
f(S) := \max\{\rho_{uv} \mid u \in S, v \notin S\} \tag{3.4}
$$

The problems MST, STP, SFP, and SNDP all fall in this class of NDPs and can be modeled this way. The correctness of formulation ($\mathrm{NDP}_{\mathrm{uc}}$) for each of these problems follows from the "max flow = min cut" theorem, cf. e.g., Grötschel, Monma, and Stoer [80].

Another MIP formulation for this type of network design problems is based on *multi-commodity flows*, cf. e.g., Ahuja, Magnanti, and Orlin [2]. The idea is to send ρ_{uv} units of flow from vertex u to v, for each ordered vertex pair $u, v \in V$, while each edge has capacity one. Then, every edge with non-zero flow is used in the solution (set $x_e = 1$).

Contrary to ($\mathrm{NDP}_{\mathrm{uc}}$) this formulation is a *compact* model. A *compact* formulation has a size that is polynomial in the graph size. ($\mathrm{NDP}_{\mathrm{uc}}$) is obviously not compact since the number of constraints is exponential in $|V|$.

Let A be the arc set of the bidirection of G. To distinguish the different flows we use K commodities, one for each pair of vertices u, v with $\rho_{uv} > 0$; we denote the connectivity requirement of commodity k corresponding to vertices u, v with q_k and set $q_k := \rho_{uv}$. For each commodity $k \in \{1, \ldots, K\}$ let the vertex with smaller index be the source and the other the target.

To model the flow we use flow variables f_{ij}^k linked to the directed arcs and to the kth commodity, indicated by the upper index k. Moreover, we use binary decision variables x_e for each undirected edge. By adopting the notation of Magnanti and Raghavan [130] the

flow formulation for network design problems reads as follows:

$$
\begin{aligned}
(\mathrm{NDP_{uf}}) \quad & \min \sum_{e \in E} c_e x_e \\
& \text{s.t. } f_{ij}^k \leq x_e, \\
& \quad f_{ji}^k \leq x_e \qquad \forall k \in \{1, \ldots, K\}, \forall e = \{i, j\} \in E \qquad (3.5) \\
f^k(\delta^-(i)) - f^k(\delta^+(i)) &= \begin{cases} -q_k, & \text{if } i = \text{source}(k) \\ q_k, & \text{if } i = \text{target}(k) \\ 0, & \text{otherwise} \end{cases} \Bigg\} \begin{array}{l} \forall k \in \{1, \ldots, K\}, \\ \forall i \in V \end{array} \qquad (3.6) \\
f &\in [0, 1]^{|A| \cdot K} \qquad (3.7) \\
x &\in \{0, 1\}^{|E|} \qquad (3.8)
\end{aligned}
$$

The objective function contains only x variables and is identical to the cut-based formulation. Arcs which are used for routing flow are payed for through capacity constraints (3.5). Constraints (3.6) are classical flow-conservation constraints. For each source (target) the outgoing (ingoing) flow has to meet the connectivity requirement and for non-source and non-target vertices the sum of the ingoing and outgoing flow has to be identical.

Comparing the two formulations ($\mathrm{NDP_{uc}}$) and ($\mathrm{NDP_{uf}}$) is twofold. From a polyhedral point of view both formulations are equally strong (cf. Section 2.2.2):

Theorem 3.6 (e.g. [130]). *(NDP_{uf}) is equivalent to (NDP_{uc}).*

On the other hand, in practice it makes a difference which model is solved. Although formulation ($\mathrm{NDP_{uf}}$) is compact it does contain a lot more variables than formulation ($\mathrm{NDP_{uc}}$), i.e., $|E| + 2 \cdot |E| \cdot K$ versus $|E|$. Moreover, separating undirected cuts (3.1) can be done in polynomial time by computing minimum cuts, cf. e.g., Grötschel, Monma, and Stoer [81]. It is known that cut-based formulations perform in general better and scale better with an increasing graph size. For example, computational results for the Steiner tree problem can be found in the PhD theses by Daneshmand [55] and Polzin [139].

3.3 Steiner tree problem

The Steiner tree problem—in particular its stochastic version—is one of the central problems studied in this thesis. It is also one of the classical network design problems with a vast literature. For a first overview we refer the reader to the books by Hwang, Richards, and Winter [96] and Prömel and Steger [145], and to a dedicated homepage [91]. Further references are given in the following for each specific subtopic. We focus on the topics which are relevant to the *stochastic Steiner tree problem* and discuss them in more detail.

3.3.1 Complexity, polynomially solvable cases, and parameterized algorithms

The decision version of the Steiner tree problem—is there a Steiner tree which costs at most a given value?—is one of the famous 21 NP-complete problems studied by Karp [106]. Bern and Plassmann [17] gave a proof based on a constructed graph with edge weights 1 and 2. Therefore, they could show that the STP is strongly NP-hard and MAX SNP-hard. Moreover, the STP is still NP-hard on planar graphs, cf. Garey and Johnson [70].

On the other hand, there exist special cases for which the Steiner tree problem can be solved in polynomial time:

- *Trees*: Since paths in trees are unique the minimum Steiner tree is the union of paths connecting any two terminals.
- *Partial 2-trees, series-parallel graphs*: Wald and Colbourn [170] described a linear time algorithm for partial 2-trees.
- *Partial k-trees*: For partial k-trees, which are graphs with treewidth k, the STP can be solved with the FPT algorithm described in Section 5.4; this is a polynomial-time algorithm if k is a constant.
- *Bounded number of terminals*: The case $|T| = 2$ is the shortest path problem. For a constant $|T|$ the algorithm by Dreyfus and Wagner [63] solves the STP in time $O(3^{|T|}|V|)$; this is again an FPT algorithm which is polynomial for constant $|T|$.
- *Bounded number of non-terminals*: The case $T = V$ is the MST. For a constant number of non-terminals $|V \backslash T|$ a simple enumeration of all possible subsets of $V \backslash T$ combined with the MST gives the optimum Steiner tree.
- For further special cases we refer the reader to Winter [174] and Hwang, Richards, and Winter [96].

For series-parallel graphs or partial 2-trees, respectively, complete LP formulations for the Steiner tree and related problems were given by Margot, Prodon, and Liebling [132] and Goemans [72, 73].

Approximation algorithms. Besides the long list of literature on exact approaches and polyhedral studies there exists a vast literature on approximation algorithms, too. The classical algorithm is the *distance network heuristic* (MST heuristic) with an approximation ratio of $2 - 2/|T|$, cf. e.g., Vazirani [169] or Prömel and Steger [145]. Another $(2 - 2/|T|)$-approximation is based on the primal-dual scheme by Goemans and Williamson [75].

Over the years, the approximation ratio was improved several times to the currently best $\ln(4) + \varepsilon \approx 1.39$ for an arbitrary small $\varepsilon > 0$ by Byrka, Grandoni, Rothvoß, and Sanità [31]. Another preceding milestone was the algorithm by Robins and Zelikovsky [153] which achieves an approximation ratio that converges to $1 + \ln(3)/2 \approx 1.55$. Both algorithms are based on the enumeration of *(restricted) components* of size k and the approximation ratio converges to the given value if k converges to ∞. Unfortunately, the running time is exponential in k which makes the algorithms inapplicable in practice. A nice overview of (more) approximation algorithms and heuristics with an experimental study was given by Chimani and Woste [37].

On the other hand, an inapproximability result by Chlebík and Chlebíková [43] states that the STP is not approximable within a factor of $96/95$ unless NP = P.

3.3.2 IP formulations

IP formulations and polyhedra of the Steiner tree problem have been studied intensely in the 1990s by Chopra and Rao [44, 45], Chopra and Tsai [46], Goemans and Myung [74], Koch

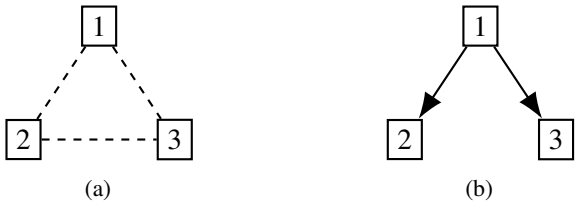


Figure 3.1: An instance of the STP where all three vertices are terminals and all edge costs are 1. (a) shows the optimum solution of the relaxed undirected cut formulation with value 1.5 by setting $x_e = 0.5$ for all edges e. There is no valid solution for the directed cut formulation with the same value. Instead, two arcs have to be selected, e.g., for root node 1 $z_{12} = z_{13} = 1$, as depicted in (b).

and Martin [112], Polzin and Daneshmand [140, 142], Daneshmand [55], and Polzin [139]. We like to highlight [55, 112, 139, 140, 142] since the authors also described branch&cut algorithms and provided results from computational studies.

Cut-based formulations. With the IP formulation (NDP_{uc}) at hand the undirected cut formulation for the Steiner tree problem follows directly by setting $f(S) := 1$ if S contains at least one, but not all, terminals: $x(\delta(S)) \geq 1, \forall S \subset V \colon \emptyset \neq S \cap T \neq T$. In this case there has to be at least one edge connecting these vertices with the remaining part of the graph. The related constraints are called *Steiner cuts* and we denote the formulation by (STP_{uc}); this undirected cut formulation was first introduced by Aneja [5].

Another possibility for formulating the Steiner tree problem is by considering arborescences in the bidirection: it is easy to see that a minimum cost arborescence—rooted at an arbitrary terminal and containing all terminals—in the bidirection gives a cost minimal Steiner tree.

Let $G = (V, E, c)$ be the undirected weighted input graph with the weighted bidirection $\bar{G} = (V, A, c)$. Moreover, let the root r be a designated terminal. With the notation $V_r := V \backslash \{r\}$ and binary arc variables $z_a, \forall a \in A$, the directed cut formulation (first described by Wong [175]) for the Steiner tree problem is given by:

$$
\begin{aligned}
(\text{STP}_{\text{dc}}) \quad & \min \sum_{a \in A} c_a z_a \\
& \text{s.t. } z(\delta^-(S)) \geq 1 \qquad \forall S \subseteq V_r \colon S \cap T \neq \emptyset && (3.9) \\
& z \in \{0,1\}^{|A|} && (3.10)
\end{aligned}
$$

The constraints (3.9) are directed ingoing cuts and assure the construction of an r-rooted arborescence. The induced STP solution is given by the set $\{e = \{i, j\} \in E \mid z_{ij} = 1 \text{ or } z_{ji} = 1\}$.

By projecting the directed z variables into the space of undirected edge variables x by $x_e := z_{ij} + z_{ji}, \forall e = \{i, j\} \in E$, it follows directly that the undirected cut formulation is not stronger than the directed cut formulation. The classical example given by Figure 3.1 shows that there exist instances where model (STP_{dc}) gives a stronger lower bound. A formal proof can be found in, e.g., Chopra and Rao [44].

Theorem 3.7. *(STP_{dc}) is stronger than (STP_{uc}).*

Flow-based formulations. Analogously to formulation (NDP_{uf}) one can find equally strong flow formulations for the Steiner tree problem. One simply has to set one terminal as root node r and route a commodity from r to each terminal $t \in T_r$ ($T_r := T \backslash \{r\}$); hence, the number of commodities is $|T| - 1$. We denote the resulting formulation by (STP_{uf}).

$$(\text{STP}_{\text{uf}}) \quad \min \sum_{e \in E} c_e x_e$$

$$\text{s.t. } x_e \geq f_{ij}^t, x_e \geq f_{ji}^t \qquad \forall t \in T_r, \forall e = \{i,j\} \in E \tag{3.11}$$

$$f^t(\delta^-(i)) - f^t(\delta^+(i)) = \begin{cases} -1, & \text{if } i = r \\ 1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \quad \forall t \in T_r, \forall i \in V \tag{3.12}$$

$$f \in [0,1]^{|A| \cdot (|T|-1)} \tag{3.13}$$

$$x \in \{0,1\}^{|E|} \tag{3.14}$$

(STP_{uf}) is equivalent to (STP_{uc}). Moreover, there exists a stronger flow formulation which is equivalent to (STP_{dc}). This formulation can be obtained from (STP_{uf}) by replacing the undirected edge variables x by their directed counterparts z.

$$(\text{STP}_{\text{df}}) \quad \min \sum_{a \in A} c_a z_a$$

$$\text{s.t. } f \text{ satisfies (3.12)}$$

$$z_{ij} \geq f_{ij}^t \qquad \forall t \in T_r, \forall (i,j) \in A \tag{3.15}$$

$$f \in [0,1]^{|A| \cdot (|T|-1)} \tag{3.16}$$

$$z \in \{0,1\}^{|A|} \tag{3.17}$$

Theorem 3.8. *(STP_{df}) is equivalent to (STP_{dc}).*

By transitivity it follows that (STP_{df}) is stronger than (STP_{uf}). For formal proofs regarding the strength of the formulations we refer the reader again to, e.g., [44].

It is possible to formulate an equivalent directed flow-based model by replacing (i) directed arc variables by undirected edge variables x and (ii) constraints (3.15) by constraints $f_{ij}^{t_1} + f_{ji}^{t_2} \leq x_e$, for each edge $e = \{i,j\} \in E$ and each pair $t_1, t_2 \in T_r$, see, e.g., Goemans and Myung [74]:

$$(\text{STP}_{\text{df2}}) \quad \min \sum_{e \in E} c_e x_e$$

$$\text{s.t. } f \text{ satisfies (3.12)}$$

$$x_e \geq f_{ij}^{t_1} + f_{ji}^{t_2} \qquad \forall t_1, t_2 \in T_r, \forall e = \{i,j\} \in E \tag{3.18}$$

$$f \in [0,1]^{|A| \cdot (|T|-1)} \tag{3.19}$$

$$x \in \{0,1\}^{|E|} \tag{3.20}$$

Lemma 3.9 ([74]). *(STP_{df}), (STP_{dc}), and (STP_{df2}) are equivalent.*

3.4 Survivable network design problem

In this section we discuss the survivable network design problem. We mention polynomially solvable special cases and give references to approximation algorithms as well as parameterized algorithms. The main part contains the description of the well-known undirected cut and flow formulation for the SNDP, and we discuss how these formulations can be strengthened by considering the bidirection of the input graph.

3.4.1 Complexity, polynomially solvable cases, and parameterized algorithms

Complexity and approximation algorithms. Due to the Steiner tree problem the node- and edge-connectivity versions of the SNDP are strongly NP-hard and inapproximable within a factor of $96/95$. The best known approximation algorithm is due to Jain [99] and achieves a ratio of 2. In fact, this algorithm can deal with all network design problems from Section 3.2 where the connectivity function f is *weakly submodular*. These problems were introduced by Goemans and Williamson [75] who described approximation algorithms with ratio $2 \cdot \max\{\rho_{uv}\}$ for *proper* functions f. An overview of approximations, polynomially solvable cases, and the SNDP in general, was given by Kerivin and Mahjoub [108].

Polynomially solvable cases. The following two special cases of the ncSNDP as well as the SNDP can be solved in linear time on series-parallel graphs: Raghavan [147] presented an algorithm when connectivity requirements are in $\{0, 1, 2\}$ and Winter [173] showed how to solve the problem with connectivity requirements in $\{0\} \cup \{2\ell \mid \ell \in \mathbb{N}\}$. For an overview of further special cases we refer the reader to Kandyba-Chimani [105].

Parameterized algorithms. To the best of our knowledge, parameterized algorithms for the SNDP have not been studied so far. The only considered special cases are Steiner tree and Steiner forest problems, cf. Section 3.3.

Regarding treewidth, it was shown that the Steiner forest problem is already NP-hard for graphs with a treewidth of 3. This result was published independently by Gassner [71] and Bateni, Hajiaghayi, and Marx [13]. On the other hand, [13] described a polynomial-time algorithm for the SFP on graphs with treewidth 2.

Guo, Niedermeier, and Suchý [82] considered directed "Steiner problems" including the *directed Steiner tree problem*, *directed Steiner forest problem*, and *strongly connected subgraph problems*. The authors achieve some hardness results as well as some FPT results; one of the considered parameters is the number of terminals.

3.4.2 IP formulations

We introduce basic IP formulations based on undirected cuts and multicommodity flows for the SNDP in Section 3.2. For the STP the directed models provide stronger formulations, cf. Section 3.3.2. Magnanti and Raghavan [130] were able to strengthen the formulations (NDP_{uc}) and (NDP_{uf}) by using a famous theorem by Nash-Williams [134] about graph orientations that we restate here:

Theorem 3.10 (Nash-Williams [134]). *Let $G = (V, E)$ be an undirected graph and let κ_{uv} be the maximum number of edge-disjoint paths from u to v, where $u, v \in V$, $u \neq v$. Then G has an orientation such that for every pair of nodes u and v in G, the number of pairwise edge-disjoint directed paths from u to v in the resulting directed graph is at least $\lfloor \frac{\kappa_{uv}}{2} \rfloor$.*

If connectivity requirements are in $\{0,1\} \cup \{2\ell \mid \ell \in \mathbb{N}\}$ then it is possible to orient any optimal SNDP solution as follows: Since we are dealing with the unitary SNDP, any optimal SNDP solution consists of edge-biconnected components connected with each other by cutvertices or bridges. Using the result of Theorem 3.10, each of those edge-biconnected components can be oriented such that for each pair of distinct nodes u and v from the same component there exist $\rho_{uv}/2$ edge-disjoint directed paths from u to v and $\rho_{uv}/2$ edge-disjoint directed paths from v to u.

To orient possible bridges, a node r is chosen for which we know that it is a part of an edge-biconnected component and each bridge is oriented away from this component. To this end, the edge-biconnected components are oriented, shrunk into single nodes, and the obtained tree is oriented away from the "root" r. These orientation properties can be used to derive a MIP model that uses binary arc variables z_{ij} associated to the bidirection. By projecting the arc variables into the space of undirected edges as $x_e = z_{ij} + z_{ji}$, for all $e = \{i,j\} \in E$, it is not difficult to see that the obtained directed model is stronger than the undirected one. In fact, the directed model is stronger if and only if there exists a pair of distinct nodes $u, v \in V$, such that $\rho_{uv} = 1$, cf. Magnanti and Raghavan [130].

To model the general SNDP, i.e., the SNDP with arbitrary connectivity requirements $\rho_{uv} \in \mathbb{N}$, Magnanti and Raghavan [130] presented an extended MIP formulation which is similar to the one described above with the only difference that the binary arc variables z_{ij} are relaxed to be continuous. This minor change makes the model valid for arbitrary values of ρ_{uv} and provably stronger than its undirected counterpart.

By using fractional arc variables $z_a, \forall a \in A$, and $f(S) := \max\{\rho_{uv} \mid u \in S, v \notin S\}, \forall S \subset V$, the resulting model by [130] reads as follows:

$$
\begin{aligned}
(\mathrm{SNDP_{dc}}) \quad & \min \sum_{e \in E} c_e x_e & & \\
\text{s.t. } & z(\delta^-(S)) \geq f(S)/2 & \forall \emptyset \neq S \subseteq V\colon f(S) \geq 2 & \quad (3.21)\\
& z(\delta^-(S)) \geq 1 & \forall \emptyset \neq S \subseteq V_r\colon f(S) = 1 & \quad (3.22)\\
& x_e \geq z_{ij} + z_{ji} & \forall e = \{i,j\} \in E & \quad (3.23)\\
& z \geq \mathbf{0} & & \quad (3.24)\\
& x_e \in \{0,1\}^{|E|} & & \quad (3.25)
\end{aligned}
$$

Constraints (3.22) are the known directed Steiner cuts implying connectivity of the solution, i.e., the existence of a directed path from r to each "terminal". The directed cuts (3.21) ensure that $\rho_{uv}/2$ directed path are selected from u to v and from v to u, respectively. The capacity constraints (3.23) enforce that each edge is oriented in (at most) one direction and that each used arc is payed for.

The formal proof of correctness for ($\mathrm{SNDP_{dc}}$) as well as an equally strong model based on multicommodity flows can be found in Magnanti and Raghavan [130].

Lemma 3.11 ([130]). *($SNDP_{dc}$) is a valid formulation for the survivable network design problem and stronger than (NDP_{uc}).*

Part II

The two-stage stochastic Steiner tree problem

Chapter 4

Introduction

This chapter introduces the *two-stage stochastic Steiner tree problem* and its related rooted version. The problem definitions are given in Section 4.1. Afterwards, in Section 4.2, some basic examples are discussed which lead to some simple observations. Section 4.3 contains an overview of the complexity and the related work. We close the chapter with Section 4.4 by describing linear time algorithms when the input graph is a tree.

4.1 Definitions

The *stochastic Steiner tree problem* is a natural extension of the classical STP to a two-stage stochastic combinatorial optimization problem. In the first stage, it is possible to buy some "profitable" edges while the terminal set and the edge costs are subject to uncertainty. Then, in the second stage, one of the given scenarios is realized and additional edges have to be purchased in order to interconnect the now known set of terminals. The objective is to make a decision about edges to be selected in the first stage, while minimizing the expected cost of the overall solution.

For the used notations see page vii and Section 2.3. Moreover, let edge set E^0 denote the selected edges in the first stage and E^k the edges in the second stage in scenario k, $\forall k \in \mathcal{K}$, with $\mathcal{K} := \{1, \dots, K\}$. The *stochastic Steiner tree problem* is defined as follows:

Problem 4.1 (Stochastic Steiner tree problem (SSTP)):
Given: *undirected graph* $G = (V, E)$*, first-stage edge cost* $c_e^0 \in \mathbb{R}^{>0}, \forall e \in E$*, and a set of* $K \geq 1$ *scenarios. Each scenario* $k \in \mathcal{K}$ *is defined by its probability* $p^k \in (0; 1]$*, second-stage edge cost* $c_e^k \in \mathbb{R}^{>0}, \forall e \in E$*, and a set of terminals* $\emptyset \neq T^k \subseteq V$*. Moreover, it holds* $\sum_{k \in \mathcal{K}} p^k = 1$.
Solution: $K + 1$ *edge sets* $E^0, \dots, E^K \subseteq E$ *such that* $G[E^0 \cup E^k]$ *connects* $T^k, \forall k \in \mathcal{K}$
Objective: *minimize the expected cost* $\sum_{e \in E^0} c_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E^k} c_e^k$

The *expected cost of an edge* $e \in E$ is defined as $c_e^* := \sum_{k \in \mathcal{K}} p^k c_e^k$. W.l.o.g. one can assume that $c_e^0 < c_e^*, \forall e \in E$; otherwise, this edge would never be purchased in the first stage since it can be installed in every scenario at the same or cheaper cost. On the other hand it is also valid to assume $c_e^0 > \min_{k \in \mathcal{K}}\{p^k c_e^k\}, \forall e \in E$, since such an edge would never be installed in any scenario.

In the literature, cf. Section 4.3, the SSTP is mostly considered to have some special terminal $r \in V$. This vertex r is a designated root node and is contained in every terminal

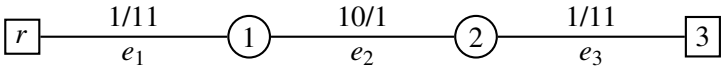


Figure 4.1: A simple example for the rooted stochastic Steiner tree problem where applying the assumption "$c_e^0 < c_e^*, \forall e \in E$" is not feasible, cf. text. There exists only one scenario (connect terminals r and 3) and edge costs are written above the edges (as first stage cost/scenario cost). The optimum solution selects all edges in the first stage with overall cost 12. Disabling e_2 in the first stage would imply cost 13.

set $T^k, \forall k \in \mathcal{K}$. We refer to this special case of the stochastic Steiner tree problem as *SSTP with global terminal*—in particular, we want to make the distinction to the *rooted SSTP* clear which is motivated and defined in the following.

Notice that for the SSTP the first-stage solution E^0 does not have to be connected. In particular, it is easy to construct instances with the optimum first-stage solution being disconnected, cf. Section 4.2. However, fragmented solutions might be unreasonable in practical settings. For example, if new cables or pipes are installed underground one would prefer starting at one point and connecting adjacent streets first and not by digging in several parts of the city simultaneously. These ideas lead to the problem definition of the *rooted SSTP*:

Problem 4.2 (Rooted stochastic Steiner tree problem (rSSTP)):

Given: *undirected graph* $G = (V, E)$*, first-stage edge cost* $c_e^0 \in \mathbb{R}^{>0}, \forall e \in E$*, root node* $r \in V$*, and a set of* $K \geq 1$ *scenarios. Each scenario* $k \in \mathcal{K}$ *is defined by its probability* $p^k \in (0; 1]$*, second-stage edge cost* $c_e^k \in \mathbb{R}^{>0}, \forall e \in E$*, and a set of terminals* $\emptyset \neq T^k \subseteq V$ *with* $r \in T^k$*. Moreover, it holds* $\sum_{k \in \mathcal{K}} p^k = 1$.

Solution: $K + 1$ *edge sets* $E^0, \ldots, E^K \subseteq E$ *such that* $G[E^0]$ *is a tree containing* r *and* $G[E^0 \cup E^k]$ *connects* $T^k, \forall k \in \mathcal{K}$

Objective: *minimize the expected cost* $\sum_{e \in E^0} c_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E^k} c_e^k$

The two problems SSTP and rSSTP are the main stochastic Steiner tree variants considered in this thesis.

Notice that the assumption $c_e^0 < c_e^*, \forall e \in E$, as for the SSTP, is not valid for the rSSTP due to the necessary first-stage tree. This is shown by Figure 4.1; here, edge e_2 would be disabled in the first stage which prohibits the optimum solution. By swapping first- and second-stage edge costs this example shows that this holds for assumption $c_e^0 > \min_{k \in \mathcal{K}}\{p^k c_e^k\}$ as well.

Relationship between SSTP and rSSTP. Although both variants of the stochastic Steiner tree problem are related and seem similar there are some differences. Obviously, one can interpret any rSSTP instance as SSTP instance and the optimum unrooted stochastic Steiner tree is at least as cheap as the rooted one. Hence, the SSTP can be seen as a relaxation of the rSSTP since the rooted version adds the constraint for a connected first-stage solution. On the other hand, instances for the SSTP do not necessarily contain a root node and do not have to be feasible for the rSSTP.

Moreover, the (known) complexity varies for some classes considered in this thesis. For example, the rooted version allows for an FPT algorithm parameterized by the overall number of terminals; for the unrooted version it is an open question if such an algorithm exists. Moreover, it is possible to formulate the rooted SSTP by a fully directed MIP

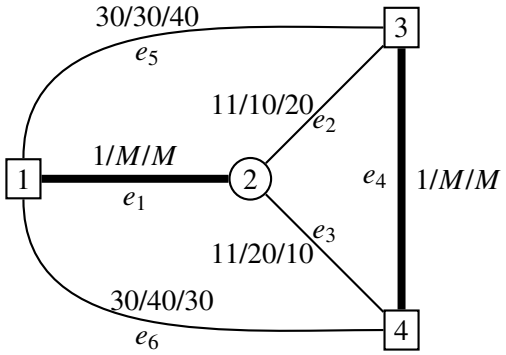


Figure 4.2: Example SSTP instance with two equally probable scenarios having the same set of terminals {1,3,4}. Edge costs for the first stage and the two scenarios are written next to the edges (as $c^0/c^1/c^2$) with M representing a sufficiently large positive value. The optimum solution edges of the first stage are highlighted by thick edges; scenario 1 and 2 additionally purchase edge e_2 and e_3, respectively. Hence, the optimum solution has cost 12.

formulation but for the unrooted SSTP this attempt causes some difficulties—as discussed in Section 6.3. But there are also similar results concerning trees, cf. Section 4.4, and the FPT complexity on partial 2-trees and on treewidth-bounded graphs, cf. Section 5.2, 5.5, and 5.6.

4.2 Observations and examples

In this section we like to discuss some observations and examples to give a better insight into the stochastic Steiner tree problems. For this purpose, let E^0 denote the optimal first stage and $E^1, \ldots, E^K$ the optimal scenario edges of an instance.

Observation 4.3. *The following edge sets are all cycle-free: E^0, $E^k, \forall k \in \mathcal{K}$, and $E^0 \cup E^k$, $\forall k \in \mathcal{K}$.*

Otherwise one can simply remove one edge from the cycle—either in the first stage or in a scenario—and obtain a cheaper solution; recall that all edge costs are positive. This observation obviously holds for the SSTP as well for the rSSTP.

Observation 4.4. *For the SSTP edge set E^0 may be disconnected.*

An example is already given by Figure 4.1 when interpreted as SSTP instance. Here, the optimum solution with (expected) cost 3 is $E^0 = \{e_1, e_3\}$ and $E^1 = \{e_2\}$. Figure 4.2 presents a slightly more complex example with 2 scenarios. Here, edges e_1 and e_4 are selected in the first stage and e_2 and e_3 each in one scenario, respectively.

Moreover, when solving both scenarios of the example in Figure 4.2 independently as Steiner tree problems the optimum solutions use edges e_2, e_3, e_5 for scenario 1 and edges e_2, e_3, e_6 for scenario 2. Hence, both independent scenario solutions are disjoint to the optimum first-stage solution for the SSTP.

Observation 4.5. *Let $F^k, \forall k \in \mathcal{K}$, denote the optimum solution edges to the Steiner tree problem induced by scenario k. It possibly holds $F^k \cap E^0 = \emptyset$, for all $k \in \mathcal{K}$.*

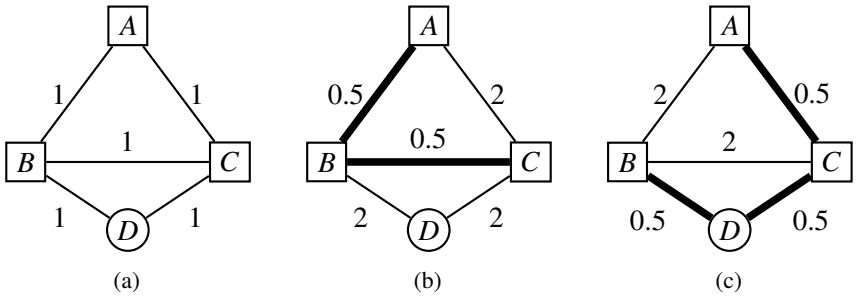


Figure 4.3: An instance for the SSTP with two equally probable scenarios which have the same terminal sets $T^1 = T^2 = \{A, B, C\}$. However, the optimum solution selects no edge in the first stage. (a) gives the edge costs of the first stage which are all 1. (b) and (c) depict the two scenarios with the selected edges of the optimum solution being drawn as bold lines. The overall solution cost is 1.25 whereas the selection of any first-stage edge would lead to higher cost.

In other words, solving the scenarios independently does not give any information about the corresponding first-stage solution which might not be used in any scenario. Contrarily, using an edge in all scenarios does not indicate a profitable edge for the first stage neither.

Observation 4.6. *Let $F^k, \forall k \in \mathcal{K}$, denote the optimum solution edges to the Steiner tree problem induced by scenario k with an edge e^* such that $e^* \in \bigcap_{k\in\mathcal{K}} F^k \neq \emptyset$. There exist instances with $e^* \notin E^0$.*

The same example gives such an instance where edges e_2 and e_3 are selected in both scenarios but they are not contained in the optimum first-stage solution.

Moreover, Figure 4.2 shows that it might not be optimal to connect all coinciding terminals already in the first stage. Here, every scenario has the same terminal set but E^0 is not a Steiner tree for this set.

Notice that although these latter three properties do not hold for the rooted SSTP and this very example in Figure 4.2 it is easy to see that there exist instances where the statements are also true for the rSSTP.

The stochastic Steiner tree problems do not contain terminals in the first stage and in general it is not possible to derive connectivity requirements from the scenarios into the first stage. Consider the example given by Figure 4.3 which consists of two equally probable scenarios: Although both terminal sets are identical the optimum first-stage solution is empty. Hence, the connectivity requirements in the scenarios do not imply any connectivity requirements in the first stage and cannot be imposed in the first stage.

Observation 4.7. *The terminal sets do not imply connectivity requirements in the first stage.*

For the deterministic STP it is valid to assume that the edge costs satisfy the triangle inequality, i.e., the input graph $G = (V, E, c)$ is a complete graph and for any three distinct vertices $u, v, w \in V$ it holds $c_{uv} \leq c_{uw} + c_{wv}$, see, e.g., Vazirani [169]. The STP with this property is called the *metric Steiner tree problem*. We use the same naming and call the (r)SSTP the *metric (rooted) stochastic Steiner tree problem* if the edge costs in the first stage c^0 and if the edge costs c^k in every scenario $k \in \mathcal{K}$ satisfy the triangle inequality.

Observation 4.8. *The SSTP is equivalent to the metric SSTP and the rSSTP is equivalent to the metric rSSTP.*

Proof. The proof is similar to the one for the STP as given by, e.g., [169]. We argue that the original instance $\mathcal{I}$ and the *metric closure* $\mathcal{I}'$ (the complete graph where edge costs in the first stage and each scenario equal the costs of the shortest paths) have the same optimum solution cost $\text{opt}_{\mathcal{I}} = \text{opt}_{\mathcal{I}'}$.

The first direction "$\text{opt}_{\mathcal{I}} \geq \text{opt}_{\mathcal{I}'}$" holds since edge costs in $\mathcal{I}'$ are no more than in $\mathcal{I}$. Hence, a solution in $\mathcal{I}$ can be directly translated into a solution for $\mathcal{I}'$ with the same or cheaper cost.

The opposite direction "$\text{opt}_{\mathcal{I}} \leq \text{opt}_{\mathcal{I}'}$" follows from the following transformation. For an optimal solution in $\mathcal{I}'$ all edges can be replaced by the shortest paths in the original graph; notice that the cost of this solution is not worse than $\text{opt}_{\mathcal{I}'}$. Afterwards, cycles can be removed by deleting edges from the solution and a selected second-stage edge which is already used in the first stage can be discarded, too. Of course, these modifications do not increase the overall cost. □

4.3 Complexity and related work

Since the stochastic Steiner tree problem is obviously a generalization of the classical Steiner tree problem the known complexity results can be carried over directly. Hence, the stochastic STPs are NP-hard and there is no polynomial time approximation scheme unless NP = P.

Although the STP allows constant factor approximations the stochastic problems are harder to approximate. Ravi and Sinha [150] showed that the *group Steiner tree problem* can be reduced to the *stochastic shortest path problem (SSP)*; the SSP is identical to the rSSTP or the SSTP with a global terminal, respectively, and with exactly one additional terminal in each scenario. The *group Steiner tree problem* is $\Omega(\log^{2-\varepsilon} n)$-hard to approximate due to a reduction from the *label cover problem*, cf. Halperin and Krauthgamer [90] and Dinur and Safra [62].

However, in literature the stochastic Steiner tree problem has been mostly investigated for approximation algorithms. Due to the inapproximability results restricted versions of the stochastic Steiner tree problem have been considered to obtain approximation algorithms.

The *classical* approach for making the problem better tractable is by introducing an *inflation factor* for the second-stage edges. Instead of independent edge costs in the scenarios each scenario has a fixed inflation factor $\sigma^k, \forall k \in \mathcal{K}$, mostly with $\sigma^k > 1$, such that all edge costs are increased by the same factor and edge e is assigned cost $\sigma^k c_e^0$ in scenario k. It is also possible to allow only one *uniform inflation factor* σ for the second-stage costs, i.e., $\sigma = \sigma^1 = \ldots = \sigma^K$.

Moreover, there are several other possibilities for reducing the problem complexity. For example, it is common to assume a special vertex being a terminal in all scenarios. In the literature, this vertex is referred to as root node but due to the overloaded naming w.r.t. the rSSTP we call such a vertex a global terminal. This assumption lets the problem become similar to the rSSTP without introducing the requirement for a first-stage tree.

Scenario representations. We assume that the random variables of the stochastic problems have finite support, compare the definition in Section 2.3. Hence, it is possible to describe the possible outcomes with a finite set of scenarios for which the probabilities are known. This stochastic model is known under the term *finite scenario model* or *polynomial*

scenario model. Running times of algorithms designed for the finite scenario model are measured in the input size which contains the full scenario representations.

A different approach is the *black box model* or *oracle model* where the number of scenarios might be exponential. In this model, the set of scenarios is not given explicitly, i.e., the distribution is not known. Instead, there exists an oracle that returns one scenario—depending on its probability—in polynomial time. Moreover, the sampling procedure of asking the oracle is considered to be an elementary operation. *Polynomial time algorithms* for the black box model are algorithms with polynomial running time and a polynomial number of oracle calls. For a nice overview and further references we refer the reader to the article by Shmoys and Swamy [163].

Finite scenario model approximations. As far as we know there are two results concerning the stochastic Steiner tree problem in the finite scenario model as considered in this thesis. Further approaches deal with the black box/oracle model which are listed in the following paragraph.

Gupta, Ravi, and Sinha [87] considered the SSTP with K inflation factors and a global terminal. The authors present a 40-approximation based on rounding of the relaxed LP followed by a primal-dual phase which is inspired by the classical approximation algorithm for the Steiner tree and forest problem by Goemans and Williamson [75]. As an interesting side note Gupta, Ravi, and Sinha [87] could show that the additional restriction of a valid first stage being a tree rooted at the global terminal at most doubles the cost of the optimum solution.

Gupta and Kumar [83] considered the so-called *stochastic Steiner forest problem* which is a special case of the SSTP. The main difference to the previously mentioned work is the non-existence of a global terminal. On the other hand, their problem contains a uniform fixed inflation factor $\sigma > 1$, the edge costs in the first stage have to be positive integers, and the scenarios are equally probable. For this version Gupta and Kumar [83] described a constant factor approximation based on a primal-dual scheme.

Black box model approximations. All following approximation algorithms are based on the common idea of *scenario sampling*. In this approach a specified number of scenarios is sampled by calling the scenario oracle. The exact number mostly depends on the uniform inflation factor and for an inflation factor σ the sample size is mostly $\lfloor \sigma \rfloor$. Approximating the sampled scenarios allows for an approximate solution to the related stochastic problem.

To the best of our knowledge, Immorlica, Karger, Minkoff, and Mirrokni [98] presented the first approximation algorithm. Their problem is restricted by a uniform inflation factor and the algorithm achieves an approximation guarantee of $O(\log n)$. Gupta, Pál, Ravi, and Sinha [85, 89] introduced the concept of *boosted sampling* which is a sophisticated version of the described sampling approach. The authors consider the problem with a global terminal and a uniform inflation factor and present a constant factor approximation algorithm with ratio 3.55. The same problem is considered by Shmoys and Swamy [163] who presented a 4-approximation.

Contrarily to the other problems, Gupta and Pál [84] approximated the stochastic Steiner tree without global terminal. Their problem has a fixed uniform inflation factor and the presented algorithm has a ratio of 12.6.

The only approach without uniform inflation factor is due to Gupta, Hajiaghayi, and Kumar [86]. The considered SSTP does not have inflation factors but there are only two

cost functions for the first-stage edges and one for the second-stage edges. Hence, in this problem the second-stage costs are independent of the realized terminal sets. The authors show that this restricted problem is at least $\Omega(\log \log n)$-hard and give an approximation algorithm with a polylogarithmic approximation ratio.

Heuristic. Hokama, San Felice, Bracht, and Usberti [95] introduced a heuristic for the SSTP and present results of a computational study. The heuristic is a biased random-key genetic algorithm which in its core is a general search metaheuristic. Although the method is described only for the SSTP it can be modified to work for the rSSTP as well. To populate the initial set of solutions for the genetic algorithm two heuristics were introduced: (i) a buy-none heuristic where the first stage is always empty and (ii) a procedure which iteratively adds profitable edges to the first stage. In both cases the scenarios are solved with the MST heuristic. Hokama, San Felice, Bracht, and Usberti [95] evaluated the genetic algorithm experimentally on the instances from Bomze, Chimani, Jünger, Ljubić, Mutzel, and Zey [27] and from the 11th DIMACS challenge [101]. To the best of our knowledge these two publications are the only ones which present computational results for the SSTP. We discuss and compare the approaches in our computational study in Section 8.3.6.

4.4 Stochastic Steiner tree problems on trees

Solving the deterministic Steiner tree problem on trees is trivial—and can be done in linear time—because the (shortest) path between any two vertices is unique: Comprising the edges of the paths connecting all pairs of terminals gives the minimum Steiner tree. The same result holds for the Steiner forest problem.

Hence, trees seem to be a simple graph class for the stochastic Steiner tree problem. This is true for the SSTP since there exists a straight-forward linear time algorithm, as discussed in the next paragraph. Afterwards, we see that solving the rSSTP on a tree is not that trivial but can be done in linear time, too.

SSTP on trees. Like already mentioned the deterministic Steiner tree problem is solved by combining the shortest paths between all terminal pairs. By using this method independently for each scenario one can find all *relevant* edges $\tilde{E} := \bigcup_{k \in \mathcal{K}} \tilde{E}^k$; thereby $\tilde{E}^1, \dots, \tilde{E}^K$ are the required edges for each of the K scenarios.

Obviously, an optimum solution uses only edges from $\tilde{E}$ and it only remains to decide for each edge if this edge is bought in the first or in the second stage. Consider any edge $e \in \tilde{E}$: If $c_e^0 < \sum_{k \in \mathcal{K}: e \in \tilde{E}^k} p^k c_e^k$ it is cheaper to select e as a first-stage edge. Otherwise e is used as a second-stage edge in all scenarios k with $e \in \tilde{E}^k$.

The constructed solution obviously connects all terminals and hence, it is feasible. Moreover the solution is optimal since every other solution has to use the edges from $\tilde{E}$. The running time is linear in the size of the stochastic instance and dominated by the time required for computing edge sets $\tilde{E}^k, \forall k \in \mathcal{K}$, which can be done in $O(K \cdot |V|)$ time.

rSSTP on trees. Solving the rooted stochastic Steiner tree problem on trees is not as simple as the unrooted version due to the required first-stage tree. Hence, it is not possible to decide for each edge independently if it should be used in the first stage or in the scenario solutions. However, by considering the input tree rooted at r (recall that r is the root node

from the rSSTP) and by using a bottom-up traversal allows for a simple optimality equation which can be evaluated for each node. For the description we assume that the edge sets $\tilde{E}, \tilde{E}^1, \ldots, \tilde{E}^K$—as described in the preceding paragraph—are already known. Moreover, let v be the current node with u being its parent in the tree; the corresponding edge is denoted by $e = \{u, v\}$.

One can observe that in any solution v is reachable from the root solely by first-stage edges, or the path consists of an (empty) sub-path of first-stage edges followed by some scenario edges. To this end we store two values $b_1(v)$ and $b_2(v)$ which represent both choices for connecting vertex v. $b_1(v)$ gives the minimum cost for connecting v via e as first-stage edge and all terminals in the subtree rooted at v with a minimum v-rooted stochastic Steiner tree. $b_2(v)$ stores the minimum cost for connecting v by e and all terminals in the subtree rooted at v by second-stage edges only.

As base case we set $b_1(v) := b_2(v) := 0$ for all nodes $v \in V$ with $v \notin G[\tilde{E}]$, i.e., v is not a terminal and the subtree rooted at v does not contain any terminal. Otherwise and in case $v \neq r$ the two following optimality equations hold:

$$b_1(v) := c_e^0 + \sum_{\substack{w \in V: \\ w \text{ child of } v}} \min\{b_1(w), b_2(w)\} \tag{4.1}$$

$$b_2(v) := \sum_{k \in \mathcal{K}: e \in \tilde{E}^k} p^k c_e^k + \sum_{\substack{w \in V: \\ w \text{ child of } v}} b_2(w) \tag{4.2}$$

The overall optimum solution value $b(r)$ can be computed by evaluating the following equation in the root node: $b(r) := \sum_{w \in V: w \text{ child of } r} \min\{b_1(w), b_2(w)\}$. Using a bottom-up traversal, each edge is considered only a constant number of times per scenario; Hence, the overall running time is $O(K \cdot |V|)$. The constructed solution is feasible since all terminals are obviously connected and first-stage edges are only selectable when the current node is connected by first-stage edges (case (4.1)) or incident to the root node (case $b(r)$) which leads to a first-stage tree rooted at r.

The optimality of the constructed solution can be seen by a simple inductive proof on the tree: leaves of $G[\tilde{E}]$ are assigned correct values and the values of the other nodes are computed correctly since all child values are correct.

After extracting the optimum solution value in the root node one can use a backtracking procedure for computing the actual edge sets. This procedure also runs in linear time of the input size.

Lemma 4.9. *The SSTP as well as the rSSTP can be solved in time $O(K \cdot |V|)$ on trees.*

We close the discussion by referring the interested reader to another version of the stochastic Steiner tree problem which was introduced by Kurz [115]. His problem definition requires a first-stage tree but there does not exist a common root node, contrarily to the rSSTP. Moreover, each scenario solution combined with the selected edge set from the first stage has to consist of one connected component. Kurz showed that this problem can be solved in time $O(|V|^3 + K \cdot |V|)$ on trees.

Notice that by calling our algorithm $|V|$ times—by considering each vertex as root node—this problem can be solved in time $O(K \cdot |V|^2)$ as well.

Chapter 5

Parameterized algorithms

In this chapter we introduce FPT algorithms for the stochastic Steiner tree problems by considering the following parameters: the overall number of terminals, the treewidth, and the number of scenarios. We start in Section 5.1 with the number of terminals. Afterwards, we consider tree decompositions where we first take a look at graphs with treewidth 2, the partial 2-trees (Section 5.2). Section 5.3 shows that the SSTP is already NP-hard on graphs with treewidth 3. Afterwards, we consider the general case and describe treewidth-based algorithms for the SSTP. We summarize the ideas of the deterministic case in Section 5.4. Section 5.5 deals with the expansion of this approach to the SSTP and Section 5.6 summarizes the necessary modifications for the rSSTP.

5.1 Number of terminals

The famous algorithm by Dreyfus and Wagner [63] is based on dynamic programming and solves the deterministic STP on a graph $G = (V, E)$ with terminals T in time $O(3^{|T|}n + 2^{|T|}n^2 + nm)$, with $n = |V|, m = |E|$. This result places the STP in the complexity class FPT parameterized by the number of terminals. In fact, this algorithm also works for the *directed Steiner tree problem*, see Kurz [115], Kurz, Mutzel, and Zey [116].

Problem 5.1 (Directed Steiner tree problem (dSTP)):
Given: *directed graph* $G = (V, A, c)$*, arc cost* $c_a \in \mathbb{R}^{>0}, \forall a \in A$*, set of terminal vertices* $\emptyset \neq T \subseteq V$*, root node* $r \in T$
Solution: *arc set* $A' \subseteq A$ *such that there exists a directed path from* r *to each terminal* $t \in T \backslash \{r\}$
Objective: *minimize overall cost* $\sum_{a \in A'} c_a$

Obviously, the STP can be reduced to the dSTP by considering the weighted bidirection of the undirected input graph and by setting the root node r to an arbitrary terminal. Moreover, it is easy to see that the algorithm by Dreyfus and Wagner can be adapted for solving the dSTP. In fact, the algorithm already considers directed components and hence, only minor modifications are necessary.

As we will discuss in the following the rooted stochastic Steiner tree problem can be solved by the directed STP. Hence, the algorithm by Dreyfus and Wagner gives an FPT algorithm for the rSSTP parameterized by the number of terminals.

For solving the rSSTP we describe a polynomial reduction to the dSTP. Let the input of the rSSTP be the undirected weighted graph $G = (V, E, c)$ with root node r and let A be the

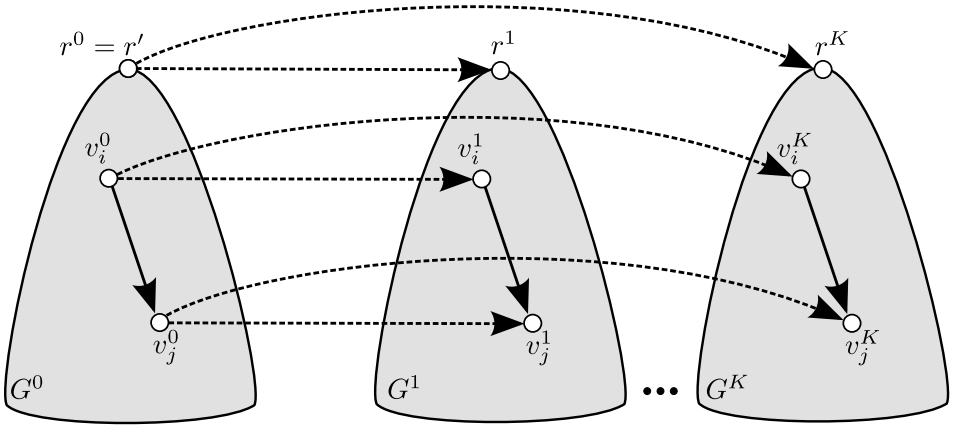


Figure 5.1: Sketch of the constructed graph G' in the reduction from the rSSTP to the dSTP, cf. text. Dashed arcs are transition arcs with cost 0.

arc set of the bidirection of G. We construct a dSTP instance $G' = (V', A')$ with root node r', terminal set T', and arc costs w such that the optimum solution values of both instances are identical.

The dSTP instance basically consists of $K+1$ copies of the bidirection of G: a first-stage copy $G^0 := (V^0, A^0)$ and K scenario copies $G^1, \dots, G^K$ with $G^k := (V^k, A^k), \forall k \in \mathcal{K}$. An arc (i, j) in G^0 has cost $c^0_{\{i,j\}}$ and in the kth scenario copy it has cost $p^k c^k_{\{i,j\}}$. The first-stage copy of the root is the overall root. Moreover, each vertex in a scenario copy has one additional ingoing arc with the source of this arc being the corresponding vertex in the first-stage copy; these *transition* arcs have cost 0. Figure 5.1 depicts the structure of the graph G'. Formally, the dSTP instance is constructed as follows:

- $V' := \bigcup_{\ell=0}^{K} V^\ell$ with $V^\ell := \{v_1^\ell, \dots, v_n^\ell\}$; v_i^ℓ is the copy of vertex v_i in the first stage ($\ell = 0$) or in the ℓth scenario ($\ell \in \mathcal{K}$)
- $A' := A^* \cup \bigcup_{\ell=0}^{K} A^\ell$ with:
  - $A^0 := \left\{(v_i^0, v_j^0) \,\middle|\, (i, j) \in A\right\}$; (v_i^0, v_j^0) is the first-stage copy of arc (i, j) with cost $w^0_{ij} := c^0_{\{i,j\}}, \forall (i, j) \in A$
  - $A^k := \left\{(v_i^k, v_j^k) \,\middle|\, (i, j) \in A\right\}$; (v_i^k, v_j^k) is the kth scenario copy of arc (i, j) with cost $w^k_{ij} := p^k c^k_{\{i,j\}}, \forall (i, j) \in A, \forall k \in \mathcal{K}$
  - $A^* := \left\{(v_i^0, v_i^k) \,\middle|\, i \in \{1, \dots, n\}, k \in \mathcal{K}\right\}$; the cost of each transition arc is 0
- $r' := r^0$; r^0 is the first-stage copy of the root node r
- $T' := \bigcup_{k \in \mathcal{K}} \left\{v_i^k \,\middle|\, v_i \in T^k\right\}$

Now, consider an optimum solution for this dSTP instance which is an arborescence rooted at r'. Since there exists no arc entering the first-stage copy from the other copies the subgraph of the optimum solution in A^0 is a connected tree. Moreover, all terminals in T' are connected which are vertices in the scenario copies. Hence, a terminal v_i^k in G^k

is connected by a directed path consisting of arcs in A^0, followed by one transition arc in A^*, and arcs in A^k. Since the transition arcs have zero cost the solution can be transformed into a solution for the rSSTP with the same cost. Arcs in the subgraph G^0 are transformed into first-stage edges and arcs in the graphs $G^1, \ldots, G^K$ are scenario edges. This solution obviously consists of an r-rooted first-stage tree and all terminals in all scenarios are connected.

On the other hand, an optimum solution to the rooted stochastic Steiner tree problem can be transformed easily into an equivalent solution for the corresponding dSTP instance. One simply has to orient the arcs from the root node away and whenever a path switches from first to second-stage edges a transition arc is used. This obviously occurs exactly once per path in the optimum rSSTP solution.

Hence, applying the algorithm by Dreyfus and Wagner to the constructed dSTP instance solves the rooted stochastic Steiner tree problem. Let $t^* := \sum_{k \in \mathcal{K}} |T^k|$ denote the overall number of terminals. The constructed directed graph consists of $(K + 1)n$ vertices and $2m(K + 1) + Kn$ arcs, respectively.

Lemma 5.2. *The rooted stochastic Steiner tree problem (rSSTP) can be solved by an* FPT *algorithm parameterized by the overall number of terminals t^* with running time $O(3^{t^*}Kn + 2^{t^*}K^2n^2 + K^2nm)$.*

Since this approach uses the algorithm by Dreyfus and Wagner as a black box it is possible to take advantage of known faster algorithms. Björklund, Husfeldt, Kaski, and Koivisto [21] described how to speed up the algorithm by Dreyfus and Wagner directly by fast subset convolution. Their approach has a running time of $\tilde{O}(2^{|T|}nm)$ for the Steiner tree problem with integer edge weights which are bounded by a constant; notice that polylogarithmic factors are hidden by $\tilde{O}$, cf. [21]. Moreover, Fuchs, Kern, Mölle, Richter, Rossmanith, and Wang [69] described a different algorithm for the STP based on dynamic programming with running time $O((2 + \delta)^{|T|}n^{O(1/(\delta/\ln(1/\delta))^{\zeta})})$, for any $1/2 < \zeta < 1$ and sufficiently small $\delta > 0$.

It is also possible to solve the SSTP variant defined by Kurz [115] and Kurz, Mutzel, and Zey [116] (cf. Section 4.4) by adopting the described transformation: recall that this problem has no designated root node but still requires the first stage solution being a tree. One simply has to add a virtual root node and connect it with directed arcs to all vertices of G^0. The costs of these arcs are set to a sufficiently large value such that an optimum solution chooses exactly one of these arcs. More detailed proofs and further discussions can be found in [115, 116]. For example, it was shown that the *rooted stochastic prize-collecting Steiner tree problem* can be solved with the same transformation.

We like to close the discussion by mentioning three open problems.

Open problem 5.1. Is the stochastic Steiner tree problem (SSTP) parameterized by the overall number of terminals in FPT?

Since the dSTP algorithm is used as a black box in our approach it is an interesting question if it is possible to adapt the algorithm for the stochastic problems.

Open problem 5.2. Is it possible to adapt the algorithm by Dreyfus and Wagner [63] directly for the stochastic Steiner tree problems?

Although there exists a simple FPT algorithm with parameter number of non-terminals for the STP it is an open problem if this result holds for the stochastic problems, too.

Open problem 5.3. Are the stochastic Steiner tree problems (SSTP and rSSTP) parameterized by the overall number of non-terminals in FPT?

Remark 5.3. The previously described reduction from the rSSTP to the dSTP is approximation preserving. As already discussed in Section 4.3 this relationship does not yield *good* approximation algorithms since the dSTP is as hard to approximate as the *label cover problem* which is $\Omega(\log^{2-\varepsilon} n)$-hard.

5.2 Partial 2-trees

Wald and Colbourn [170] showed that the Steiner tree problem can be solved in linear time on partial 2-trees. The idea of the algorithm is based on the recursive way of constructing these graphs: start with a simple edge and in each step add a vertex that is adjacent to both endpoints of an edge. Therefore, in a 2-tree there always exists a vertex with degree 2 which is removed by the algorithm in each iteration. While removing a vertex all necessary information about the attached subgraphs are stored in a constant number of variables representing the best solutions w.r.t. the removed subgraphs. Because of the restricted structure of these graphs there are only 6 relevant cases (variables) to be considered in each iteration; for details we refer the reader to [170].

For the stochastic Steiner tree problem this idea can be adopted, as described in Bökler, Mutzel, and Zey [25] and Bökler [26]. The basic idea remains unchanged: In each step a vertex with degree 2 is removed and all necessary solution values are stored. However, the number of distinguishable cases increases highly and is not a constant anymore. The difficulty arises from the set of scenarios. Although the K scenarios are somehow independent they influence, and are influenced by, the first-stage decisions. Therefore, all relevant combinations of the first stage and the K scenarios have to be considered.

The complete description of all cases is quite technical and lengthy. For the SSTP there are 9^K variables and the evaluation for each degree-2 vertex takes time $\Theta(16^K)$. Overall, the running time is $O(16^K|V|)$.

Theorem 5.4 ([25]). *The stochastic Steiner tree problem on partial 2-trees and parameterized by the number of scenarios is in* FPT. *There exists an algorithm with running time* $O(16^K|V|)$.

For more details we refer the reader to Bökler, Mutzel, and Zey [25] or Bökler [26]. Although the problem was not investigated in [25, 26] the rooted stochastic Steiner tree problem on partial 2-trees can be parameterized by the number of scenarios, too. We describe an algorithm for general treewidth in Section 5.6.

We like to close the discussion on partial 2-trees with an open problem regarding the complexity. As we will see in the next part the SSTP is already NP-hard on graphs with treewidth 3. For treewidth 2, the complexity is unknown.

Open problem 5.4. Are the stochastic Steiner tree problems (SSTP and rSSTP) on partial 2-trees solvable in polynomial time?

5.3 Treewidth-bounded graphs

The stochastic Steiner tree problems are solvable in polynomial time on trees and for treewidth 2 the SSTP is fixed parameter tractable with the parameter being the number of scenarios.

However, it is still open if the problems are already NP-hard for treewidth 2. For treewidth 3, we can show that the SSTP is indeed NP-hard. We use a polynomial reduction from the Steiner forest problem which was shown to be NP-hard on graphs with any fixed treewidth ≥ 3 independently by Bateni, Hajiaghayi, and Marx [13] and Gassner [71].

The main idea of our reduction is to encode each terminal set of the SFP into a single scenario and to use unprofitably large edge costs in the second stage. The first-stage edge costs are the given edge costs by the SFP instance. Then, the optimum solution to this SSTP instance buys exclusively edges in the first stage which correspond to the optimum solution for the SFP.

Theorem 5.5 ([25]). *The Steiner forest problem (SFP) is polynomial time reducible to the stochastic Steiner tree problem (SSTP).*

Proof. The constructed SSTP instance uses the same graph with first-stage edge costs $c_e^0 := c_e, \forall e \in E$, and K equally probable scenarios: $p^k := 1/K, \forall k \in \mathcal{K}$. The terminal set of scenario k is set to $T^k, \forall k \in \mathcal{K}$, and all edge costs to "big M"; here it suffices to set $M := 1 + K \cdot \max_{e \in E} c_e$. Therefore, buying an edge e in the first stage is even cheaper than buying e in a single scenario since $1/K \cdot M > c_e^0$.

Intuitively, the SSTP now constructs the minimum Steiner forest by connecting all terminal sets already in the first stage. We argue that the SFP instance has a solution with value z if and only if the SSTP instance has a solution with value z.

Let $\hat{E}^0, \hat{E}^1, \ldots, \hat{E}^K, \hat{E}^\ell \subseteq E, \forall \ell \in \{0, \ldots, K\}$, denote a solution to the constructed SSTP instance with cost z. Due to the previous observation concerning the high costs of the edges in the second stage we can set $\tilde{E}^0 := \hat{E}^0 \cup \bigcup_{k \in \mathcal{K}} \hat{E}^k$ and $\tilde{E}^1 := \ldots := \tilde{E}^k := \emptyset$ and obtain a (new) solution with value at most z. Moreover, this solution connects all terminal sets in the first stage and hence, $\tilde{E}^0$ is a feasible solution for the SFP instance with the same (or better) value.

Now consider a solution $\tilde{E} \subseteq E$ to the SFP instance with value z. Since $\tilde{E}$ connects all terminal sets the solution $\hat{E}^0 := \tilde{E}, \hat{E}^1 := \ldots := \hat{E}^K := \emptyset$ with value z is valid for the SSTP instance.

To conclude the proof notice that the constructed instance for the stochastic Steiner tree problem is polynomial in the size of the SFP instance. □

Notice that the preceding reduction does not modify the input graph of the SFP; hence, Theorem 5.5 and the result by Bateni, Hajiaghayi, and Marx [13] and Gassner [71] implies the following result.

Corollary 5.6. *The stochastic Steiner tree problem (SSTP) is* NP*-hard on graphs with any fixed treewidth* ≥ 3.

The reduction from the SFP to the SSTP depends on the property that first-stage solutions may be disconnected. Hence, the arguments are not transferable to the rooted SSTP.

Open problem 5.5. Is the rooted stochastic Steiner tree problem (rSSTP) on graphs with treewidth ≥ 3 NP-hard?

Deterministic STP. Concerning treewidth-based algorithms for the STP the strongest result is due to Korach and Solel [114] who described an algorithm with running time $O(tw^{4tw} \cdot |V|)$. More recent publications, in particular those dealing with polynomial approximation schemes (see next paragraph) where the STP on bounded-treewidth graphs arises as a subproblem, mostly propose own, yet weaker, results.

For the unweighted STP, i.e., the objective is to minimize the number of edges of the Steiner tree, Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, and Wojtaszczyk [53] gave a Monte Carlo algorithm for the decision problem with a one-sided error—false negatives occur with probability of at most $1/2$—requiring only $O(3^{tw}|V|^{O(1)})$ time; however, its derandomization is considered an open problem.

Recently, the STP and related problems for graphs with bounded treewidth have received more attention due to their applicability to approximate network optimization problems in planar graphs: In multiple papers by Bateni, Chekuri, Ene, Hajiaghayi, Korula, and Marx [12], Chekuri, Ene, and Korula [35], Bateni, Hajiaghayi, and Marx [11], Borradaile, Kenyon-Mathieu, and Klein [28], Borradaile, Klein, and Mathieu [29], and Bateni, Hajiaghayi, and Marx [13], polynomial time approximation schemes were proposed which use dynamic programming on bounded-treewidth graphs as a subroutine. Hence, the development of faster algorithms for the problem on bounded-treewidth graphs directly leads to faster PTASes for the corresponding problem on planar graphs.

For the STP, the approximation scheme of [29] uses an algorithm for solving the problem on graphs with bounded carving-width (a relative of treewidth) as a black box. Chekuri et al. [35] (later merged into [12]) gave an algorithm for the prize-collecting Steiner tree problem with running time $O(B_\ell^3 \cdot s_\ell \cdot |V|)$, where $\ell := tw + 1$, B_ℓ is the number of partitions of a set with ℓ elements (the ℓth Bell number), and s_ℓ is the number of subgraphs of a ℓ-vertex graph. Since $s_\ell = O(2^{(\ell^2)})$, this leads to a running time of $O(2^{(tw^2)} \cdot B_{tw+1}^3 \cdot |V|)$. This algorithm then allows PTASes for prize-collecting Steiner tree and forest problems. Bateni et al. [11] (also later merged into [12]) described PTASes for prize-collecting network design problems on planar graphs by using a similar approach. They investigated the prize-collecting STP (the solution is a tree), prize-collecting TSP (the solution is a cycle), and the prize-collecting stroll (the solution is a path). To this end they described a $(1 + \varepsilon)$-approximation for the prize-collecting STP problem with a running time of order $O(tw^{tw} \cdot 2^{tw} \cdot |V|)$ that can be adapted to solve the other two considered problems as well.

Furthermore, Polzin and Daneshmand [141] introduced an algorithm with running time $O(2^{3b} \cdot |V|)$ where b is the size of a "border" obtained in the algorithm and a parameter similar to pathwidth. Yet note that even for simple trees—with natural treewidth 1—the pathwidth is unbounded.

5.4 Treewidth-based algorithm for the STP

In the following, we describe the ideas of a new treewidth-based parameterized algorithm for the deterministic STP. We expand this approach for solving the SSTP (Section 5.5) and the rSSTP (Section 5.6).

Our algorithm for the deterministic STP runs in time $O(B_{tw+2}^2 \cdot tw \cdot |V|)$ on a graph with vertex set V and treewidth tw. The ℓth *Bell number* B_ℓ is the number of partitions of a set with ℓ elements and can be recursively defined as $B_0 = 1$, $B_{\ell+1} = \sum_{i=0}^{\ell} \binom{\ell}{i} B_i$, for $\ell \geq 0$, cf. Riordan [151]. It holds $B_\ell < \ell! < \ell^\ell < 2^{(\ell^2)}$, for $\ell \geq 3$, and $B_\ell > 2^\ell$, for

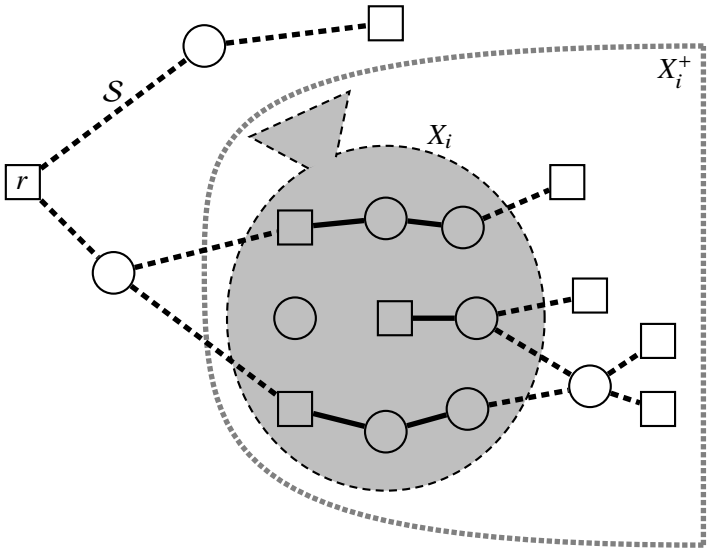


Figure 5.2: A Steiner tree $\mathcal{S}$ decomposes into a forest when restricted to the vertices of bag X_i. The bag X_i is indicated by the grey shape, terminals are drawn as rectangles and Steiner vertices as circles. For simplicity, we omit edge costs as well as all edges not in $\mathcal{S}$.

$\ell \geq 5$, and Berend and Tassa [16] showed that $B_\ell < (0.792\ell/\ln(\ell+1))^\ell$. This algorithm is hence linear for graphs with fixed treewidth and requires $O(|V|^3 \log|V|/\log\log|V|)$ time for $tw \in O(\log|V|/\log\log|V|)$.

Our algorithm follows the classical bottom-up approach for algorithms based on tree decompositions: Starting from the leaves of a nice tree decomposition $(\mathcal{T} = (I, F), \mathcal{X})$ we enumerate a sufficient number of sub-solutions per tree node $i \in I$, using only the information previously computed for the children of i. Such information is stored in a table tab_i, for each node $i \in I$. The final optimum solution of the original problem can then be read from the table tab_r of $\mathcal{T}$s root node r.

Overall, given any tree decomposition, we can easily transform it into a nice tree decomposition where we pick the root r such that its bag X_r contains at least one terminal vertex. While the latter property is not necessary, it allows us to give a simpler description of our algorithm.

Since the tree traversal requires only $O(|V|)$ time, the algorithms time complexity mainly depends on two aspects: The amount of information to be stored per node, and the necessary effort to establish the sub-solutions at a node based on its childrens data. We start by concentrating on the first question and describe how to represent the necessary solutions efficiently by using a coloring-based scheme. Afterwards, we describe how to efficiently combine our coloring with the bottom-up traversal to solve the STP.

5.4.1 Representing sub-solutions

The general idea of using the rooted tree decomposition is the following: Let i be any node in $\mathcal{T}$ with the corresponding bag X_i. We define X_i^+ to be the set of all vertices that are in descendent bags of X_i, with X_i being a descendent of itself. Then, let G_i (G_i^+) describe the subgraph of G induced by the vertices X_i (X_i^+, respectively) and let T_i (T_i^+) be the set of terminals in X_i (X_i^+, respectively).

When we consider any node $i \in I$, we observe, based on the third property of a tree decomposition, that no vertex of $X_i^+ \setminus X_i$ will appear in any other bag than the ones descending from node i. For our bottom-up approach this means that these vertices are not considered in other parts of the algorithm and will never be considered again. Hence, the sub-solutions at node i have to ensure that all terminals $T_i^+ \setminus T_i$ are properly connected with other vertices. Consider any Steiner tree $\mathcal{S}$ in G. The subgraph of $\mathcal{S}$ in G_i^+ then forms a forest, with the property that any terminal $T_i^+ \setminus T_i$ is connected to some vertex in X_i, cf. Figure 5.2.

Our table tab_i stores multiple rows, each row representing a different solution: the rows of tab_i are indexed by solution patterns and the columns are indexed by the vertices of X_i. Thereby, each row of tab_i indicates the corresponding solution pattern. Observe that we do not have to consider all possible subgraphs of a bag X_i but can use the fact that a forest in G_i contains at most $|X_i| - 1$ edges. It remains how to describe these forests uniquely and compactly and allow for fast merging operations within the dynamic programming.

We show that it (roughly) suffices to consider all possible partitions of the at most $tw+1$ many vertices in X_i by assigning colors to them. Each color then indicates the set of vertices that lie in a connected component (a tree, in fact) in G_i^+. To obtain such a description scheme, we first consider some arbitrary but fixed total numbering $\Phi\colon V \xrightarrow{1:1} \{1,\ldots,|V|\}$ of all vertices of the given graph. Based thereon, we assign—locally for each bag X_i—the unique secondary index $\varphi_i\colon X_i \xrightarrow{1:1} \{1,\ldots,|X_i|\}$ which satisfies $\varphi_i(v) < \varphi_i(w) \Leftrightarrow \Phi(v) < \Phi(w)$ for all $v, w \in X_i$.

We now introduce a coloring function $\gamma_j\colon X_i \to \{0,\ldots,|X_i|\}$ whereby any vertex $v \in X_i$ may only be colored by a color at most as large as its local index, i.e., $\gamma_j(v) \leq \varphi_i(v)$. We use the coloring to encode the following information: all vertices of color 0 are not contained in the represented sub-solution and all vertices with a common color > 0 are already connected in the graph G_i^+. Note that these connections do not have to exist in G_i (cf. Figure 5.2). Finally, in order to be a feasible coloring, we require all terminals T_i in X_i to be colored > 0.

Hence, the color of a connected component C of the sub-solution is exactly the smallest secondary index of all vertices contained in C. We observe that a vertex v with $\varphi_i(v) = z$ has $z+1$ possible colors. Moreover, the intersection of any tree $\mathcal{S}$ of G with G_i^+ provides a natural partition of X_i. For each connected component C of $\mathcal{S} \cap G_i^+$, $C \cap X_i$ gives a partition set. The last partition set is formed by the vertices $X_i \setminus \mathcal{S}$ not contained in $\mathcal{S}$. To represent this special set, we (conceptually) add an additional "ghost" element to X_i with secondary index 0. Thus, the solution patterns for X_i are given by all $B_{|X_i|+1}$ partitions of the set X_i plus the ghost element.

For each possible partition, table tab_i stores a row with the unique corresponding coloring, i.e., a color index for each vertex of X_i. Additionally, we will store a solution value $val(\gamma)$ for each coloring γ, to be set by the subsequent algorithm. Hence, the size of any table tab_i can be bounded by $O(B_{tw+2} \cdot tw)$. It is crucial that all the rows of tab_i are held in a canonical order, to allow efficient look-up operations. The next section will discuss the efficient enumeration and look-up strategy for the table. This will then be used in the subsequent algorithm description.

5.4.2 Enumeration and look-up of table rows

Enumeration. We start with showing how to enumerate all possible rows (i.e., colorings) in $O(tw)$ time per coloring. Let X_i be the bag under consideration, $w := |X_i|$, and $v_1, ..., v_w$ the nodes of X_i in increasing order of their secondary index. We have to generate B_{w+1} colorings.

Assume we would want to enumerate all numbers between 0 and 999 in base-10 representation. We would start with "000" and for each next number, we would increment the least significant, right-most, digit, until we would have to increment on the digit "9". In this case we set the digit to "0" and increment the second-right-most digit, etc. In the worst case, one incrementation step looks at each digit once.

Our enumeration scheme is similar to this traditional counting, but when incrementing a "digit"—in our setting changing the color of a vertex—not all higher numbers are valid. The first row of the table tab_i colors all its vertices with color 0. It remains to describe how to obtain the coloring γ_j of row j from the coloring γ_{j-1} of row $j-1$. Initially, set $\gamma_j := \gamma_{j-1}$.

Let $\tau := w$. We increment the color of vertex v_τ unless it is already colored with its highest feasible color, i.e., its own secondary index $\varphi_i(v_\tau)$. We hence distinguish between two cases: If $\gamma_j(v_\tau) = \varphi_i(v_\tau)$, we set $\gamma_j(v_\tau) := 0$ and continue the incrementation with "the next digit" $\tau := \tau - 1$. Otherwise we have to identify to which color we can increment $\gamma_j(v_\tau)$. Observe that we may only use a color c if the vertex with secondary index c is itself colored with c. Therefore, we scan γ_j from vertex $v_{\gamma_j(v_\tau)+1}$ (the potentially next color) with increasing indices until we find the first entry $> \gamma_j(v_\tau)$ or arrive at index $\varphi_i(v_\tau)$ itself. This index is then the next feasible color for v_τ. We observe that this scan only considers vertices with indices less than τ. Therefore, each vertex is looked at at most once when constructing the coloring of row j.

Due to this enumeration scheme, it is clear that the colorings are sorted increasingly, when interpreting the colors of the vertices as digits of a number in base-w representation, where v_w is the right-most digit. Furthermore, consider the first B_w of those colorings (where vertex v_1 is colored 0): When decrementing each non-0 color by 1 and deleting the first color we obtain exactly the colorings in the exact same order one would obtain when enumerating colorings for a set of size $w-1$.

Look-up. In the algorithm, we will often have to find the row index j for some feasible given coloring γ, in order to obtain the solution value stored for this rows sub-solution. Assume for now that γ is a coloring for some set X_i with $|X_i| = tw + 1$. Let $v_1, ..., v_{tw+1}$ be the vertices of X_i in increasing order of their secondary index.

We want to support this operation with the best-possible running time of $O(tw)$. Since the tables at the decomposition trees nodes have the size $O(B_{tw+2} \cdot tw)$, we can allow for a look-up datastructure of size $O(B_{tw+2} \cdot tw)$. In fact, we will also be able to construct it in time $O(B_{tw+2} \cdot tw)$.

Conceptually, start with a search tree D with $tw + 1$ levels. The root node constitutes on level 1. Each node, on some level a, holds an array of size at most $a + 1$. For inner nodes, the entries of these arrays point to corresponding tree nodes of the next level, i.e., they form the search tree. In the leaves, the array entries are row indices. In order to find the row index for γ we traverse D from its root downwards: Assume we are at some node α on level a, we use the color $\gamma(v_a)$ as the index in the array of α to select the next node

(or, if α is a leaf, read the row index). While this datastructure hence allows a look-up time of $O(tw)$, it would require more space than we want to allow.

During a look-up, assume we are at a node on some level a and pick the edge corresponding to the vertex v_a having a color distinct from $\varphi(v_a)$ (possibly 0)—we may say v_a has a *foreign color*. This rules out that any of the following vertices will ever be colored a. Hence it suffices to consider the following shrunken structure $\mathcal{D}$ with corresponding look-up operation: For any node α on level a, let $\mathrm{Z}(\alpha)$ denote the number of foreign colored vertices with index smaller than a. Observe that this number is trivial to establish when traversing the datastructure from the root to α. Then, the array at α only holds $a + 1 - \mathrm{Z}(\alpha)$ entries. As $\mathcal{D}$ now encodes exactly only all feasible colorings, it follows that its size is bounded by $O(B_{tw+2} \cdot tw)$.

Now, when using $\mathcal{D}$ for a look-up, we use an auxiliary array A with one entry per possible color and initialize $A[0] := 0$. Furthermore, we use a counter z—initialized to zero—which counts the number of encountered foreign colored vertices during the traversal. Assume we are at some node α on level a and want to pick the edge corresponding to the vertex v_a having a color c. If $a = c$, i.e. v_a does not have a foreign color, we set $A[c] := a + 1 - z$; otherwise color a is unused and $A[a]$ is irrelevant. In any case, we use $A[c]$ to identify the index in the shrunken child array at α in $\mathcal{D}$, to know how to continue the traversal.

Recall that the enumeration scheme for a w-element set also includes the rows (correctly ordered) for any smaller set—one only has to shift the colors by the size difference. Hence, we can use this datastructure to look up the row indices for bags smaller than $tw + 1$ as well. We summarize:

Lemma 5.7. *Given some coloring γ, we find the corresponding row index in $O(tw)$ time, using a static global look-up datastructure $\mathcal{D}$ of size $O(B_{tw+2} \cdot tw)$. As a preprocessing, $\mathcal{D}$ can be constructed in $O(B_{tw+2} \cdot tw)$ time.*

5.4.3 Processing the decomposition tree

Having our coloring concept at hand, we can now describe how to ensure its properties when computing the actual sub-solution tables in a bottom-up fashion. Our recursion can be described by distinguishing between the different types of nodes of $\mathcal{T}$. Recall that for each row, representing some coloring γ, we store the cost $val(\gamma)$ of the represented sub-solution.

Leaf node. Let $i \in I$ be a leaf, and hence a base case for our algorithm. The table tab_i contains only two rows corresponding to the two possible colors 0 and 1, respectively, for the only vertex $v \in X_i$. If $v \in T$ but is colored 0, the sub-solutions cost is $+\infty$; in the other case the cost is 0.

Introduce node. Let $i \in I$ be an introduce node, and $j \in I$ its only child. We have $X_j \subset X_i$, $|X_i| = |X_j| + 1$, and let v be the additional vertex, cf. Figure 5.3.

As a preprocessing, we create tab_i and set the value entries for each row to $+\infty$. Then, we iterate over each row in tab_j, in order to generate multiple solutions for tab_i; since we sometimes generate the same coloring more than once we store and update the best found solution value for each row in tab_i. Let γ_j be the currently considered coloring from tab_j. First, we adapt this coloring to satisfy the coloring scheme of i, instead of j: By the fact that both secondary indices stem from a common primary index Φ all colors $\geq \varphi_i(v)$ have to be

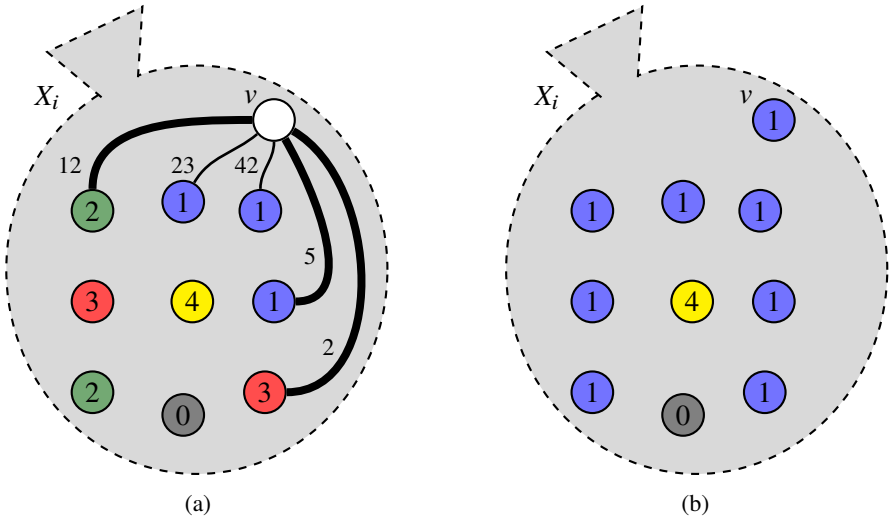


Figure 5.3: (a) An introduce node i with the additional vertex v. The numbers inside the vertices describe the color partitions, i.e., vertices with the same number are already connected in $G_i^+ \setminus \{v\}$ in this solution. Only edges incident to v are shown. (b) shows the resulting coloring when the algorithm picks the thick edges to connect the vertex sets with colors 1, 2, and 3 via v.

increased by one. This modification takes $O(tw)$ time giving us an intermediate coloring γ_i' where v is yet uncolored.

We now have to consider not only all possible colorings of v, but also all possibilities of joining several color partitions in γ_j via the new vertex v. First consider the solutions where v does not essentially change the given partitions, i.e., v is colored with color 0 or $\varphi_i(v)$. Then, the obtained coloring has the same solution value as γ_j, or $+\infty$ if v is a 0-colored terminal vertex. We write these solutions into tab_i, identifying the correct row via our look-up datastructure in $O(tw)$ time.

Now, we enumerate all possible—$O(2^{tw})$ many—non-empty sets of incident edges for connecting the new vertex v to one or more existing color partitions. For each such edge set, we generate a solution γ_i from γ_i' by coloring the newly connected vertex sets with the smallest contained secondary index. We compute the cost $val(\gamma_i)$ for this solution by summing the known cost $val(\gamma_j)$ and the costs of the new edges inserted between the connected color partitions and v.

Thereby, two cases are invalid and penalized with cost $+\infty$. First, a solution containing a 0-colored vertex which is now connected by a new edge is not allowed—the related feasible solution where this very vertex is in its own partition and colored with its local index will be considered anyway. Second, new edges inducing a cycle, i.e., in case at least two edges connect the same color partition, are also prohibited.

Again, we find the row corresponding to this coloring in tab_i in $O(tw)$ time and update $val(\gamma_i)$ accordingly. Overall, we generate $O(2^{tw})$ solutions from each of the $O(B_{tw+1})$ solutions in the child node. Each solutions takes $O(tw)$ time to process.

Hence, given a correct solution table for its child node we compute a correct solution table for an introduce node in $O(B_{tw+1} \cdot 2^{tw} \cdot tw)$ time.

Forget node. Let $i \in I$ be a forget node, and $j \in I$ its only child. We have $X_i \subset X_j$, $|X_i| = |X_j| - 1$, and let v be the discarded vertex.

As a preprocessing, we generate all rows of tab_i and set each solution cost to $+\infty$. We then look at the rows of tab_j one by one; let γ_j be the corresponding coloring, and $c := \gamma_j(v)$. We say γ_j *induces* a coloring γ_i of the vertices X_i, by simply dropping the vertex v and shifting the color index by -1 for all colors $> \varphi_j(v)$; the vertices colored with color $\varphi_j(v)$ in γ_j obtain the color matching the smallest secondary index $\varphi_i(.)$ among themselves.

The case $c = 0$ is trivial because the induced coloring gets the same cost as the solution in tab_j. If $c > 0$ but there is no other vertex with color c, we cannot easily remove this vertex from the solution, as it represents a component. In general, this component contains terminals which have to be connected to the final Steiner tree $\mathcal{S}$. Hence we cannot use this sub-solution to improve the solution value of the induced coloring of X_i. Otherwise, we can safely drop the vertex and set $val(\gamma_i) := val(\gamma_j)$ if the current value of $val(\gamma_i)$ is not already smaller. If the computed cost of γ_i is smaller than the current $val(\gamma_i)$ entry for this coloring in tab_i we update $val(\gamma_i)$ accordingly.

To summarize, we compute a correct solution table for a forget node in $O(B_{tw+2} \cdot tw)$ time.

Join node. Let $i \in I$ be a join node, and $j, j' \in I$ its two children. We have $X_j = X_{j'} = X_i$.

Again, we first construct all rows of tab_i and set the solution values to $+\infty$. Then we consider all possible combinations of solutions from X_j and $X_{j'}$. Let γ_j and $\gamma_{j'}$ be colorings (rows) of tab_j and $tab_{j'}$, respectively. We want to construct a merged solution γ_i that resembles the combined connectivities of both solutions, i.e., two vertices $v_s, v_t \in X_i$ should be in the same color partition if and only if there is a vertex sequence $\langle v_s := v_1, v_2, \ldots, v_\beta := v_t \rangle$ in X_i such that, for all $1 \leq \alpha < \beta$, the vertices $v_\alpha, v_{\alpha+1}$ have the same color in either γ_j or $\gamma_{j'}$.

Note that, a priori, such a merge might lead to cycles in the solution: Assume two vertices v_1, v_2 are colored with identical color c_j in γ_j and furthermore, they have a common color $c_{j'}$ in $\gamma_{j'}$. Hence the vertices are connected in both sub-solutions, but the connection paths do not need to coincide. Even if the paths do coincide, we would have to identify them to not count their cost twice for the combined solution. Hence, we only want to combine two solutions if each pair of vertices is connected in at most one sub-solution. Then, the value of the combined solution can be given as $val(\gamma_i) := val(\gamma_j) + val(\gamma_{j'})$, which we can store into tab_i.

We are able to perform the merge operation, including the check of validity, in linear time $O(tw)$: Consider a helper array $recol\colon \{1, \ldots, |X_i|\} \to \{1, \ldots, |X_i|\}$ and construct a graph C with a vertex c_r per used color r. Then, for each vertex $v \in X_i$, add an edge between the two colors $c_{\gamma_j(v)}$ and $c_{\gamma_{j'}(v)}$ of v in X_j and $X_{j'}$, respectively. Clearly, the graph has only $O(tw)$ vertices and edges, cf. Figure 5.4 (a) and (b).

We observe that we can merge two sub-solutions iff the following three properties are satisfied:

(i) vertex c_0 representing color 0 represents an isolated connected component in C, i.e., no vertex is colored 0 in one and > 0 in the other sub-solution,

(ii) no vertex in $C \backslash \{c_0\}$ has more than one incident selfloop, i.e., any pair of vertices has a common color > 0 in at most one of the two colorings $\gamma_j, \gamma_{j'}$, and

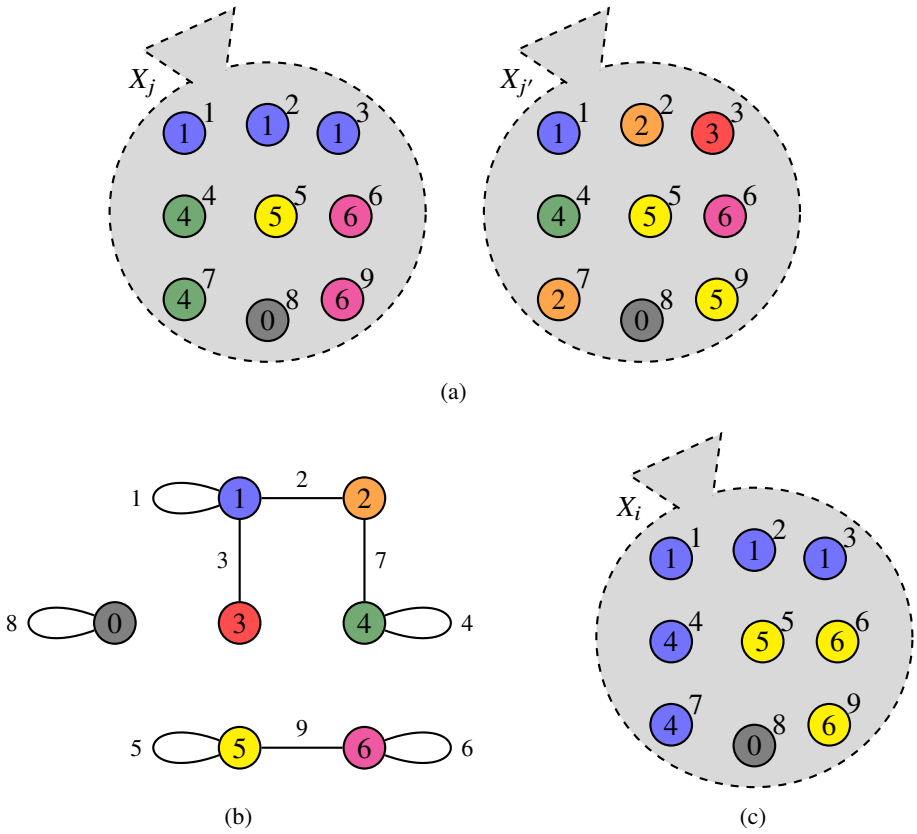


Figure 5.4: (a) The child bags X_j, $X_{j'}$ of a join node X_i with two feasible colorings. The numbers inside the vertices describe the color partitions, the numbers to the top right of each vertex the local indices. (b) depicts the constructed auxiliary graph C used for the DFS sub-algorithm with edge-labels indicating the corresponding vertices. (c) shows the resulting coloring in X_i.

(iii) C is acyclic (disregarding selfloops), i.e., any pair of vertices is connected in at most one sub-solution.

For merging and recoloring we remove vertex c_0 together with its incident self-loops and mark all other vertices in C as unvisited. Then, for increasing $r \in \{1, \ldots, |X_i|\}$, we start a depth-first search (DFS) in C at any unvisited c_r and set $recol(c_{r'}) := r$ for any vertex $c_{r'}$ visited in this DFS run. Hence, in the end, $recol$ gives the new color for any color in either γ_j or $\gamma_{j'}$. Whenever a DFS run revisits an already visited vertex, we identified a cycle (including the special case of multiple edges), and the merge operation should be aborted; thereby, self-loops are simply ignored. If no cycles are detected, we can finally again consider each $v \in X_i$ and set $\gamma_i(v) := 0$ if $\gamma_j(v) = \gamma_{j'}(v) = 0$, and $\gamma_i(v) := recol(\gamma_j(v)) = recol(\gamma_{j'}(v))$ otherwise. Figure 5.4 shows the outcome of a feasible combination step in a join node.

We deduce that a correct solution table for a join node can be computed in $O(B^2_{tw+2} \cdot tw)$ time.

Extracting the solution at the root node. From the described construction process it is clear that each solution of a bag X_i describes the minimum cost of a forest where all

terminals from X_i^+ are connected to some vertex of X_i. Also recall that it can be safely assumed that at least one terminal is contained in the root bag X_r of $\mathcal{T}$. Hence the optimum solution value for the whole graph can be found in the root bag X_r of $\mathcal{T}$, identifying a cheapest solution where all vertices with color $\neq 0$ are contained in the same connected component (i.e., have the same color).

5.4.4 Analysis and remarks

In the following, we summarize the main results concerning the algorithms running time and its correctness. For detailed proofs we refer to Chimani, Mutzel, and Zey [41].

Lemma 5.8. *The above algorithm requires $O(B_{tw+2}^2 \cdot tw \cdot |V|)$ time and space.*

Proof. A table tab_i at some tree node i stores $O(B_{tw+2})$ rows and requires $O(B_{tw+2} \cdot tw)$ storage. The look-up datastructure has size $O(B_{tw+2} \cdot tw)$.

During the bottom-up traversal of $\mathcal{T}$ we consider all possible row combinations for two tables in the case of the join node which requires $O(B_{tw+2}^2 \cdot tw)$ time; this dominates the other cases. Due to the linear size of $\mathcal{T}$ the overall running time follows. □

Lemma 5.9. *The above algorithm correctly computes an optimum solution to the Steiner tree problem.*

The correctness can be shown by an inductive proof on the decomposition tree, cf. [41].

Theorem 5.10. *Given a graph with vertex set V and a tree decomposition with treewidth tw, the Steiner tree problem can be solved to optimality in $O(B_{tw+2}^2 \cdot tw \cdot |V|)$ time.*

The previously described algorithm can be adopted to solve other network design problems on treewidth-bounded graphs. [41] showed that the prize-collecting STP and the k-cardinality tree problem can be solved by similar FPT algorithms in $O(B_{tw+2}^2 \cdot tw \cdot |V|)$ and $O(B_{tw+2}^2 \cdot (tw + k^2) \cdot |V|)$ time, respectively.

We close the discussion on the deterministic STP with two remarks.

Remark 5.11. The described FPT algorithm works for negative edge weights, too. In case the given graph G has only positive edge weights, we do not need to actively identify cycles or multiedges: the solutions objective value will be greater than the alternative cycle and multiedge-free combination, which will also be considered.

Remark 5.12. We can also run the algorithm on a tree decomposition where the root node does not contain any terminal. In this case, whenever we process a tree node i with $T \subseteq X_i^+$ we check for the best solution where all vertices with color > 0 belong to the same color partition and store a reference to it. After processing the root node this reference gives the optimum solution. Note that this algorithm has the same time complexity.

5.5 Treewidth-based algorithm for the SSTP

The parameterized treewidth-based algorithm for the deterministic STP can be adopted for solving the rooted and unrooted version of the stochastic Steiner tree problem. The main

difficulty consists of dealing with the set of scenarios; all other ideas like the coloring scheme for representing sub-solutions work similarly.

We start with the algorithm for the SSTP and afterwards, we summarize the necessary modifications for the rSSTP. The resulting algorithms are parameterized by the combination of treewidth and number of scenarios. For the best of our knowledge, this is the first approach placing the SSTP in the complexity class FPT with respect to the parameter treewidth.

To simplify the description we multiply the probability of a scenario into the second-stage edge costs, i.e., the cost of edge $e \in E$ in scenario $k \in \mathcal{K}$ is $p^k c_e^k$.

5.5.1 Coloring

Representing sub-solutions works similarly to the deterministic case by using a coloring scheme. But instead of one coloring function we now need K coloring functions, one for each scenario. Let $E^0, E^1, \ldots, E^K$ be the edge set of an optimum stochastic Steiner tree. When considering the vertices X_i of a bag $i \in I$ and a scenario $k \in \mathcal{K}$ the edges $E^0 \cup E^k$ induce a forest in X_i (and X_i^+). Thereby, all terminals in $T^k \cap X_i^+$ are connected to vertices in X_i. The main idea is to encode a solution by K colorings representing the connected components induced by the edge sets $E^0 \cup E^k$, one for each scenario $k \in \mathcal{K}$. We will see that it is not necessary to explicitly store the first stage solution.

To make the description clearer we will speak of a coloring as the union of K *sub-colorings*. Hence, a sub-coloring is always related to a certain scenario while a *coloring* represents a global solution.

Notice that the sub-colorings are independent of each other: coloring a vertex v in one sub-coloring with color c does not influence the set of feasible colors for v in any other sub-coloring.

We again use the following two types of numberings: the total vertex numbering $\Phi\colon V \xrightarrow{1:1} \{1, \ldots, |V|\}$ and the secondary numbering $\varphi_i\colon X_i \xrightarrow{1:1} \{1, \ldots, |X_i|\}$ being assigned locally to each bag $X_i, i \in I$, with $\varphi_i(v) < \varphi_i(w) \Leftrightarrow \Phi(v) < \Phi(w)$ for all $v, w \in X_i$. A coloring/solution γ_i in bag $X_i, i \in I$, is a vector of K sub-colorings $\gamma_i := (\gamma_i^1, \ldots, \gamma_i^K)$ with $\gamma_i^k\colon X_i \to \{0, 1, \ldots, |X_i|\}, \forall k \in \mathcal{K}, \forall i \in I$. Again, a vertex can only be colored by a color at most as large as its local index, i.e., $\gamma_i^k(v) \leq \varphi_i(v), \forall v \in X_i$.

The interpretation of > 0-colored vertices is identical to the deterministic case: All vertices sharing the same color are already connected by $E^0 \cup E^k$ in G_i^+. On the other hand, assigning a vertex color 0 has a slightly different meaning: (a) either a 0-colored vertex is not used in a solution or (b) it is *irrelevant* for a sub-solution and the corresponding scenario.

This latter property is described and motivated as follows. Consider an edge e being used in the optimum solution as a first-stage edge but $E^0 \backslash \{e\} \cup E^k$ still connects T^k for a scenario $k \in \mathcal{K}$. Such an edge is called *irrelevant* for this scenario. Moreover, edge e might even form (be contained in, respectively) an isolated connected component in $E^0 \cup E^k$. In the deterministic STP algorithm all colored vertices (color > 0) are used and have to be connected. This requirement is not valid anymore since the first stage might be disconnected or some of its parts are irrelevant for certain scenarios. Hence, although being connected by E^0 we allow vertices being assigned color 0 (and named *irrelevant*) if they do not need to be connected to the remaining part of the solution. But of course, we still require all vertices in T^k being colored > 0 in a sub-coloring of scenario $k \in \mathcal{K}$.

5.5.2 Solutions, enumeration, tables, and look-up

We again denote the table corresponding to bag $X_i, i \in I$, by tab_i. The number of possible solutions is defined by all possible combinations of the sub-colorings, i.e., the cartesian product of the K sub-colorings. For a bag X_i with $w := |X_i|$ the number of possible partitions is B_{w+1}. Hence, combining the scenarios leads to B_{w+1}^K possible solutions, which is the number of rows of tab_i. Each solution itself stores K colors per vertex leading to a table size of $B_{w+1}^K \cdot K \cdot w$. Overall, the size of a table can be bounded by $O(B_{tw+2}^K \cdot K \cdot tw)$.

The rows/solutions of a bag $i \in \mathcal{I}$ can be enumerated by using K nested enumerations for each sub-coloring. Each sub-enumeration uses the very same procedure from the deterministic algorithm. Hence, coloring γ_j from row j can be obtained from the preceding coloring γ_{j-1} of row $j-1$ in linear time w.r.t. the size of the instance and the treewidth, i.e., $O(K \cdot tw)$.

Looking up the index of a given coloring also works similarly by using the same techniques as in the deterministic setting. Conceptually, the look-up datastructure is the concatenation of K nested single-scenario search trees. Hence, we use an expanded search tree with $K \cdot (tw + 1)$ levels. Thereby, every $tw + 1$ levels represent a scenario, a search tree of size $O(B_{tw+2} \cdot tw)$. This leads to an overall search tree of size $O(B_{tw+2}^K \cdot tw^K)$ which can be constructed in a preprocessing step in $O(B_{tw+2}^K \cdot tw^K)$ time.

5.5.3 Processing the decomposition tree

The optimum solution can be computed by using dynamic programming in a bottom-up traversal of the decomposition tree. The description breaks down to the case distinction of the types of bags in the nice tree decomposition.

In all cases we denote the currently considered node by $i \in I$ with vertex set X_i. As a preprocessing we always generate the corresponding table tab_i and set each solution cost to $+\infty$.

Leaf node. The base case is a leaf node with $|X_i| = 1$; let $X_i = \{v\}$. For each scenario $k \in \mathcal{K}$, v can be colored 0 or 1. Hence, the number of rows of tab_i is 2^K. The cost of a solution γ_i is $+\infty$ if one terminal is colored 0, i.e., $\exists k \in \mathcal{K}\colon v \in T^k$ and $\gamma_i^k(v) = 0$, and 0, otherwise.

Introduce node. Let $i \in I$ be an introduce node, and $j \in I$ its only child. We have $X_j \subset X_i$, $|X_i| = |X_j| + 1$, and let v be the additional vertex.

As in the deterministic case, we iterate over the rows in tab_j in order to generate multiple solutions for tab_i. Let γ_j be the currently considered coloring from tab_j. Again, we first adapt this coloring to satisfy the coloring scheme of i by increasing all colors $\geq \varphi_i(v)$ by one for each sub-coloring. This modification can be done in $O(K \cdot tw)$ time with a resulting intermediate coloring γ_i' with an uncolored vertex v in each sub-coloring.

Next, all possibilities of connecting/joining color partitions by first- and second-stage edges to/via the new vertex v are considered. For this purpose we simply enumerate all subsets of incident edges of v as first-stage edges. For each such subset we enumerate—and combine—all subsets for all scenarios as well. There are at most $O(2^{tw})$ subsets leading to $O(2^{(K+1)\cdot tw})$ combinations overall. Invalid solutions can be discarded right away,

e.g., solutions with coinciding first- and second-stage edges, cycles, or 0-colored vertices connected by second-stage edges, are not valid.

Then, we combine each intermediate coloring γ_i' with the $O(2^{(K+1)\cdot tw})$ possible edge sets to generate multiple solutions for tab_i. Thereby, all scenarios can be considered independently and the sub-colorings can be updated similarly to the deterministic case. However, we have to deal with the special property of connected, but irrelevant, 0-colored vertices. It is fundamental that a connected vertex can only be colored 0 if it is no terminal and all selected incident edges are first-stage edges. If this property is violated in any sub-coloring the solution cost is $+\infty$. Overall, we have to deal with more cases than in the deterministic setting. We describe the resulting cases (1)–(4) in the following; they are distinguished by the number of selected edges in the first and second stage, independently for each scenario.

Consider a scenario $k \in \mathcal{K}$ with intermediate sub-coloring $\gamma_i'^k$ and let $\hat{E}^0 \cup \hat{E}^k$ be the set of selected first- and second-stage edges.

(1) If $\hat{E}^0 \cup \hat{E}^k = \emptyset$ vertex v can be assigned color 0 or $\varphi_i(v)$. This case is identical to the deterministic algorithm, as is the following. (2) If $\hat{E}^0 = \emptyset, \hat{E}^k \neq \emptyset$ all newly connected vertices get a common color > 0 which is the lowest common index. Of course, any such vertex having color 0 in $\gamma_i'^k$ induces an infeasible solution.

The two remaining cases (3) and (4) are induced by a non-empty first stage. (3) If $\hat{E}^0 \neq \emptyset$ and $\hat{E}^k = \emptyset$ we consider two sub-cases (3a) and (3b), respectively, by declaring v irrelevant, or not. (3a) First, we consider v being irrelevant by assigning color 0—then, all other vertices keep their color. This case is only feasible if at most one adjacent vertex is colored > 0. Otherwise, the irrelevant vertex v would connect several components and then, v has to be colored > 0, which is covered by the following case. (3b) Second, vertex v is not irrelevant and we assign v a color $c > 0$. If all adjacent vertices are colored 0, v has to get color $\varphi_i(v)$. Otherwise, the colors are updated as usual such that v is contained in the partition colored by $c > 0$.

Last but not least, if (4) $\hat{E}^0 \neq \emptyset$ and $\hat{E}^k \neq \emptyset$, v cannot be colored 0. Therefore, the colors can be updated with the known procedure and all 0-colored vertices—irrelevant or not used—are treated identically and can be connected by first-stage edges.

Again, when finding a feasible solution γ_i the solution cost $val(\gamma_i)$ can be calculated by summing the cost of the related coloring in tab_j plus the costs of the newly added edges in the first stage and all scenarios. Moreover, we update the solution value in tab_i, in case the solution value improves, by identifying the corresponding row via the look-up datastructure. As discussed by cases (1)–(4) any edge set $\hat{E}^0 \cup \hat{E}^k$ induces at most two feasible sub-colorings. Processing one new solution, i.e., computing new color indices and finding the correct row in the table, can be done in linear time $O(K \cdot tw)$.

Hence, given a solution table for its child node, we compute a solution table for an introduce node in $O(B^K_{tw+1} \cdot 2^{(K+1)tw} \cdot 2^K \cdot K \cdot tw)$ time.

Forget node. Let $i \in I$ be a forget node, and $j \in I$ its only child. We have $X_i \subset X_j$, $|X_i| = |X_j| - 1$, and let v be the discarded vertex.

In fact, the handling of a forget node is almost identical to the deterministic algorithm and basically repeated K times for each sub-coloring. After generating tab_i we again look at each solution γ_j from tab_j. In case the cost of γ_j is $+\infty$ we skip this solution. Otherwise, we try to drop the discarded vertex v and compute the induced sub-colorings γ_i^k, independently for each scenario.

Consider a scenario $k \in \mathcal{K}$ and let $c := \gamma_j^k(v)$. In case $c = 0$, vertex v can be safely removed; independently of v being irrelevant or unused. If $c > 0$ there are two more cases to distinguish. First, if v is not the only c-colored vertex its connected component is still present and hence, everything is fine. Second, in case v is the last remaining vertex of a color partition it represents a component which, in general, connects some terminals. If this component contains the whole set T^k, i.e., the sub-coloring contains only colors 0 and c, and $T^k \subseteq X_i^+$, this scenarios connectivity requirements are already fulfilled and the removal of v is feasible. Otherwise, the constructed sub-coloring induces several connected components and hence, it is infeasible.

Overall, after removing v from each sub-coloring the induced solution is feasible iff every sub-coloring is feasible. The cost of this solution is taken from the child table. Again, the running time for these steps is linear for each scenario. Therefore, a solution table for a forget node can be computed in $O(B_{tw+2}^K \cdot K \cdot tw)$ time.

Join node. Let $i \in I$ be a join node, and $j, j' \in I$ its two children with $X_j = X_{j'} = X_i$. After constructing tab_i we combine each solution from tab_j with each solution from $tab_{j'}$ to obtain a new solution for tab_i representing the combined connectivities. The procedure for computing new color indices is identical to the deterministic algorithm and directly applied to each scenario independently. Thereby, an infeasible sub-solution implies an infeasible overall solution. Moreover, the new solution cost is calculated by summing both combined solution costs. For details we refer to the description in Section 5.4.3. We conclude that, given a solution table for its child nodes, we are able to compute a solution table for a join node in $O(B_{tw+2}^{2K} \cdot K \cdot tw)$ time.

Extracting the optimum solution. A solution γ_j in a table tab_i, $i \in I$, is globally feasible if and only if each sub-coloring γ_j^k is feasible (solution cost $< +\infty$), each γ_j^k constitutes at most one color partition, and the terminals of all scenarios are contained in G_i^+, i.e., $T^k \subseteq X_i^+$, $\forall k \in \mathcal{K}$. Hence, during the traversal of the decomposition tree we check for these properties and store a reference to the best found feasible solution. In the end, this reference gives the optimum solution.

5.5.4 Analysis

Lemma 5.13. *The above algorithm for the stochastic Steiner tree problem requires* $O(B_{tw+2}^{2K} \cdot K \cdot tw \cdot |V|)$ *time and space.*

Proof. Each table tab_i, $i \in I$, stores $O(B_{tw+2}^K)$ colorings/rows and requires overall $O(B_{tw+2}^K \cdot K \cdot tw)$ storage since each row contains the solution value and a column for each vertex in each scenario. Moreover, the look-up datastructure requires $O(B_{tw+2}^K \cdot tw^K)$ space.

The most time-consuming step during the bottom-up traversal of $\mathcal{T}$ is the join node where all possible combinations of two tables are considered; the required time can be bounded by $O(B_{tw+2}^{2K} \cdot K \cdot tw)$. This bound dominates the time required for introduce, forget, and leaf nodes, as well as all other extra effort. Due to the linear size of the decomposition tree $\mathcal{T}$, we can deduce the overall running time and the required space. □

Before discussing the correctness of the algorithm please notice that the introduce node is the only case where edges are actually added to the solutions. In the forget node a vertex

is discarded and the representations of solutions shrink, and in the join node two solutions are merged.

Lemma 5.14. *The above algorithm correctly computes an optimum solution for the stochastic Steiner tree problem.*

Proof. We argue the correctness by an inductive proof on the decomposition tree. Let $\Gamma_i^{k,c} := \{v \in X_i \mid \gamma_i^k(v) = c\}$ be the vertices colored $c \in \{0, \ldots, |X_i|\}$ in the kth sub-coloring γ_i^k in a bag $i \in \mathcal{I}$, for each scenario $k \in \mathcal{K}$.

Our induction hypothesis (IH) states that, for each processed bag X_i, each coloring $\gamma_i = (\gamma_i^1, \ldots, \gamma_i^K)$ implies a cost-minimal solution among the solutions satisfying the following conditions:

1. $\forall k \in \mathcal{K}$: Each sub-coloring γ_i^k corresponds to a forest F_i^k in G_i^+.

2. $\forall k \in \mathcal{K}$: F_i^k consists of (pairwise disconnected) trees $F_i^{k,c}$, one for each color $c > 0$, with $\Gamma_i^{k,c} \neq \emptyset$, $\Gamma_i^{k,c} \subseteq V(F_i^{k,c})$, and $|\Gamma_i^{k,c'} \cap V(F_i^{k,c})| = 0$ for all $c' \neq c$, i.e., each tree connects only vertices of the same color partition.

3. $\forall k \in \mathcal{K}$: F_i^k contains all terminals of the kth scenario of G_i^+, i.e., $(T^k \cap X_i^+) \subseteq V(F_i^k)$.

If one of the properties does not hold and any sub-solution is infeasible, the overall solution is infeasible and has cost $+\infty$.

As always, the base case is the *leaf node* where the hypothesis obviously holds. Now consider any internal bag $X_i, i \in I$, and assume that the induction hypothesis is true for all descendants of i. By using contradictions we show that IH still holds for bag i.

Forget node. Each solution of a forget bag is induced by one solution from the child bag which is obtained by removing the special vertex, updating the indices of sub-colorings, and without changing its solution value.

Assume there exists a better solution value for coloring γ_i than the one identified by the algorithm. In this case we can add the forget vertex to this solution and change colors according to this solution in each sub-coloring. The resulting coloring $\gamma_{i'}$ is a feasible solution for the child bag with the same value. By using the induction hypothesis this solution would be considered by our algorithm without changing its solution value.

Introduce node. Each solution of an introduce bag is obtained by using a coloring from the child node and coloring the special vertex 0, with its own local index, or connecting it by every possible first- and second-stage edge combination, for each scenario. Again, assume the computed value $val(\gamma_i)$ of coloring γ_i is too high and consider its dominating solution.

By removing the introduced vertex and all its incident edges in each scenario a feasible and minimal solution for the child bag is created. Since we consider this solution (IH), as well as all possible edge sets, this solution would be generated by the algorithm.

Join node. Consider a solution γ_i with an assumably wrong solution value. For each scenario $k \in \mathcal{K}$, we can decompose this solution into two disjoint sub-forests F_j^k and $F_{j'}^k$ such that these two forests induce feasible sub-colorings for the children j and j', respectively. Our tables contain all possible combinations of sub-colorings; hence, the induction hypothesis implies that both solutions are contained in the child bags with the correct solution values. In the algorithm we combine all possible solutions and obtain the correct objective value. □

Finally, the following theorem summarizes the above lemmata.

Theorem 5.15. *Given a graph $G = (V, E)$ and a tree decomposition with treewidth tw, the stochastic Steiner tree problem (SSTP) can be solved to optimality in $O(B_{tw+2}^{2K} \cdot K \cdot tw \cdot |V|)$ time. This places the SSTP into the complexity class* FPT *parameterized by the combination of treewidth and number of scenarios.*

5.5.5 Treewidth-based algorithm for the Steiner forest problem

Our discussions reveal that the Steiner forest problem is closely related to the SSTP: Theorem 5.5 shows that the SFP can be solved by the (unrooted version of the) stochastic Steiner tree problem. Therefore, the result concerning the parameterized complexity can be transferred directly to the SFP. To the best of our knowledge, this is the first result concerning the parameterized complexity of the SFP with respect to treewidth.

Theorem 5.16. *Given a graph $G = (V, E)$ and a tree decomposition with treewidth tw, the Steiner forest problem can be solved to optimality in $O(B_{tw+2}^{2K} \cdot K \cdot tw \cdot |V|)$ time. Hence, the Steiner forest problem belongs to the complexity class* FPT *parameterized by the combination of treewidth and number of terminal sets.*

Proof. Follows directly from Theorem 5.5, the FPT algorithm for the SSTP, and Theorem 5.15. □

5.6 Treewidth-based algorithm for the rooted SSTP

The rooted version of the stochastic Steiner tree problem can be solved with a similar approach. However, the first stage is not unconstrained anymore and we have to ensure that the first stage is empty or a tree containing the root node. In the following we describe the algorithm for the rooted SSTP in a compact way and mainly point out the necessary modifications compared to the SSTP algorithm.

Coloring. We again use K coloring functions $\gamma^1, \ldots, \gamma^K$ for the K scenarios. Additionally, we introduce a coloring function γ^0 which describes the connected components of the first stage.

The interpretation of colors in the first stage is identical to the deterministic setting. A 0-colored vertex is not connected by first-stage edges at all and the set of vertices with a common color > 0 induces a connected component in the first stage.

The sub-colorings $\gamma^1, \ldots, \gamma^K$ represent the same information as in the SSTP algorithm: All vertices sharing the same color are already connected by $E^0 \cup E^k$ and either a 0-colored vertex is not used in a solution or it is irrelevant for the corresponding scenario. Notice that a vertex v can be colored > 0 in γ^0 and 0 in some γ^k if v is irrelevant, and the other way around is possible, too.

Dynamic Programming. To simplify the description we assume that the root node r is contained in the root node of the decomposition tree $\mathcal{T}$. Recall that r is a terminal in every scenario and the first stage needs to be a (possibly empty) tree containing r. Hence, any feasible solution implies a single connected first-stage component.

The number of rows, i.e., solutions, of a table tab_i for bag $X_i, i \in I$, with $w := |X_i|$ is B_{w+1}^{K+1}. Each solution stores a color for each vertex in the first stage and every scenario. Hence, the size of a table can be bounded by $O(B_{tw+2}^{K+1} \cdot K \cdot tw)$.

The index of a coloring can be found by using a similar look-up datastructure. Here, it is expanded by one level representing the first-stage coloring. Overall, the size of the search tree is $O(B_{tw+2}^{K+1} \cdot tw^{K+1})$ and it can be constructed in the same time.

Let $i \in I$ denote the currently considered node in the traversal of $\mathcal{T}$. For each case we first generate the full table tab_i with default costs $+\infty$. Then we consider each solution γ_j in the child bag (introduce and forget node) or each combination of two solutions $\gamma_j, \gamma_{j'}$ from both child bags (join node) in order to generate solutions for the current node i.

Leaf node. The table of a leaf node $i \in I$ simply contains all 2^{K+1} possible 0/1-colorings for the only vertex. The cost of a solution is $+\infty$, if a terminal in any scenario is colored 0, and 0 otherwise.

Introduce node. In the introduce node edges are added and the property of the first-stage tree cannot be violated. Hence, the description of the introduce node is basically identical to the description in the SSTP algorithm. The only difference is the additional coloring for the first stage, but updating γ_i^0 works with the known procedure. Hence, given a solution table for its child node, we compute a solution table for an introduce node in $O(B_{tw+1}^{K+1} \cdot 2^{(K+1)tw} \cdot 2^K \cdot K \cdot tw)$ time.

Forget node. A solution γ_j of tab_j induces exactly one coloring γ_i for the current bag X_i by dropping the forget vertex and updating indices accordingly. In case every sub-coloring remains feasible the solution itself is feasible and the solution cost is unchanged; otherwise the cost is set to $+\infty$. Infeasibility occurs when the forget vertex v is the last representative of a connected component—either in the first-stage coloring γ^0 or a sub-coloring $\gamma^k, k \in \mathcal{K}$. The running time for computing the table of a forget node is $O(B_{tw+2}^{K+1} \cdot K \cdot tw)$.

Join node. A join node combines two solutions from the two child tables tab_j and $tab_{j'}$ to obtain a new solution for tab_i representing the combined connectivities. Thereby, the deterministic algorithm can be used to compute the first-stage coloring γ^0. Each sub-coloring can be computed by using the method from the SSTP algorithm. Therefore, processing a join node takes time $O(B_{tw+2}^{2K+2} \cdot K \cdot tw)$.

Extracting the optimum solution. Feasible solutions can be found in the root node of the tree decomposition since it contains the root r which is a terminal in every scenario. Hence, we search for the cost-minimal solution in the final table of $\mathcal{T}$s root node satisfying the following conditions. First, coloring γ^0 has to induce exactly one connected component containing r or all vertices are assigned color 0, respectively; the latter case represents a solution with an empty first stage. Moreover, each sub-coloring has to comprise one connected component containing all terminals of the root bag. Last but not least, the solution cost has to be $< +\infty$.

Analysis. The size of each table is bounded by $O(B_{tw+2}^{K+1} \cdot K \cdot tw)$ and our look-up datastructure requires $O(B_{tw+2}^{K+1} \cdot tw^{K+1})$ space (same time bound for the construction). The most time-consuming step during the bottom-up traversal is the join node where two tables are combined. Hence, the overall running time can be bounded by $O(B_{tw+2}^{2K+2} \cdot K \cdot tw \cdot |V|)$.

To argue the correctness of the algorithm we observe the following three properties. First, the terminal sets of all scenarios are connected appropriately which can be proven by adopting the proof of Lemma 5.14. Second, the first stage is a tree containing the root node r. This property is only crucial for the forget node because it is the only case with

shrunken connected components. But notice that in this case any disconnected first stage is penalized with cost $+\infty$. Third, the cost of the optimum solution is computed correctly. Again this can be seen by adopting the inductive proof of Lemma 5.14 where each solution can be traced back to the child node (introduce and forget node) or it can be decomposed into two solutions from both children (join node).

Theorem 5.17. *Given a graph $G = (V, E)$ and a tree decomposition with treewidth tw, the rooted stochastic Steiner tree problem (rSSTP) can be solved to optimality in $O(B_{tw+2}^{2K+2} \cdot K \cdot tw \cdot |V|)$ time. This places the rSSTP into the complexity class* FPT *parameterized by the combination of treewidth and number of scenarios.*

Proof. Follows from the preceding discussion and in particular Lemma 5.13, 5.14, and Theorem 5.15. □

Chapter 6

IP formulations

This chapter is dedicated to the IP formulations for the stochastic Steiner tree problems. We recall known flow- and cut-based models in Section 6.1. Then, we adopt orientation properties from the deterministic STP and develop stronger semi-directed models in Section 6.2. For the rooted SSTP we present directed formulations in Section 6.3. Section 6.4 contains the comparison of the strength of the introduced models. Since the Steiner forest problem is a special case of the SSTP we apply similar and further ideas and introduce stronger models for the SFP in Section 6.5.

6.1 Undirected formulations

We start by introducing undirected cut- and flow-based formulations for the unrooted version of the SSTP. Since the rooted version can be formulated by stronger directed formulations it is considered in Section 6.3.

Undirected cut formulation. The following IP is a formulation based on undirected cuts and it was already considered in the literature, e.g., by Gupta, Ravi, and Sinha [87]. It is the classical expansion of formulation (STP_{uc}) for the deterministic STP. Binary decision variables for the first-stage edges are denoted by $x^0_e, \forall e \in E$, and scenario edges of the kth scenario by $x^k_e, \forall e \in E, \forall k \in \mathcal{K}$. The objective is to minimize the expected cost which is the sum of the selected first-stage edges plus the sum of second-stage edges weighted by the scenario probability.

$$(\text{SSTP}_{\text{uc}}) \min \sum_{e\in E} c^0_e x^0_e + \sum_{k\in\mathcal{K}} p^k \sum_{e\in E} c^k_e x^k_e$$

$$\text{s.t. } (x^0 + x^k)(\delta(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V : \emptyset \neq T^k \cap S \neq T^k \tag{6.1}$$

$$x^0 \in \{0,1\}^{|E|} \tag{6.2}$$

$$x^{1\ldots K} \in \{0,1\}^{|E|\cdot K} \tag{6.3}$$

Constraints (6.1) are undirected cuts ensuring the connectivity of each scenario terminal set. Thereby, first-stage and second-stage edges can be used to satisfy a cut.

Undirected flow formulation. Here, we present a similar model to the one introduced by Gupta, Ravi, and Sinha [87]. We modify the model such that we have a flow only in

the second stage. Thereby, the flow can be constructed by using selected first-stage and second-stage edges.

We again use variables x^0 and $x^k, \forall k \in \mathcal{K}$, for modeling the solution edges. Moreover, the bidirection with arc set A is considered and a flow is computed in each scenario $k \in \mathcal{K}$ from a designated root node $r^k \in T^k$ to each terminal. We use variables $f_{ij}^{k,t}$ for each scenario $k \in \mathcal{K}$, arc $(i,j) \in A$, and terminal $t \in T_r^k$ with $T_r^k := T^k \setminus \{r^k\}$. Let $t_r^* := \sum_{k \in \mathcal{K}} |T_r^k|$ denote the number of commodities. The undirected flow model for the SSTP then reads as follows:

$$
\begin{aligned}
(\mathrm{SSTP_{uf}})\ \min \ & \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} c_e^k x_e^k \\
\text{s.t. } & x_e^0 + x_e^k \geq f_{ij}^{k,t}, \\
& x_e^0 + x_e^k \geq f_{ji}^{k,t} \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E, \forall t \in T_r^k \qquad (6.4)
\end{aligned}
$$

$$
f^{k,t}(\delta^-(i)) - f^{k,t}(\delta^+(i)) = \begin{cases} -1, & \text{if } i = r^k \\ 1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \quad \begin{matrix} \forall k \in \mathcal{K}, \forall t \in T_r^k, \\ \forall i \in V \end{matrix} \qquad (6.5)
$$

$$
f \in [0,1]^{|A| \cdot t_r^*} \qquad (6.6)
$$

$$
x^0 \in \{0,1\}^{|E|} \qquad (6.7)
$$

$$
x^{1 \ldots K} \in \{0,1\}^{|E| \cdot K} \qquad (6.8)
$$

In this model there has to be one unit of flow in each scenario from the root to each terminal. This is enforced by the flow conservation constraints (6.5); the root has one outgoing flow (first case), the terminal one ingoing flow (second case), and for all other vertices the ingoing flow equals the outgoing flow. Edges which are used for routing the flow are selected as solution edges by the capacity constraints (6.4), either as first-stage or as second-stage edges. It is easy to see that the formulation ($\mathrm{SSTP_{uf}}$) is valid and that it is equivalent to the flow model introduced by [87].

Due to the discussion on the deterministic STP it is not surprising that the cut-based formulation is equivalent to the flow formulation, c.f. Section 6.4. However, there exist stronger formulations based on orientation properties.

6.2 Semi-directed formulations

Semi-directed cut formulations. In the following we introduce semi-directed cut-based formulations for the SSTP which are based on the application of orientation properties like in the directed cut formulation for the deterministic STP. However, edge variables x^0 for the first stage remain undirected in all semi-directed formulations. As will be discussed at the beginning of Section 6.3, using a directed first stage is difficult and no stronger formulation is known. On the other hand, it is possible to consider the bidirected input graph in the second stage.

In the first semi-directed model we use edge variables x^0 and arc variables $z_a^k, \forall a \in A, \forall k \in \mathcal{K}$. We search for a first-stage edge set E^0 and second-stage arc sets $A^1, \ldots, A^K$ such that $E^0 \cup A^k$ contains a semi-directed path from a designated terminal $r^k \in T^k$ to each terminal in $T_r^k = T^k \setminus \{r^k\}$, for all scenarios $k \in \mathcal{K}$. In other words, $A^0 \cup A^k$ has to contain a feasible arborescence for all scenarios $k \in \mathcal{K}$, with $A^0 := \bigcup_{\{i,j\} \in E^0} \{(i,j),(j,i)\}$. To shorten

the notation we use $V_r^k := V \backslash \{r^k\}$ and we write $(x^0 + z^k)(\delta^-(S)) := x^0(\delta(S)) + z^k(\delta^-(S)) = \sum_{(i,j)\in\delta^-(S)} x^0_{\{i,j\}} + z^k_{ij}$ for semi-directed cuts.

$$\text{(SSTP}_{\text{sdc1}}\text{)} \quad \min \sum_{e\in E} c_e^0 x_e^0 + \sum_{k\in\mathcal{K}} p^k \sum_{e=\{i,j\}\in E} c_e^k (z_{ij}^k + z_{ji}^k)$$

$$\text{s.t. } (x^0 + z^k)(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V_r^k : S \cap T_r^k \neq \emptyset \tag{6.9}$$

$$x^0 \in \{0,1\}^{|E|} \tag{6.10}$$

$$z^{1\ldots K} \in \{0,1\}^{|A|\cdot K} \tag{6.11}$$

This first formulation uses semi-directed cuts, i.e., each cut (6.9) for scenario $k \in \mathcal{K}$ can be fulfilled by first-stage edges or by second-stage arcs from this scenario.

Lemma 6.1. *Formulation* ($SSTP_{sdc1}$) *models the stochastic Steiner tree problem correctly.*

Proof. Let $\tilde{E}^0, \tilde{E}^1, \ldots, \tilde{E}^K$ be an optimum solution for the stochastic Steiner tree problem. Since this solution connects all terminals in all scenarios we can easily find 0/1-values for x^0 and $z^k, \forall k \in \mathcal{K}$, respectively, by using (and orienting) exactly the edges $\tilde{E}^0, \ldots, \tilde{E}^K$ such that there is a semi-directed path from r^k to each terminal in $T_r^k, \forall k \in \mathcal{K}$.

On the other hand, due to constraints (6.9), an optimum solution $(\tilde{x}^0, \tilde{z}^{1\ldots K})$ to ($\text{SSTP}_{\text{sdc1}}$) connects the designated root node r^k by semi-directed paths to each terminal in T_r^k. Hence, using the selected undirected first-stage edges plus the undirected counterparts of the second-stage arcs gives a feasible solution to the SSTP. In both cases the objective value remains the same. □

In formulation ($\text{SSTP}_{\text{sdc1}}$) a selected first-stage edge fulfills all related semi-directed cuts. Hence, in the extreme case when all terminals are connected via first-stage edges this model does not give stronger bounds than the undirected model.

This drawback is overcome by the second semi-directed formulation. It is based on additional capacity constraints which enforce that selected first-stage edges have to be incorporated into the second-stage solution and in particular, each selected first-stage edge has to be oriented such that a feasible arborescence is established in each scenario. Due to this change, the cut constraints are now purely directed and contain only second-stage arc variables. Because of the different meaning of the second-stage arc variables we use the identifier $y^{1\ldots K}$ instead of $z^{1\ldots K}$ as in ($\text{SSTP}_{\text{sdc1}}$). The second semi-directed cut formulation for the SSTP reads as follows:

$$\text{(SSTP}_{\text{sdc2}}\text{)} \quad \min \sum_{e\in E} c_e^0 x_e^0 + \sum_{k\in\mathcal{K}} p^k \sum_{e=\{i,j\}\in E} c_e^k (y_{ij}^k + y_{ji}^k - x_e^0)$$

$$\text{s.t. } y^k(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V_r^k : S \cap T_r^k \neq \emptyset \tag{6.12}$$

$$y_{ij}^k + y_{ji}^k \geq x_e^0 \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E \tag{6.13}$$

$$x^0 \in \{0,1\}^{|E|} \tag{6.14}$$

$$y^{1\ldots K} \in \{0,1\}^{|A|\cdot K} \tag{6.15}$$

This model is basically a union of K directed Steiner tree formulations joined by the first stage through capacity constraints (6.13). Compared to the previous cut-based formulations the objective function contains a corrective term for subtracting the additional cost that results from these constraints.

Lemma 6.2. *Formulation ($SSTP_{sdc2}$) models the stochastic Steiner tree problem correctly.*

Proof. An optimum solution $\tilde{E}^0, \tilde{E}^1, \ldots, \tilde{E}^K$ to the SSTP can be easily translated into a feasible solution for model ($\text{SSTP}_{\text{sdc2}}$) by using the edge set $\tilde{E}^0 \cup \tilde{E}^k$ for finding a feasible arborescence in each scenario $k \in \mathcal{K}$; then let variables x^0 represent $\tilde{E}^0$ and set arc variables y^k according to the arborescences, $\forall k \in \mathcal{K}$. Due to the corrective term the objective values are identical.

Contrarily, due to the correctness of (STP_{dc}) for the deterministic STP an optimum solution $(\tilde{x}^0, \tilde{y}^{1\ldots K})$ to ($\text{SSTP}_{\text{sdc2}}$) contains an r^k-rooted arborescence in each scenario $k \in \mathcal{K}$. Hence, $\tilde{E}^0, \tilde{E}^1, \ldots, \tilde{E}^K$, with $\tilde{E}^0 := \{e \in E \mid \tilde{x}^0_e = 1\}$ and $\forall k \in \mathcal{K}$: $\tilde{E}^k := \{e = \{i,j\} \in E \mid \tilde{x}^0_e = 0 \wedge (\tilde{y}^k_{ij} = 1 \vee \tilde{y}^k_{ji} = 1)\}$, is a feasible solution with the same objective value. □

Let ($\text{SSTP}^{\text{rel:}x^0}_{\text{sdc2}}$) denote formulation ($\text{SSTP}_{\text{sdc2}}$) with the integrality constraint (6.14) being relaxed to $x^0 \in [0,1]^{|E|}$.

Lemma 6.3. *The optimum solution to ($SSTP^{rel:x^0}_{sdc2}$) is integer.*

Proof. Assume there exists an optimum solution $(\tilde{x}^0, \tilde{y}^{1\ldots K})$ to ($\text{SSTP}^{\text{rel:}x^0}_{\text{sdc2}}$) that is non-integer. Let variable $\tilde{x}^0_e$ corresponding to edge $e = \{i,j\} \in E$ be fractional, i.e., $0 < \tilde{x}^0_e < 1$. The term in the objective function corresponding to edge e is:

$$\begin{aligned}
&c^0_e \tilde{x}^0_e + \sum_{k \in \mathcal{K}} p^k c^k_e (\tilde{y}^k_{ij} + \tilde{y}^k_{ji} - \tilde{x}^0_e) \\
&= c^0_e \tilde{x}^0_e - \sum_{k \in \mathcal{K}} p^k c^k_e \tilde{x}^0_e + \sum_{k \in \mathcal{K}} p^k c^k_e (\tilde{y}^k_{ij} + \tilde{y}^k_{ji}) \\
&= (c^0_e - c^*_e)\tilde{x}^0_e + \sum_{k \in \mathcal{K}} p^k c^k_e (\tilde{y}^k_{ij} + \tilde{y}^k_{ji})
\end{aligned}$$

In case $c^0_e < c^*_e$ set $\tilde{x}^0_e := 1$ and if $c^0_e > c^*_e$ set $\tilde{x}^0_e := 0$. In both cases the resulting solution is still feasible: Constraint (6.13) together with the integrality of $y^{1\ldots K}$ ensures that for all scenarios $k \in \mathcal{K}$ it holds $\tilde{y}^k_{ij} + \tilde{y}^k_{ji} \geq 1$ and hence, (6.13) is still satisfied. Moreover, the objective value improves which is a contradiction.

In case $c^0_e = c^*_e$ variable x^0_e has coefficient 0 in the objective function and can be fixed to $\tilde{x}^0_e := 0$. □

Notice that this property only holds for the non-decomposed formulation. It can only be applied when the model gets solved directly with a single branch&cut algorithm. However, when applying a decomposition approach like the two-stage branch&cut algorithm the integrality constraints cannot be relaxed anymore.

We close the discussion on semi-directed formulations by rewriting the objective function of ($\text{SSTP}_{\text{sdc2}}$). First, consider only the parts concerning the first-stage variables:

$$\begin{aligned}
&\min \sum_{e \in E} c^0_e x^0_e + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} -c^k_e x^0_e = \min \sum_{e \in E} c^0_e x^0_e + \sum_{k \in \mathcal{K}} \sum_{e \in E} -p^k c^k_e x^0_e \\
&= \min \sum_{e \in E} c^0_e x^0_e + \sum_{e \in E} x^0_e \sum_{k \in \mathcal{K}} -p^k c^k_e = \min \sum_{e \in E} c^0_e x^0_e + \sum_{e \in E} -x^0_e c^*_e \\
&= \min \sum_{e \in E} (c^0_e - c^*_e) x^0_e
\end{aligned}$$

Now, we rewrite the objective function and call the resulting formulation ($\text{SSTP}_{\text{sdc2}^*}$):

$$(\text{SSTP}_{\text{sdc2}^*}) \min \sum_{e \in E} (c_e^0 - c_e^*) x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e=\{i,j\} \in E} c_e^k (y_{ij}^k + y_{ji}^k)$$
$$\text{s.t. } (x^0, y^{1\dots K}) \text{ satisfies (6.12)–(6.15)}$$

Obviously, ($\text{SSTP}_{\text{sdc2}^*}$) is identical to ($\text{SSTP}_{\text{sdc2}}$). However, when the models get decomposed the modified objective function does matter. Then, the master problem of formulation ($\text{SSTP}_{\text{sdc2}^*}$) has negative coefficients (due to assumption $c_e^* > c_e^0$) whereas the coefficients in the master problem of ($\text{SSTP}_{\text{sdc2}}$) are positive. Moreover, this change affects the primal and dual subproblems and in particular, the generated L-shaped optimality cuts. We discuss the differences w.r.t. the decomposition in Section 7.3 and the impact on the performance in the computational study, cf. Section 8.3.2.

Remark 6.4. We like to shortly revisit formulation (SSTP_{uc}) based on undirected cuts. Notice that by adding similar capacity constraints $x_e^k \geq x_e^0, \forall k \in \mathcal{K}, \forall e \in E$, the undirected cuts (6.1) contain only second-stage variables, as in model ($\text{SSTP}_{\text{sdc2}}$). Then, it is possible to relax the first-stage variables to $x^0 \in [0,1]^{|E|}$ without violating overall integrality; the proof is similar to the one of Lemma (6.3). On the other hand, these modifications do not influence the strength of the LP relaxation and this formulation is as strong as (SSTP_{uc}).

Semi-directed flow formulation. The flow formulation can be strengthened as in the deterministic setting. One simply has to enforce that a selected undirected edge cannot be used for routing flow in both directions for the same commodity. Therefore, directed arc variables $y^k, \forall k \in \mathcal{K}$, are used and constraints (6.4) are replaced by the stronger constraints (6.16). To highlight the connection to formulation ($\text{SSTP}_{\text{sdc2}}$) we use the same capacity constraints (6.17).

$$(\text{SSTP}_{\text{sdf}}) \min \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e=\{i,j\} \in E} c_e^k (y_{ij}^k + y_{ji}^k - x_e^0)$$

$$\text{s.t. } f \text{ satisfies (6.5)}$$

$$y_{ij}^k \geq f_{ij}^{k,t} \qquad \forall k \in \mathcal{K}, \forall (i,j) \in A, \forall t \in T_r^k \tag{6.16}$$

$$y_{ij}^k + y_{ji}^k \geq x_e^0 \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E \tag{6.17}$$

$$f \in [0,1]^{|A| \cdot t_r^*} \tag{6.18}$$

$$x^0 \in \{0,1\}^{|E|} \tag{6.19}$$

$$y^{1\dots K} \in \{0,1\}^{|A| \cdot K} \tag{6.20}$$

Formulation (SSTP_{sdf}) is the equivalent to ($\text{SSTP}_{\text{sdc2}}$): instead of satisfying directed cuts one has to find a feasible flow in each scenario and moreover, the scenarios are linked to the first stage by capacity constraints (6.17).

Observation 6.5. *Formulation (SSTP_{sdf}) models the stochastic Steiner tree problem correctly.*

6.3 Directed formulations

The last section presents semi-directed formulations for the stochastic Steiner tree problem with first-stage decisions still being represented by undirected edges. The natural question is

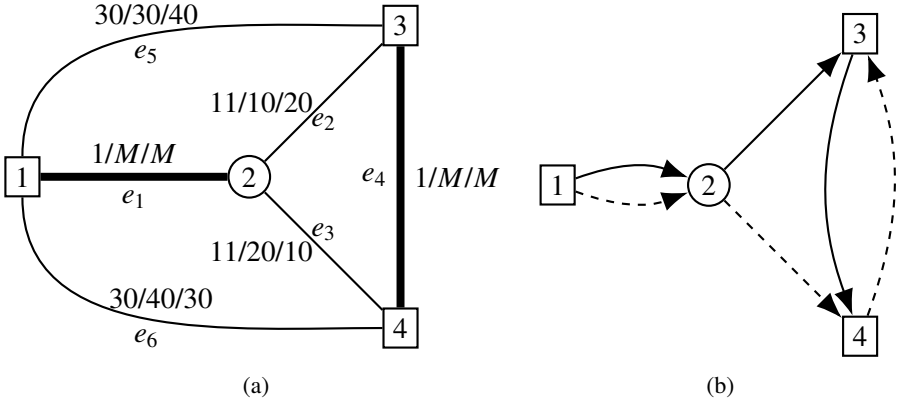


Figure 6.1: (a) The same SSTP instance as in Figure 4.2. There are two equally probable scenarios with identical terminal set {1,3,4}; the edge costs for the first stage and the two scenarios are written next to the edges (i.e., $c^0/c^1/c^2$) with M being a sufficiently large positive value. The optimum solution edges of the first stage are highlighted by thick edges; scenario 1 and 2 additionally purchase edge e_2 and e_3, respectively. The optimum solution has cost 12. (b) Minimum arborescences (y^k-values) in the scenarios for formulation ($\text{SSTP}_{\text{sdc2}}$). Solid arcs represent the first and dashed arcs the second scenario.

whether the undirected edges are necessary or if it is possible to model the SSTP with a fully directed and stronger formulation. In the following, we briefly discuss difficulties coming along with the unrooted SSTP and afterwards, we describe fully directed formulations for the rSSTP.

Difficulties for the unrooted SSTP. Formulating the SSTP with a directed first stage causes difficulties since first-stage solutions may be disconnected. Consider Figure 6.1 which depicts such an example. Here, the optimum first-stage solution is disconnected as shown in Figure 6.1 (a). The optimum arborecences of the two scenarios are given in (b). In particular, edge e_4 is used in direction $(3, 4)$ in one and direction $(4, 3)$ in the other scenario. Hence, already fixing an orientation in the first stage omits an optimum scenario solution—or at least, makes the corresponding solution more expensive.

Open problem 6.1. Does there exist a directed model for the stochastic Steiner tree problem (SSTP) which is stronger than the semi-directed model ($\text{SSTP}_{\text{sdc2}}$)?

Remark 6.6. Of course, it is possible to replace the undirected edge variables in formulation ($\text{SSTP}_{\text{sdc2}}$) by directed arc variables y^0 in the first stage and then, in the second stage, one can use capacity constraints $y^k_{ij} + y^k_{ji} \geq y^0_{ij} + y^0_{ji}, \forall k \in \mathcal{K}, \forall (i, j) \in A$. However, this does not yield a stronger formulation since the orientation of an edge is lost in the second stage; it is easy to see that the arc variables can be replaced by undirected edge variables by $x^0_e := y^0_{ij} + y^0_{ji}$.

Directed cut formulations for the rSSTP. While we are not aware of a fully directed and stronger cut-based formulation for the SSTP the rooted version of the SSTP permits a stronger model with directed variables only. For the following formulations we again consider the weighted bidirection $\bar{G} = (V, A, c)$ of the input graph.

The first formulation is called ($\text{rSSTP}_{\text{dc1}}$); afterwards, we introduce two more formulations ($\text{rSSTP}_{\text{dc2}}$) and ($\text{rSSTP}_{\text{dc2}^*}$), respectively, similar to the semi-directed SSTP models. We use directed arc variables z^0 and z^k for the first and second stage in scenario $k \in \mathcal{K}$, respectively. Let $V_r := V \backslash \{r\}$ and $T_r^k := T^k \backslash \{r\}, \forall k \in \mathcal{K}$.

$$(\text{rSSTP}_{\text{dc1}}) \min \sum_{a \in A} c_a^0 z_a^0 + \sum_{k \in \mathcal{K}} p^k \sum_{a \in A} c_a^k z_a^k$$

$$\text{s.t. } z^0(\delta^-(S)) \geq z^0(\delta^-(v)) \qquad \forall \emptyset \neq S \subseteq V_r, \forall v \in S \tag{6.21}$$

$$(z^0 + z^k)(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V_r : S \cap T_r^k \neq \emptyset \tag{6.22}$$

$$z^0 \in \{0,1\}^{|A|} \tag{6.23}$$

$$z^{1\ldots K} \in \{0,1\}^{|A| \cdot K} \tag{6.24}$$

Constraints (6.22) are directed cuts ensuring a feasible arborescence in each scenario consisting of first- and second-stage arcs. Moreover, the additional directed cuts (6.21) are used to enforce the required first-stage tree.

Lemma 6.7. *Formulation ($rSSTP_{dc1}$) models the rooted stochastic Steiner tree problem correctly.*

Proof. Let $\tilde{E}^0, \tilde{E}^1, \ldots, \tilde{E}^K$ describe an optimum rSSTP solution. Since $\tilde{E}^0$ induces a tree the edges can be oriented from the root r outwards. Then, it is clear that for each scenario $k \in \mathcal{K}$ the edge set $\tilde{E}^k$ can be oriented such that $\tilde{E}^0 \cup \tilde{E}^k$ contains an arborescence with directed paths from r to each terminal. This orienting procedure gives a solution to ($\text{rSSTP}_{\text{dc1}}$) with the same cost.

On the other hand, an optimum solution to ($\text{rSSTP}_{\text{dc1}}$) guarantees that every terminal is reachable by a directed path from the root node due to constraints (6.22). Moreover, constraints (6.21) plus the objective function ensure that the first stage is a tree rooted at r. Hence, the related undirected edges yield a feasible solution to the rSSTP; again with identical objective value. □

It is possible to use the same idea leading to the semi-directed formulation ($\text{SSTP}_{\text{sdc2}}$) for another directed formulation for the rSSTP. The variable identifier for the first-stage arcs is z^0 and the arc variables for the K scenarios are $y^{1\ldots K}$. Again, we use identifier y due to the different meaning such that scenario arcs contain already selected first-stage arcs.

$$(\text{rSSTP}_{\text{dc2}}) \min \sum_{a \in A} c_a^0 z_a^0 + \sum_{k \in \mathcal{K}} p^k \sum_{a \in A} c_a^k (y_a^k - z_a^0)$$

$$\text{s.t. } z^0(\delta^-(S)) \geq z^0(\delta^-(v)) \qquad \forall \emptyset \neq S \subseteq V_r, \forall v \in S \tag{6.25}$$

$$y^k(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V_r : S \cap T_r^k \neq \emptyset \tag{6.26}$$

$$y_a^k \geq z_a^0 \qquad \forall k \in \mathcal{K}, \forall a \in A \tag{6.27}$$

$$z^0 \in \{0,1\}^{|A|} \tag{6.28}$$

$$y^{1\ldots K} \in \{0,1\}^{|A| \cdot K} \tag{6.29}$$

Constraints (6.25) are identical to constraints (6.21) in ($\text{rSSTP}_{\text{dc1}}$) and model the first-stage tree. Capacity constraints (6.27) enforce the selection of first-stage arcs in each scenario. Then, the directed cuts (6.26) in the scenarios contain only variables y.

The preceding discussions allow us to observe the correctness of this formulation.

Observation 6.8. *Formulation ($rSSTP_{dc2}$) models the rooted stochastic Steiner tree problem correctly.*

The objective function of model ($\text{rSSTP}_{\text{dc2}}$) can be rewritten analogously to the semi-directed formulation. We call the resulting formulation ($\text{rSSTP}_{\text{dc2}^*}$) which is equivalent to ($\text{rSSTP}_{\text{dc2}}$) but the change in the objective function matters when a decomposition is applied.

$$(\text{rSSTP}_{\text{dc2}^*}) \min \sum_{a\in A}(c_a^0 - c_a^*)z_a^0 + \sum_{k\in\mathcal{K}} p^k \sum_{a\in A} c_a^k y_a^k$$
$$\text{s.t. } (z^0, y^{1\dots K}) \text{ satisfies (6.25)–(6.29)}$$

If $c_a^0 < c_a^* := \sum_{k\in\mathcal{K}} p^k c_a^k$ holds for all arcs $a \in A$ we can again relax the integrality restrictions on the first-stage variables without losing overall integrality. Let ($\text{rSSTP}_{\text{dc2}}^{\text{rel:}z^0}$) denote formulation ($\text{rSSTP}_{\text{dc2}}$) with the integrality constraint (6.28) being relaxed to $z^0 \in [0,1]^{|A|}$.

Theorem 6.9. *If $c_a^0 < c_a^*$ holds for all arcs $a \in A$ the optimum solution to ($rSSTP_{dc2}^{rel:z^0}$) is integer.*

Proof. Let $(\tilde{z}^0, \tilde{y}^{1\dots K})$ denote an optimum solution to ($\text{rSSTP}_{\text{dc2}}^{\text{rel:}z^0}$) which is non-integer. We consider an arc $\alpha \in A$ with $0 < \tilde{z}_\alpha^0 < 1$ defined as follows. If there exists a fractional arc (r, j) we set $\alpha := (r, j)$. Otherwise, we set $\alpha := (i, j)$ such that the directed path P from the root r to vertex i consists only of selected arcs, i.e., $\tilde{z}_a^0 = 1, \forall a \in P$. Notice that arc α is well-defined due to constraints (6.25).

We consider three main cases. In each case we construct a feasible solution $(\hat{z}^0, \hat{y}^{1\dots K})$ with a better objective value than by $(\tilde{z}^0, \tilde{y}^{1\dots K})$. We always start with $\hat{z}^0 := \tilde{z}^0, \hat{y}^{1\dots K} := \tilde{y}^{1\dots K}$ and describe the necessary modifications.

Case 1: $\alpha = (i, r)$. Since α is an ingoing arc of the root r it is not contained in any directed cut. Hence, setting $\hat{z}_\alpha^0 := 0$ and $\hat{y}_\alpha^k := 0, \forall k \in \mathcal{K}$, gives a feasible and better solution.

Case 2: $\alpha = (r, j)$. In this case set $\hat{z}_\alpha^0 := 1$. First, notice that the objective value improves since the term in the objective function with respect to arc α is $c_\alpha^0 \tilde{z}_\alpha^0 + \sum_{k\in\mathcal{K}} p^k c_\alpha^k (\tilde{y}_\alpha^k - \tilde{z}_\alpha^0) = c_\alpha^0 \tilde{z}_\alpha^0 + \sum_{k\in\mathcal{K}} p^k c_\alpha^k (1 - \tilde{z}_\alpha^0) = (c_\alpha^0 - c_\alpha^*)\tilde{z}_\alpha^0 + c_\alpha^*$ and $c_\alpha^0 < c_\alpha^*$.

Second, we argue that the solution $(\hat{z}^0, \hat{y}^{1\dots K})$ is feasible. Since $\hat{y}^{1\dots K} = \tilde{y}^{1\dots K}$ we do not need to consider constraints (6.26). Constraints (6.25) are only crucial for vertex j since for all other vertices the right-hand side does not change and the left-hand side does not decrease. For vertex j notice that z_α^0 is contained in the left-hand and in the right-hand side of any constraint; hence, the constraints are still satisfied. Constraint (6.27) is also only interesting for arc α; but since $\tilde{z}_\alpha^0 > 0$ it holds $\hat{y}_\alpha^k = \tilde{y}_\alpha^k = 1$ and the constraint is also still satisfied.

Case 3: $\alpha = (i, j)$ *with* $i \neq r, j \neq r$. Let $\mathcal{L} := \{\ell \in V \mid (\ell, j) \in A, \ell \neq i, \tilde{z}_{\ell j}^0 > 0\}$, i.e., $\mathcal{L}$ is the set of vertices $\ell \neq i$ with a (fractionally) selected arc (ℓ, j).

Case 3.1: $\mathcal{L} = \emptyset$. Hence, arc α is the only ingoing arc of j with $\tilde{z}_{\cdot,j}^0 > 0$. In this case we set $\hat{z}_\alpha^0 := 1$.

The arguments are similar to Case 2. Again, the objective value improves and constraints (6.26) and (6.27) are still satisfied. Constraints (6.25) are again only crucial for

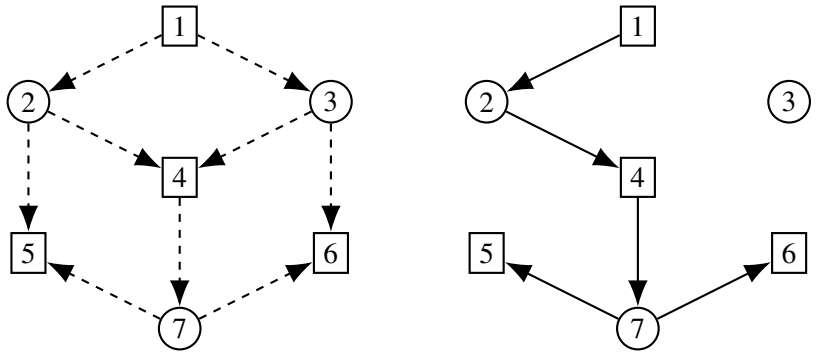


Figure 6.2: Instance for the STP where (STP_{dc}) has an integrality gap of 10/9. All edge costs are 1 and terminals are drawn as rectangles. (a) shows the optimum fractional solution (dashed arcs are set to 0.5) whereas (b) depicts an optimum integer solution. Moreover, this graph can be used to construct an rSSTP instance where the optimum solution to ($\text{rSSTP}^{\text{rel:}z^0}_{\text{dc1}}$) is fractional but ($\text{rSSTP}^{\text{rel:}z^0}_{\text{dc2}}$) is integer.

vertex j and are satisfied due to the properties of arc α: Recall that we set α such that the directed path P from r to i consists of arcs a with $\tilde{z}^0_a = 1, \forall a \in P$. Hence, any cut S with $j \in S, r \notin S$, satisfies $\hat{z}^0(\delta^-(S)) \geq \tilde{z}^0(\delta^-(S)) \geq 1 = \hat{z}^0(\delta^-(j))$.

Case 3.2: $\mathcal{L} \neq \emptyset$. Since $\mathcal{L} \neq \emptyset$ there exists at least one arc (ℓ, j) with $\tilde{z}^0_{\ell j} > 0, \ell \neq i$.

Hence, due to capacity constraints (6.27) and integrality of y^k in any scenario $k \in \mathcal{K}$ it holds $\tilde{y}^k(\delta^-(j)) = 1 + |\mathcal{L}| \geq 2$. Since directed cuts have a right-hand side of 1 it is clear that this solution is non-optimal.

Now, set $\hat{z}^0_\alpha := 1$, $\hat{z}^0_{\ell j} := 0, \forall \ell \in \mathcal{L}$, and $\hat{y}^k_{\ell j} := 0, \forall \ell \in \mathcal{L}, \forall k \in \mathcal{K}$. First, we argue that this solution has a better objective value and afterwards, we discuss its feasibility.

As discussed in Case 2 increasing $\hat{z}^0_\alpha$ leads to a decrease of the objective value. Moreover, deleting arcs from the solution by setting $\hat{z}^0_{\ell j} := 0, \forall \ell \in \mathcal{L}$, and $\hat{y}^k_{\ell j} := 0, \forall \ell \in \mathcal{L}, \forall k \in \mathcal{K}$, improves the objective, too. Hence, the newly constructed solution has a better objective value.

To show the feasibility of this solution we consider the constraints one by one. Capacity constraints (6.27) are satisfied by construction. The directed cuts in the scenarios (6.26) are satisfied for every valid cut $S \ni j$ since S crosses the path P or arc α where each arc $a \in P \cup \alpha$ has a value $\hat{y}^k_a = 1, \forall k \in \mathcal{K}$, such that it holds $\hat{y}^k(\delta^-(S)) \geq 1$. All other valid cuts $S \not\ni j$ are still satisfied since the arc variables crossing these cuts are not modified.

Last but not least, we have to consider constraints (6.25) where the arguments are again similar. Consider any valid cut S for constraint (6.25). If $j \notin S$ the constraint is still satisfied since the related arc variables are unchanged. In case $j \in S$ the cut S crosses $P \cup \alpha$ such that (i) $\hat{z}^0(\delta^-(S)) \geq 1$. Since arc costs are non-negative and the right-hand side of the directed cuts is 1 any optimum solution satisfies (ii) $z^0(\delta^-(v)) \leq 1, \forall v \in V$. We modified z^0 such that (iii) $\hat{z}^0(\delta^-(j)) = 1$. Combining (i)–(iii) shows that constraints (6.25) are satisfied. □

We like to shortly revisit the first directed cut formulation ($\text{rSSTP}_{\text{dc1}}$) and show that ($\text{rSSTP}^{\text{rel:}z^0}_{\text{dc1}}$) does not have the latter property; let ($\text{rSSTP}^{\text{rel:}z^0}_{\text{dc1}}$) denote model ($\text{rSSTP}_{\text{dc1}}$) with relaxed first-stage variables $z^0 \in [0,1]^{|A|}$. An example is given by Figure 6.2 (a). The

corresponding undirected graph depicts a classical instance for the deterministic STP (cf. e.g., Polzin and Daneshmand [142]) where the directed cut formulation has an integrality gap—here it is $10/9$. Now, consider an rSSTP instance on that graph that contains one scenario with the four terminals $\{1, 4, 5, 6\}$ and with vertex 1 being the root node r. Moreover, let the cost in the first stage be 1 for each edge and in the scenario 2 for each edge such that ($\text{rSSTP}^{\text{rel:}z0}_{\text{dc1}}$) connects all terminals already in the first stage. Figure 6.2 (a) gives the optimum solution with cost 4.5 for model ($\text{rSSTP}^{\text{rel:}z0}_{\text{dc1}}$) where each dashed arc a is set to $z^0_a := 0.5$; moreover, $z^{1\dots K} := \mathbf{0}$ is integer. Figure 6.2 (b) depicts the first stage of an optimum solution with cost 5 for the described rSSTP instance which is also the optimum solution to ($\text{rSSTP}^{\text{rel:}z0}_{\text{dc2}}$).

Directed flow formulations for the rSSTP. We close the discussion on the formulations for the rooted stochastic Steiner tree problem by introducing a polynomially sized model. This formulation is again flow-based. Compared to the previously introduced flow formulations it requires additional node variables $w^0_v \in \{0, 1\}, \forall v \in V$, and additional first-stage flow variables $f^{0,v}_{ij}, \forall v \in V_r, \forall (i, j) \in A$, for ensuring the first-stage tree.

The description of the formulation is split into several parts for better readability. First, we introduce the variables. The solution is represented by arc variables z^0 for the first stage and $y^{1\dots K}$ for the K scenarios. Similar to the flow formulations (SSTP_{uf}) and (SSTP_{sdf}) and the cut-based formulation ($\text{rSSTP}_{\text{dc2}}$) we use capacity constraints to ensure that each first-stage arc is also used in each scenario. Hence, we use the same identifier $y^{1\dots K}$ for the second stage.

As for the semi-directed flow formulations we have flow variables $f^{k,t}_{ij}$ for each scenario $k \in \mathcal{K}$, terminal $t \in T^k_r$, and arc $(i, j) \in A$. Moreover, we use the already mentioned flow variables $f^{0,v}_{ij}$ and binary node variables w^0_v for the first stage.

$$f^0 \in [0, 1]^{|V_r|\cdot|A|} \tag{6.30}$$

$$f^{1\dots K} \in [0, 1]^{|A|\cdot t^*_r} \tag{6.31}$$

$$z^0 \in \{0, 1\}^{|A|} \tag{6.32}$$

$$w^0 \in \{0, 1\}^{|V_r|} \tag{6.33}$$

$$y^{1\dots K} \in \{0, 1\}^{|A|\cdot K} \tag{6.34}$$

The constraints which contain first-stage variables are given as follows. Thereby, $w^0_v = 1$ implies that vertex v is contained in the first-stage tree. In this case a flow of unit one needs to be send from the root to this vertex. This is ensured by the classical flow conservation constraints (6.37); here with the right-hand side $-w^0_v$ and w^0_v, respectively. Constraints (6.36) ensure the correct assignment of the node variables.

$$z^0_{ij} \geq f^{0,v}_{ij} \qquad \forall v \in V_r, \forall (i, j) \in A \tag{6.35}$$

$$w^0_v \geq z^0(\delta^-(v)) \qquad \forall v \in V_r \tag{6.36}$$

$$f^{0,v}(\delta^-(i)) - f^{0,v}(\delta^+(i)) = \begin{cases} -w^0_v, & \text{if } i = r \\ w^0_v, & \text{if } i = v \\ 0, & \text{otherwise} \end{cases} \quad \forall v \in V_r, \forall i \in V \tag{6.37}$$

Again, we use capacity constraints (6.38) to ensure that each first-stage arc is also used in each scenario. These constraints link the first and second stage and they are the only

constraints using both first- and second-stage variables.

$$y_{ij}^k \geq z_{ij}^0 \qquad \forall k \in \mathcal{K}, \forall (i,j) \in A \tag{6.38}$$

The remaining constraints are identical to the constraints in the semi-directed flow formulation. They ensure that all arcs used for routing flow are also purchased in the objective function and that the constructed flow is valid.

$$y_{ij}^k \geq f_{ij}^{k,t} \qquad \forall k \in \mathcal{K}, \forall (i,j) \in A, \forall t \in T_r^k \tag{6.39}$$

$$f^{k,t}(\delta^-(i)) - f^{k,t}(\delta^+(i)) = \left.\begin{cases} -1, & \text{if } i = r \\ 1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}\right\} \begin{array}{l} \forall k \in \mathcal{K}, \forall t \in T_r^k, \\ \forall i \in V \end{array} \tag{6.40}$$

Finally, the directed flow-based formulation reads as follows:

$$\begin{aligned} (\text{rSSTP}_{\text{df}}) \ \min & \sum_{a \in A} c_a^0 z_a^0 + \sum_{k \in \mathcal{K}} p^k \sum_{a \in A} c_a^k (y_a^k - z_a^0) \\ \text{s.t. } & (z^0, y^{1\ldots K}, w^0, f) \text{ satisfies (6.30)–(6.40)} \end{aligned}$$

Due to the preceding discussion we skip the proof about the correctness.

Observation 6.10. *Formulation ($rSSTP_{df}$) models the rooted stochastic Steiner tree problem correctly.*

6.4 Strength of the formulations

This section provides a comparison of the introduced formulations from a polyhedral point of view. We consider the undirected and semi-directed formulations for the SSTP in Section 6.4.1 and Section 6.4.2, respectively, while Section 6.4.3 focusses on the directed models for the rooted problem.

6.4.1 Undirected formulations for the SSTP

We start by comparing the undirected formulations based on cuts and flows, respectively. The related polytopes of the relaxed formulations are denoted by

$$\begin{aligned} \mathcal{P}_{\text{uc}}^{\text{SSTP}} &= \left\{ x^{0\ldots K} \in [0,1]^{|E| \cdot (K+1)} \,\middle|\, x^{0\ldots K} \text{ satisfies (6.1)} \right\} \\ \mathcal{P}_{\text{uf}}^{\text{SSTP}} &= \Big\{ (x^{0\ldots K}, f) \in [0,1]^{|E| \cdot (K+1)} \times [0,1]^{|A| \cdot t_r^*} \,\Big| \\ &\qquad\qquad (x^{0\ldots K}, f) \text{ satisfies (6.4) and (6.5)} \Big\} \end{aligned}$$

In order to compare the formulations we project the variables of the flow formulation onto the space of undirected edge variables, i.e.,

$$\text{Proj}_{x^{0\ldots K}} \left(\mathcal{P}_{\text{uf}}^{\text{SSTP}} \right) = \left\{ x^{0\ldots K} \,\middle|\, \exists f \colon (x^{0\ldots K}, f) \in \mathcal{P}_{\text{uf}}^{\text{SSTP}} \right\}$$

As for the undirected cut-based and flow-based formulations of the deterministic STP the two formulations for the SSTP are equally strong.

Lemma 6.11. *$\mathcal{P}^{SSTP}_{uc} = Proj_{x^{0\dots K}}\left(\mathcal{P}^{SSTP}_{uf}\right)$, i.e., the undirected cut- and flow-based formulations ($SSTP_{uc}$) and ($SSTP_{uf}$) are equivalent.*

Proof. This lemma follows directly from the classical "max flow = min cut" theorem, cf. Theorem 2.1, applied to each scenario. If there is a flow of one unit from the root node to each terminal then every cut separating the terminal from the root node is satisfied. On the other hand, if every undirected cut is satisfied it is possible to find a feasible flow from the root node to every terminal using exactly those edges. In both models either first- or second-stage edges can be used. □

6.4.2 Semi-directed formulations for the SSTP

Before comparing the formulations we expand the semi-directed cut formulations by *subtour elimination constraints of size two (SEC2)* in the second stage; constraints (6.41) are added to ($\text{SSTP}_{\text{sdc1}}$) and (6.42) to ($\text{SSTP}_{\text{sdc2}}$), respectively:

$$z^k_{ij} + z^k_{ji} \leq 1 \qquad \forall k \in \mathcal{K}, \forall \{i,j\} \in E \tag{6.41}$$

$$y^k_{ij} + y^k_{ji} \leq 1 \qquad \forall k \in \mathcal{K}, \forall \{i,j\} \in E \tag{6.42}$$

We introduce the additional constraints to make the comparison of polytopes easier. Although these constraints cut the polytopes of the LP relaxations they are not binding, i.e., any optimum solution satisfies the SEC2s anyway. As noted by Chopra and Rao [44] for the STP: "the inequalities [. . .] are redundant but are added to make projection easier".

Then, the polytopes of the relaxed cut formulations are denoted by

$$\mathcal{P}^{\text{SSTP}}_{\text{sdc1}} = \Big\{(x^0, z^{1\dots K}) \in [0,1]^{|E|} \times [0,1]^{|A|\cdot K} \Big| \\ (x^0, z^{1\dots K}) \text{ satisfies (6.9) and (6.41)}\Big\}$$

$$\mathcal{P}^{\text{SSTP}}_{\text{sdc2}} = \Big\{(x^0, y^{1\dots K}) \in [0,1]^{|E|} \times [0,1]^{|A|\cdot K} \Big| \\ (x^0, y^{1\dots K}) \text{ satisfies (6.12), (6.13), and (6.42)}\Big\}$$

Again, we consider the projections onto the space of undirected edge variables $x^{0\dots K}$:

$$\text{Proj}_{x^{0\dots K}}\left(\mathcal{P}^{\text{SSTP}}_{\text{sdc1}}\right) = \Big\{x^{0\dots K} \Big| \exists z^{1\dots K} : (x^0, z^{1\dots K}) \in \mathcal{P}^{\text{SSTP}}_{\text{sdc1}}, \\ x^k_e = z^k_{ij} + z^k_{ji}, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E\Big\}$$

$$\text{Proj}_{x^{0\dots K}}\left(\mathcal{P}^{\text{SSTP}}_{\text{sdc2}}\right) = \Big\{x^{0\dots K} \Big| \exists y^{1\dots K} : (x^0, y^{1\dots K}) \in \mathcal{P}^{\text{SSTP}}_{\text{sdc2}}, \\ x^k_e = y^k_{ij} + y^k_{ji} - x^0_e, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E\Big\}$$

We start by comparing the undirected and the first semi-directed cut formulation. Not surprisingly, the additional directed parts of the formulation make it stronger.

Theorem 6.12. *$\mathcal{P}^{SSTP}_{uc} \supsetneq Proj_{x^{0\dots K}}\left(\mathcal{P}^{SSTP}_{sdc1}\right)$, i.e., the semi-directed cut-based formulation ($SSTP_{sdc1}$) is stronger than the undirected cut-based formulation ($SSTP_{uc}$).*

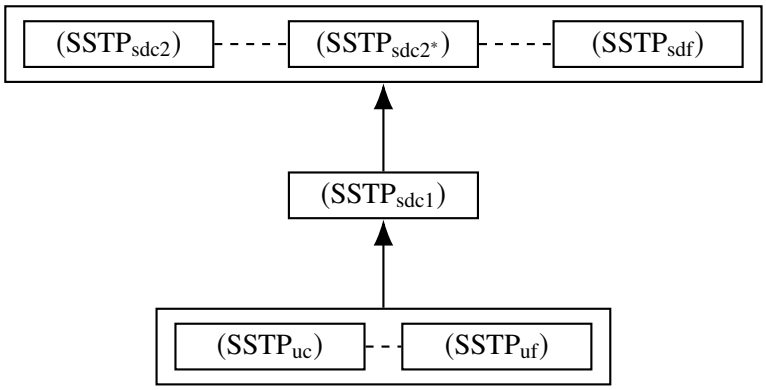


Figure 6.3: Hierarchy of undirected and semi-directed formulations for the SSTP. The dashed line and the additional clusters indicate that formulations are equivalent. An arrow indicates that the target cluster contains stronger formulations than the formulations in the source cluster.

Proof. Let $(\tilde{x}^0, \tilde{z}^{1\dots K}) \in \mathcal{P}^{\text{SSTP}}_{\text{sdc1}}$ and set $\hat{x}^0 := \tilde{x}^0, \hat{x}^k_e := \tilde{z}^k_{ij} + \tilde{z}^k_{ji}, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E$. We obtain a solution $\hat{x}^{0\dots K}$ for (SSTP$_{\text{uc}}$); its validity is discussed in the following.

Bounds of the first-stage variables $\hat{x}^0$ are obviously satisfied. Moreover, it clearly holds $\hat{x}^k_e \geq 0$ and due to constraints (6.41): $\hat{x}^k_e = \tilde{z}^k_{ij} + \tilde{z}^k_{ji} \leq 1$. Hence, $\hat{x}^{0\dots K} \in [0,1]^{|E|\cdot(K+1)}$.

We now show that the undirected cuts (6.1) are also satisfied by $\hat{x}^{0\dots K}$. Let $S \subseteq V$ represent a feasible cut set in scenario $k \in \mathcal{K}$, i.e., $\emptyset \neq S \cap T^k \neq T^k$. Since cuts in (SSTP$_{\text{sdc1}}$) are semi-directed and ingoing we assume w.l.o.g. that it holds $r^k \notin S$. Otherwise one can simply consider the complementary set $V\backslash S$, since $\delta(S) = \delta(V\backslash S)$ and then, it holds $r^k \notin (V\backslash S)$.

$$\begin{aligned}(\hat{x}^0 + \hat{x}^k)(\delta(S)) &= \sum_{e\in\delta(S)} \hat{x}^0_e + \hat{x}^k_e \\ &= \sum_{e\in\delta(S)} \tilde{x}^0_e + \sum_{\{i,j\}\in\delta(S)} \tilde{z}^k_{ij} + \tilde{z}^k_{ji} \\ &\geq \tilde{x}^0(\delta(S)) + \tilde{z}^k(\delta^-(S)) \geq 1\end{aligned}$$

The last inequality holds since $(\tilde{x}^0, \tilde{z}^{1\dots K})$ satisfies constraint (6.9) for cut set S.

Intuitively, the strict inequality of the formulations results from the directed arcs in the scenarios and the strength of the directed cut formulation for the deterministic STP. Figure 6.4 gives a small example with this property. Here, everything is purchased in the second stage and the semi-directed model gives a better lower bound. □

The following theorem shows that formulation (SSTP$_{\text{sdc2}}$) is stronger than formulation (SSTP$_{\text{sdc1}}$).

Theorem 6.13. *$Proj_{x^{0\dots K}}\left(\mathcal{P}^{SSTP}_{sdc1}\right) \supsetneq Proj_{x^{0\dots K}}\left(\mathcal{P}^{SSTP}_{sdc2}\right)$, i.e., the semi-directed cut-based formulation (SSTP$_{sdc2}$) is stronger than the semi-directed cut-based formulation (SSTP$_{sdc1}$).*

Proof. Let $(\tilde{x}^0, \tilde{y}^{1\dots K}) \in \mathcal{P}^{\text{SSTP}}_{\text{sdc2}}$ and set $\hat{x}^0_e := \tilde{x}^0_e, \forall e \in E$, and $\hat{x}^k_e := \tilde{y}^k_{ij} + \tilde{y}^k_{ji} - \tilde{x}^0_e, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E$. We argue that $\hat{x}^{0\dots K} \in \text{Proj}_{x^{0\dots K}}(\mathcal{P}^{\text{SSTP}}_{\text{sdc1}})$ by showing that there exists a variable assignment $\hat{z}^{1\dots K} \in [0,1]^{K\cdot|A|}$, with $\hat{z}^k_{ij} + \hat{z}^k_{ji} = \hat{x}^k_e, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E$, such that $(\hat{x}^0, \hat{z}^{1\dots K}) \in \mathcal{P}^{\text{SSTP}}_{\text{sdc1}}$.

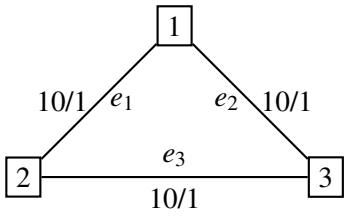


Figure 6.4: Example where the LP relaxation of ($\text{SSTP}_{\text{sdc1}}$) gives a better lower bound than (SSTP_{uc}). There is one scenario and all vertices are terminals. Edge costs for the first stage are all 10 and for the scenario 1. Both formulations purchase edges only in the second stage. The optimum solution to the undirected formulation has cost 1.5 with $x_e^1 = 0.5, \forall e \in E$. Since there is no valid orientation using 0.5 of each edge the semi-directed formulation selects two arcs in the second stage to connect the two remaining vertices to a root node leading to overall cost 2, e.g., for root node 1 set $z^1_{(1,2)} = z^1_{(1,3)} = 1$.

This solution is obtained by transforming $(\tilde{x}^0, \tilde{y}^{1\ldots K})$ into a feasible ($\text{SSTP}_{\text{sdc1}}$)-solution. Thereby, the parameter $\alpha_{ij}^k \in [0,1], \forall k \in \mathcal{K}, \forall (i,j) \in A$, is used:

$$\alpha_{ij}^k := \begin{cases} \frac{\tilde{y}_{ij}^k}{\tilde{y}_{ij}^k + \tilde{y}_{ji}^k} & \text{if } \tilde{y}_{ij}^k + \tilde{y}_{ji}^k > 0 \\ 0 & \text{otherwise} \end{cases}$$

This parameter allows us to split up the first-stage values among the two corresponding directed arcs, independently for each scenario. With α at hand the directed arc variables are set to $\hat{z}_{ij}^k := \tilde{y}_{ij}^k - \alpha_{ij}^k \tilde{x}_e^0, \forall k \in \mathcal{K}, \forall (i,j) \in A$, with $e = \{i,j\} \in E$.

First we show that this is a valid projection. Notice that $\forall k \in \mathcal{K}$, $\forall e = \{i,j\} \in E$: $\alpha_{ij}^k + \alpha_{ji}^k \in \{0,1\}$; if $\tilde{y}_{ij}^k + \tilde{y}_{ji}^k > 0$ this value is 1 and 0 otherwise. Now, consider edge $e = \{i,j\} \in E$ in scenario $k \in \mathcal{K}$ with $\tilde{y}_{ij}^k + \tilde{y}_{ji}^k > 0$. Then, $\hat{z}_{ij}^k + \hat{z}_{ji}^k = \tilde{y}_{ij}^k - \alpha_{ij}^k \tilde{x}_e^0 + \tilde{y}_{ji}^k - \alpha_{ji}^k \tilde{x}_e^0 = \tilde{y}_{ij}^k + \tilde{y}_{ji}^k - \tilde{x}_e^0$. In case $\alpha_{ij}^k = \alpha_{ji}^k = 0$, due to $\tilde{y}_{ij}^k + \tilde{y}_{ji}^k = 0$ and constraints (6.13), i.e., $y_{ij}^k + y_{ji}^k \geq x_e^0$, it follows $\tilde{x}_e^0 = 0$. Hence, it always holds $\hat{z}_{ij}^k + \hat{z}_{ji}^k = \tilde{y}_{ij}^k + \tilde{y}_{ji}^k - \tilde{x}_e^0 = \hat{x}_e^k, \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E$.

Now we are able to prove $\hat{x}^{0\ldots K} \in \text{Proj}_{x^{0\ldots K}}(\mathcal{P}^{\text{SSTP}}_{\text{sdc1}})$. Due to the preceding discussion it is clear that the subtour elimination constraints (6.41) are satisfied. Moreover, it obviously holds $\hat{x}_e^0 \in [0,1], \forall e \in E$.

Next, we consider the bounds for the arc variables $\hat{z}_{ij}^k, \forall k \in \mathcal{K}, \forall (i,j) \in A$. It holds $\hat{z}_{ij}^k \leq 1$ since $\hat{z}_{ij}^k \leq \tilde{y}_{ij}^k \leq 1$. Non-negativity can be seen by considering two cases. (i) If $\alpha_{ij}^k > 0$:

$$\hat{z}_{ij}^k = \tilde{y}_{ij}^k - \alpha_{ij}^k \tilde{x}_e^0 = \tilde{y}_{ij}^k - \tilde{x}_e^0 \frac{\tilde{y}_{ij}^k}{\tilde{y}_{ij}^k + \tilde{y}_{ji}^k} = \tilde{y}_{ij}^k \left(1 - \frac{\overbrace{\tilde{x}_e^0}^{\leq 1}}{\tilde{y}_{ij}^k + \tilde{y}_{ji}^k} \right) \geq 0$$

Inequality $\frac{\tilde{x}_e^0}{\tilde{y}_{ij}^k + \tilde{y}_{ji}^k} \leq 1$ is true due to capacity constraints (6.13). (ii) If $\alpha_{ij}^k = 0$ the non-negativity follows directly since $\hat{z}_{ij}^k = \tilde{y}_{ij}^k \geq 0$.

It remains to show that a valid cut $S \subseteq V_r$ in scenario $k \in \mathcal{K}$ is satisfied by $(\hat{x}^0, \hat{z}^{1\dots K})$:

$$\begin{aligned}(\hat{x}^0 + \hat{z}^k)(\delta^-(S)) = \sum_{(i,j)\in\delta^-(S)} \hat{x}^0_{\{i,j\}} + \hat{z}^k_{ij} &= \sum_{(i,j)\in\delta^-(S)} \tilde{x}^0_{\{i,j\}} + \tilde{y}^k_{ij} - \alpha^k_{ij}\tilde{x}^0_{\{i,j\}} \\ &= \sum_{(i,j)\in\delta^-(S)} (1-\alpha^k_{ij})\tilde{x}^0_{\{i,j\}} + \tilde{y}^k_{ij} \\ &\geq \sum_{(i,j)\in\delta^-(S)} \tilde{y}^k_{ij} \geq 1\end{aligned}$$

The last inequality is true due to the validity of solution $\tilde{y}^k$ for scenario k and constraints (6.12). This completes the "$\supseteq$"-part of the proof.

An example showing the strict inequality can be constructed by exploiting the different *meaning* of the first-stage variables. In formulation ($\text{SSTP}_{\text{sdc1}}$) a first stage edge $e = \{i, j\}$ contributes its value to cuts in both directions, i.e., $\delta^-(S)$ and $\delta^+(S)$. Contrarily, a feasible solution for formulation ($\text{SSTP}_{\text{sdc2}}$) has to find an orientation for this edge and distribute its value to the related arcs. In a sloppy way, the same edge has a lesser value in the second semi-directed formulation.

Hence, the same example from Figure 6.4 can be utilized to show the strict inequality; one simply has to set the edge cost to 1 for all first-stage and 10 for the scenario edges, respectively. There is still one scenario with all three vertices being terminals. Then, formulation ($\text{SSTP}_{\text{sdc1}}$) selects all three edges at 0.5 in the first stage satisfying all cuts in the scenario. On the other hand, this solution is not valid for ($\text{SSTP}_{\text{sdc2}}$) and there is none with overall cost 1.5. □

To complete the hierarchy of SSTP formulations given in Figure 6.3 it remains to show the equivalence of the semi-directed flow and cut-based formulations. To give the formal proof we denote the polytope of the relaxed flow formulation and the projection onto the same variable space as follows.

$$\begin{aligned}\mathcal{P}^{\text{SSTP}}_{\text{sdf}} = \Big\{(x^0, y^{1\dots K}, f) \in [0,1]^{|E|} \times [0,1]^{|A|\cdot K} \times [0,1]^{|A|\cdot t^*_r} \,\Big|\, \\ (x^0, y^{1\dots K}, f) \text{ satisfies (6.5), (6.16), and (6.17)}\Big\}\end{aligned}$$

$$\text{Proj}_{(x^0, y^{1\dots K})}\left(\mathcal{P}^{\text{SSTP}}_{\text{sdf}}\right) = \left\{(x^0, y^{1\dots K}) \,\middle|\, \exists f\colon (x^0, y^{1\dots K}, f) \in \mathcal{P}^{\text{SSTP}}_{\text{sdf}}\right\}$$

Lemma 6.14. *$\mathcal{P}^{SSTP}_{sdc2} = Proj_{(x^0,y^{1\dots K})}\left(\mathcal{P}^{SSTP}_{sdf}\right)$, i.e., the semi-directed cut- and flow-based formulations ($SSTP_{sdc2}$) and ($SSTP_{sdf}$) are equivalent.*

Proof. Restricting both models to one particular scenario, i.e., for one $k \in \mathcal{K}$ to y^k and (y^k, f^k), respectively, results in the related formulations (STP_{dc}) and (STP_{df}). Since the formulations for the deterministic STP are equivalent and the remaining parts of the stochastic models are identical the lemma follows. □

6.4.3 Directed formulations for the rSSTP

To make the comparison of the polytopes easier we add the following constraints to the directed cut formulations: ($\text{rSSTP}_{\text{dc1}}$) is expanded by both constraints and ($\text{rSSTP}_{\text{dc2}}$) by

the second type of constraints (6.44):

$$z_a^0 + z_a^k \leq 1 \qquad \forall k \in \mathcal{K}, \forall a \in A \tag{6.43}$$

$$z^0(\delta^-(v)) \leq 1 \qquad \forall v \in V_r \tag{6.44}$$

Both constraints (6.43) and (6.44) are obviously redundant since the right-hand side of the directed cuts is 1.

The considered polytopes of the relaxed formulations are denoted as follows.

$$\mathcal{P}_{\text{dc1}}^{\text{rSSTP}} = \left\{ z^{0\ldots K} \in [0,1]^{|A|\cdot(K+1)} \,\middle|\, z^{0\ldots K} \text{ satisfies (6.21), (6.22), (6.43), and (6.44)} \right\}$$

$$\mathcal{P}_{\text{dc2}}^{\text{rSSTP}} = \left\{ (z^0, y^{1\ldots K}) \in [0,1]^{|A|\cdot(K+1)} \,\middle|\, (z^0, y^{1\ldots K}) \text{ satisfies (6.25)–(6.27), and (6.44)} \right\}$$

We use a projection for the second formulation to compare both models:

$$\text{Proj}_{z^{0\ldots K}}\left(\mathcal{P}_{\text{dc2}}^{\text{rSSTP}}\right) = \left\{ (z^0, z^{1\ldots K}) \,\middle|\, (z^0, y^{1\ldots K}) \in \mathcal{P}_{\text{dc2}}^{\text{rSSTP}}, \; z_a^k = y_a^k - z_a^0, \forall k \in \mathcal{K}, \forall a \in A \right\}$$

Theorem 6.15. *$\mathcal{P}_{dc1}^{rSSTP} = Proj_{z^{0\ldots K}}(\mathcal{P}_{dc2}^{rSSTP})$, i.e., the directed cut-based formulations ($rSSTP_{dc1}$) and ($rSSTP_{dc2}$) are equivalent.*

Proof. "$\subseteq$": Let $\tilde{z}^{0\ldots K} \in \mathcal{P}_{\text{dc1}}^{\text{rSSTP}}$. We show that $(\hat{z}^0, \hat{y}^{1\ldots K}) \in \mathcal{P}_{\text{dc2}}^{\text{rSSTP}}$ with $\hat{z}^0 := \tilde{z}^0, \hat{y}^k := \tilde{z}^k + \tilde{z}^0, \forall k \in \mathcal{K}$.

First, we consider the variable bounds. Since $\hat{z}^0 = \tilde{z}^0$ we have $\hat{z}^0 \in [0,1]^{|A|}$. Moreover, $\hat{y}^k$ is obviously non-negative and due to (6.43): $\hat{y}^k = \tilde{z}^k + \tilde{z}^0 \leq \mathbf{1}, \forall k \in \mathcal{K}$.

Second, the directed cuts in the first stage, i.e., constraints (6.25), and constraints (6.44), are identical in both formulations and hence, they are satisfied. This is also true for the capacity constraints (6.27) since $\hat{y}_a^k = \tilde{z}_a^k + \tilde{z}_a^0 \geq \hat{z}_a^0, \forall k \in \mathcal{K}, \forall a \in A$.

Third, consider a valid cut set $S \subseteq V_r$ in scenario $k \in \mathcal{K}$. Since $\tilde{z}^{0\ldots K}$ is a valid solution for ($\text{rSSTP}_{\text{dc1}}$) it satisfies the directed cuts (6.22) which leads to $\hat{y}^k(\delta^-(S)) = (\tilde{z}^k + \tilde{z}^0)(\delta^-(S)) \geq 1$. Hence, the directed cuts (6.26) are satisfied by $\hat{y}^{1\ldots K}$.

"$\supseteq$": The opposite direction is similar. Let $(\tilde{z}^0, \tilde{y}^{1\ldots K}) \in \mathcal{P}_{\text{dc2}}^{\text{rSSTP}}$. We set $\hat{z}^0 := \tilde{z}^0, \hat{z}^k := \tilde{y}^k - \tilde{z}^0, \forall k \in \mathcal{K}$, such that $\hat{z}^{0\ldots K} \in \mathcal{P}_{\text{dc1}}^{\text{rSSTP}}$.

Again, directed cuts in the first stage are obviously satisfied and the variable bounds trivially hold for the first-stage variables. For the second-stage variables we have $\hat{z}^k \leq \tilde{y}^k \leq \mathbf{1}$ and $\hat{z}^k = \tilde{y}^k - \tilde{z}^0 \geq \mathbf{0}, \forall k \in \mathcal{K}$, due to constraints (6.27).

The added constraints (6.43) are satisfied since $\hat{z}_a^0 + \hat{z}_a^k = \tilde{z}_a^0 + \tilde{y}_a^k - \tilde{z}_a^0 = \tilde{y}_a^k \leq 1, \forall a \in A, \forall k \in \mathcal{K}$, and last but not least, a valid cut set $S \subseteq V_r$ in scenario $k \in \mathcal{K}$ is satisfied since $(\hat{z}^0 + \hat{z}^k)(\delta^-(S)) = (\tilde{z}^0 + \tilde{y}^k - \tilde{z}^0)(\delta^-(S)) = \tilde{y}^k(\delta^-(S)) \geq 1$. □

We close the discussion by comparing the directed flow formulation (rSSTP_{df}) to the second directed cut formulation ($\text{rSSTP}_{\text{dc2}}$).

$$\mathcal{P}_{\text{df}}^{\text{rSSTP}} = \left\{ (z^0, y^{1\ldots K}, w^0, f) \in [0,1]^{|A|\cdot(K+1)} \times [0,1]^{|V_r|} \times [0,1]^{|A|\cdot(|V_r|+t_r^*)} \,\middle|\, (z^0, y^{1\ldots K}, w^0, f) \text{ satisfies (6.35)–(6.40)} \right\}$$

$$\text{Proj}_{(z^0, y^{1\ldots K})}\left(\mathcal{P}_{\text{df}}^{\text{rSSTP}}\right) = \left\{ (z^0, y^{1\ldots K}) \,\middle|\, \exists (w^0, f) \colon (z^0, y^{1\ldots K}, w^0, f) \in \mathcal{P}_{\text{df}}^{\text{rSSTP}} \right\}$$

Theorem 6.16. *$\mathcal{P}^{rSSTP}_{dc2} = Proj_{(z^0,y^{1\dots K})}\left(\mathcal{P}^{rSSTP}_{df}\right)$, i.e., the directed cut- and flow-based formulations ($rSSTP_{dc2}$) and ($rSSTP_{df}$) are equivalent.*

Proof. "$\subseteq$": Let $(\tilde{z}^0, \tilde{y}^{1\dots K}) \in \mathcal{P}^{\text{rSSTP}}_{\text{dc2}}$. We use $(\hat{z}^0, \hat{y}^{1\dots K}) := (\tilde{z}^0, \tilde{y}^{1\dots K})$ to construct a solution $(\hat{z}^0, \hat{y}^{1\dots K}, \hat{w}^0, \hat{f}) \in \mathcal{P}^{\text{rSSTP}}_{\text{df}}$. First, constraints (6.38) are contained in both models and hence satisfied for $(\hat{z}^0, \hat{y}^{1\dots K})$. Second, since w^0 gives the connected vertices in the first stage we set $\hat{w}^0_v := \tilde{z}^0(\delta^-(v)), \forall v \in V_r$; due to (6.44) we have $\tilde{z}^0(\delta^-(v)) \in [0,1]$. Hence, bounds on $\hat{w}^0$ are satisfied and moreover, (6.36) is satisfied with equality. Third, the remaining part of formulation (rSSTP_{df}) contains the construction of the flows: we set the flow variables $\hat{f}$ such that in the first stage a flow with value $\hat{w}^0_v$ is send from the root to every vertex and in every scenario a flow of value 1 from the root to every terminal. The feasibility and correctness follows again from "max flow = min cut".

"$\supseteq$": Let $(\tilde{z}^0, \tilde{y}^{1\dots K}, \tilde{w}^0, \tilde{f}) \in \mathcal{P}^{\text{rSSTP}}_{\text{df}}$. Again, set $(\hat{z}^0, \hat{y}^{1\dots K}) := (\tilde{z}^0, \tilde{y}^{1\dots K})$. First, (6.44) is satisfied for all vertices $v \in V_r$ since $\hat{z}^0(\delta^-(v)) \leq \tilde{w}^0_v$ due to (6.36). Second, due to (6.35)–(6.37) there is a flow with value $\tilde{w}^0_v$ in the first stage from the root to a vertex $v \in V_r$ with $\tilde{w}^0_v > 0$ and moreover, arcs used for routing flow are selected by $\tilde{z}^0$ through (6.35). Hence, again due to "max flow = min cut", the directed cuts (6.25) are satisfied for $\hat{z}^0$ for all $v \in V_r$. The same holds for the directed cuts (6.26) in the scenarios and variables $\hat{y}^k, \forall k \in \mathcal{K}$. Last but not least, (6.27) is satisfied since the constraints are contained in both models. □

6.5 The Steiner forest problem

Since the Steiner forest problem can be directly solved by the stochastic Steiner tree problem, compare Section 5.3, Theorem 5.5, the semi-directed formulation for the SSTP can be utilized for modeling the Steiner forest problem. In this section we first introduce semi-directed cut- and flow-based formulations for the SFP, we show their equivalence, and we prove that these models are stronger than the undirected formulations. Afterwards, we present a directed flow-based model with additional variables for modeling a hierarchy on the terminal sets. We give a proof that this model is stronger than the semi-directed models. Last but not least, we develop an equivalent cut-based formulation whose existence was stated an open problem in the literature.

Semi-directed formulations. Our semi-directed formulation is based on the idea of reducing the Steiner forest problem to the stochastic Steiner tree problem as in Theorem 5.5. We use this reduction for describing the idea of the new model. Moreover, we use the same terminology from the SSTP and consider two stages for constructing the solution. In a "first stage" undirected edges are selected which need to be payed for. In a "second stage" this set of undirected edges can be used to find a feasible arborescence for each terminal set: each arborescence is an orientation of (and in particular, restricted to) the edge set of the "first stage". Hence, although the connectivity of a terminal set is primarily ensured by the arborescence the "first-stage" edges already connect all terminal sets.

We use edge variables $x_e, \forall e \in E$, for representing the solution and $y^k_a, \forall a \in A$, for the kth arborescence and terminal set $T^k, k \in \mathcal{K} = \{1, \dots, K\}$. Moreover, an arbitrary terminal $r^k \in T^k$ is set as root node for terminal set $k, \forall k \in \mathcal{K}$. Again, let $V^k_r := V \backslash \{r^k\}$ and

$T_r^k := T^k \setminus \{r^k\}, \forall k \in \mathcal{K}$. The semi-directed cut formulation for the SFP reads as follows.

$$\begin{aligned}
(\text{SFP}_{\text{sdc}}) \ \min & \sum_{e \in E} c_e x_e & & \\
\text{s.t. } x_e &\geq y_{ij}^k + y_{ji}^k & \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E & \quad (6.45)\\
y^k(\delta^-(S)) &\geq 1 & \forall k \in \mathcal{K}, \forall S \subseteq V_r^k : S \cap T_r^k \neq \emptyset & \quad (6.46)\\
x &\in \{0,1\}^{|E|} & & \quad (6.47)\\
y^{1\ldots K} &\in \{0,1\}^{|A| \cdot K} & & \quad (6.48)
\end{aligned}$$

Although it is possible to transform an SFP instance into an SSTP instance (Theorem 5.5) and directly use the semi-directed formulation ($\text{SSTP}_{\text{sdc2}}$)—which is stronger than the undirected model—we prefer this (cleaner) formulation instead which differs slightly from the SSTP formulation. Basically, the capacity constraints (6.45) work in the opposite direction: here, only *relevant* selected "first-stage" edges have to be oriented—an edge is *relevant* if its removal disconnects T^k, otherwise it is *irrelevant*. Moreover, only selected "first-stage" edges can be used for building valid arborescences.

Lemma 6.17. *Formulation (SFP_{sdc}) models the Steiner forest problem correctly.*

Proof. Let $\tilde{E} \subseteq E$ be an optimum solution to an SFP instance. Since any terminal set $T^k, k \in \mathcal{K}$, is connected by $\tilde{E}$ it is possible to construct an r^k-rooted arborescence containing all vertices T^k by using only edges in $\tilde{E}$, for each set $k \in \mathcal{K}$. Thereby, all irrelevant edges can be ignored. Orienting the edges $\tilde{E}$ in this way induces values for variables $y^{1\ldots K}$. By setting x to the incidence vector representing $\tilde{E}$ we obtain a solution which is valid for (SFP_{sdc}) and has the same objective value.

On the other hand, an optimum solution to (SFP_{sdc}) implies that each designated root node connects its related terminal set by directed paths (constraints (6.46)). The constraints (6.45) enforce that an edge is payed for whenever a related arc is used. Hence, the edge set $\tilde{E} = \{e \in E \mid x_e = 1\}$ gives a feasible solution to the Steiner forest problem with the same objective value. □

Remark 6.18. This model and all following SFP models can be expanded and strengthened by flow-balance constraints, as for the STP. For (SFP_{sdc}) the flow-balance constraints read:

$$y^k(\delta^+(v)) \geq y^k(\delta^-(v)) \qquad \forall k \in \mathcal{K}, \forall v \in V \setminus T^k \qquad \text{(Fb)}$$

Polzin and Daneshmand [142] mentioned that constraints (Fb) strengthen the directed STP models (STP_{dc}) and (STP_{df}). Unfortunately, the presented instance, which should show the strict inequality, works only for the directed Steiner tree problem (where the input graph does not need to be a bidirection). However, an instance can be found in the article by Ljubić, Weiskircher, Pferschy, Klau, Mutzel, and Fischetti [125] (cf. Figure 3).

Although these constraints make the formulations stronger, we consider all following SFP models without adding constraints (Fb) since they obviously do not influence the relative strength and the relationships of the SFP models.

Let (SFP_{uc}) denote the undirected cut-based formulation for the SFP, i.e., formulation (NDP_{uc}) from Section 3.2 with connectivity function (3.3) for the right-hand side of the undirected cuts.

The polytopes of the LP-relaxations of the SFP formulations are defined as follows:

$$\mathcal{P}^{\text{SFP}}_{\text{uc}} = \left\{ x \in [0,1]^{|E|} \,\middle|\, x \text{ satisfies (3.1) with connectivity function (3.3)} \right\}$$
$$\mathcal{P}^{\text{SFP}}_{\text{sdc}} = \left\{ (x, y^{1\dots K}) \in [0,1]^{|E|} \times [0,1]^{|A|\cdot K} \,\middle|\, (x, y^{1\dots K}) \text{ satisfies (6.45) and (6.46)} \right\}$$

To compare the formulations we consider the linear projection of the semi-directed cut formulation onto the space of undirected x variables, denoted by $\text{Proj}_x\left(\mathcal{P}^{\text{SFP}}_{\text{sdc}}\right)$.

Lemma 6.19. *$\mathcal{P}^{SFP}_{uc} \supsetneq Proj_x(\mathcal{P}^{SFP}_{sdc})$, i.e., the semi-directed cut-based formulation (SFP_{sdc}) is stronger than the undirected cut-based formulation (SFP_{uc}).*

Proof. The arguments are the same as in the proof of Lemma 6.17: any solution $(\tilde{x}, \tilde{y}^{1\dots K}) \in \mathcal{P}^{\text{SFP}}_{\text{sdc}}$ contains feasible arborescences for all terminal sets due to constraints (6.46). These arborescences only use selected x variables (constraints (6.45)). Hence, dropping the y variables leaves a feasible solution to formulation (SFP_{uc}) with the same objective value.

The strict inequality follows from any example showing that the directed cut formulation of the STP is stronger than the undirected cut formulation, cf. e.g., Figure 3.1. When interpreted as SFP instance there is one terminal set and (SFP_{sdc}) is identical to (STP_{dc}) whereas (SFP_{uc}) is identical to (STP_{uc}). □

Next, we introduce an equivalent flow-based formulation. Thereby, we again use undirected edge variables which can be used for routing flow for every terminal set.

$$(\text{SFP}_{\text{sdf}}) \quad \min \sum_{e \in E} c_e x_e$$
$$\text{s.t. } x_e \geq f^{k,t_1}_{ij} + f^{k,t_2}_{ji} \qquad \begin{array}{l} \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E, \\ \forall t_1, t_2 \in T^k_r \end{array} \tag{6.49}$$
$$f^{k,t}(\delta^-(i)) - f^{k,t}(\delta^+(i)) = \begin{cases} -1, & \text{if } i = r^k \\ 1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \qquad \begin{array}{l} \forall k \in \mathcal{K}, \forall t \in T^k_r, \\ \forall i \in V \end{array} \tag{6.50}$$
$$f \in [0,1]^{|A|\cdot t^*_r} \tag{6.51}$$
$$x \in \{0,1\}^{|E|} \tag{6.52}$$

Restricting this model to a terminal set basically results in the flow-based Steiner tree model (STP_{df2}). Hence, formulation (SFP_{sdf}) models the Steiner forest problem correctly.

The polytope of this formulation is given as follows.

$$\mathcal{P}^{\text{SFP}}_{\text{sdf}} = \left\{ (f,x) \in [0,1]^{|A|\cdot t^*_r} \times [0,1]^{|E|} \,\middle|\, (f,x) \text{ satisfies (6.49), (6.50)} \right\}$$

Let $\text{Proj}_x\left(\mathcal{P}^{\text{SFP}}_{\text{sdf}}\right)$ denote the linear projection of $\mathcal{P}^{\text{SFP}}_{\text{sdf}}$ into the x variable space.

Observation 6.20. *$Proj_x\left(\mathcal{P}^{SFP}_{sdc}\right) = Proj_x\left(\mathcal{P}^{SFP}_{sdf}\right)$, i.e., the semi-directed cut- and flow-based formulations (SFP_{sdc}) and (SFP_{sdf}) are equivalent.*

Proof. Restricting both formulations to one $k \in \mathcal{K}$ results in models (STP_{dc}) and (STP_{df2}), respectively; Lemma 3.9 states the equivalence of these two models. □

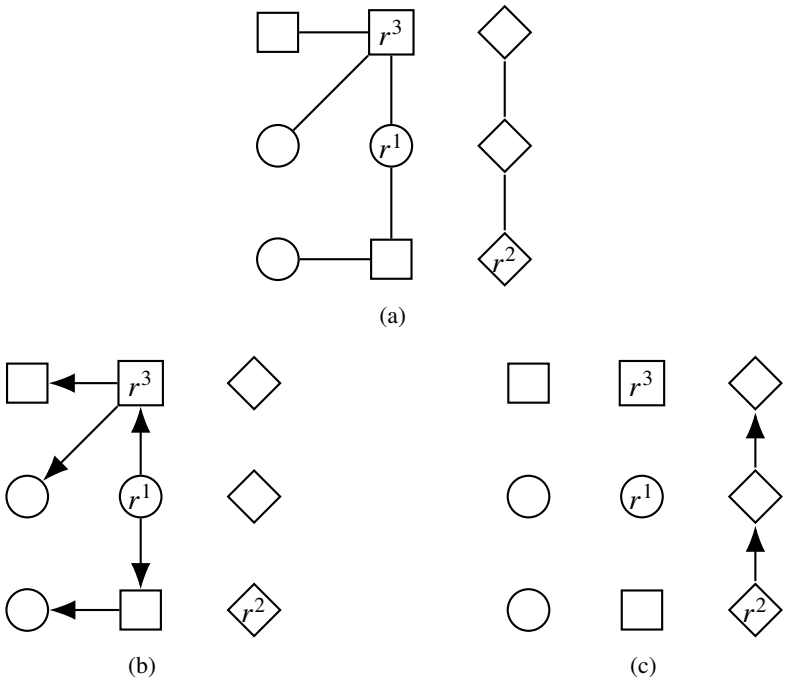


Figure 6.5: (a) Optimum solution to an SFP instance with 3 terminal set: circles (set 1), diamonds (set 2), and rectangles (set 3), and their designated root nodes r^1, r^2, r^3. (b) and (c) show optimal arborescences for "scenario" 1 and "scenario" 2, respectively, w.r.t. the following α values of an optimal solution to ($\text{SFP}_{\alpha:\text{dc}}$): $\alpha_{11} = 1, \alpha_{22} = 1, \alpha_{13} = 1$. All remaining α variables are 0 and moreover, "scenario" 3 does not use any arcs.

Notice that—analogously to the STP—it is possible to describe an equivalent flow-based formulation by using arc variables $z^k_{ij}, z^k_{ji}, \forall\{i,j\} \in E, \forall k \in \mathcal{K}$, with constraints $z^k_{ij} \geq f^{k,t}_{ij}, \forall k \in \mathcal{K}, \forall t \in T^k_r, \forall (i,j) \in A$, and $x_e \geq z^k_{ij} + z^k_{ji}, \forall e = \{i,j\} \in E, \forall k \in \mathcal{K}$, which replace constraints (6.49).

Directed formulations. A long-standing open problem is the existence of a directed cut-based formulation for the Steiner forest problem. Magnanti and Raghavan [130] remark that "there does not seem to be a straightforward way to formulate a directed cut model". We (almost) close this gap by presenting an expanded directed cut-based model in the following. Again, we assign an arbitrary terminal $r^k \in T^k$ as designated root node for terminal set $k, \forall k \in \mathcal{K}$.

Notice that if it is known which terminal sets are interconnected it is easy to find feasible arborescences which connect all terminal sets by constructing disjoint Steiner trees. Figure 6.5 illustrates a small example with a solution given by Figure 6.5 (a). Here, sets 1 and 3 are connected whereas set 2 is contained in another connected component. By using the root nodes with minimal indices the connected components can be oriented and we obtain the disjoint directed trees depicted in Figure 6.5 (b) and (c), respectively.

There are two key ideas to our new formulation. First, we model a hierarchy of the terminal sets and their designated root nodes, respectively, which indicates which sets are connected. Second, for each terminal set $k \in \mathcal{K}$ we introduce a "scenario" in which r^k is a root node of an arborescence. In the following we describe both ideas in more detail.

To model the hierarchy, we introduce additional variables $\alpha_{ij} \in \{0,1\}$ for each pair of terminal sets $i, j \in \mathcal{K}$, with $i \leq j$. Thereby, $\alpha_{ij} = 1$ if r^j (and all vertices in T^j) are contained in the arborescence rooted at r^i; $\alpha_{ii} = 1$ implies that r^i is a root node of an arborescence itself, cf. Figure 6.5. We call r^i with $\alpha_{ii} = 1$ a *parent node* and say that r^i is the *parent of* r^j if $\alpha_{ij} = 1$ and contrarily, r^j is the *child of* r^i; the same notation is used for the sets. Each root node is either a parent or a child node and r^1 is always a parent node—we set $\alpha_{11} = 1$ (this is implied in the model by constraints (6.54)).

Now, we introduce a "scenario" for each terminal set such that the kth "scenario" contains exactly those edges of the tree rooted at r^k. Again, consider Figure 6.5: "scenario" 1 contains the tree (arborescence) rooted at r^1 which connects T^1 and T^3 as shown by Figure 6.5 (b). Moreover, "scenario" 3 is empty since T^3 is connected to r^1 in "scenario" 1 and the solution of "scenario" 2 is shown in Figure 6.5 (c).

Although we mainly focus on cut-based formulations in this thesis we first introduce the flow-based formulation here since the comparison of the formulations seems to be more intuitive for flow-based models. Afterwards, we present an equivalent cut-based formulation.

We need some additional notations. Let $T^{k\dots K} := \bigcup_{k \leq \ell \leq K} T^\ell, \forall k \in \mathcal{K}$, denote the union of terminal sets $k, \dots, K$, and let $T_r^{k\dots K} := T^{k\dots K} \backslash \{r^k\}$ be the same set of terminals without the kth root node. In the following model we have a flow from root node r^k in scenario $k \in \mathcal{K}$ to each terminal $t \in T_r^{k\dots K}$. Therefore, the overall number of commodities is $|A| \cdot \sum_{k \in \mathcal{K}} T_r^{k\dots K} =: F$. Last but not least, set(t) gives the index of the set containing a terminal $t \in T^{1\dots K}$, i.e., set$(t) = i \Leftrightarrow t \in T^i$.

The directed flow-based formulation reads as follows.

$$
\begin{aligned}
(\mathrm{SFP}_{\alpha:\mathrm{df}}) \ \min \ & \sum_{e \in E} c_e x_e & & \\
\text{s.t. } x_e \geq & \sum_{k \in \mathcal{K}} x_e^k & \forall e \in E & \quad (6.53)\\
\sum_{1 \leq j \leq k} & \alpha_{jk} = 1 & \forall k \in \mathcal{K} & \quad (6.54)\\
\alpha_{kk} \geq & \alpha_{k\ell} & \forall k \in \mathcal{K}\backslash\{1, K\}, \ \forall \ell \in \{k+1, \dots, K\} & \quad (6.55)\\
x_e^k \geq & f_{ij}^{k,t_1} + f_{ji}^{k,t_2} & \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E, \ \forall t_1, t_2 \in T_r^{k\dots K} & \quad (6.56)\\
f^{k,t}(\delta^-(i)) - f^{k,t}(\delta^+(i)) = & \begin{cases} -\alpha_{k\ell}, & \text{if } i = r^k \\ \alpha_{k\ell}, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} & \forall k \in \mathcal{K}, \forall t \in T_r^{k\dots K}, \text{ with set}(t) = \ell, \ \forall i \in V & \quad (6.57)\\
f \in & [0,1]^{|A| \cdot F} & & \quad (6.58)\\
\alpha \in & \{0,1\}^{0.5 \cdot K \cdot (K+1)} & & \quad (6.59)\\
x \in & \{0,1\}^{|E|} & & \quad (6.60)\\
x^{1\dots K} \in & \{0,1\}^{|E| \cdot K} & & \quad (6.61)
\end{aligned}
$$

(6.54) and (6.55) imply a valid assignment of the α variables by modeling a flat hierarchy between the sets: a root node is either a parent or it is a child of exactly one other root node. (6.54) states that every root r^k, $k \in \mathcal{K}$, has to be a parent ($\alpha_{kk} = 1$), or it has to be a child

and it has to be contained in another arborescence ($\exists j < k \colon \alpha_{jk} = 1$). (6.55) states that if a root node r^ℓ is a child of another root node r^k then r^k has to be a parent node.

Constraints (6.56) and (6.57) model a valid flow: a flow of value $\alpha_{k\ell}$ is send from a root node r^k to a terminal $t \in T_r^{k\ldots K}$ with $\text{set}(t) = \ell$. Hence, if r^k is the parent of set ℓ ($\ell \geq k$) each terminal in T^ℓ is reachable from r^k.

The constraints (6.56) ensure the correct assignment of flow- and edge variables, similar to the Steiner tree problem and formulation (STP_{df2}). Here, the constraint affects each "scenario" separately. The fact that an optimum solution to the SFP consists of a disjoint set of trees—hence, all "scenario" solutions are disjoint—is represented by the sum in constraints (6.53): any edge used in any "scenario" needs to be payed for.

Lemma 6.21. *($SFP_{\alpha:df}$) models the Steiner forest problem correctly.*

Proof. Let $\tilde{E} \subseteq E$ be an optimal solution to the SFP. We construct a solution to ($\text{SFP}_{\alpha:\text{df}}$) and start with $\tilde{x}$ being the incidence vector of $\tilde{E}$ and with $\tilde{\alpha} := \mathbf{0}$. Now, for each connected component C in $G[\tilde{E}]$ set $\tilde{\alpha}_{ii} = 1$ if r^i is the root node with lowest index contained in C and for all other root nodes $r^j \in C, j > i$, set $\tilde{\alpha}_{ij} = 1$. Notice that $\tilde{\alpha}$ satisfies (6.54) and (6.55) and that each terminal (set) has exactly one assigned parent node. After fixing the α variables the remaining part of the model describes a union of disjoint Steiner trees, one for each connected component. First, a connected component with parent node $r^k, k \in \mathcal{K}$, is represented by edge variables $\tilde{x}^k$. Second, $\tilde{E}$ can be oriented such that each connected component is an arborescence rooted at its parent node. Then, the arcs of the arborescences can be used for constructing flows from each parent node r^k to each terminal $t \in T_r^k$ or $t \in T^\ell$ with $\ell > k$ and $\alpha_{k\ell} = 1$. Since the connected components are disjoint constraint (6.53) is satisfied. Overall, the constructed solution is feasible for ($\text{SFP}_{\alpha:\text{df}}$) and has the same objective value.

An optimum solution $(\tilde{f}, \tilde{\alpha}, \tilde{x}, \tilde{x}^{1\ldots K})$ to ($\text{SFP}_{\alpha:\text{df}}$) implies a hierarchy of the terminal sets with parent and child sets. Thereby, every set has exactly one assigned parent; in particular, every terminal of a set has the same parent. Hence, due to (6.57) there exists a flow of one unit from each terminal to the assigned parent such that every terminal is connected. Constraints (6.56) and (6.53) collect the used edges, ensure the trees being disjoint, and hence, $\tilde{E} := \{e \in E \mid \tilde{x}_e = 1\}$ is a feasible solution to the SFP with the same cost. □

The polytope of the flow-based model ($\text{SFP}_{\alpha:\text{df}}$) is defined as follows.

$$\mathcal{P}^{\text{SFP}}_{\alpha:\text{df}} = \Big\{(f, \alpha, x, x^{1\ldots K}) \in [0,1]^{|A|\cdot F} \times [0,1]^{0.5\cdot K\cdot(K+1)} \times [0,1]^{|E|} \times [0,1]^{|E|\cdot K} \Big| \\ (f, \alpha, x, x^{1\ldots K}) \text{ satisfies (6.53)–(6.57)}\Big\}$$

Let $\text{Proj}_x\left(\mathcal{P}^{\text{SFP}}_{\alpha:\text{df}}\right)$ denote the linear projection of $\mathcal{P}^{\text{SFP}}_{\alpha:\text{df}}$ into the x variable space.

Lemma 6.22. *$Proj_x(\mathcal{P}^{SFP}_{sdf}) \supsetneq Proj_x(\mathcal{P}^{SFP}_{\alpha:df})$, i.e., the directed flow formulation ($SFP_{\alpha:df}$) is stronger than the semi-directed flow formulation (SFP_{sdf}) .*

Proof. Let $(\tilde{f}, \tilde{\alpha}, \tilde{x}, \tilde{x}^{1\ldots K}) \in \mathcal{P}^{\text{SFP}}_{\alpha:\text{df}}$. For better overview we divide the proof into several parts. Parts (A)–(D) show that $\text{Proj}_x(\mathcal{P}^{\text{SFP}}_{\text{sdf}}) \supseteq \text{Proj}_x(\mathcal{P}^{\text{SFP}}_{\alpha:\text{df}})$ and (E) gives an example where the strict inequality holds.

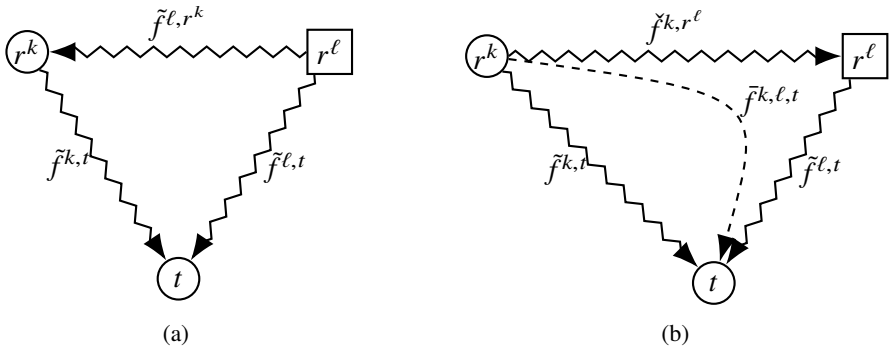


Figure 6.6: Schematic view on the involved flows in the proof of Lemma 6.22. r^k and r^ℓ are root nodes for sets T^k and T^ℓ, respectively, with $\ell < k$, and $t \in T^k_r$. (a) The original flows. (b) The reverse flow $\check{f}^{k,r^\ell}$ from r^k to r^ℓ, cf. part (B) of the proof, and the combined flow $\bar{f}^{k,\ell,t}$ from r^k to t over r^ℓ, cf. part (C).

(A) Flows are acyclic. We assume that any flow $\tilde{f}^{k,t}, \forall k \in \mathcal{K}, \forall t \in T^{k\dots K}_r$, is acyclic and satisfies $\tilde{f}^{k,t}_{ij} = 0 \vee \tilde{f}^{k,t}_{ji} = 0, \forall \{i,j\} \in E$. Otherwise it is valid to modify the flow $\tilde{f}$ such that the assumption is satisfied. Let $a_1 \in \{(i,j),(j,i)\}$ and let a_2 be the reverse arc, denoted by $\mathrm{rev}(a_1)$. Now, if (w.l.o.g.) $\tilde{f}^{k,t}_{a_1} \geq \tilde{f}^{k,t}_{a_2} > 0$ we can set $\tilde{f}^{k,t}_{a_1} := \tilde{f}^{k,t}_{a_1} - \tilde{f}^{k,t}_{a_2}$ and $\tilde{f}^{k,t}_{a_2} := 0$. Afterwards, $\tilde{f}$ is still a valid flow (both $\tilde{f}(\delta^-(\cdot))$ and $\tilde{f}(\delta^+(\cdot))$ decrease by $\tilde{f}^{k,t}_{a_2}$ for i and j) with the same value and all constraints in ($\mathrm{SFP}_{\alpha:\mathrm{df}}$) are still satisfied.

(B) Reverse flow. We introduce additional flow variables $\check{f}^{k,r^\ell}, \forall \ell \in \{1,\dots,K-1\}, \forall k \in \{\ell+1,\dots,K\}$, i.e., with $k > \ell$. Notice that these flow variables do not exist since we have only flow variables $f^{k,t}$ for a set k and terminal $t \in T^{k\dots K}_r$, i.e., $k \leq \mathrm{set}(t)$. The values of the new variables are set such that the flow from r^ℓ to r^k is reversed, cf. Figure 6.6:

$$\forall a \in A, \forall \ell \in \{1,\dots,K-1\}, \forall k \in \{\ell+1,\dots,K\}\colon \check{f}^{k,r^\ell}_a := \tilde{f}^{\ell,r^k}_{\mathrm{rev}(a)}$$

(C) Flow from r^k to t *over* r^ℓ. Now, we construct a flow $\bar{f}^{k,\ell,t}$ for a set $k \in \mathcal{K}\backslash\{1\}$, a terminal $t \in T^k_r$, and a set $\ell \in \{1,\dots,k-1\}$. This flow will send $\tilde{\alpha}_{\ell k}$ from r^k to t (over r^ℓ) by using the reverse flow from r^ℓ to r^k (cf. Figure 6.6):

$$\bar{f}^{k,\ell,t} := \tilde{f}^{\ell,t} + \check{f}^{k,r^\ell}$$

(C.1) Feasibility and value. We show that $\bar{f}^{k,\ell,t}$ is a feasible flow from r^k to t with value $\tilde{\alpha}_{\ell k}, \forall k \in \mathcal{K}\backslash\{1\}, \forall t \in T^k_r, \forall \ell \in \{1,\dots,k-1\}$. Let $i \in V$. We have:

$$\begin{aligned}&\bar{f}^{k,\ell,t}(\delta^-(i)) - \bar{f}^{k,\ell,t}(\delta^+(i))\\ &= \tilde{f}^{\ell,t}(\delta^-(i)) + \check{f}^{k,r^\ell}(\delta^-(i)) - \tilde{f}^{\ell,t}(\delta^+(i)) - \check{f}^{k,r^\ell}(\delta^+(i))\end{aligned}$$

Case $i = r^k$: $\tilde{f}^{\ell,t}(\delta^-(r^k)) - \tilde{f}^{\ell,t}(\delta^+(r^k)) = 0$ since r^k is an internal node under flow $\tilde{f}^{\ell,t}$. Moreover, $\check{f}^{k,r^\ell}(\delta^-(r^k)) - \check{f}^{k,r^\ell}(\delta^+(r^k)) = -\tilde{\alpha}_{\ell k}$ (the reverse flow).

Case $i = t$: Similar arguments: $\check{f}^{k,r^\ell}(\delta^-(t)) - \check{f}^{k,r^\ell}(\delta^+(t)) = 0$ since t is an internal node under $\check{f}^{k,r^\ell}$ and $\tilde{f}^{\ell,t}(\delta^-(t)) - \tilde{f}^{\ell,t}(\delta^+(t)) = \tilde{\alpha}_{\ell k}$.

Case $i = r^\ell$: $\tilde{f}^{\ell,t}(\delta^-(r^\ell)) - \tilde{f}^{\ell,t}(\delta^+(r^\ell)) = -\tilde{\alpha}_{\ell k}$ and $\check{f}^{k,r^\ell}(\delta^-(r^\ell)) - \check{f}^{k,r^\ell}(\delta^+(r^\ell)) = \tilde{\alpha}_{\ell k}$. Hence, the sum is 0.

Otherwise : Since $\tilde{f}^{\ell,t}$ and $\check{f}^{k,r^\ell}$ are flows the sum is 0.

Hence, $\bar{f}^{k,\ell,t}$ is a feasible flow from r^k to t with value $\tilde{\alpha}_{\ell k}$.

(C.2) Acyclic $\bar{f}^{k,\ell,t}$. Again, we assume that $\bar{f}$ is acyclic. Otherwise, we modify the flow similar to before. Consider an edge $\{i,j\} \in E$. Let $a_1 \in \{(i,j),(j,i)\}$ with $a_2 = \text{rev}(a_1)$ and with $\bar{f}^{k,\ell,t}_{a_1} \geq \bar{f}^{k,\ell,t}_{a_2} > 0$. Then, we set $\bar{f}^{k,\ell,t}_{a_1} := \bar{f}^{k,\ell,t}_{a_1} - \bar{f}^{k,\ell,t}_{a_2} = \tilde{f}^{\ell,t}_{a_1} + \check{f}^{k,r^\ell}_{a_1} - \tilde{f}^{\ell,t}_{a_2} - \check{f}^{k,r^\ell}_{a_2}$ and $\bar{f}^{k,\ell,t}_{a_2} := 0$. Notice that for any arc a_1, with $a_2 = \text{rev}(a_1)$, it holds $\bar{f}^{k,\ell,t}_{a_1} \leq \max\{0, \tilde{f}^{\ell,t}_{a_1} + \check{f}^{k,r^\ell}_{a_1} - \tilde{f}^{\ell,t}_{a_2} - \check{f}^{k,r^\ell}_{a_2}\}$

(C.3) $\bar{f}^{k,\ell,t_1}_{ij} + \bar{f}^{k,\ell,t_2}_{ji} \leq \tilde{x}^\ell_e$. Now, consider a set $k \in \mathcal{K}\backslash\{1\}$, two terminals $t_1, t_2 \in T^k_r$, $\ell \in \{1,\dots,k-1\}$, and an edge $e = \{i,j\} \in E$, again with the two related arcs $a_1 \in \{(i,j),(j,i)\}$ and $a_2 = \text{rev}(a_1)$. We argue that $\bar{f}^{k,\ell,t_1}_{a_1} + \bar{f}^{k,\ell,t_2}_{a_2} \leq \tilde{x}^\ell_e$.

If one flow is zero the inequality holds: E.g., if $\bar{f}^{k,\ell,t_2}_{a_2} = 0$ we have: $\bar{f}^{k,\ell,t_1}_{a_1} + \bar{f}^{k,\ell,t_2}_{a_2} = \bar{f}^{k,\ell,t_1}_{a_1} = \tilde{f}^{\ell,t_1}_{a_1} + \check{f}^{k,r^\ell}_{a_1} = \tilde{f}^{\ell,t_1}_{a_1} + \tilde{f}^{\ell,r^k}_{a_2} \leq \tilde{x}^\ell_e$. The last inequality is true due to constraint (6.56). The part with $\bar{f}^{k,\ell,t_1}_{a_1} = 0$ works analogously.

Otherwise, if both flows are > 0, we have: $\bar{f}^{k,\ell,t_1}_{a_1} + \bar{f}^{k,\ell,t_2}_{a_2} \overset{\text{(C.2)}}{=} \tilde{f}^{\ell,t_1}_{a_1} + \check{f}^{k,r^\ell}_{a_1} - \tilde{f}^{\ell,t_1}_{a_2} - \check{f}^{k,r^\ell}_{a_2} + \tilde{f}^{\ell,t_2}_{a_2} + \check{f}^{k,r^\ell}_{a_2} - \tilde{f}^{\ell,t_2}_{a_1} - \check{f}^{k,r^\ell}_{a_1} = \tilde{f}^{\ell,t_1}_{a_1} - \tilde{f}^{\ell,t_1}_{a_2} + \tilde{f}^{\ell,t_2}_{a_2} - \tilde{f}^{\ell,t_2}_{a_1} \leq \tilde{x}^\ell_e$, again by constraint (6.56).

(D) Solution to $\mathcal{P}^{\text{SFP}}_{\text{sdf}}$. Due to the previous discussion we are now able to construct a solution $(\hat{f},\hat{x}) \in \mathcal{P}^{\text{SFP}}_{\text{sdf}}$ with the same objective value.

(D.1) Variable assignment. We use the same values for the undirected edges by assigning $\hat{x} := \tilde{x}$. Trivially, $\hat{x} \in [0,1]^{|E|}$.

The flow variables $\hat{f}^{k,t}, \forall k \in \mathcal{K}, \forall t \in T^k_r$, are assigned the following values:

$$\hat{f}^{k,t} := \tilde{f}^{k,t} + \sum_{1 \leq \ell < k} \bar{f}^{k,\ell,t}$$

In case $k = 1$ the sum over ℓ is empty. Obviously, it holds $\hat{f}^{k,t} \geq 0$; the upper bound of 1 follows from (D.3).

(D.2) Flow conservation and flow value 1. Consider a set $k \in \mathcal{K}$, a terminal $t \in T^k_r$, and a vertex $i \in V$. By inserting the definition we get:

$$\begin{aligned}
&\hat{f}^{k,t}(\delta^-(i)) - \hat{f}^{k,t}(\delta^+(i)) \\
&= \tilde{f}^{k,t}(\delta^-(i)) + \sum_{1 \leq \ell < k} \bar{f}^{k,\ell,t}(\delta^-(i)) - \tilde{f}^{k,t}(\delta^+(i)) - \sum_{1 \leq \ell < k} \bar{f}^{k,\ell,t}(\delta^+(i))
\end{aligned}$$

Case $i = r^k$: $\tilde{f}^{k,t}(\delta^-(r^k)) - \tilde{f}^{k,t}(\delta^+(r^k)) = -\tilde{\alpha}_{kk}$ and for each $\ell, 1 \leq \ell < k$, it holds $\bar{f}^{k,\ell,t}(\delta^-(r^k)) - \bar{f}^{k,\ell,t}(\delta^+(r^k)) = -\tilde{\alpha}_{\ell k}$ (due to (C.1)). Overall we get $-\tilde{\alpha}_{kk} + \sum_{1 \leq \ell < k} -\tilde{\alpha}_{\ell k} = -1$ (due to constraint (6.54)).

Case $i = t$: Analogously, $\tilde{f}^{k,t}(\delta^-(t)) - \tilde{f}^{k,t}(\delta^+(t)) = \tilde{\alpha}_{kk}$ and for each $\ell, 1 \leq \ell < k$, it holds $\bar{f}^{k,\ell,t}(\delta^-(t)) - \bar{f}^{k,\ell,t}(\delta^+(t)) = \tilde{\alpha}_{\ell k}$ (due to (C.1)). Overall we get $\tilde{\alpha}_{kk} + \sum_{1 \leq \ell < k} \tilde{\alpha}_{\ell k} = 1$ (due to constraint (6.54)).

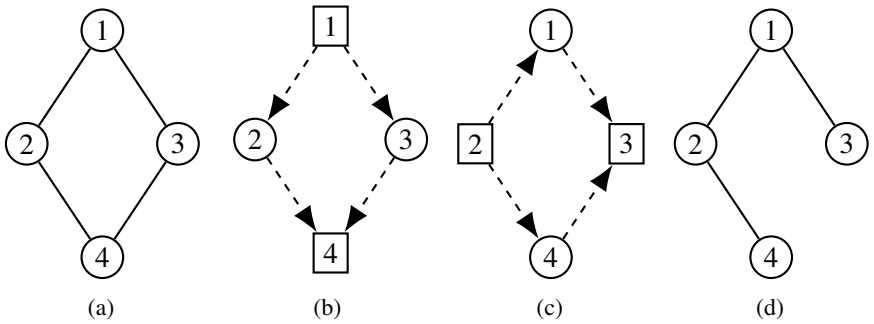


Figure 6.7: Instance for the Steiner forest problem where the LP relaxation of the directed flow formulation gives a better bound than the semi-directed flow formulation, cf. part (E) in the proof of Lemma 6.22. (a) depicts the input graph, (b) and (c) give valid flows for sets 1 and 2 (dashed arcs route flow of 0.5), and (d) gives an optimum integer solution which is also an optimum solution (projected to undirected x variables) of the LP relaxation of ($\text{SFP}_{\alpha:\text{df}}$).

Otherwise: Since $\tilde{f}^{k,t}$ and $\bar{f}^{k,\ell,t}(\delta^-(i)), 1 \le \ell < k$, are flows from r^k to t (see (C.1)) the sum is 0.

We conclude that $\hat{f}^{k,t}$ is a flow with value 1 from r^k to t, $\forall k \in \mathcal{K}, \forall t \in T_r^k$.

(D.3) $\hat{x}_e \ge \hat{f}_{ij}^{k,t_1} + \hat{f}_{ji}^{k,t_2}$. Last but not least, we need to show that constraints (6.49) are satisfied. Let $e = \{i, j\} \in E$, $k \in \mathcal{K}$, and $t_1, t_2 \in T_r^k$.

$$\begin{aligned}
\hat{f}_{ij}^{k,t_1} + \hat{f}_{ji}^{k,t_2} &\le \tilde{f}_{ij}^{k,t_1} + \sum_{1\le\ell<k} \bar{f}_{ij}^{k,\ell,t_1} + \tilde{f}_{ji}^{k,t_2} + \sum_{1\le\ell<k} \bar{f}_{ji}^{k,\ell,t_2} \\
&\overset{(6.56)}{\le} \tilde{x}_e^k + \sum_{1\le\ell<k} \left(\bar{f}_{ij}^{k,\ell,t_1} + \bar{f}_{ji}^{k,\ell,t_2}\right) \\
&\overset{(C.3)}{\le} \tilde{x}_e^k + \sum_{1\le\ell<k} \tilde{x}_e^\ell \le \sum_{1\le k\le K} \tilde{x}_e^k \overset{(6.53)}{\le} \tilde{x}_e = \hat{x}_e
\end{aligned}$$

Notice that in case $k = 1$ the sum(s) over $\ell < k$ are again empty.

(E) Example for strict inequality. Figure 6.7 gives an example with $x \in \text{Proj}_x(\mathcal{P}_{\text{sdf}}^{\text{SFP}})$ but $x \notin \text{Proj}_x(\mathcal{P}_{\alpha:\text{df}}^{\text{SFP}})$. The instance for the SFP has unit edge costs and two terminal sets $T^1 = \{1, 4\}$ and $T^2 = \{2, 3\}$ with $r^1 = 1, r^2 = 2$. The optimum solution to (SFP_{sdc}) sets $x_e := 0.5, \forall e \in E$, and the flows are given by Figure (b) and (c) with dashed arcs routing a flow of value 0.5. Hence, the optimum solution value of the LP relaxation of (SFP_{sdf}) is 2.

On the other hand, this solution is not valid for model ($\text{SFP}_{\alpha:\text{df}}$). A value of 0.5 for each edge implies a flow for the first terminal set as depicted in Figure (b). Then, it is not possible to route any flow for the second set (from node 2 to 3) without increasing the x variables. Hence, it has to hold $\alpha_{12} = 1$. However, sending a flow with value 1 from node 1 to nodes 2 and 3 while using the same arcs as in (b) is not possible. It is easy to see that the optimum solution to the LP relaxation of ($\text{SFP}_{\alpha:\text{df}}$) has a value of 3 (e.g., as in Figure (d)). □

With ($\text{SFP}_{\alpha:\text{df}}$) at hand we are able to formulate a directed cut-based formulation for the Steiner forest problem. Thereby, constraints (6.63) and (6.64) for the correct assignment of the α variables are the same as in the flow-based model ($\text{SFP}_{\alpha:\text{df}}$).

$$
\begin{aligned}
(\text{SFP}_{\alpha:\text{dc}})\ \min & \sum_{e\in E} c_e x_e & & \\
\text{s.t. } x_e &\geq \sum_{k\in\mathcal{K}} (z^k_{ij} + z^k_{ji}) && \forall e = \{i,j\} \in E && (6.62)\\
\sum_{1\leq j\leq k} \alpha_{jk} &= 1 && \forall k \in \mathcal{K} && (6.63)\\
\alpha_{kk} &\geq \alpha_{k\ell} && \forall k \in \mathcal{K}\backslash\{1,K\}, \forall \ell \in \{k+1,\ldots,K\} && (6.64)\\
z^k(\delta^-(S)) &\geq \alpha_{k\ell} && \begin{array}{l}\forall k \in \mathcal{K}, \forall \ell \in \{k,\ldots,K\},\\ \forall S \subseteq V\colon S \cap T^\ell \neq \emptyset, r^k \notin S\end{array} && (6.65)\\
\alpha &\in \{0,1\}^{0.5\cdot K\cdot(K+1)} && && (6.66)\\
x &\in \{0,1\}^{|E|} && && (6.67)\\
z^{1\ldots K} &\in \{0,1\}^{|A|\cdot K} && && (6.68)
\end{aligned}
$$

Constraints (6.65) are the directed cuts which depend on the "scenario" k, the terminal set ℓ, and the related $\alpha_{k\ell}$ variable. If a root node r^k is an assigned parent node for terminal set T^ℓ, i.e., $\alpha_{k\ell} > 0$, then all directed cuts separating r^k from any terminal in T^ℓ need to have a value of at least $\alpha_{k\ell}$.

Constraint (6.62) is a simple capacity constraint which implies that any used arc in any "scenario" is payed for in the objective function.

Lemma 6.23. *($SFP_{\alpha:dc}$) models the Steiner forest problem correctly.*

Proof. This proof is similar to the proof of Lemma 6.21 (correctness of ($\text{SFP}_{\alpha:\text{df}}$)). Let $\tilde{E} \subseteq E$ be an optimal solution to the SFP. Start with $\tilde{\alpha} := \mathbf{0}$. For each connected component C in $G[\tilde{E}]$ set $\tilde{\alpha}_{ii} = 1$ if r^i is the root node with lowest index contained in C and $\forall r^j \in C, j > i$, set $\tilde{\alpha}_{ij} = 1$. Now, $\tilde{E}$ can be oriented such that each connected component is an arborescence rooted at its parent node. This procedure gives values to variables $z^{1\ldots K}$ and x and induces a feasible solution to ($\text{SFP}_{\alpha:\text{dc}}$) with the same objective value.

The opposite direction follows directly from the preceding discussion: an optimum solution $(\tilde{\alpha}, \tilde{x}, \tilde{z}^{1\ldots K})$ to ($\text{SFP}_{\alpha:\text{dc}}$) implies a hierarchy of the terminal sets with parent and child sets and constraints (6.65) ensure that each terminal set is connected to its parent node. Hence, $\tilde{E} := \{e \in E \mid \tilde{x}_e = 1\}$ is a feasible solution to the SFP with the same cost. □

As final step we compare both directed flow- and cut-based models. The polytope of the cut-based model is denoted by:

$$
\mathcal{P}^{\text{SFP}}_{\alpha:\text{dc}} = \Big\{(\alpha, x, z^{1\ldots K}) \in [0,1]^{0.5\cdot K\cdot(K+1)} \times [0,1]^{|E|} \times [0,1]^{|A|\cdot K} \Big| \\ (\alpha, x, z^{1\ldots K}) \text{ satisfies (6.62)–(6.65)}\Big\}
$$

Let $\text{Proj}_x\left(\mathcal{P}^{\text{SFP}}_{\alpha:\text{dc}}\right)$ denote the linear projection of $\mathcal{P}^{\text{SFP}}_{\alpha:\text{dc}}$ into the x variable space.

Lemma 6.24. *$Proj_x\left(\mathcal{P}^{SFP}_{\alpha:df}\right) = Proj_x\left(\mathcal{P}^{SFP}_{\alpha:dc}\right)$, i.e., the directed flow- and cut-based formulations ($SFP_{\alpha:df}$) and ($SFP_{\alpha:dc}$) are equivalent.*

Proof. The constraints concerning the α variables are identical in both models. When considering one particular terminal set $k \in \mathcal{K}$ constraints (6.57) model a flow of value $\alpha_{k\ell}$ from r^k to each terminal $t \in T^\ell$, for each $\ell \in \{k, \dots, K\}$ (except r^k itself). On the other hand, the directed cuts (6.65) ensure that each directed cut separating r^k and t has a value of at least $\alpha_{k\ell}$. This is obviously equivalent. Moreover, similar to the equivalence of the STP models (STP_{df}), (STP_{dc}), and (STP_{df2}), constraints (6.53) and (6.56) on the one hand and constraint (6.62) on the other hand are equivalent, too. □

Remark 6.25. We like to shortly mention a possibility for modeling a cut-based formulation with fewer variables. This model contains the same α variables and variables $z \in \{0,1\}^{|A|}$ instead of x and $z^{1\dots K}$. Then, the objective function minimizes $\sum_{e=\{i,j\}\in E} c_e(z_{ij} + z_{ji})$. The constraints are the previously used constraints (6.63) and (6.64) combined with the directed cuts $z(\delta^-(S)) \geq \sum_{1\leq i\leq k:\, r^i\notin S} \alpha_{ik}, \forall k \in \mathcal{K}, \forall S \subseteq V\colon S \cap T^k \neq \emptyset$.

We remark that this model contains fewer variables, but it is also weaker than ($\text{SFP}_{\alpha:\text{dc}}$).

Improved undirected flow formulation. Magnanti and Raghavan [130] and Raghavan [146] introduced an *improved undirected flow formulation* for the SFP; we denote this model by (SFP_{iuf}). The formulation builds a set of arborescences which connect the terminal sets. Thereby, several sets can be contained in the same arborescence. This flow formulation is shown to be stronger than the undirected cut and flow formulation. A drawback is the large size of the formulation: the number of commodities is about $\sum_{k\in\mathcal{K}} k \cdot |T^k|$ and the number of constraints is at least $\prod_{1\leq k\leq K} |T^{k\dots K}|$ and hence, exponential in K.

It is not clear how this formulation is related to our formulations. On the one hand, (SFP_{iuf}) is not weaker than (SFP_{sdc}) and (SFP_{sdf}) since it gives a better bound on the instance presented in Figure 6.7. However, the strict inequality is not clear and moreover, we are not aware of any instances where ($\text{SFP}_{\alpha:\text{df}}$) and ($\text{SFP}_{\alpha:\text{dc}}$) give different bounds than (SFP_{iuf}).

Open problem 6.2. What is the relationship of the semi-directed and directed flow- and cut-based formulations and the *improved undirected flow formulation* [130] for the Steiner forest problem?

6.5.1 Partial 2-trees

Partial 2-trees are an interesting graph class since many NP-hard problems can be solved in polynomial time on such graphs. Moreover, many problems can be characterized completely by an LP formulation.

For the Steiner tree problem there exists a polynomial-time algorithm by Wald and Colbourn [170] and complete descriptions were given by Margot, Prodon, and Liebling [132] and Goemans [72, 73].

The Steiner forest problem is solvable in polynomial time, too, due to an algorithm by Bateni, Hajiaghayi, and Marx [13]. However, to the best of our knowledge, there does not exist a complete LP characterization of the SFP on partial 2-trees. The strongest known formulation (SFP_{iuf}) by Magnanti and Raghavan [130]—see also Section 6.5—does not have this property; an instance is given by Figure 6.8. Here, all edges have unit cost and

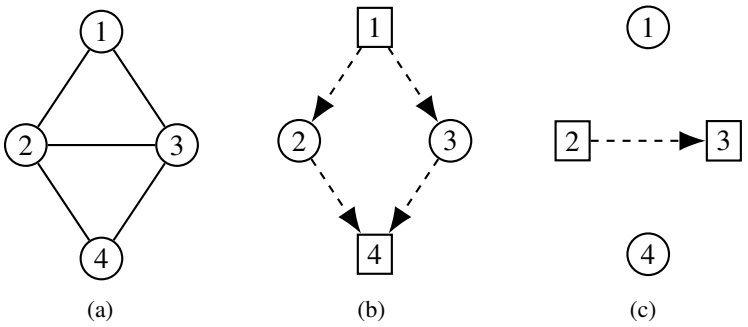


Figure 6.8: Instance for the SFP where the input graph is a partial 2-tree and the LP relaxations of the directed formulation ($\text{SFP}_{\alpha:\text{dc}}$) and the *improved undirected flow formulation* [130] have a fractional optimum solution. We have unit edge costs and $T^1 = \{1, 4\}$, $T^2 = \{2, 3\}$, and $r^1 = 1, r^2 = 2$. (a) shows the input graph. (b) and (c) depict valid assignments for the z variables in model ($\text{SFP}_{\alpha:\text{dc}}$) and for terminal sets 1 and 2, respectively. Thereby, every dashed arc has value 0.5 and the α variables are assigned values $\alpha_{11} = 1, \alpha_{12} = 0.5, \alpha_{22} = 0.5$.

the terminal sets are $T^1 = \{1, 4\}$ and $T^2 = \{2, 3\}$. As discussed by Raghavan [146] the optimum solution to the LP relaxation sets $x_e := 0.5$ for all edges e leading to a solution value of 2.5.

Notice that the optimum solution to the LP relaxation of (SFP_{sdc}) has a value of 2; here, $x_e := 0.5$ for all edges e except $x_{\{2,3\}} := 0$. Moreover, the optimum solution to the LP relaxation of ($\text{SFP}_{\alpha:\text{dc}}$) is fractional, too. The solution is given by Figure 6.8 (b) and (c).

However, if vertex sets are instead $T^1 = \{2, 3\}$ and $T^2 = \{1, 4\}$, with the same root nodes, the optimum solution to the LP relaxation of ($\text{SFP}_{\alpha:\text{dc}}$) and ($\text{SFP}_{\alpha:\text{df}}$) is integer: here, three arcs are selected, e.g., $\{(2, 1), (2, 3), (2, 4)\}$. The same holds for the *improved undirected flow formulation*, see [130].

Open problem 6.3. Does there exist a complete linear description for the Steiner forest problem on partial 2-trees?

For the stochastic Steiner tree problems there is no polynomial-time algorithm known, cf. Section 5.2 and Open problem 5.4. Moreover, since the SFP is a special case of the SSTP there is no complete description known.

Open problem 6.4. Does there exist a complete linear description for the (rooted) stochastic Steiner tree problem on partial 2-trees?

Chapter 7

Two-stage branch&cut algorithm

This chapter introduces the decomposition of the stochastic cut-based models and contains further details of the implemented two-stage branch&cut algorithm. We describe and concentrate on the semi-directed formulation ($\text{SSTP}_{\text{sdc2}}$). In Section 7.1, we start with the master problem and the primal and dual subproblem. In the second part, Section 7.2, several types of optimality cuts for the two-stage b&c algorithm are introduced: L-shaped optimality cuts with the method for strengthening these cuts (Section 7.2.1), integer optimality cuts (Section 7.2.2), and further optimality cuts (Section 7.2.3). Moreover, we consider the disaggregation of all constraints (Section 7.2.4) and a possibility for improving the lower bound L (Section 7.2.5). In the last part, Section 7.3, the necessary adaptations for—and differences to—the other semi-directed and directed formulations are described.

7.1 Decomposition

We start by recalling the semi-directed formulation ($\text{SSTP}_{\text{sdc2}}$) introduced in Section 6.2. Although it is possible to expand all stochastic models by additional constraints—in the first stage as well as in the second stage—these inequalities do not strengthen the formulations. We restrict the description to the base version and consider the additional inequalities and their impacts on the algorithm in Section 8.1.4.

We follow the description of the two-stage b&c algorithm given in Section 2.3.3 and use the same notations as before: K denotes the number of scenarios, $\mathcal{K}$ is the set of scenario indices, x^0 are the undirected first-stage variables, and y^k the directed arc variables of the kth scenario with $k \in \mathcal{K}$.

$$(\text{SSTP}_{\text{sdc2}}) \min \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e=\{i,j\} \in E} c_e^k (y_{ij}^k + y_{ji}^k - x_e^0)$$

$$\text{s.t. } y^k(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \subseteq V_r^k : S \cap T_r^k \neq \emptyset \tag{6.12}$$

$$y_{ij}^k + y_{ji}^k \geq x_e^0 \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E \tag{6.13}$$

$$x^0 \in \{0,1\}^{|E|} \tag{6.14}$$

$$y^{1\ldots K} \in \{0,1\}^{|A| \cdot K} \tag{6.15}$$

Following the description of Benders' decomposition (Section 2.3.2) and the two-stage b&c algorithm (Section 2.3.3) this model can be decomposed by introducing $K + 1$ new

variables $\theta, \theta^1, \ldots, \theta^K$ which represent and replace the K scenarios as lower bounds for the second-stage cost.

To improve readability we remove the specifier "SSTP" and the current first-stage solution $\tilde{x}^0$ from the identifier of the linear programs. The relaxed master problem reads as follows.

$$\begin{aligned}
(\mathrm{RMP}_{\mathrm{sdc2}})\ \min\ & \sum_{e \in E} c_e^0 x_e^0 + \theta \\
\text{s.t. } & \theta \geq \sum_{k \in \mathcal{K}} p^k \theta^k && (7.1)\\
& \text{Optimality cuts} && (7.2)\\
& x^0 \in [0,1]^{|E|} && (7.3)\\
& \theta, \theta^1, \ldots, \theta^K \geq 0 && (7.4)
\end{aligned}$$

Constraint (7.1) relates the overall second-stage cost represented by θ and the K second-stage costs for the scenarios represented by $\theta^1, \ldots, \theta^K$. Constraints (7.2) are separated optimality cuts as described in Section 7.2: L-shaped optimality cuts, integer optimality cuts, and further optimality cuts. Notice that we again focus on the description of the disaggregated L-shaped optimality cuts and moreover, notice that the master problem in its basic version does not contain any further constraints.

The relaxed primal of the kth subproblem, $k \in \mathcal{K}$, for a (fractional) first-stage solution $\tilde{x}^0 \in [0,1]^{|E|}$ looks as follows.

$$\begin{aligned}
(\mathrm{RSP}_{\mathrm{sdc2}})\ \min\ & \sum_{e=\{i,j\} \in E} c_e^k (y_{ij}^k + y_{ji}^k) - \sum_{e \in E} c_e^k \tilde{x}_e^0 \\
\text{s.t. } & y^k(\delta^-(S)) \geq 1 && \forall S \subseteq V_r^k : S \cap T_r^k \neq \emptyset && (7.5)\\
& y_{ij}^k + y_{ji}^k \geq \tilde{x}_e^0 && \forall e = \{i,j\} \in E && (7.6)\\
& y^k \geq \mathbf{0} && && (7.7)
\end{aligned}$$

The LPs of the subproblems are similar to the directed cut-based model for the STP, see Section 3.3.2. The models differ in the capacity constraints (7.6) such that any solution is enforced to select some arcs corresponding to the selected first-stage edges and find a feasible orientation. On the other hand, the directed cuts (7.5) are classical Steiner cuts. Hence, we refer to these subproblems as *restricted Steiner arborescence problems*.

Notice that the constraint $y^k \leq \mathbf{1}$ is redundant since all right-hand sides are at most 1 and any solution $\hat{y}^k \not\leq \mathbf{1}$ is non-optimal (recall that edge costs are positive).

7.2 Optimality cuts

This part introduces several types of optimality cuts for the two-stage b&c algorithm: L-shaped optimality cuts together with a strengthening procedure and a heuristic for improving the cuts (Section 7.2.1), integer optimality cuts (Section 7.2.2), and further optimality cuts (Section 7.2.3), respectively. Moreover, we consider the disaggregation of all introduced constraints in Section 7.2.4. Last but not least, Section 7.2.5 discusses a possibility for improving the lower bound L for the second-stage cost.

7.2.1 L-shaped optimality cuts

To generate a disaggregated L-shaped optimality cut (Lc) as described in Section 2.3.2 an optimum solution to the dual subproblem is required. Consider a scenario $k \in \mathcal{K}$ and let $\mathcal{S}^k$ denote the set of valid directed cuts, i.e., $\mathcal{S}^k := \{S \subseteq V_r^k \mid S \cap T_r^k \neq \emptyset\}$. The dual subproblem of scenario $k \in \mathcal{K}$ is obtained by assigning variable α_S^k to each directed cut $S \in \mathcal{S}^k$ and a constraint of type (7.5), and β_e^k to each edge $e \in E$ and a capacity constraint (7.6). Moreover, there exists a constraint for each directed arc $(i, j) \in A$.

Notice that the term $-\sum_{e \in E} c_e^k \tilde{x}_e^0$ in the objective function of (RSP_{sdc2}) is a constant. The kth dual problem for the first-stage solution $\tilde{x}^0$ reads as follows.

$$\text{(D:RSP}_{\text{sdc2}}\text{)} \quad \max \sum_{S \in \mathcal{S}^k} \alpha_S^k + \sum_{e \in E} \tilde{x}_e^0 \beta_e^k - \sum_{e \in E} c_e^k \tilde{x}_e^0$$

$$\text{s.t.} \sum_{S \in \mathcal{S}^k \,:\, (i,j) \in \delta^-(S)} \alpha_S^k + \beta_e^k \leq c_e^k \qquad \forall (i, j) \in A, e = \{i, j\} \tag{7.8}$$

$$\alpha^k, \beta^k \geq \mathbf{0} \tag{7.9}$$

Let $(\tilde{\alpha}^k, \tilde{\beta}^k)$ denote an optimum solution to (D:RSP_{sdc2}) w.r.t. $\tilde{x}^0$ with solution value $R^k(\tilde{x}^0)$. An L-shaped optimality cut for formulation ($\text{SSTP}_{\text{sdc2}}$) and scenario $k \in \mathcal{K}$ looks as follows:

$$\theta^k \geq \sum_{S \in \mathcal{S}^k} \tilde{\alpha}_S^k + \sum_{e \in E} x_e^0 \tilde{\beta}_e^k - \sum_{e \in E} c_e^k x_e^0$$

$$\Leftrightarrow \theta^k + \sum_{e \in E} \left(c_e^k - \tilde{\beta}_e^k\right) x_e^0 \geq \sum_{S \in \mathcal{S}^k} \tilde{\alpha}_S^k \tag{7.10}$$

Let $\tilde{\theta}^k$ denote the current kth scenario cost. If $\tilde{\theta}^k < \sum_{S \in \mathcal{S}^k} \tilde{\alpha}_S^k + \sum_{e \in E} \tilde{x}_e^0 (\tilde{\beta}_e^k - c_e^k)$, the cut is violated and can be added to the master problem (RMP_{sdc2}).

Notice that the right-hand side of the L-shaped cut and the coefficient next to $x_e^0, e \in E$, is non-negative. The right-hand side is non-negative since $\alpha^k \geq \mathbf{0}$ and due to the dual constraints (7.8) it holds $\beta_e^k \leq c_e^k, \forall e \in E$.

For the solution $x^0 = \tilde{x}^0$ the cut states that $\theta^k \geq R^k(\tilde{x}^0)$ which is the desired lower bound on θ^k. Modifying the first stage gives different bounds. A component-wise larger first stage (possibly) decreases the bound on θ^k whereas a smaller first-stage solution (possibly) increases the bound. Let $\hat{x}^0, \check{x}^0 \in [0, 1]^{|E|}$ such that $\hat{x}^0 \gneq \tilde{x}^0$, i.e., $\hat{x}_e^0 \geq \tilde{x}_e^0, \forall e \in E$, and $\exists \hat{e} \in E \colon \hat{x}_{\hat{e}}^0 > \tilde{x}_{\hat{e}}^0$, and $\check{x}^0 \lneq \tilde{x}^0$ be defined analogously with $\exists \check{e} \in E \colon \check{x}_{\check{e}}^0 < \tilde{x}_{\check{e}}^0$. Then, inserting $\hat{x}^0$ into the cut gives $\theta^k \geq \hat{r}$, with $\hat{r} \leq R^k(\tilde{x}^0) - (\hat{x}_{\hat{e}}^0 - \tilde{x}_{\hat{e}}^0)(c_{\hat{e}}^k - \tilde{\beta}_{\hat{e}}^k)$ and hence, the right-hand side is at most $R^k(\tilde{x}^0)$. Similarly for $\check{x}^0$ the cut yields $\theta^k \geq \check{r}$, with $\check{r} \geq R^k(\tilde{x}^0) + (\tilde{x}_{\check{e}}^0 - \check{x}_{\check{e}}^0)(c_{\check{e}}^k - \tilde{\beta}_{\check{e}}^k)$ which is at least $R^k(\tilde{x}^0)$.

Deriving stronger L-shaped optimality cuts. Since the (relaxed) master problem mainly consists of L-shaped optimality cuts, the number of master iterations of the two-stage b&c approach—and hence, the overall running time—is highly influenced by the strength of the generated L-shaped cuts. Here, we propose a new and fast way of strengthening the generated L-shaped cuts.

Most of the previously proposed strengthening approaches (cf. e.g., Fischetti, Salvagnin, and Zanette [65], Magnanti and Wong [131], Papadakos [138], Sherali and Lunday [162], Wentges [171]) require solving an auxiliary LP in order to generate a stronger L-shaped

cut. Contrarily, our new approach is very efficient and it is able to find a stronger L-shaped cut in linear time (with respect to the number of primal variables).

Let $(\tilde{x}^0, \tilde{\theta}, \tilde{\theta}^1, \ldots, \tilde{\theta}^K)$ denote the current solution to the master problem (RMP_{sdc2}). Consider any scenario $k \in \mathcal{K}$ with the current optimum dual solution $(\tilde{\alpha}^k, \tilde{\beta}^k)$. Instead of optimizing an additional problem, the L-shaped cuts (7.10) can be strengthened by modifying $(\tilde{\alpha}^k, \tilde{\beta}^k)$ directly.

Notice that if for an edge $e \in E$ the current first-stage solution satisfies $\tilde{x}^0_e = 0$ then the corresponding dual variable β^k_e does not appear in the objective function of the dual ($\text{D:RSP}_{\text{sdc2}}$). Hence, it is not difficult to see that we deal with a highly degenerate LP and one can expect that the optimum solutions to the dual subproblem ($\text{D:RSP}_{\text{sdc2}}$) usually produce positive slacks in the constraints (7.8); typically, if possible, a dual variable with coefficient zero in the objective function will be set to zero by an LP solver. The idea is to produce another LP-optimal solution $(\hat{\alpha}^k, \hat{\beta}^k)$ of the dual subproblem such that the corresponding slacks are reduced to zero. Therefore, the values of the dual multipliers in the associated L-shaped cut will be increased as follows:

For each edge $e = \{i, j\} \in E$ set

$$\hat{\beta}^k_e := \begin{cases} c^k_e - \max\limits_{a \in \{(i,j),(j,i)\}} \left\{ \sum\limits_{S \in \mathcal{S}^k \,:\, a \in \delta^-(S)} \tilde{\alpha}^k_S \right\} & \text{if } \tilde{x}^0_e = 0 \\ \tilde{\beta}^k_e & \text{otherwise.} \end{cases} \tag{7.11}$$

If $\hat{\beta}^k_e > \tilde{\beta}^k_e$ holds for at least one edge $e \in E$ the strengthened L-shaped cut is given as

$$\theta^k + \sum_{e \in E} \left(c^k_e - \hat{\beta}^k_e \right) x^0_e \geq \sum_{S \in \mathcal{S}^k} \tilde{\alpha}^k_S \tag{7.12}$$

Theorem 7.1. *The strengthened L-shaped optimality cut* (7.12) *is valid and stronger than the standard L-shaped optimality cut* (7.10)*.*

Proof. Consider two L-shaped cuts: the standard one implied by the optimum dual solution $(\tilde{\alpha}^k, \tilde{\beta}^k)$ and the strengthened one implied by $(\tilde{\alpha}^k, \hat{\beta}^k)$ with $\hat{\beta}^k$ being set by (7.11). Obviously, $(\tilde{\alpha}^k, \hat{\beta}^k)$ is a feasible and LP-optimal solution to the dual subproblem ($\text{D:RSP}_{\text{sdc2}}$) since $\hat{\beta}^k$ is set without violating any dual constraints.

Furthermore, notice that $\hat{\beta}^k_e \geq \tilde{\beta}^k_e$, for all $e \in E$, and that the right-hand side of both cuts is identical. Since there exists $e_1 \in E$ such that $\hat{\beta}^k_{e_1} > \tilde{\beta}^k_{e_1}$, the coefficient of $x^0_{e_1}$ is strictly smaller for the strengthened L-shaped cut than for the standard one which concludes the proof. □

Similar approaches were used for stabilizing a column generation approach for constrained tree problems, see Leitner, Ruthmair, and Raidl [121] and Leitner [120]. Moreover, Álvarez-Miranda, Fernández, and Ljubić [4] adopt our method for a Benders' decomposition approach for the robust facility location problem.

Strengthening cuts through laminarity. Our method for strengthening L-shaped cuts modifies only the β^k variables of the current optimum dual solution. In particular, the values of the α^k variables remain unchanged and moreover, the set of α^k variables remains unchanged. By considering a different set of directed cuts the set of α^k variables changes and the dual solution may imply a different and stronger L-shaped cut. In this part we

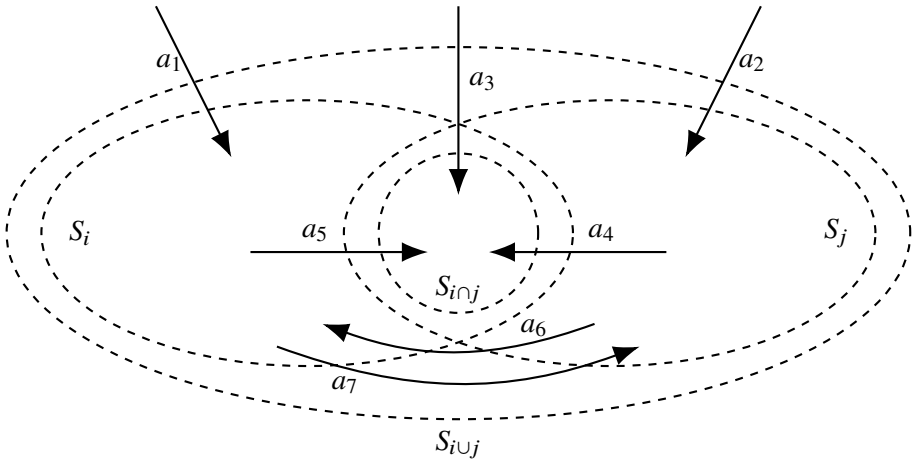


Figure 7.1: Sketch of two crossing cuts S_i and S_j with $S_{i\cup j}$ and $S_{i\cap j}$, and all possible arcs lying in the cuts. The cuts are drawn as dashed circles and arcs as solid lines.

discuss a heuristic for strengthening the L-shaped cuts. This approach is based on the property of a *laminar set of cuts* and *non-crossing cuts*, cf. Figure 7.1.

Let $S_i, S_j \subset V, S_i \neq S_j$, be two non-empty and valid cuts. We say that S_i *covers* S_j iff $\delta(S_i) \subsetneq \delta(S_j)$. Let $\mathcal{S}^k$ be the set of valid cuts which are separated for scenario $k \in \mathcal{K}$. If no cut in $\mathcal{S}^k$ is covered we call $\mathcal{S}^k$ *cover-free*.

Observation 7.2. *One may assume w.l.o.g. that $\mathcal{S}^k$ is cover-free.*

Proof. We can check the property for all pairs of cuts in polynomial time in the size of $|\mathcal{S}^k|$ and $|E|$. Now assume there exist $S_i, S_j \in \mathcal{S}^k$ such that S_i covers S_j. Let $(\tilde{\alpha}^k, \tilde{\beta}^k)$ denote the current optimum dual solution. It is easy to see that there exists a dual solution for cut set $\mathcal{S}^k \setminus \{S_j\}$ with the same objective value implying an L-shaped cut that is not weaker.

Consider the alternative solution $(\hat{\alpha}^k, \hat{\beta}^k)$ with $\hat{\alpha}^k_{S_i} := \hat{\alpha}^k_{S_i} + \tilde{\alpha}^k_{S_j}$, $\hat{\alpha}^k_{S_j} := 0$, and $\hat{\alpha}^k_{S_\ell} := \tilde{\alpha}^k_{S_\ell}, \forall S_\ell \in \mathcal{S}^k \setminus \{S_i, S_j\}$. Each α variable has the same objective function coefficient and hence, $(\hat{\alpha}^k, \hat{\beta}^k)$ has the same objective value. All dual constraints are also still satisfied since the set of constraints $\alpha^k_{S_i}$ is contained in is a subset of the constraints $\alpha^k_{S_j}$ is contained in. Hence, $(\hat{\alpha}^k, \hat{\beta}^k)$ is valid and still optimum. The L-shaped cut cannot be weaker since the right-hand side remains unchanged and the values of the β^k variables are unchanged—it might even be possible to strengthen the cut since $\sum_{S \in \mathcal{S}^k : (i,j) \in \delta^-(S)} \alpha^k_S$ decreases for edges contained in $\delta(S_j \setminus S_i)$. □

Due to this observation we assume from now on that any separated and considered set of cuts is cover-free.

Definition 7.3 (*Laminarity*, *(non)-crossing* Cuts). Two non-empty cuts $S_i, S_j \subset V, S_i \neq S_j$ are *non-crossing* if and only if $S_i \subset S_j$, or $S_j \subset S_i$, or $S_i \cap S_j = \emptyset$. Do none of the properties hold we call the two cuts *crossing*. A set of cuts $\mathcal{S} \subseteq 2^V$ is called *laminar* if each pair of cuts $S_i, S_j \in \mathcal{S}, S_i \neq S_j$, is non-crossing.

Now, consider a scenario $k \in \mathcal{K}$ and two crossing cuts $S_i, S_j \in \mathcal{S}^k$, with $S_i \neq S_j$, such that the intersection contains a terminal, i.e., with the notation $S_{i\cap j} := S_i \cap S_j$ and

$S_{i\cup j} := S_i \cup S_j$, it holds $T_r^k \cap S_{i\cap j} \neq \emptyset$. Figure 7.1 sketches the situation and depicts a set of relevant arcs $\{a_1, \ldots, a_7\}$ which are contained in $\delta^-(S_i), \delta^-(S_j), \delta^-(S_{i\cap j})$, and $\delta^-(S_{i\cup j})$, respectively.

If $\tilde{\alpha}_{S_i}^k > 0$ and $\tilde{\alpha}_{S_j}^k > 0$ it is possible to utilize the current optimum dual solution $(\tilde{\alpha}^k, \tilde{\beta}^k)$ and construct a new dual solution $(\hat{\alpha}^k, \hat{\beta}^k)$ by the following method called *Laminarize* which sets $\alpha_{S_i}^k = 0$ or $\alpha_{S_j}^k = 0$ (or both) and results in a laminar set (at least w.r.t. $S_i, S_j, S_{i\cup j}, S_{i\cap j}$).

0. *Input*: Two crossing cuts $S_i, S_j \in \mathcal{S}^k$, $S_i \neq S_j$, with $T_r^k \cap S_{i\cap j} \neq \emptyset$, $\tilde{\alpha}_{S_i}^k > 0$, and $\tilde{\alpha}_{S_j}^k > 0$
1. *Initialization*: $(\hat{\alpha}^k, \hat{\beta}^k) := (\tilde{\alpha}^k, \tilde{\beta}^k)$, $\alpha_{\min}^k := \min\{\tilde{\alpha}_{S_i}^k, \tilde{\alpha}_{S_j}^k\}$
2. *Increase-Step*: $\hat{\alpha}_{S_{i\cup j}}^k := \tilde{\alpha}_{S_{i\cup j}}^k + \alpha_{\min}^k$, $\hat{\alpha}_{S_{i\cap j}}^k := \tilde{\alpha}_{S_{i\cap j}}^k + \alpha_{\min}^k$
3. *Decrease-Step*: $\hat{\alpha}_{S_j}^k := \tilde{\alpha}_{S_j}^k - \alpha_{\min}^k$, $\hat{\alpha}_{S_i}^k := \tilde{\alpha}_{S_i}^k - \alpha_{\min}^k$
4. *Cut strengthening*: Apply the cut strengthening (equation (7.11)) to $(\hat{\alpha}^k, \hat{\beta}^k)$

The following lemmata show that the generated dual solution leads to a feasible and stronger L-shaped optimality cut.

Lemma 7.4. *After Step 1–3 of* Laminarize *the solution* $(\hat{\alpha}^k, \hat{\beta}^k)$ *is an optimum dual solution.*

Proof. We verify the non-negativity of the variables and the feasibility of the dual constraints (7.8). First, $\hat{\alpha}^k \geq \mathbf{0}$ follows directly since $\alpha_{\min}^k$ is the minimum of $\tilde{\alpha}_{S_i}^k$ and $\tilde{\alpha}_{S_j}^k$ and these are the only two variables with a decreasing value.

Second, we obviously only need to consider a constraint (7.8) corresponding to an arc (u, v) which lies in one of the modified cuts $\delta^-(S_i), \delta^-(S_j)$, $\delta^-(S_{i\cup j})$, or $\delta^-(S_{i\cap j})$. Now, consider such an arc (u, v) with the related constraint $\sum_{S\in\mathcal{S}^k\,:\,(u,v)\in\delta^-(S)} \hat{\alpha}_S^k + \hat{\beta}_e^k \leq c_e^k$. All such types of arcs are depicted in Figure 7.1. The following table also summarizes the relationship of the arcs and the related directed cuts: If an arc $a_\ell \in A$ is contained in a cut $S \in \mathcal{S}^k$ we denote this with a filled or empty circle in the corresponding cell $[S, a_\ell]$ of the matrix. Thereby, each row contains either filled or empty circles which depends on the modification of the α^k variable (decreased $\hat{=}$ empty circle, increased $\hat{=}$ filled circle).

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
S_i	○		○	○		○	
S_j		○	○		○		○
$S_{i\cup j}$	●	●	●				
$S_{i\cap j}$			●	●	●		

As one can see, the number of filled circles in each column is not larger than the number of empty circles. Hence, the value $\sum_{S\in\mathcal{S}^k\,:\,(u,v)\in\delta^-(S)} \hat{\alpha}_S^k$ does not increase for any arc $(u, v) \in A$ (in particular, the sum decreases for arcs a_6 and a_7, respectively, which will be important for Lemma 7.5). Since we consider $\hat{\beta}^k = \tilde{\beta}^k$ it follows that $(\hat{\alpha}^k, \hat{\beta}^k)$ is a feasible dual solution.

Last but not least, we show that the objective value does not decrease. First, it is clear that the second sum in the objective function does not change: $\sum_{e\in E} \tilde{x}_e^0 \tilde{\beta}_e^k = \sum_{e\in E} \tilde{x}_e^0 \hat{\beta}_e^k$.

For the first sum we need to recall that $S_{i\cup j}$ and $S_{i\cap j}$ are valid directed cuts since $S_{i\cup j}, S_{i\cap j} \subseteq V_r^k$ and both sets contain at least one terminal. Moreover, since $\alpha^k_{S_i} + \alpha^k_{S_j}$ decreases by $2\alpha^k_{\min}$ and $\alpha^k_{S_{i\cup j}} + \alpha^k_{S_{i\cap j}}$ increases by $2\alpha^k_{\min}$ it follows $\sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S = \sum_{S\in\mathcal{S}^k} \hat{\alpha}^k_S$.□

Lemma 7.5. *If there exists an edge $\{u, v\} \in E$ with $u \in S_i, v \in S_j, u, v \notin S_{i\cap j}$ then $(\hat{\alpha}^k, \hat{\beta}^k)$, as the result of* Laminarize*, implies a stronger L-shaped optimality cut than $(\tilde{\alpha}^k, \tilde{\beta}^k)$.*

Proof. The final application of the cut strengthening does not affect the validity of the dual solution, cf. Theorem 7.1. Hence, $(\hat{\alpha}^k, \hat{\beta}^k)$ is a valid L-shaped optimality cut.

Now assume there exists an edge $e = \{u, v\} \in E$ with $u \in S_i, v \in S_j, u, v \notin S_{i\cap j}$. Notice that the arcs (u, v) and (v, u) of the bidirection are arcs of type a_6 and a_7, respectively, as depicted by Figure 7.1. As discussed in the proof of Lemma 7.4 for both arcs $a \in \{(u, v), (v, u)\}$ it holds $\sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \hat{\alpha}^k_S = \sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \tilde{\alpha}^k_S - \alpha^k_{\min}$. Hence, $\hat{\beta}^k_e$ can be increased by $\alpha^k_{\min} > 0$ without violating both dual constraints. This procedure gives a stronger optimality cut. □

Pareto optimal L-shaped cuts. The definition of *Pareto optimal* cuts was introduced by Magnanti and Wong [131]. Informally, an L-shaped cut dominates another L-shaped cut if the obtained bounds for the second-stage cost are never worse, and the bound has to be strictly better for at least one first-stage solution. Originally, [131] described the aggregated version of the optimality cuts (Lc1) but it is possible to extend this definition to the multicut version (Lc), which we consider in the following.

Let the primal and dual subproblem be defined by (RSP($k, \tilde{x}^0$)) and (D:RSP($k, \tilde{x}^0$)), respectively. Let $\mathcal{X}^0$ denote the set of feasible solutions for the master problem (RMP) and $\widetilde{\Pi}^k_{\text{opt}}$ the set of optimal dual solutions for (D:RSP($k, \tilde{x}^0$)).

Definition 7.6 (Dominance, Pareto optimality). Let $\hat{\pi}^k, \tilde{\pi}^k \in \widetilde{\Pi}^k_{\text{opt}}$. The L-shaped optimality cut $\theta^k + (\hat{\pi}^k)^\top T^k x^0 \geq (\hat{\pi}^k)^\top h^k$ *dominates* the cut $\theta^k + (\tilde{\pi}^k)^\top T^k x^0 \geq (\tilde{\pi}^k)^\top h^k$ if and only if the two following properties hold:

1. $\forall \bar{x}^0 \in \mathcal{X}^0\colon (\hat{\pi}^k)^\top h^k - (\hat{\pi}^k)^\top T^k \bar{x}^0 \geq (\tilde{\pi}^k)^\top h^k - (\tilde{\pi}^k)^\top T^k \bar{x}^0$
2. $\exists \bar{x}^0 \in \mathcal{X}^0\colon (\hat{\pi}^k)^\top h^k - (\hat{\pi}^k)^\top T^k \bar{x}^0 > (\tilde{\pi}^k)^\top h^k - (\tilde{\pi}^k)^\top T^k \bar{x}^0$

An L-shaped optimality cut $\theta^k + (\hat{\pi}^k)^\top T^k x^0 \geq (\hat{\pi}^k)^\top h^k$ is *Pareto optimal* if and only if there exists no dominating cut. Since a cut is implied by its dual solution we use both terms analogously for dual solutions.

Magnanti and Wong [131] introduced a linear program for generating a Pareto optimal L-shaped cut, which is an expansion of the dual problem (D:RSP($k, \tilde{x}^0$)):

$$\begin{aligned} (\text{MW}(k, \tilde{x}^0, \check{x}^0)) \quad & \max\ \pi^k(h^k - T^k \check{x}^0) \\ & \text{s.t. } \pi^k \in \widetilde{\Pi}^k_{\text{opt}} \end{aligned} \tag{7.13}$$

This LP is parameterized by the current first-stage solution $\tilde{x}^0$ and another first-stage solution $\check{x}^0$. The objective is to maximize the same objective function of the dual subproblem but with $\check{x}^0$ as multipliers. Moreover, the solution has to be optimal for the dual subproblem w.r.t. $\tilde{x}^0$, which is ensured by (7.13). If $\check{x}^0$ is a *core-point*, i.e., it lies in the relative interior

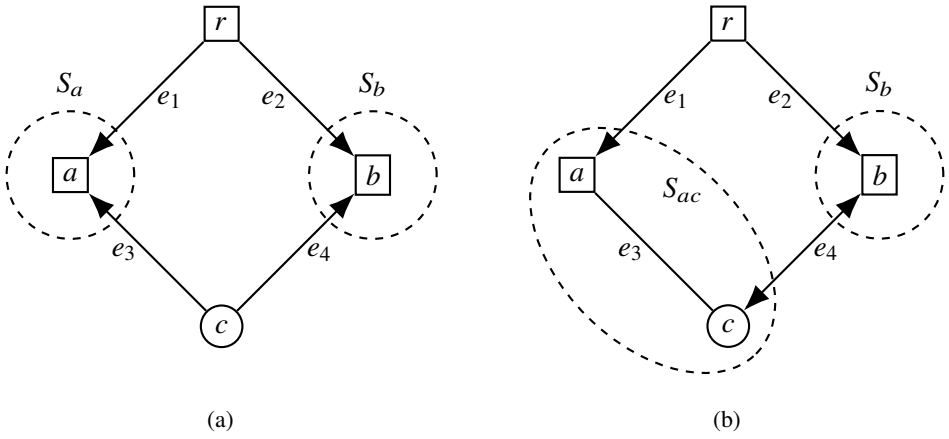


Figure 7.2: Counter example for Pareto optimality of the generated (and strengthened) L-shaped optimality cuts for formulation ($\text{SSTP}_{\text{sdc2}}$), cf. text. The directed cuts are indicated by dashed circles.

of the convex hull of $\mathcal{X}^0$, then the optimal solution to (MW($k, \tilde{x}^0, \check{x}^0$)) implies a Pareto optimal L-shaped cut, cf. [131].

Although the strengthened L-shaped optimality cuts perform very well in practice, see Section 8.3 and 10.2, the cuts are not Pareto optimal in the sense of Magnanti and Wong. A counter example is given by Figure 7.2. We consider the empty first-stage solution $\tilde{x}^0 = \mathbf{0}$, a scenario with terminal set $\{r, a, b\}$, and all edge costs are 1. Clearly, the optimum primal solution to the subproblem is integer with $x^0_{e_1} = x^0_{e_2} = 1$ and cost 2. Figure 7.2 (a) and (b) depict three valid directed cuts S_a, S_b, S_{ac}. (a) shows an optimum dual solution $(\tilde{\alpha}^k, \tilde{\beta}^k)$ by using S_a and S_b and by setting $\tilde{\alpha}^k_{S_a} = \tilde{\alpha}^k_{S_b} = 1$. Analogously, (b) gives a solution $(\hat{\alpha}^k, \hat{\beta}^k)$ with $\hat{\alpha}^k_{S_{ac}} = \hat{\alpha}^k_{S_b} = 1$. Hence, the dual β^k variables for e_1, e_2, e_4 are 0 for both dual solutions. However, edge e_3 is affected only by the first solution such that $\tilde{\beta}^k_{e_3} = 0$ whereas $\hat{\beta}^k_{e_3} = 1$. Hence, the L-shaped optimality cut induced by solution $(\tilde{\alpha}^k, \tilde{\beta}^k)$ reads $\theta^k + x^0_{e_1} + x^0_{e_2} + x^0_{e_3} + x^0_{e_4} \geq 2$ and for solution $(\hat{\alpha}^k, \hat{\beta}^k)$ the cut states $\theta^k + x^0_{e_1} + x^0_{e_2} + x^0_{e_4} \geq 2$. Clearly, the second cut dominates the first one and both solutions are feasible and optimal solutions to (D:RSP($k, \tilde{x}^0$)). Moreover, notice that the cut set of this example is laminar and hence, the described method *Laminarize* does not give Pareto optimal L-shaped cuts either.

Open problem 7.1. Does there exist a combinatorial polynomial-time algorithm that generates a Pareto optimal L-shaped cut (for the SSTP)?

We close the discussion on Pareto optimal L-shaped cuts by mentioning two further related publications. Sherali and Lunday [162] considered *maximal nondominated cuts* with the property that "*a Pareto optimal [. . .] cut generated in the sense of Magnanti and Wong (1981) is also maximal provided that the core point x^0 is positive, but the converse is not necessarily true*". The example of Figure 7.2 shows that the strengthened and laminarized L-shaped cuts are also not maximal nondominated.

Papadakos [138] described enhancements to the method of Magnanti and Wong [131] by showing that the constraint $\pi^k \in \widetilde{\Pi}^k_{\text{opt}}$ in the linear program (MW($k, \tilde{x}^0, \check{x}^0$)) is not necessary for generating a Pareto optimal L-shaped cut. However, the generated cut does not cut off any particular first-stage solution anymore and the idea is to add some of these cuts at the beginning of an L-shaped algorithm.

7.2.2 Integer optimality cuts

In this section we use a similar notation as in Section 2.3.3. Moreover, we consider several binary first-stage solutions $\tilde{x}^0, \hat{x}^0, \check{x}^0$, etc. with the associated 1-index sets $\tilde{S}, \hat{S}, \check{S}$, etc., with, e.g., $\tilde{S} = \{i \mid \tilde{x}^0_i = 1\}$. In the following we skip these definitions and we always use the same accent to indicate a solution and its corresponding index set. We use an index set as representative for a solution such that the index set equals an edge set, and we use both index set and solution interchangeably.

Let $\tilde{q}$ denote the value of the second-stage recourse function w.r.t. a binary first-stage solution $\tilde{x}^0$, i.e., $\tilde{q} := \sum_{k\in\mathcal{K}} p^k \tilde{q}^k$ with $\tilde{q}^k = Q^k(\tilde{x}^0)$ being the optimum solution value of the subproblem $(\mathrm{SP}(k, \tilde{x}^0))$. Last but not least, let L be a valid lower bound for the expected second-stage cost.

The standard integer optimality cut w.r.t. $\tilde{x}^0$ and $\tilde{S}$ reads as follows.

$$\begin{aligned} &\theta \geq (\tilde{q} - L)\left(\sum_{e\in\tilde{S}} x^0_e - \sum_{e\notin\tilde{S}} x^0_e - |\tilde{S}| + 1\right) + L \\ \Leftrightarrow\ &\theta + \sum_{e\in\tilde{S}} (L - \tilde{q}) x^0_e + \sum_{e\notin\tilde{S}} (\tilde{q} - L) x^0_e \geq (L - \tilde{q})(|\tilde{S}| - 1) + L \end{aligned} \tag{Ic}$$

Deriving stronger integer optimality cuts. By considering a binary first-stage solution it is possible to derive new and stronger integer optimality cuts. The idea is based on the following observation.

Observation 7.7. *Let $\tilde{S}$ denote a first-stage solution with second-stage cost $\tilde{q}$. Then, $\tilde{q}$ is a lower bound for the second-stage cost of any solution $\check{S} \subseteq \tilde{S}$.*

This is obviously true since removing an edge from the first stage cannot decrease the cost in any scenario. By removing an edge existing connections are lost which can be ignored (if an edge is not necessary for satisfying connections in a scenario) or need to be newly established. In the latter case the new solution cannot be cheaper than the previous one; this would be a contradiction to the optimality.

By following Observation 7.7 we introduce new integer optimality cuts:

$$\theta + (\tilde{q} - L) \sum_{e\notin\tilde{S}} x^0_e \geq \tilde{q} \tag{Ic$^-$}$$

The idea is that any sub-solution of $\tilde{S}$ has the same lower bound $\tilde{q}$ on the second-stage cost and the lower bound for any other solution is L.

Lemma 7.8. *Constraint* (Ic$^-$) *is a valid integer optimality cut.*

Proof. For proving validity and to improve readability we reformulate the cuts as follows:

$$\theta \geq \tilde{q} + (L - \tilde{q}) \sum_{e\notin\tilde{S}} x^0_e. \tag{7.14}$$

Inserting the first-stage solution $\tilde{S}$ gives the correct bound $\theta \geq \tilde{q}$ (since $\sum_{e\notin\tilde{S}} \tilde{x}^0_e = 0$). Now, consider a first-stage solution $\check{S}$ with $\check{S} \subsetneq \tilde{S}$. It holds $e \notin \tilde{S} \Rightarrow e \notin \check{S}$ and hence, $\sum_{e\notin\tilde{S}} \check{x}^0_e = 0$, and the cut states $\theta \geq \tilde{q}$; this is a valid bound due to Observation 7.7.

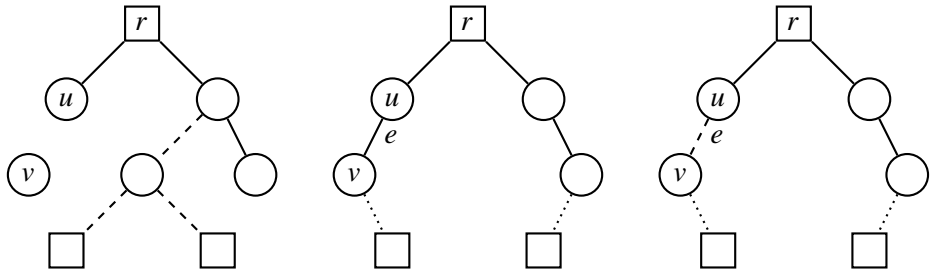


Figure 7.3: Sketch showing the validity of the lower bound for the second-stage cost when the first stage is expanded by E'. In this example we consider the simplest case $E' = \{e\}$. (a) A fist-stage solution $\tilde{S}$ (solid edges) with an optimum scenario solution $\tilde{S}^k$ (dashed edges). (b) A fist-stage solution $\hat{S}$ with $\hat{S} = \tilde{S} \cup \{e\}, e \notin \tilde{S}$, and the corresponding optimum scenario solution $\hat{S}^k$ (dotted edges). (c) The solution $\hat{S}^k \cup \{e\}$ for the same scenario under $\tilde{S}$. Since (c) depicts a heuristic scenario solution to $\tilde{S}$ its scenario cost is at least as expensive as the cost of the solution in (a).

Last but not least, consider a solution $\bar{S}$ with $\bar{S} \backslash \tilde{S} \neq \emptyset$. With $\Delta := \sum_{e \notin \tilde{S}} \bar{x}_e^0$ the right-hand side of (Ic$^-$) is $\tilde{q} + (L - \tilde{q})\Delta = \Delta L - (\Delta - 1)\tilde{q} = L + (\Delta - 1)(L - \tilde{q})$. Since $L \leq \tilde{q}$ and $\Delta \geq 1$ it follows that $(\Delta - 1)(L - \tilde{q}) \leq 0$ and hence, the right-hand side of (Ic$^-$) is at most L. □

Lemma 7.9. *If it holds $\tilde{q} > L$ and $|\tilde{S}| \geq 1$, then the integer optimality cut* (Ic$^-$) *is stronger than* (Ic).

Proof. We show (i) that a solution which is feasible for (Ic$^-$) is feasible for (Ic), too, and (ii) that there exists a solution which is feasible for (Ic) but it is infeasible for (Ic$^-$).

(i) Let $(\bar{\theta}, \bar{x}^0)$ be feasible for (Ic$^-$). We start by inserting $(\bar{\theta}, \bar{x}^0)$ into (7.14):

$$\begin{aligned}
&\bar{\theta} \geq \tilde{q} + (L - \tilde{q}) \sum_{e \notin \tilde{S}} \bar{x}_e^0 \\
\Rightarrow\ &\bar{\theta} \geq (\tilde{q} - L) \left(- \sum_{e \notin \tilde{S}} \bar{x}_e^0 + 1 \right) + L \\
\Rightarrow\ &\bar{\theta} \geq (\tilde{q} - L) \left(|\tilde{S}| - \sum_{e \notin \tilde{S}} \bar{x}_e^0 - |\tilde{S}| + 1 \right) + L \\
\Rightarrow\ &\bar{\theta} \geq (\tilde{q} - L) \left(\sum_{e \in \tilde{S}} \bar{x}_e^0 - \sum_{e \notin \tilde{S}} \bar{x}_e^0 - |\tilde{S}| + 1 \right) + L
\end{aligned}$$

This concludes the first part of the proof since this is equal to inserting $(\bar{\theta}, \bar{x}^0)$ into (Ic).

(ii) Consider a binary solution $\check{x}^0$ with $\check{S} \subsetneq \tilde{S}$ and $|\check{S}| = |\tilde{S}| - 1$. The bound on θ given by (Ic) for $\check{x}^0$ is $\theta \geq L$. On the other hand, (Ic$^-$) implies $\theta \geq \tilde{q} > L$. □

Constraints (Ic$^-$) are derived by considering a sub-solution and by removing edges. In the following we consider the opposite direction.

Again, $\tilde{S} \subsetneq E$ denotes the current first-stage solution and $\tilde{q} := \sum_{k \in \mathcal{K}} p^k \tilde{q}^k$, with $\tilde{q}^k$ being the optimum solution value of (SP$(k, \tilde{x}^0)$). Now, let $\hat{S} \supsetneq \tilde{S}$ be a first-stage solution

with $E' = \hat{S} \setminus \tilde{S}$. Consider any scenario $k \in \mathcal{K}$ and let $\hat{S}^k$ be the optimum second-stage solution w.r.t. $\hat{S}$ with cost $\hat{q}^k$; analogously, $\hat{q} := \sum_{k\in\mathcal{K}} p^k \hat{q}^k$. Clearly, $\hat{S}^k \cup E'$ is a feasible solution to scenario k and first-stage solution $\tilde{S}$, as depicted by Figure 7.3.

This relationship allows us to deduce the following lemma.

Lemma 7.10. *Let $\hat{S}, \tilde{S}$ denote first-stage solutions with $\hat{S} \supsetneq \tilde{S}$, $E' = \hat{S} \setminus \tilde{S}$, and second-stage costs $\hat{q}$ and $\tilde{q}$, respectively. It holds $\hat{q} \geq \tilde{q} - \sum_{e\in E'} c^*_e$.*

Proof. Consider any scenario $k \in \mathcal{K}$ and first-stage solution $\tilde{S}$. Since $\tilde{q}^k$ is the optimum solution cost of (SP(k, $\tilde{x}^0$)) and $\hat{S}^k$ is a feasible solution to scenario k and first-stage solution $(\tilde{S} \cup E')$ it holds $\tilde{q}^k \leq \hat{q}^k + \sum_{e\in E'} c^k_e$. Hence:

$$\begin{aligned}\tilde{q} = \sum_{k\in\mathcal{K}} p^k \tilde{q}^k &\leq \sum_{k\in\mathcal{K}} p^k \left(\hat{q}^k + \sum_{e\in E'} c^k_e\right) \\ &= \sum_{k\in\mathcal{K}} p^k \hat{q}^k + \sum_{e\in E'} c^*_e = \hat{q} + \sum_{e\in E'} c^*_e\end{aligned}$$

Rearranging the equation gives the desired statement. □

The preceding Lemma 7.10 leads to another type of integer optimality cuts:

$$\theta + \sum_{e\notin\tilde{S}} c^*_e x^0_e \geq \tilde{q} \tag{7.15}$$

Lemma 7.11. *Constraint* (7.15) *is a valid integer optimality cut.*

Proof. Inserting a first-stage solution $\check{S} \subseteq \tilde{S}$ gives the valid bound $\theta \geq \tilde{q}$ (Observation 7.7). The bound for a solution $\hat{S} \supsetneq \tilde{S}$, with $E' := \hat{S}\setminus\tilde{S}$, is $\theta \geq \tilde{q} - \sum_{e\in E'} c^*_e$, which is also valid due to Lemma 7.10.

Now consider any other solution $\bar{S}$ and the intersection S_1 of $\bar{S}$ and $\tilde{S}$, $S_1 := \bar{S} \cap \tilde{S}$, which is possibly empty. Due to Observation 7.7 it holds that $\tilde{q}$ is a valid lower bound for the second-stage cost of S_1. Then, we can apply Lemma 7.10 with S_1 as base solution, with $\tilde{q}$ as lower bound on the second-stage cost, and with $E' := \bar{S} \setminus S_1$. It follows that $\tilde{q} - \sum_{e\in E'} c^*_e$ is a valid lower bound for the second-stage cost of S_1 and hence, since $S_1 \subseteq \bar{S}$, it is a valid lower bound for $\bar{q}$. □

By considering the minimum of c^*_e and $\tilde{q} - L$, for all $e \in E$, constraints (7.15) can be strengthened as follows:

$$\theta + \sum_{e\notin\tilde{S}} \min\left\{c^*_e, \tilde{q} - L\right\} x^0_e \geq \tilde{q} \tag{Ic$^+$}$$

Lemma 7.12. *Constraint* (Ic$^+$) *is a valid integer optimality cut.*

Proof. If for all edges $e \notin \tilde{S}$ it holds $\min\{c^*_e, \tilde{q} - L\} = c^*_e$ then (Ic$^+$) is equivalent to (7.15) and Lemma 7.11 implies the validity. Hence, assume there exists an edge $e_1 \notin \tilde{S}$ with $\tilde{q} - L < c^*_{e_1}$ and consider any solution $\bar{S}$ with $e_1 \in \bar{S}$. Then, constraint (Ic$^+$) states that $\theta \geq L - r$, with $r \geq 0$, which is a valid bound on θ. If the number of edges with the property $\tilde{q} - L < c^*_e$ is greater than one then the right-hand side gets even smaller (since $\tilde{q} \geq L$). □

Lemma 7.13. *Constraint* (Ic$^+$) *is not weaker than constraint* (Ic$^-$). *If there exists an edge* $e_1 \notin \tilde{S}$ *with* $c^*_{e_1} < \tilde{q} - L$ *then* (Ic$^+$) *is stronger than* (Ic$^-$).

Proof. First, the right-hand side of both constraints is equal and θ has the same coefficient. Second, the coefficient of every edge $e \in E$ in (Ic$^+$) is not greater than the one in (Ic$^-$). Third, if there exists an edge $e_1 \notin \tilde{S}$ with $c^*_{e_1} < \tilde{q} - L$ then the coefficient of e_1 is strictly smaller in (Ic$^+$). □

The next Lemma follows directly from Lemmata 7.9 and 7.13.

Lemma 7.14. *If it holds* $\tilde{q} > L$ *and* $|\tilde{S}| \geq 1$*, then the integer optimality cut* (Ic$^+$) *is stronger than* (Ic).

7.2.3 Further optimality cuts

No-good cuts. During the two-stage b&c algorithm, cf. Section 2.3.3, it is sometimes possible to identify and forbid the current (non-optimal) integer first-stage solution. Notice that after *Step 3.3.1* a feasible and overall integer solution is computed. In particular, the exact second-stage cost w.r.t. the current first-stage solution is known. Hence, it is valid to insert another optimality cut in *Step 3.3.3* to prevent the algorithm from considering the same first-stage solution again.

Consider the following optimality cut which we refer to as *no-good cut*, see, e.g., D'Ambrosio, Frangioni, Liberti, and Lodi [54] and references therein.

$$\sum_{i \in \tilde{S}} x^0_i - \sum_{i \notin \tilde{S}} x^0_i \leq |\tilde{S}| - 1 \tag{Ng}$$

These cuts are considered in several publications, e.g., Codato and Fischetti [48] used the constraints in a similar way and called them *combinatorial Benders cuts*.

The constraints (Ng) do not contain the θ variable and are important when implementing an exact algorithm since the coefficients of these cuts are all binary and hence, they are numerically more stable, see Section 10.2. The validity of these cuts is obvious since only the current first-stage solution sums up to $|\tilde{S}|$ on the left-hand side; for any binary first-stage solution $\bar{x}^0 \neq \tilde{x}^0$, i.e., $\exists i \colon \bar{x}^0_i \neq \tilde{x}^0_i$, it holds $\sum_{i \in \tilde{S}} \bar{x}^0_i - \sum_{i \notin \tilde{S}} \bar{x}^0_i \leq |\tilde{S}| - 1$.

The integer optimality cut (Ic) gives only a strong bound for θ for the current first-stage solution $\tilde{x}^0$; for every other solution the constraint implies at most the trivial bound L. On the other hand, (Ng) cuts off any solution $(\theta, \tilde{x}^0)$.

Lemma 7.15. *Constraint* (Ng) *together with the trivial constraint* $\theta \geq L$ *is stronger than constraint* (Ic).

Proof. Consider the following reformulation of (Ic):

$$\theta \geq (\tilde{q} - L)\left(\sum_{e \in \tilde{S}} x^0_e - \sum_{e \notin \tilde{S}} x^0_e\right) - (\tilde{q} - L)(|\tilde{S}| - 1) + L$$

For every $(\bar{\theta}, \bar{x}^0)$ valid for (Ng) with $\bar{\theta} \geq L$ it holds $\sum_{i \in \tilde{S}} \bar{x}^0_i - \sum_{i \notin \tilde{S}} \bar{x}^0_i \leq |\tilde{S}| - 1$. When inserting $\bar{x}^0$ into the corresponding integer optimality cut (Ic) the right-hand side of (Ic) is at most L, i.e., $\theta \geq L$. Hence, $(\bar{\theta}, \bar{x}^0)$ is valid for (Ic).

On the other hand, the solution $(\tilde{\theta}, \tilde{x}^0)$ with $\tilde{\theta} = \tilde{q}$ is valid for (Ic) but not valid for (Ng). □

Notice that constraints (Ng) can only be inserted into the first-stage master problem when the exact second-stage cost is known. The integer optimality cuts can even be inserted when the second-stage cost is only given as a lower bound, e.g., when the second stage is solved as relaxed problem or by a heuristic.

Cut-based constraints. The following first-stage constraints can be derived from valid cuts in the scenarios. Consider any scenario $k \in \mathcal{K}$ and a valid cut $\emptyset \neq S \subseteq V_r^k$. A feasible overall solution has to satisfy S with a first-stage edge $e_1 \in \delta(S)$ or by using an edge e_2 in scenario k at cost $p^k c_e^k$. Now, let $\mathcal{K}_S$ denote the set of scenarios for which S is a valid cut and let $c_{\delta(S)}^k$ denote the minimum cost of an edge in S for scenario $k \in \mathcal{K}_S$, i.e., $c_{\delta(S)}^k := \min\{c_e^k \mid e \in \delta(S)\}, \forall k \in \mathcal{K}_S$.

Therefore, a feasible first-stage solution (x^0, θ) has to satisfy $\sum_{e\in\delta(S)} x_e^0 \geq 1$ or $\theta \geq \sum_{k\in\mathcal{K}_S} p^k c_{\delta(S)}^k$. With $c_{\delta(S)}^* := \sum_{k\in\mathcal{K}_S} p^k c_{\delta(S)}^k$ this leads to the following *cut-based constraints*:

$$\theta + \sum_{e\in\delta(S)} c_{\delta(S)}^* x_e^0 \geq c_{\delta(S)}^* \tag{Cc}$$

Lemma 7.16. (Cc) *is a valid constraint for the first-stage master problem.*

Proof. If a first-stage solution $\bar{x}^0$ does not contain any edge $e \in \delta(S)$ the cut states $\theta \geq c_{\delta(S)}^*$. This cost is the lowest possible cost to satisfy the given cut in each scenario and hence, this cost is a valid lower bound for the second-stage cost. On the other hand, if $\exists e \in \delta(S)\colon \bar{x}_e^0 = 1$ then (Cc) gives at most the trivial bound $\theta \geq 0$. □

Notice that—in contrast to all other optimality cuts—constraints (Cc) do not depend on the current first-stage solution. Hence, these constraints are the only constraints that can be inserted into the first-stage master problem at any time. In particular, they can be used to strengthen the initial first-stage master problem; one only needs to find a valid cut set for at least one scenario.

The constraints (Cc) can be expanded by considering several edge-disjoint cuts. Let $S_1, S_2, \ldots, S_\ell, \ell \geq 2$, be valid and pairwise edge-disjoint cuts and $c_{\delta(S_i)}^*$ be defined as above, $\forall i \in \{1, \ldots, \ell\}$. Then, the cut-based constraint reads as follows:

$$\theta + \sum_{1\leq i\leq\ell} \sum_{e\in\delta(S_i)} c_{\delta(S_i)}^* x_e^0 \geq \sum_{1\leq i\leq\ell} c_{\delta(S_i)}^* \tag{Cc}$$

Open problem 7.2. Is there another type of constraints or optimality cuts that is independent of the current first-stage solution?

Improved integer optimality cuts. Laporte and Louveaux [117] introduced *improved integer optimality cuts* which can be derived when "more information is available on $Q(x^0)$"; here, we have $\tilde{q} = Q(\tilde{x}^0)$. With a parameter a defined later the cut reads as follows:

$$\begin{aligned} \theta &\geq a\left(\sum_{e\in\tilde{S}} x_e^0 - \sum_{e\notin\tilde{S}} x_e^0\right) + \tilde{q} - a|\tilde{S}| \\ \Leftrightarrow \theta - \sum_{e\in\tilde{S}} a x_e^0 + \sum_{e\notin\tilde{S}} a x_e^0 &\geq \tilde{q} - a|\tilde{S}| \end{aligned} \tag{iIc}$$

Thereby, $a := \max\{\tilde{q} - \lambda(1,\tilde{S}), (\tilde{q} - L)/2\}$ with $\lambda(1,\tilde{S}) \leq Q(\bar{x}^0) = \bar{q}$ for all feasible first-stage solutions $\bar{x}^0$ with $\sum_{i\in\tilde{S}} \bar{x}_i^0 - \sum_{i\notin\tilde{S}} \bar{x}_i^0 = |\tilde{S}| - 1$ (all *1-neighbors* of $\tilde{x}^0$). Hence, $\lambda(1,\tilde{S})$ is a valid lower bound for the second-stage cost for all 1-neighbors. Compared to (Ic) these constraints improve the bound on all 1-neighbors of $\tilde{x}^0$ if $\lambda(1,\tilde{S}) > L$: for $\tilde{x}^0$ the cut states $\theta \geq \tilde{q}$, for all 1-neighbors the bound is $\theta \geq \lambda(1,\tilde{S})$, and for alle s-neighbors, $s \geq 2$, the bound is at most $\theta \geq L$.

These constraints highly depend on a good estimate for $\lambda(1,\tilde{S})$ which is a lower bound on the second-stage cost for all feasible 1-neighbors of the current first-stage solution. This is obviously problem-dependent and may be difficult to compute in practice.

For the SSTP it is possible to combine the bounds given by Observation 7.7 and Lemma 7.10. Let $N(1,\tilde{S})$ denote the set of 1-neighbors of $\tilde{S}$: $N(1,\tilde{S}) := \{x^0 \mid \sum_{e\in\tilde{S}} x_e^0 - \sum_{e\notin\tilde{S}} x_e^0 = |\tilde{S}| - 1\}$. Moreover, let $\tilde{c}^*_{\max} := \max\{c_e^* \mid e \in E\backslash\tilde{S}\}$.

Lemma 7.17. *Let $\bar{x}^0 \in N(1,\tilde{S})$. It holds $\bar{q} \geq \tilde{q} - \tilde{c}^*_{\max}$.*

Proof. Since $\bar{x}^0$ is a 1-neighbor of $\tilde{x}^0$ there exists exactly one edge e_1 with $\bar{x}^0_{e_1} \neq \tilde{x}^0_{e_1}$. We distinguish between two cases.

(i) $e_1 \in \tilde{S}$, $\tilde{x}^0_{e_1} = 1$, and $\bar{x}^0_{e_1} = 0$. By Observation 7.7 we get $\bar{q} \geq \tilde{q}$.

(ii) $e_1 \notin \tilde{S}$, $\tilde{x}^0_{e_1} = 0$, and $\bar{x}^0_{e_1} = 1$. Using Lemma 7.10 with $E' = \{e_1\}$ we get $\bar{q} \geq \tilde{q} - c^*_{e_1}$. Since we need a lower bound for all 1-neighbors $\bar{q} \geq \tilde{q} - \tilde{c}^*_{\max}$ is valid. □

The preceding lemma gives a valid bound for $\lambda(1,\tilde{S})$ and a feasible value for the parameter a in (iIc): $\lambda(1,\tilde{S}) \geq \tilde{q} - \tilde{c}^*_{\max}$ and constraint (iIc) holds with $a = \max\{\tilde{c}^*_{\max}, (\tilde{q} - L)/2\}$. Unfortunately, using this value for a does not improve the bounds given by (Ic^+) which will be argued in the following.

Laporte and Louveaux [117] showed that the bound for an s-neighbor of $\tilde{S}$ is $\tilde{q} - as$. Hence, for $\tilde{S}$ the bound is still $\theta \geq \tilde{q}$. With a defined as above, for a 1-neighbor the bound is at most $\theta \geq \tilde{q} - c^*_{\max}$. And for an s-neighbor, $s \geq 2$, the bound is still at most $\theta \geq L$.

On the other hand, (Ic^+) implies the same bound for $\tilde{S}$ and obviously not weaker bounds for s-neighbors, $s \geq 2$. Notice that in the latter case it is possible that the bound is stronger.

The case of a 1-neighbor deserves a closer look. First, consider $\bar{S}$ with $\tilde{S} = \bar{S} \cup \{e_1\}$. In this case ($\text{Ic}^+$) gives the stronger bound $\theta \geq \tilde{q}$.

Second, for $\bar{S} = \tilde{S} \cup \{e_1\}$ (Ic^+) states that $\theta \geq \tilde{q} - \min\{c^*_{e_1}, \tilde{q} - L\}$. If $c^*_{e_1}$ is the minimum then (Ic^+) does not give a weaker bound since $c^*_{e_1} \leq c^*_{\max}$. If $\tilde{q} - L$ is the minimum we have $\tilde{q} - L \leq c^*_{e_1} \leq c^*_{\max} \Leftrightarrow L \geq \tilde{q} - c^*_{\max}$; therefore, (iIc) does not improve the trivial bound $\theta \geq L$.

Open problem 7.3. Is there a better lower bound for the second-stage cost of 1-neighbors leading to stronger improved integer optimality cuts?

7.2.4 Disaggregated optimality cuts

As for the L-shaped optimality cuts (cf. Section 2.3.2 and Section 7.2.1) it is possible to disaggregate all θ-related (integer optimality) cuts into K cuts, one for each scenario. We always use the prefix "D-" to indicate a disaggregated constraint. As $\theta = \sum_{k\in\mathcal{K}} p^k\theta^k$ one has to consider $\tilde{q}^k = Q^k(\tilde{x}^0)$ instead of $\tilde{q} = \sum_{k\in\mathcal{K}} p^k\tilde{q}^k$. Analogously to the global lower bound L we assume there exists a lower bound L^k for the second-stage cost of each scenario k. Again, it is valid to assume $L^k \geq 0, \forall k \in \mathcal{K}$.

Then, the classical integer optimality cuts (Ic) as disaggregated cuts read as follows:

$$\theta^k \geq (\tilde{q}^k - L^k)\left(\sum_{e\in\tilde{S}} x_e^0 - \sum_{e\notin\tilde{S}} x_e^0 - |\tilde{S}| + 1\right) + L^k$$
$$\Leftrightarrow \theta^k + \sum_{e\in\tilde{S}}(L^k - \tilde{q}^k)x_e^0 + \sum_{e\notin\tilde{S}}(\tilde{q}^k - L^k)x_e^0 \geq (L^k - \tilde{q}^k)(|\tilde{S}| - 1) + L^k \tag{D-Ic}$$

Observation 7.18. *Constraint* (D-Ic) *is a valid integer optimality cut.*

Proof. The proof is similar to the one given in [20] about the validity of (Ic) and sketched as follows: For $\tilde{x}^0$ the cut states $\theta^k \geq \tilde{q}^k$ which is a valid bound since $\tilde{q}^k = Q^k(\tilde{x}^0)$. For every other first-stage solution the bound on θ^k is at most L^k. □

Similarly, one can decompose the new integer optimality cuts (Ic^-) and (Ic^+). For the constraints (Ic^-) we can observe a similar result as given by Observation 7.7, namely: $\tilde{q}^k$ is a lower bound for the second-stage cost of scenario k for any solution $\check{S}$ with $\check{S} \subseteq \tilde{S}$. Then, the disaggregated cut is the result of a simple replacement of $\tilde{q}$ by $\tilde{q}^k$:

$$\theta^k + (\tilde{q}^k - L^k)\sum_{e\notin\tilde{S}} x_e^0 \geq \tilde{q}^k \tag{D-Ic$^-$}$$

Following our discussions it is clear that constraints (D-Ic^-) are valid integer optimality cuts. We conclude without proof:

Observation 7.19. *Constraint* (D-Ic^-) *is a valid integer optimality cut.*

For the disaggregation of (Ic^+) we recall a result of Lemma 7.10: $\tilde{q}^k \leq \hat{q}^k + \sum_{e\in E'} c_e^k$ which holds for a first-stage solution $\hat{S}$ with $\hat{S} = \tilde{S} \cup E'$, $E' \neq \emptyset$, and $E' \cap \tilde{S} = \emptyset$. This leads to the disaggregation of constraints (7.15):

$$\theta^k + \sum_{e\notin\tilde{S}} c_e^k x_e^0 \geq \tilde{q}^k. \tag{D-7.15}$$

Analogously to the aggregated version, these cuts can be strengthened by using the minimum of c_e^k and $\tilde{q}^k - L^k$ as coefficient for each edge $e \notin \tilde{S}$:

$$\theta^k + \sum_{e\notin\tilde{S}} \min\left\{c_e^k, \tilde{q}^k - L^k\right\} x_e^0 \geq \tilde{q}^k \tag{D-Ic$^+$}$$

Again, we skip the proof of validity which is very similar.

Observation 7.20. *Constraint* (D-Ic^+) *is a valid integer optimality cut.*

Last but not least, a disaggregated cut-based constraint for a scenario k w.r.t. the cuts $S_1, \ldots, S_\ell$, $\ell \geq 1$, with $c^k_{\delta(S_i)} := \min\{c_e^k \mid e \in \delta(S_i)\}, \forall i \in \{1, \ldots, \ell\}$, reads as follows:

$$\theta^k + \sum_{1\leq i\leq \ell}\sum_{e\in\delta(S_i)} c^k_{\delta(S_i)} x_e^0 \geq \sum_{1\leq i\leq\ell} c^k_{\delta(S_i)} \tag{D-Cc}$$

Observation 7.21. (D-Cc) *is a valid constraint for the first-stage master problem.*

7.2.5 Obtaining lower bounds for the second-stage cost

The integer optimality cuts (Ic), (Ic$^-$), (Ic$^+$), and (iIc) contain the global lower bound L on the second-stage cost. Improving this bound leads to stronger optimality cuts. However, a feasible solution for the stochastic Steiner tree problem can be constructed by connecting all terminals already in the first stage, and with empty second-stage edge sets. Since all edge costs are assumed to be positive a feasible value is $L = 0$.

During the two-stage b&c algorithm branching on the first-stage variables allows for improvements on this bound. Consider a branching step on a first-stage variable, say $x^0_{e_1}$ corresponding to edge $e_1 \in E$. In the first branch $x^0_{e_1} = 1$ is imposed. By adding an edge to the first stage the second-stage cost cannot increase and L cannot be improved.

However, the second branch with the constraint $x^0_{e_1} = 0$ might lead to a stronger L. Consider a scenario $k \in \mathcal{K}$ and a valid cut $\emptyset \neq S \subseteq V^k_r$ and assume that we are in a node of the b&b tree where all edge variables contained in S are fixed to 0. In this case, the graph induced by the current solution is disconnected and any scenario k where S is a valid cut, i.e., $k \in \mathcal{K}_S$, has to install at least one edge crossing the cut. Hence, for this branch L can be increased by $c^*_{\delta(S)} = \sum_{k\in\mathcal{K}_S} p^k c^k_{\delta(S)}$ with $c^k_{\delta(S)} = \min\{c^k_e \mid e \in \delta(S)\}, \forall k \in \mathcal{K}_S$; for the definitions compare with the paragraph introducing the cut-based constraints (Cc).

Notice that this setting is indeed possible: Since the first-stage master problem contains integer and L-shaped optimality cuts it is possible that the current first-stage solution $\tilde{x}^0$ satisfies $0 < \tilde{x}^0_{e_1} < 1$ although e_1 is the only edge in the cut with an unfixed variable.

Since the experiments on the SSTP showed that the two-stage b&c mostly has a low number of b&b nodes, see Section 8.3, we do not investigate further methods for improving the lower bound L and close this discussion with the following open problem.

Open problem 7.4. Is it possible to further improve the lower bound L (during the two-stage branch&cut algorithm)?

7.3 Adaptation to other formulations

In the previous descriptions the focus lies on the semi-directed formulation (SSTP$_{\text{sdc2}}$). In this section we describe the differences and necessary modifications for the other introduced formulations. Section 7.3.1 considers the semi-directed SSTP models and Section 7.3.2 contains the description of the directed rSSTP models.

7.3.1 Semi-directed formulations

(SSTP$_{\text{sdc1}}$). In (SSTP$_{\text{sdc1}}$) a subproblem $k \in \mathcal{K}$ contains (in its basic version) only the directed cuts (6.9). For a first-stage solution $\tilde{x}^0$ and a valid cut $S \subseteq V^k_r \colon S \cap T^k_r \neq \emptyset$, the constraint is $z^k(\delta^-(S)) \geq 1 - \tilde{x}^0(\delta(S))$. This implies two main differences: (i) the dual and hence, the L-shaped cuts, and (ii) the separation of the directed cuts.

Regarding (i) the dual only contains variables α^k_S, one for each cut $S \in \mathcal{S}^k$, with $\mathcal{S}^k$ being the set of valid cuts. The objective function then reads $\max \sum_{S\in\mathcal{S}^k} \alpha^k_S(1 - \tilde{x}^0(\delta(S))$ and the generated disaggregated L-shaped cut for the dual solution $\tilde{\alpha}^k$ is

$$\theta^k + \sum_{e\in E} x^0_e \left(\sum_{S\in\mathcal{S}^k \colon e\in\delta(S)} \tilde{\alpha}^k_S \right) \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S \tag{7.16}$$

Hence, the proposed cut strengthening (Section 7.2.1) is not applicable and we are not aware of any similar and fast method for strengthening the optimality cuts. The method *Laminarize*, on the other hand, can be used for improving the dual solution heuristically.

(ii) The separation of directed cuts (6.9), described in detail in Section 8.1.3, needs a minor modification. Here, the capacity of an arc $(i, j) \in A$, with $e = \{i, j\} \in E$, is set to $\tilde{z}^k_{ij} + \tilde{x}^0_e$ for the computation of the minimum cuts.

Theorem 6.13 shows that ($\text{SSTP}_{\text{sdc1}}$) is weaker than ($\text{SSTP}_{\text{sdc2}}$). This statement also holds for the decomposition. The subproblem of ($\text{SSTP}_{\text{sdc1}}$) is weaker (gives worse bounds) and hence, the generated L-shaped optimality cuts are weaker.

($\text{SSTP}_{\text{sdc2}^*}$). The obvious difference to model ($\text{SSTP}_{\text{sdc2}}$) lies in the objective function. In ($\text{SSTP}_{\text{sdc2}}$) the term $-\sum_{e\in E} c^*_e x^0_e$ is contained in the subproblem whereas in ($\text{SSTP}_{\text{sdc2}^*}$) the term is part of the master problem, i.e, the objective function reads min $\sum_{e\in E}(c^0_e - c^*_e)x^0_e + \theta$.

This influences several parts of the decomposition. In the first stage, the edge coefficients are negative and, in particular at the beginning of a b&c algorithm with few L-shaped optimality cuts, the first-stage solutions contain many edges. Moreover, the primal and dual subproblems do not subtract the constant term anymore. For scenario $k \in \mathcal{K}$ and a first-stage solution $\tilde{x}^0 \in [0, 1]^{|E|}$ the primal objective function reads as follows: min $\sum_{e=\{i,j\}\in E} c^k_e(y^k_{ij} + y^k_{ji})$; the constraints are identical to (RSP_{sdc2}).

Then, with $(\tilde{\alpha}^k, \tilde{\beta}^k)$ as optimum solution to the dual subproblem a disaggregated L-shaped optimality cut looks as follows:

$$\theta^k + \sum_{e\in E}(-\tilde{\beta}^k_e)x^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S \tag{7.17}$$

Notice that the same cut strengthening method as described in (7.11) is still valid and, if applicable, it leads to a stronger L-shaped cut.

Similar to before, let ($\text{RMP}_{\text{sdc2}^*}$) and ($\text{RSP}_{\text{sdc2}^*}$) denote the relaxed master problem and relaxed subproblem of ($\text{SSTP}_{\text{sdc2}^*}$) Although both models are equivalent it is not reasonable to compare the polytopes of (RMP_{sdc2}) and ($\text{RMP}_{\text{sdc2}^*}$) directly since the master polytope of ($\text{SSTP}_{\text{sdc2}}$) is a strict superset of the master polytope of ($\text{SSTP}_{\text{sdc2}^*}$). The reason is that both models enforce the selection of used first-stage edges in the second stage due to constraints $y^k_{ij} + y^k_{ji} \geq \tilde{x}^0_e, \forall e = \{i, j\} \in E$, but ($\text{RSP}_{\text{sdc2}}$) subtracts this cost and gives the *true* second-stage cost whereas ($\text{RSP}_{\text{sdc2}^*}$) gives the cost of the solution as if it would be bought solely in the second stage. Hence, for any $\tilde{x}^0 \in [0, 1]^{|E|}$ the second-stage cost (the lowest feasible value for θ) for ($\text{RMP}_{\text{sdc2}^*}$) is not lower than the cost for (RMP_{sdc2}).

This comparison raises the question if both models are equally strong under the decomposition. To answer this question we remark that the relaxed master problem ($\text{RMP}_{\text{sdc2}^*}$) of ($\text{SSTP}_{\text{sdc2}^*}$) is basically the same as the one of ($\text{SSTP}_{\text{sdc2}}$). However, it is possible to expand ($\text{RMP}_{\text{sdc2}^*}$) by the following constraints, one for each scenario $k \in \mathcal{K}$, which induce stronger lower bounds for the second-stage costs.

$$\theta^k \geq \sum_{e\in E} c^k_e x^0_e \tag{7.18}$$

First, we observe the correctness of the newly added constraints.

Observation 7.22. *Constraints* (7.18) *are valid for* ($\text{RMP}_{\text{sdc2}^*}$).

Proof. ($\text{RSP}_{\text{sdc2}^*}$) contains the constraints $y^k_{ij} + y^k_{ji} \geq \tilde{x}^0_e, \forall e = \{i,j\} \in E$. Hence, for a first-stage solution $\tilde{x}^0 \in [0,1]^{|E|}$ the objective value of the kth subproblem is $\sum_{e\in E} c^k_e(y^k_{ij}+y^k_{ji}) \geq \sum_{e\in E} c^k_e \tilde{x}^0_e$. □

Now, we are able to compare both decompositions.

Lemma 7.23. *The decompositions and in particular, the L-shaped optimality cuts, of models ($SSTP_{sdc2}$) and ($SSTP_{sdc2^*}$) are equally strong.*

Proof. We show (a stronger result) that (RMP_{sdc2}) and ($\text{RMP}_{\text{sdc2}^*}$) consider the same first-stage solutions, if three assumptions concerning the computation of optimum solutions are satisfied. We highlight the assumptions by (A1)–(A3) in the following.

In the first iteration the set of optimality cuts is empty. Then, the optimum solution to (RMP_{sdc2}) is ($x^0 = \mathbf{0}, \theta = \mathbf{0}$) and the optimum solution value to ($\text{RMP}_{\text{sdc2}^*}$) is also 0. We assume that (A1) the selected optimum solution to ($\text{RMP}_{\text{sdc2}^*}$) is ($x^0 = \mathbf{0}, \theta = \mathbf{0}$).

Since (RSP_{sdc2}) and ($\text{RSP}_{\text{sdc2}^*}$) and their duals are equivalent (except for the constant term), the generated L-shaped optimality cuts are equivalent in the sense that the values to the dual variables α, β are identical. Here, we assume that (A2) for each first-stage solution $\tilde{x}^0$ both models always generate the same directed cuts and find the same optimum (primal and dual) solution. Hence, if an L-shaped cut $\theta^k + \sum_{e\in E}(c^k_e - \tilde{\beta}^k_e)x^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S$ is generated for (RMP_{sdc2}) and some $\tilde{x}^0$ then there exists an L-shaped cut $\theta^k + \sum_{e\in E}(-\tilde{\beta}^k_e)x^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S$ in ($\text{RMP}_{\text{sdc2}^*}$), and vice versa.

Now assume we are at some iteration and all previously considered first-stage solutions are identical. We argue that the next first-stage solution is identical, too.

"$\Rightarrow$": Let $(\hat{x}^0, \hat{\theta}, \hat{\theta}^1, \ldots, \hat{\theta}^K)$ be an optimum solution to (RMP_{sdc2}). We show that $(\bar{x}^0, \bar{\theta}, \bar{\theta}^1, \ldots, \bar{\theta}^K)$ with $\bar{x}^0 := \hat{x}^0$, $\bar{\theta}^k := \hat{\theta}^k + \sum_{e\in E} c^k_e \hat{x}^0_e$, and $\bar{\theta} := \sum_{k\in\mathcal{K}} p^k \bar{\theta}^k$, is valid to ($\text{RSP}_{\text{sdc2}^*}$) and has to same objective value.

Clearly, it holds $\bar{x}^0 \in [0,1]^{|E|}$ and $(\bar{\theta}, \bar{\theta}^1, \ldots, \bar{\theta}^K) \geq \mathbf{0}$ as well as the constraint $\bar{\theta} \geq \sum_{k\in\mathcal{K}} p^k \bar{\theta}^k$. Moreover, $\forall k \in \mathcal{K}$: $\bar{\theta}^k = \hat{\theta}^k + \sum_{e\in E} c^k_e \hat{x}^0_e \geq \sum_{e\in E} c^k_e \bar{x}^0_e$, hence, constraints (7.18) are satisfied. It is easy to see that the objective values are the same:

$$\sum_{e\in E}(c^0_e - c^*_e)\bar{x}^0_e + \bar{\theta} = \sum_{e\in E}(c^0_e - c^*_e)\hat{x}^0_e + \sum_{k\in\mathcal{K}} p^k(\hat{\theta}^k + \sum_{e\in E} c^k_e \hat{x}^0_e) = \sum_{e\in E} c^0_e \hat{x}^0_e + \hat{\theta}$$

Last but not least, consider an L-shaped optimality cut w.r.t. scenario k and dual solution $(\tilde{\alpha}, \tilde{\beta})$ in ($\text{RMP}_{\text{sdc2}^*}$):

$$\begin{aligned} &\bar{\theta}^k + \sum_{e\in E}(-\tilde{\beta}^k_e)\bar{x}^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S \\ \Leftrightarrow\ &\hat{\theta}^k + \sum_{e\in E} c^k_e \hat{x}^0_e + \sum_{e\in E}(-\tilde{\beta}^k_e)\hat{x}^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S \\ \Leftrightarrow\ &\hat{\theta}^k + \sum_{e\in E}(c^k_e - \tilde{\beta}^k_e)\hat{x}^0_e \geq \sum_{S\in\mathcal{S}^k} \tilde{\alpha}^k_S \end{aligned}$$

"$\Leftarrow$": For an optimum solution $(\bar{x}^0, \bar{\theta}, \bar{\theta}^1, \ldots, \bar{\theta}^K)$ to ($\text{RSP}_{\text{sdc2}^*}$) the solution given by $\hat{x}^0 := \bar{x}^0$, $\hat{\theta}^k := \bar{\theta}^k - \sum_{e\in E} c^k_e \bar{x}^0_e$, $\hat{\theta} := \sum_{k\in\mathcal{K}} p^k \hat{\theta}^k$, is valid to ($\text{RMP}_{\text{sdc2}}$). The preceding transformations show that both objective values are identical and that each L-shaped cut is satisfied. Finally, it holds $\hat{\theta}^k \geq 0, \forall k \in \mathcal{K}$, due to constraints (7.18).

We have shown that the same x^0-solution is optimal for both models. If we assume that (A3) both master problems always find this very solution both decompositions behave the same. □

Integer optimality cuts for ($\text{SSTP}_{\text{sdc2}^*}$). As mentioned before ($\text{RSP}_{\text{sdc2}^*}$) gives the cost of the second-stage as if it would be bought solely in the second stage although selected first-stage edges need to be purchased in the second-stage, too. Therefore, Observation (7.7), i.e., the second-stage cost does not decrease when a first-stage edge is removed, is not valid for this formulation. Then, integer optimality cuts (Ic^-) and (Ic^+) are not feasible, too. On the other hand, since the no-good cuts (Ng) are independent of any cost these cuts are valid for ($\text{SSTP}_{\text{sdc2}^*}$).

Now, we introduce new integer optimality cuts which are based on the following observation; it can be seen as counterpart to Observation (7.7).

Observation 7.24. *Let $\tilde{S}$ denote a first-stage solution with second-stage cost $\tilde{q}$. Then, $\tilde{q}$ is a lower bound for the second-stage cost of any solution $\hat{S} \supseteq \tilde{S}$.*

The observation holds since ($\text{RSP}_{\text{sdc2}^*}$) forces the selection of edges and hence, this may lead to a costlier solution. This leads to the following integer optimality cuts:

$$\theta + (\tilde{q} - L) \sum_{e \in \tilde{S}} (1 - x_e^0) \geq \tilde{q} \tag{Ic_*^-}$$

Inserting $\tilde{S}$ into (Ic_*^-) gives the valid bound $\theta \geq \tilde{q}$. A solution $\hat{S} \supsetneq \tilde{S}$ is assigned the bound $\tilde{q}$ which is correct due to Observation 7.24. For any other solution the bound is at most L. Hence, the cuts (Ic_*^-) are valid constraints for the first-stage master problem of model ($\text{SSTP}_{\text{sdc2}^*}$).

Analogously to (Ic^-), these cuts can also be strengthened to cuts (Ic_*^+) as follows. For this purpose, let $\check{S} \subsetneq \tilde{S}$ and $E' := \tilde{S} \setminus \check{S}$. Moreover, let $\tilde{S}^k$, $\check{S}^k$ be the optimum solution to ($\text{SP}(k, \tilde{x}^0)$) and ($\text{SP}(k, \check{x}^0)$), respectively. Notice that $\tilde{S}^k \supseteq \tilde{S}$ and $\check{S}^k \supseteq \check{S}$, due to the capacity constraints in ($\text{RSP}_{\text{sdc2}^*}$).

Lemma 7.25. *Let $\check{S}, \tilde{S}$ denote first-stage solutions with $\check{S} \subsetneq \tilde{S}$, $E' = \tilde{S} \setminus \check{S}$, and second-stage cost $\check{q}$ and $\tilde{q}$, respectively. It holds $\check{q} \geq \tilde{q} - \sum_{e \in E'} c_e^*$.*

Proof. Let $k \in \mathcal{K}$. Since $\tilde{S} \supsetneq \check{S}$ the cost of ($\tilde{S}^k \setminus \tilde{S}$) is at most the cost of ($\check{S}^k \setminus \check{S}$). Moreover, since $\tilde{S} = E' \cup \check{S}$ we get:

$$\begin{aligned} \tilde{q}^k - \sum_{e \in E'} c_e^k - \sum_{e \in \check{S}} c_e^k &\leq \check{q}^k - \sum_{e \in \check{S}} c_e^k \\ \Leftrightarrow\ \tilde{q}^k - \sum_{e \in E'} c_e^k &\leq \check{q}^k \end{aligned}$$

Hence:

$$\begin{aligned} \check{q} = \sum_{k \in \mathcal{K}} p^k \check{q}^k &\geq \sum_{k \in \mathcal{K}} p^k \left(\tilde{q}^k - \sum_{e \in E'} c_e^k \right) \\ &= \tilde{q} - \sum_{e \in E'} c_e^* \end{aligned}$$

□

The preceding lemma leads to the following optimality cuts:

$$\theta + \sum_{e \in \tilde{S}} c_e^* (1 - x_e^0) \geq \tilde{q} \tag{7.19}$$

The constraint gives the bound $\tilde{q}$ to all solutions $\hat{S} \supseteq \tilde{S}$ which is valid due to Observation 7.24. A solution $\check{S} \subseteq \tilde{S}$ is assigned the bound from Lemma 7.25. Finally, for any other solution $\bar{S}$ consider the solution $S_1 = \bar{S} \cap \tilde{S}$. Since $S_1 \subseteq \tilde{S}$ we get a valid lower bound on this cost. Adding the edges in the set $\bar{S} \setminus S_1$ does not change this bound. Hence, we get a correct bound on the cost for $\bar{S}$.

Similar to (Ic^+) it is possible to strengthen the cut to:

$$\theta + \sum_{e \in \bar{S}} \min\{c_e^*, \tilde{q} - L\}(1 - x_e^0) \geq \tilde{q} \tag{Ic^+_*}$$

The arguments for proving the validity are similar to before. If it holds $\min\{c_e^*, \tilde{q} - L\} = c_e^*, \forall e \in \tilde{S}$, the cut is identical to (7.19). Therefore, we now assume there exists $e_1 \in \tilde{S}$ such that $\tilde{q} - L < c_{e_1}^*$ and we consider a solution $\bar{S}$ with $e_1 \notin \bar{S}$. In this case the cut states $\theta \geq L - r$, for some $r \geq 0$.

Last but not least, notice that all introduced L-shaped and integer optimality cuts can be disaggregated, too, as described in Section 7.2.4 for ($\text{SSTP}_{\text{sdc2}}$).

Lower bound for ($\text{SSTP}_{\text{sdc2}^*}$). Let $lb^k, \forall k \in \mathcal{K}$, be a lower bound for the cost of solving scenario k as Steiner tree problem, for example, the value of the relaxed STP $lb^k = R^k(\mathbf{0})$, the STP $lb^k = Q^k(\mathbf{0})$, or any other lower bound from the STP. Then, $L_0 := \sum_{k \in \mathcal{K}} p^k lb^k$ is a valid lower bound for the second-stage cost of model ($\text{SSTP}_{\text{sdc2}^*}$); the reason is that the empty first-stage solution has the cheapest second-stage cost (due to Observation 7.24).

Moreover, contrarily to ($\text{SSTP}_{\text{sdc2}}$), where L can be increased in a zero-branch, it might be possible to improve L in a one-branch. Let E_1 be the set of edges with their edge variables being fixed to 1 by a branching step in the master problem and let $lb_1^k, \forall k \in \mathcal{K}$, be a lower bound for solving scenario k as STP while all edges fixed in E_1 are contracted. Then, $L_1 := \sum_{k \in \mathcal{K}} p^k lb_1^k + \sum_{e \in E_1} c_e^*$ is a valid lower bound for L.

To obtain these bounds one has to solve all scenarios (as Steiner tree problems) to (integer) optimality. However, L_0 should be given "for free" since the first solution should be $\mathbf{0}$ anyway.

7.3.2 Directed formulations

In this section we consider the decomposition of the directed formulations for the rSSTP. We first focus on model ($\text{rSSTP}_{\text{dc2}}$) and afterwards we summarize the differences for models ($\text{rSSTP}_{\text{dc1}}$) and ($\text{rSSTP}_{\text{dc2}^*}$). As we will see the decompositions of all three models are equivalent.

($\text{rSSTP}_{\text{dc2}}$). Decomposing ($\text{rSSTP}_{\text{dc2}}$) works similar to the decomposition of ($\text{SSTP}_{\text{sdc2}}$). The two main differences are (i) the arc variables instead of edge variables and (ii) the additional directed cuts in the master problem.

The relaxed master problem of (rSSTP$_{\text{dc2}}$) reads as follows.

$$\text{(RMP}_{\text{dc2}}\text{)} \ \min \sum_{a \in A} c_a^0 z_a^0 + \theta$$

$$\text{s.t. } \theta \geq \sum_{k \in \mathcal{K}} p^k \theta^k \tag{7.20}$$

$$\text{Optimality cuts} \tag{7.21}$$

$$z^0(\delta^-(S)) \geq z^0(\delta^-(v)) \qquad \forall \emptyset \neq S \subseteq V_r, \forall v \in S \tag{7.22}$$

$$z^0 \in [0,1]^{|A|} \tag{7.23}$$

$$\theta, \theta^1, \ldots, \theta^K \geq 0 \tag{7.24}$$

The relaxed subproblem for scenario $k \in \mathcal{K}$ and first-stage solution $\tilde{z}^0$ is:

$$\text{(RSP}_{\text{dc2}}\text{)} \ \min \sum_{a \in A} c_a^k y_a^k - \sum_{a \in A} c_a^k \tilde{z}_a^0$$

$$\text{s.t. } y^k(\delta^-(S)) \geq 1 \qquad \forall S \subseteq V_r \colon S \cap T_r^k \neq \emptyset \tag{7.25}$$

$$y_a^k \geq \tilde{z}_a^0 \qquad \forall a \in A \tag{7.26}$$

$$y^k \geq \mathbf{0} \tag{7.27}$$

Again, let $\mathcal{S}^k$ be the set of valid directed cuts, i.e., $\mathcal{S}^k = \{S \subseteq V_r \mid S \cap T_r^k \neq \emptyset\}$. We use dual variables α_S^k, $S \in \mathcal{S}^k$, and β_a^k, $a \in A$, assigned to the directed cuts (7.25) and capacity constraints (7.26), respectively.

$$\text{(D:RSP}_{\text{dc2}}\text{)} \ \max \sum_{S \in \mathcal{S}^k} \alpha_S^k + \sum_{a \in A} \tilde{z}_a^0 \beta_a^k - \sum_{a \in A} c_a^k \tilde{z}_a^0$$

$$\text{s.t.} \sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \alpha_S^k + \beta_a^k \leq c_a^k \qquad \forall a \in A \tag{7.28}$$

$$\alpha^k, \beta^k \geq \mathbf{0} \tag{7.29}$$

Let $(\tilde{\alpha}^k, \tilde{\beta}^k)$ denote an optimum solution to (D:RSP$_{\text{dc2}}$) w.r.t. $\tilde{z}^0$ and scenario $k \in \mathcal{K}$. A disaggregated L-shaped optimality cut for formulation (rSSTP$_{\text{dc2}}$) and scenario $k \in \mathcal{K}$ looks as follows:

$$\theta^k + \sum_{a \in A} (c_a^k - \tilde{\beta}_a^k) z_a^0 \geq \sum_{S \in \mathcal{S}^k} \tilde{\alpha}_S^k \tag{7.30}$$

The L-shaped cuts can be strengthened similarly by increasing the values of the dual β^k variables. Here, the procedure is even simpler since a β_a^k variable is affected only by one dual constraint. Hence, if this constraint has a positive slack (in case $\tilde{z}_a^0 = 0$), the related β_a^k variable can be set to $\hat{\beta}_a^k := c_a^k - \sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \tilde{\alpha}_S^k$. Then, this solution is a valid and still optimal dual solution that implies a stronger L-shaped cut.

Figure 7.4 gives an example instance where the generated and strengthened L-shaped cuts are not Pareto optimal. The current first-stage solution is $\mathbf{0}$, the considered scenario contains two terminals a, b with root node r, and all edge costs are 1. The optimum primal solution selects arcs a_1 and a_3 at cost 2. (a) indicates an optimum dual solution with $\hat{\alpha}_{S_b}^k = \hat{\alpha}_{S_{abc}}^k = 1$ and (b) gives an alternative optimum dual solution with $\tilde{\alpha}_{S_{ab}}^k = \tilde{\alpha}_{S_{bc}}^k = 1$. The arrows at the edges indicate the affected arcs and dual variables, respectively.

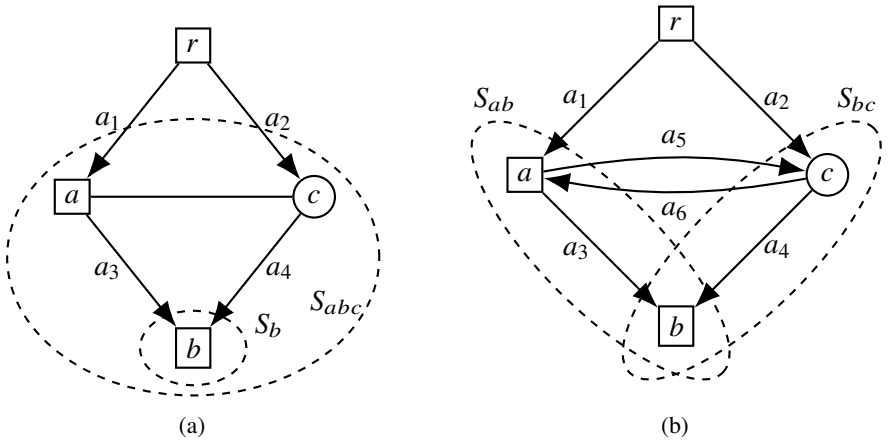


Figure 7.4: Counter example for Pareto optimality of the generated and strengthened L-shaped optimality cuts for model ($\text{rSSTP}_{\text{dc2}}$). The dashed and dotted circles represent the cut sets $S_b, S_{abc}, S_{ab}, S_{bc}$. (a) and (b) give different dual solutions such that the first L-shaped cut dominates the second one.

Hence, L-shaped cut (a) reads $\theta^k + z^0_{a_1} + z^0_{a_2} + z^0_{a_3} + z^0_{a_4} \geq 2$ which dominates cut (b) $\theta^k + z^0_{a_1} + z^0_{a_2} + z^0_{a_3} + z^0_{a_4} + z^0_{a_5} + z^0_{a_6} \geq 2$.

When we apply method *Laminarize* to this example and cut set $\{S_{ab}, S_{bc}\}$, the crossing cuts S_{ab} and S_{bc} are removed, we get the laminar cut set $\{S_b, S_{abc}\}$, and the dual solution as in (a). Hence, in this case *Laminarize* gives a Pareto optimal cut. However, we cannot give a proof—or a counter example—that a laminar and cover-free set of cuts gives a Pareto optimal cut; notice that—with straight-forward modifications—Observation 7.2 is applicable to the directed formulations, too.

Open problem 7.5. Are the strengthened L-shaped cuts for the directed rSSTP models and for a laminar and cover-free set of cuts Pareto optimal?

All introduced constraints for the first stage, as described in Sections 7.2.2–7.2.4, are still valid for the directed model. In general, one only has to replace the edge variables by arc variables. Moreover, since for the SSTP every first-stage solution is feasible and for the rSSTP a connected first stage is required the preconditions and arguments in the observations and lemmata have to be modified slightly such that only feasible tree solutions are considered.

($\text{rSSTP}_{\text{dc2}^*}$). Comparing both directed models ($\text{rSSTP}_{\text{dc2}}$) and ($\text{rSSTP}_{\text{dc2}^*}$) works analogously to the semi-directed case. Again, subproblem (RSP_{dc2}) gives the true second-stage cost whereas the cost of ($\text{RSP}_{\text{dc2}^*}$) is increased by $\sum_{a\in A} c^k_a z^0_a$. This very sum is contained in the subproblem (RSP_{dc2}) and on the other hand, in the master problem ($\text{RMP}_{\text{dc2}^*}$). As discussed for the semi-directed model it is valid to add the following constraints to the master problem of ($\text{rSSTP}_{\text{dc2}^*}$): $\theta^k \geq \sum_{e\in E} c^k_a z^0_a, \forall k \in \mathcal{K}$.

Then, both decompositions are equivalent, as stated by the next Lemma. We skip the proof since it is very similar to the one of Lemma 7.23; we just need to use directed arc variables instead.

Lemma 7.26. *The decompositions and in particular, the L-shaped optimality cuts, of models ($rSSTP_{dc2}$) and ($rSSTP_{dc2^*}$) are equally strong.*

Moreover, the directed equivalent of the new integer optimality cuts (Ic_*^-) and (Ic_*^+) are valid for model ($\text{rSSTP}_{\text{dc2}^*}$) as are the ideas on the lower bound L at the end of Section 7.3.1.

($\text{rSSTP}_{\text{dc1}}$). The decomposition of model ($\text{rSSTP}_{\text{dc1}}$) looks slightly different since this model does not contain the additional capacity constraints in the second stage. However, the relaxed master problem is identical to (RMP_{dc2})—as we will argue this even holds for the L-shaped optimality cuts. The relaxed subproblem for scenario $k \in \mathcal{K}$ and first-stage solution $\tilde{z}^0$ is:

$$
\begin{aligned}
(\text{RSP}_{\text{dc1}})\ \min\ & \sum_{a \in A} c_a^k z_a^k \\
\text{s.t. } & z^k(\delta^-(S)) \geq 1 - \tilde{z}^0(\delta^-(S)) \qquad \forall S \subseteq V_r \colon S \cap T_r^k \neq \emptyset && (7.31)\\
& z^k \geq \mathbf{0} && (7.32)
\end{aligned}
$$

The corresponding dual reads as follows:

$$
\begin{aligned}
(\text{D:RSP}_{\text{dc1}})\ \max\ & \sum_{S \in \mathcal{S}^k} \alpha_S^k (1 - \tilde{z}^0(\delta^-(S))) \\
\text{s.t. } & \sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \alpha_S^k \leq c_a^k \qquad \forall a \in A && (7.33)\\
& \alpha^k \geq \mathbf{0} && (7.34)
\end{aligned}
$$

In the following we use this transformation:

$$
\sum_{S \in \mathcal{S}^k} \alpha_S^k z^0(\delta^-(S)) = \sum_{S \in \mathcal{S}^k} \alpha_S^k \left(\sum_{a \in \delta^-(S)} z_a^0 \right) = \sum_{a \in A} \left(\sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \alpha_S^k \right) z_a^0 \tag{7.35}
$$

Using (7.35) an optimum solution $\tilde{\alpha}^k$ implies the following L-shaped optimality cut:

$$
\theta^k + \sum_{a \in A} \left(\sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \tilde{\alpha}_S^k \right) z_a^0 \geq \sum_{S \in \mathcal{S}^k} \tilde{\alpha}_S^k \tag{7.36}
$$

Lemma 7.27. *The L-shaped optimality cuts of ($rSSTP_{dc1}$) and the strengthened L-shaped cuts of ($rSSTP_{dc2}$) are equivalent.*

Proof. We argue that for any first-stage solution $\tilde{z}^0 \in [0,1]^{|A|}$ and a scenario $k \in \mathcal{K}$ both ($\text{D:RSP}_{\text{dc1}}$) and ($\text{D:RSP}_{\text{dc2}}$) are able to generate the same L-shaped cut.

"$\Rightarrow$": Let $\bar{\alpha}^k$ denote an optimum solution to ($\text{D:RSP}_{\text{dc1}}$). We use $\hat{\alpha}^k := \bar{\alpha}^k$ and $\hat{\beta}_a^k := c_a^k - \sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \bar{\alpha}_S, \forall a \in A$, as a solution to ($\text{D:RSP}_{\text{dc2}}$). As shown in the following, this solution is valid, has the same objective value, and yields the same L-shaped cut.

Validity: $(\hat{\alpha}^k, \hat{\beta}^k) \geq \mathbf{0}$ since $\bar{\alpha} \geq \mathbf{0}$ and $c_a^k - \sum_{S \in \mathcal{S}^k \colon a \in \delta^-(S)} \bar{\alpha}_S \geq 0, \forall a \in A$, due to the dual constraints (7.33).

Solution value:

$$\begin{aligned}
&\sum_{S\in\mathcal{S}^k} \hat{\alpha}_S^k + \sum_{a\in A} \tilde{z}_a^0 \hat{\beta}_a^k - \sum_{a\in A} c_a^k \tilde{z}_a^0 \\
&= \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k + \sum_{a\in A} \tilde{z}_a^0 \left(c_a^k - \sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \bar{\alpha}_S^k \right) - \sum_{a\in A} c_a^k \tilde{z}_a^0 \\
&= \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k - \sum_{a\in A} \tilde{z}_a^0 \left(\sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \bar{\alpha}_S^k \right) \\
&\overset{(7.35)}{=} \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k - \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k \tilde{z}^0(\delta^-(S)) = \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k (1 - \tilde{z}^0(\delta^-(S)))
\end{aligned}$$

L-shaped optimality cut:

$$\begin{aligned}
&\theta^k + \sum_{a\in A} (c_a^k - \hat{\beta}_a^k) z_a^0 \geq \sum_{S\in\mathcal{S}^k} \hat{\alpha}_S^k \\
\Leftrightarrow\ &\theta^k + \sum_{a\in A} \left(c_a^k - \left(c_a^k - \sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \bar{\alpha}_S^k \right) \right) z_a^0 \geq \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k \\
\Leftrightarrow\ &\theta^k + \sum_{a\in A} \left(\sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \bar{\alpha}_S^k \right) z_a^0 \geq \sum_{S\in\mathcal{S}^k} \bar{\alpha}_S^k
\end{aligned}$$

"$\Leftarrow$": With $(\hat{\alpha}^k, \hat{\beta}^k)$ as optimum (and strengthened) solution to (D:RSP$_{\text{dc2}}$) we use $\bar{\alpha}^k := \hat{\alpha}^k$ as solution to (D:RSP$_{\text{dc1}}$). Since $(\hat{\alpha}^k, \hat{\beta}^k)$ is a strengthened solution it holds $\sum_{S\in\mathcal{S}^k\,:\,a\in\delta^-(S)} \hat{\alpha}_S^k + \hat{\beta}_a^k = c_a^k, \forall a \in A$. Then, all previous transformations are still valid. □

Since (D:RSP$_{\text{dc1}}$) only contains the dual variables α the proposed cut strengthening (Section 7.2.1) is not applicable for this model. However, the L-shaped optimality cuts can be improved heuristically by *Laminarize*, see Section 7.2.1, and compare the discussion on model (rSSTP$_{\text{dc2}}$).

Chapter 8

Computational study

In the first part of this chapter we describe details about the implementation (Section 8.1) with the direct approach (Section 8.1.1), the decomposition (Section 8.1.2), the separation of the directed cuts (Section 8.1.3), additional constraints for the second stage (Section 8.1.4), and the implemented primal heuristic (Section 8.1.5). Section 8.2 describes the method for generating the stochastic instances and gives an overview of some instance statistics. The main part of this chapter is Section 8.3 which presents the results of the computational study concerning the SSTP and rSSTP. We evaluate different cutpool strategies in Section 8.3.1, the performance of the semi-directed models—direct approaches and decomposition—for the SSTP in Section 8.3.2 and the directed models for the rSSTP in Section 8.3.3. Moreover, we compare the models for the SSTP and rSSTP and the differences of both problems in Section 8.3.4. In Section 8.3.6 we discuss further experiments and Section 8.3.7 contains detailed results of the experiments.

Again, the following descriptions of the algorithms and the implementation focus on the semi-directed formulation ($SSTP_{sdc2}$) and the decomposition approach. Whenever necessary, we mention necessary modifications for the other formulations or the direct approach, respectively.

8.1 Implementation

All algorithms are implemented in C++ using the branch&cut framework ABACUS in version 3.0, see Jünger and Thienel [103] and [1]. For the representation of graphs and the implementation of further graph algorithms we use the Open Graph Drawing Framework (OGDF), see Chimani, Gutwenger, Jünger, Klau, Klein, and Mutzel [42] and [137]. The only exception is the computation of minimum cuts where we use an efficient implementation of the push-relabel method in C by Cherkassky and Goldberg [36]. Moreover, we use IBM ILOG CPLEX 12.2 [97] via ABACUS and the COIN Open Solver Interface [49] as solver for the linear programs.

8.1.1 Direct approach

We implemented each model as direct approach without using any decomposition. Hence, the direct approach is a single-stage b&c algorithm with a straight-forward implementation similar to the deterministic STP. We have one ABACUS master problem which initializes the basic IP with all first- and second-stage variables. Moreover, an ABACUS subproblem

is generated for each node in the b&b tree. The directed cuts in the scenarios—as well as the directed cuts in the first stage for the directed rSSTP formulations—are separated by the standard procedure described in Section 8.1.3.

8.1.2 Decomposition

For implementing the two-stage b&c algorithm we use two ABACUS master and two ABACUS subproblems: one pair for the first and one pair for the second stage, respectively. We distinguish them by calling the first stage "SSTP master problem" and "SSTP subproblem" and the second stage "STP master problem" and "STP subproblem", respectively.

Since all scenarios depend on the same graph and contain the same type of constraints we decided to use exactly one instance of the STP master problem which is shared by all scenarios. The benefit of this approach is the possibility to skip the initialization of the variables and the basic LP when a scenario is (re)solved and the current first-stage solution has to be initialized only once for all scenarios. Then, each time a scenario is solved the coefficients in the objective function and the basic constraints are updated. Furthermore, the right-hand sides of the separated directed cuts in the cutpool (see Section 8.3.1) are updated, too.

To achieve a good performance we modified ABACUS directly, in particular the classes `ABA_SUB` and `ABA_MASTER`. In the following we describe the modifications to the ABACUS classes and give some details of the SSTP and STP master problems and subproblems, respectively.

`ABA_SUB`. Calling a primal heuristic for the SSTP can be time consuming since all scenarios need to be solved (optimally or heuristically). To restrict the number of calls we modified the original behavior such that the primal heuristic is called at most once per subproblem. The call occurs right before a branching step.

In combination with our cutpool strategies, cf. Section 8.3.1, we implemented a procedure for cleaning the pool and removing *old* constraints. Thereby, a constraint is *old* if it has not been active in the last 5 master iterations: actually, the number of necessary iterations is implemented as a parameter and we use 5 in the experiments.

`SSTP_SUB`. The core of the stochastic master problem is the twofold separation procedure: one for the separation of L-shaped and the second one for the separation of integer optimality cuts. In case of the directed formulations and the rSSTP we have an additional separation procedure for the directed cuts in the first stage. First, directed cuts are separated, afterwards L-shaped cuts (by solving the relaxed subproblems), and—if the solution is integer—integer optimality cuts (by solving subproblems to integer optimality). In the latter case, one can obviously skip scenarios with an integer solution in the previous separation of L-shaped cuts.

`STP_SUB`. This class contains the implementation of the scenarios which is similar to the directed formulation of the deterministic STP. The difference is that we have to take the current first-stage solution $\tilde{x}^0$ (or $\tilde{z}^0$) into account. For model ($\text{SSTP}_{\text{sdc1}}$) and ($\text{SSTP}_{\text{dc1}}$) the right-hand side of the directed cuts and hence, the dual objective function, is affected. Models ($\text{SSTP}_{\text{sdc2}^{(*)}}$) and ($\text{rSSTP}_{\text{dc2}^{(*)}}$) contain additional capacity constraints which depend on $\tilde{x}^0$ (or $\tilde{z}^0$). Here, the right-hand sides of these constraints, the objective function, and the

right-hand sides of the dual constraints need to be adapted. Moreover, this class contains the procedures for the strengthening of the L-shaped cuts by modifying the dual solution values.

ABA_MASTER. We implemented a `reoptimize` procedure which is similar to the original `optimize` method and which allows for the optimizations of the second stage subproblems. Here, most parts of the initialization are skipped and a new subproblem is set as root node of the b&b tree. The `optimize` method itself remains identical. It is called only once in the very first iteration and afterwards, `reoptimize` is used.

SSTP_MASTER. As already described, the SSTP master holds exactly one instance of the STP master problem. Moreover, we have exactly one instance of the maximum flow/minimum cut algorithm which needs to be initialized once and afterwards, a faster update-procedure can be used.

STP_MASTER. During the two-stage b&c algorithm we generate one instance of the `STP_MASTER` class which represents each scenario. To solve one scenario the current first-stage solution, the set of terminals, and the edge weights have to be updated and the STP master problem can be (re)optimized.

This class manages the cutpool(s) for the separated directed cuts of the scenarios, depending on the strategy, cf. Section 8.3.1. These constraint pools are all non-duplicate pools.

8.1.3 Separation of directed cuts

Let $\tilde{y}^k \in [0;1]^{|A|}$ be an optimum solution to the kth second-stage subproblem. A violated directed cut $y^k(\delta^-(S)) \geq 1, S \subseteq V_r^k : S \cap T_r^k \neq \emptyset$, can be found in polynomial time by computing a maximum flow. The source is set to the root r^k and the sink is a terminal $t \in T_r^k$. Then, an arc $(i,j) \in A$ is assigned the capacity $\tilde{y}_{ij}^k$. If the maximum flow between r^k and t is at least one there is no violated directed cut separating r^k and t. Otherwise, there exists a maximum flow with value less than one which implies a minimum cut separating r^k and t.

For the semi-directed formulation ($\text{SSTP}_{\text{sdc1}}$) and the directed formulation ($\text{rSSTP}_{\text{dc1}}$) the arc capacities also depend on the current first-stage solution. For ($\text{SSTP}_{\text{sdc1}}$) and first-stage solution $\tilde{x}^0$ an arc (i,j) gets capacity $\tilde{z}_{ij}^k + \tilde{x}_{\{i,j\}}^0$. Similarly, in formulation ($\text{rSSTP}_{\text{dc1}}$) an arc a is assigned capacity $\tilde{z}_a^k + \tilde{z}_a^0$.

For the rooted SSTP and the directed formulations the first stage contains additional directed cuts to ensure the first-stage tree: $z^0(\delta^-(S)) \geq z^0(\delta^-(v)), \forall \emptyset \neq S \subseteq V_r, \forall v \in S$. These cuts are separated analogously. The only difference is the selection of the sink and the right-hand side. Since there are no terminals all vertices $v \in V_r$ with $z^0(\delta^-(v)) > 0$ are valid sinks.

As for the deterministic STP, the separation procedure can be influenced by several modifications, as described by e.g. Koch and Martin [112]: *shuffle terminals*, *back-cuts*, *nested-cuts*, and *minimum cardinality cuts*. In our implementations we use all modifications by default.

For the direct approach we have one additional modification. Similar to the terminal shuffling we consider the scenarios in a random sequence, i.e., we *shuffle the scenarios* at the beginning of each separation procedure.

8.1.4 Additional constraints

Here we discuss additional constraints by considering valid inequalities for the deterministic STP as described by, e.g., Koch and Martin [112] and Polzin and Daneshmand [140].

Flow-balance constraints. These constraints are derived from the flow-conservation condition and relate the in- and outdegree of non-terminal vertices. Polzin and Daneshmand [140] discussed that constraints (Fb) strengthen the directed formulations (STP_{dc}) and (STP_{df}), respectively; an instance with strict inequality was given by Ljubić, Weiskircher, Pferschy, Klau, Mutzel, and Fischetti [125].

$$y^k(\delta^+(v)) \geq y^k(\delta^-(v)) \qquad \forall v \in V \backslash T \tag{Fb}$$

Unfortunately, these constraints are not valid for our stochastic models. Since first-stage solutions might contain irrelevant parts w.r.t. one particular scenario, i.e., there might be parts of the first-stage solution that can be pruned without violating the feasibility of the solution in this scenario, these constraints would enforce the selection of unnecessary arcs. Notice that this holds for the directed formulations, too (in the first and in the second stage).

Open problem 8.1. Are there flow-balance-like constraints that can be added to, and strengthen, the stochastic Steiner tree models?

Subtour elimination constraints. For the first stage it is possible to add *(generalized) subtour elimination constraints ((G)SEC)* to forbid the construction of cycles. SECs are used for many models and problems, first by Dantzig, Fulkerson, and Johnson [57]. For example, Goemans [73] used (G)SECs for the vertex-weighted STP.

The classical *subtour elimination constraints* for undirected first-stage variables x^0 look as follows:

$$x^0(E(S)) \leq |S| - 1 \qquad \forall S \subset V, |S| \geq 2 \tag{SEC}$$

The *generalized subtour elimination constraints (GSEC)* can be obtained by introducing additional binary vertex variables $\omega_v, \forall v \in V$. These variables indicate whether a vertex v is connected by first-stage edges, i.e., $\omega_v \geq x^0_e, \forall e \in \delta(v)$.

$$x^0(E(S)) \leq \omega(S \backslash \{v\}) \qquad \forall S \subset V, |S| \geq 2, \forall v \in S \tag{GSEC}$$

These constraints can be separated in $O(|V|^4)$ time by using at most $2|V| - 2$ minimum cut computations, cf. Fischetti, Hamacher, Jørnsten, and Maffioli [64]. Unfortunately, the semi-directed models cannot take advantage of the stronger GSECs directly. The additional vertex variables appear in the semi-directed models only in the GSECs and with coefficient zero in the objective function. Hence, all ω variables can be set to 1 which gives the classical subtour elimination constraint. This can be overcome by setting the objective function coefficients of the ω variables to a small ε. Although we do not support this by

an experimental study, from our experience the first stage does not contain cycles often anyway. Hence, the separation and addition of the GSECs seems to be an unnecessary overhead.

Open problem 8.2. Do subtour elimination constraints improve the performance of the two-stage b&c algorithm for the semi-directed models?

Further constraints. All following constraints are valid for any scenario $k \in \mathcal{K}$ and—although they are not binding—they can be added to the initial model to fasten the computation time:

$$y^k_{ij} + y^k_{ji} \leq 1 \qquad \forall e = \{i, j\} \in E \tag{SEC2}$$

$$y^k(\delta^-(r^k)) = 0 \tag{8.1}$$

$$y^k(\delta^+(r^k)) \geq 1 \tag{8.2}$$

$$y^k(\delta^-(v)) = 1 \qquad \forall v \in T^k_r \tag{8.3}$$

$$y^k(\delta^-(v)) \leq 1 \qquad \forall v \in V^k_r \backslash T^k_r \tag{8.4}$$

Adding the constraints influences two parts of the decomposition approaches. First, the subproblems are directly affected due to new constraints. This might improve the running time for solving the scenarios. Second, the dual of the subproblems is modified. Although the formulations do not get stronger and hence, the L-shaped cut cannot get stronger theoretically, the constraints have an influence on the generated L-shaped cuts.

8.1.5 Primal heuristic

We implemented a rudimentary primal heuristic which is called right before a branching step and it works as follows. We start by rounding the current first-stage solution $\tilde{x}^0$ to the next integer solution, i.e., 0.5 or higher becomes 1, a value < 0.5 becomes 0. Then, we compute minimum spanning trees on the induced connected components. Thereby the cost of an edge e is set to $1 - \tilde{x}^0_e$. The union of these trees is used as first-stage solution and the scenarios are solved to integer optimality.

The primal heuristic for the rSSTP works differently since we need to construct a tree in the first-stage. Here we start a breadth-first-search from the root node r using only arcs with $\tilde{z}^0_a \geq 0.5$. Thereby, arcs closing cycles are omitted. This simple procedure results in an r-rooted tree, which is used as first-stage solution, and the scenarios are again solved to optimality.

Notice that since we solve the second stage to integer optimality it is possible to add integer optimality cuts after a call to the primal heuristic. However, we do not investigate the impacts of these additional cuts in our experimental study since rudimentary experiments did not show any positive effect.

Open problem 8.3. Do integer optimality cuts generated by the primal heuristic improve the performance of the two-stage b&c algorithm?

Group		nr.	$\|V\|$ min	$\|V\|$ max	$\|E\|$ min	$\|E\|$ max	avg $\frac{\|E\|}{\|V\|}$	avg $\|T\|$	avg $\frac{\|T\|}{\|V\|}$
COPENHAGEN	IND	5	16	114	23	228	1.79	17.60	**0.43**
	RC	5	21	247	35	486	1.83	52.00	**0.45**
LIN	01–03	3	53	57	80	84	1.49	6.00	0.11
	04–06	3	157	165	266	274	1.68	9.67	0.06
	07–10	4	307	321	526	540	1.70	12.00	0.04
PCSTP	K100	11	22	45	64	191	**3.70**	9.82	**0.34**
	P100	5	66	91	163	237	**2.52**	22.40	**0.29**
PUCN		3	64	125	192	750	**4.50**	11.00	0.14
VIENNA		2	160	290	237	439	1.50	28.50	0.13
WRP3	A	4	84	138	149	257	1.83	13.00	0.11
	B	4	132	204	230	374	1.86	18.75	0.11
	C	6	233	311	431	613	1.88	20.83	0.08

Table 8.1: Summarized statistics for the considered base instances (STP and PCSTP). The last three columns are average values over all instances of one (sub)group.

8.2 Instances

We use a representative set of instances from the literature on the deterministic STP and the prize-collecting STP to generate instances for the SSTP and rSSTP. The original instances are taken from the SteinLib [113], from the instances used at the DIMACS challenge on the STP [101], and from the prize-collecting Steiner tree problem [124].

Overall we consider 55 base instances with 16–321 vertices and 23–750 edges. An overview of some instance properties is given by Table 8.1 and further details can be found in Table 8.10 in Section 8.3.7. The instances can be categorized into 6 groups with the main characteristics described as follows.

- COPENHAGEN [101]: Instances from the obstacle-avoiding rectlinear Steiner tree problem which are transformed into STP instances. These instances have many terminals; on average 44% of the vertices are terminals. There are two similar subgroups IND and RC.
- LIN [113]: Grid graphs with holes from VLSI design. These instances are sparse (density at most 1.71) and have few terminals (at most 14%). We distinguish three groups (LIN01–LIN03, LIN04–LIN06, LIN07–LIN10) based on the graph size.
- PCSTP [124]: These instances were originally generated by Johnson, Minkoff, and Phillips [100] to simulate street networks for the prize-collecting STP. They were preprocessed by Ljubić, Weiskircher, Pferschy, Klau, Mutzel, and Fischetti [125], have a high density (average 3.33), and many terminals (on average 32% of the vertices). There are two subgroups K100 and P100.
- PUCN [101, 113]: Constructed difficult instances for the STP based on hypercubes. The instances are unweighted and the graphs have the highest density (average 4.5).
- VIENNA [101, 122, 123]: These instances originate from real-world telecommunication networks in Austrian cities and are modified by an advanced preprocessing. The graphs are quite sparse (average density is 1.5). Due to the size we only use the two smallest instances I052 ($|V| = 160, |E| = 237$) and I056 ($|V| = 290, |E| = 439$).

- WRP3 [113]: Instances obtained from wire routing problems in industry. Based on the graph size we distinguish three groups A, B, and C with instances WRP3-{11,12,14,15}, WRP3-{16,17,19,23}, and WRP3-{13,20,21,22,24,25}, respectively. The original instances contain some edges with very high edge costs (100 000) compared to the other edges (avgerage edge cost 5.77); for our instances we scale each 100 000 down to 1 000 to reduce numerical problems.

We use each base instance to generate a set of instances for the stochastic Steiner tree problems as follows. Thereby, we randomly and independently generate $\bar{K} = 1\,000$ scenarios.

1. The probabilities of the scenarios are set by distributing 10 000 points (1 point corresponds to a probability of 0.01%) over all $\bar{K}$ scenarios:
   (a) We start by assigning 1 point to each scenario.
   (b) Then, we distribute the remaining $10\,000 - \bar{K}$ points by selecting one of the $\bar{K}$ scenarios uniformly at random and increasing its number of points by 1. This procedure continues until all 10 000 points are distributed.
   (c) Hence, at the end, each scenario has a probability ≥ 0.0001 and all probabilities sum up to 1 (since 10 000 points are distributed)
2. Edge costs c^0 in the first stage are set to the original edge costs from the deterministic instance.
3. Edge costs for each edge e and a scenario $k \in \{1, \ldots, \bar{K}\}$ are randomly drawn from the interval $[1.1c_e^0, 1.3c_e^0]$.
4. For each scenario $k \in \{1, \ldots, \bar{K}\}$ we construct the terminal set T^k by independently selecting each original terminal or non-terminal vertex with probability 0.3 and 0.05, respectively. Moreover, each instance is assigned a special root node which is an original terminal with maximum degree and is contained in every terminal set. Hence, each instance is a valid input for the SSTP as well for the rSSTP.

Now, for each deterministic instance and the generated $\bar{K}$ scenarios we take the first $K \in K^*$ scenarios to obtain an stochastic instance. Here, we use 14 values for K: $K^* = \{5, 10, 20, 50, 75, 100, 150, 200, 250, 300, 400, 500, 750, 1\,000\}$. Probabilities for the scenarios of the instances with $K < \bar{K}$ are scaled appropriately.

Hence, we obtain an instance set for the stochastic Steiner tree problems which consists of $55 \cdot 14 = 770$ instances. All instances can be downloaded from our SSTP homepage [178]. The stochastic instances from the sets LIN, PCSTP, and WRP3 are contained in the benchmark library of the 11th DIMACS challenge [101].

Table 8.2 summarizes some statistics of the stochastic instances; Table 8.10 in Section 8.3.7 gives more details. Overall, the average terminal density of each subgroup is between 6% and 19% with the COPENHAGEN instances having the highest average terminal density of all instances. Moreover, since the deterministic LIN instances LIN04–LIN010 have few terminals (avg. 5%) the scenarios contain on average more terminals (avg. 64% more) than the related deterministic instances. This property only holds for this subgroup of instances; for all other instances the scenarios contain fewer terminals than the original problem.

Group		avg $\|T\|$	avg $\frac{\|T\|}{\|V\|}$	avg^2 t^*	avg^2 $\frac{t^*}{\|T\|}$	avg^2 $\frac{t^*}{\|V\|}$
COPENHAGEN	IND	17.60	0.43	8.13	0.46	**0.19**
	RC	52.00	0.45	21.13	0.42	**0.19**
LIN	01–03	6.00	0.11	5.36	0.93	0.10
	04–06	9.67	0.06	11.42	**1.29**	0.07
	07–10	12.00	0.04	19.58	**1.90**	0.06
PCSTP	K100	9.82	0.34	4.99	0.51	**0.17**
	P100	22.40	0.29	10.79	0.49	**0.14**
PUCN		11.00	0.14	8.00	0.73	0.10
VIENNA		28.50	0.13	19.79	0.69	0.09
WRP3	A	13.00	0.11	10.16	0.78	0.09
	B	18.75	0.11	14.63	0.80	0.08
	C	20.83	0.08	19.45	0.98	0.08

Table 8.2: Statistics for the generated (r)SSTP instances. The second and third column is taken from Table 8.1 and gives statistics for the deterministic instances. We use $t^* := \text{avg}_{k \in \mathcal{K}} |T^k|$ as the average number of terminals. Moreover, "avg^2" means that we take the average over all stochastic instances (implied by different number of scenarios) of one base instance and then the average over all instances of one (sub)group.

BASIC instance set. We use a smaller subset of these instances for most of the experiments; we refer to this test set as BASIC. This instance set consists of instances K100, LIN01–03, and WRP3-A and the number of scenarios is in the range from 5 to 250 scenarios. Hence, the BASIC instances contain 162 instances (18 instances times 9 scenario sizes) with the number of vertices varying between 22–138 and the number of edges between 64–257. When each instance is solved in 5 independent runs this leads to 810 calls of an algorithm.

As the experiments will show the LIN01–03 instances are very easy and can be solved by all approaches, the K100 instances are more difficult such that the slower semi-directed approaches fail on some instances (reach the time limit), and the WRP3-A set contains the hardest instances such that all semi-directed models have difficulties.

8.3 Experiments

Experimental setup. All experiments are made on the following two computers: COMP1: Intel Xeon E5-2640 2.5 GHz, six cores, 64 GB RAM, Ubuntu 12.04; COMP2: AMD Opteron 2378 2.4 GHz, four cores, 16 GB RAM, Ubuntu 12.04. Most of the experiments are made on COMP1; we use COMP2 only for the comparison of the cutpool strategies. Each run is performed on a single core. Each instance is solved independently 5 times by the same method (the randomized terminal shuffling during the separation of directed cuts affects the running time). The running times are reported in seconds.

Implemented approaches and used notation. We implemented the direct approach which uses one IP and one single b&c algorithm, and the two-stage b&c decomposition approach for the following models. To shorten the notation we use the abbreviation as placeholder for all models and write the approaches in capital letters.

- Semi-directed models for the SSTP:
  - ($\text{SSTP}_{\text{sdc1}}$): direct approach DA:SDC1 and decomposition SDC1

  - ($\text{SSTP}_{\text{sdc2}}$): decomposition SDC2
  - ($\text{SSTP}_{\text{sdc2}^*}$): direct approach DA:SDC2* and decomposition SDC2*

- Directed models for the rSSTP:

  - ($\text{rSSTP}_{\text{dc1}}$): direct approach DA:DC1 and decomposition DC1
  - ($\text{rSSTP}_{\text{dc2}}$): decomposition DC2
  - ($\text{rSSTP}_{\text{dc2}^*}$): direct approach DA:DC2* and decomposition DC2*

As described in Section 7.3 the direct approach of the formulations (s)DC2 and (s)DC2* is identical but for the two-stage b&c we distinguish between the (s)dc2 and (s)DC2* formulations (SDC2 vs. SDC2* and DC2 vs. DC2*).

To indicate different strategies, we use additional prefixes. For the direct approaches we use the prefix "DA:". For the decomposition we use "NoCS:" for the approach without the strengthened L-shaped cuts and "AGL:" for aggregated L-shaped cuts. For example, we use SDC2 for the decomposition of model ($\text{SSTP}_{\text{sdc2}}$) or DC1 for the decomposition of ($\text{rSSTP}_{\text{dc1}}$) and the corresponding direct approaches are denoted by DA:SDC2* and DA:DC1.

Standard setup for approaches. If we do not use any prefix the following standard setup is used (all other introduced modifications, constraints, and cuts are not used): the decomposition is applied, L-shaped optimality cuts are disaggregated, no-good cuts (Ng) are added, and the method for strengthening the L-shaped cuts is applied (for SDC2, SDC2*, DC2, and DC2*). Moreover, the cutpool strategy with the cleaning as described in Section 8.3.1 is used. For separating the directed cuts the classical modifications as for the STP are used, cf. Section 8.1.3: terminals are shuffled, and back, nested, and minimum cardinality cuts are computed. The constraints (SEC2) and (8.1)–(8.4) are not added. Last but not least, the primal heuristic is used and called right before a branching step.

Boxplots. To present running times visually we mostly use boxplots which can be described as follows, and which contain the following information. Consider a sorted set of data points such that Q_0 is the minimum, Q_4 is the maximum, Q_2 is the median, Q_1 is the median of the interval $[Q_0, Q_2]$ and Q_3 is the median of $[Q_2, Q_4]$. Hence, the interval $[Q_1, Q_2]$ contains the second quartile with data points from 25%–50%, and the interval $[Q_2, Q_3]$ contains the third quartile with data points from 50%–75%.

The horizontal line in a boxplot gives the median value Q_2 and the box indicates the second and third quartile. The whiskers connect the boxes with the smallest and largest data point in the interval $[Q_1 - 1.5R, Q_3 + 1.5R]$ with interquartile range $R := Q_3 - Q_1$. All other points represent outlier points. All boxplots and diagrams are generated with R [68].

Speedup in running time. When comparing two approaches we use two variants of computing the speedup. The overall speedup is computed as follows. First, for each instance and each approach we take the average running time over the 5 runs. Then, we compute the speedup for each instance by using the average values. We report the median and the average speedup, the maximum speedup, and the standard deviation over all instances.

Alternatively, we consider the running time for each approach grouped by the number of scenarios. Afterwards, for each number of scenarios (i.e., 5, 10, 20, ...) we compute

the average or median running time and then, we report the average or median speedup, respectively.

Time limit and reported values. In the experiments we mostly use a time limit of 2 hours (= 7 200 sec.); for some experiments, e.g., for the direct approaches of the directed models, the time limit is 1 hour.

If an algorithm reaches the time limit this run is nevertheless taken into account when computing and reporting the overall running time for solving a set of instances or for the average or median running time. This has to be kept in mind since this method sometimes shows some strange results. For example, if a slower approach A1 does not solve all instances the number of b&b nodes may be smaller than for a faster approach A2 which solves all instances since A1 simply is not able to process as many b&b subproblems as A2. Moreover, some table entries then seem to be odd because of this behavior; in particular since we highlight important or best values of a row, column, or category with a bold font.

8.3.1 Cutpool

Throughout the implementation we try to benefit from the scenario similarities and increase their synergy. Since the graph itself is identical and terminals may coincide generated directed cuts can be useful in several scenarios. Moreover, each scenario gets solved many times with changing first-stage solutions for each L-shaped cut generation and the previously separated directed cuts can still be violated in the next SSTP master iteration. Therefore, we implemented several strategies for managing the separated directed cuts in one or several cutpools. We distinguish between three basic strategies:

- StR1: one global cutpool; all scenarios share a global pool of separated directed cuts
- StrK: K separate cutpools; each scenario maintains a pool of separated directed cuts
- StR1K: combination of StR1 and StrK; a single pool for each scenario and one global pool

However, we did not implement StR1K because of the worse performance of StR1, compare the following results.

Moreover, due to back cuts, nested cuts, and the cutpools it is possible that cuts are generated and added to the pool several times. Therefore, we implemented all cutting pools as non-duplicate pools. Each time a constraint is added it is compared to all contained constraints through assigned hashkeys; if the hashkeys are identical both cuts need to be compared through their vertex sets.

While using a pool of separated cuts the question arises when and how often the pool should be used for separation. We implemented 3 approaches:

- SepOff: no pool separation at all; hence, this strategy turns off the whole cutpool mechanism
- Sep1: separate cuts from the pool only in the first iteration, i.e., after solving the first LP of a subproblem
- SepAll: separate cuts from the pool in every iteration of each considered STP subproblem

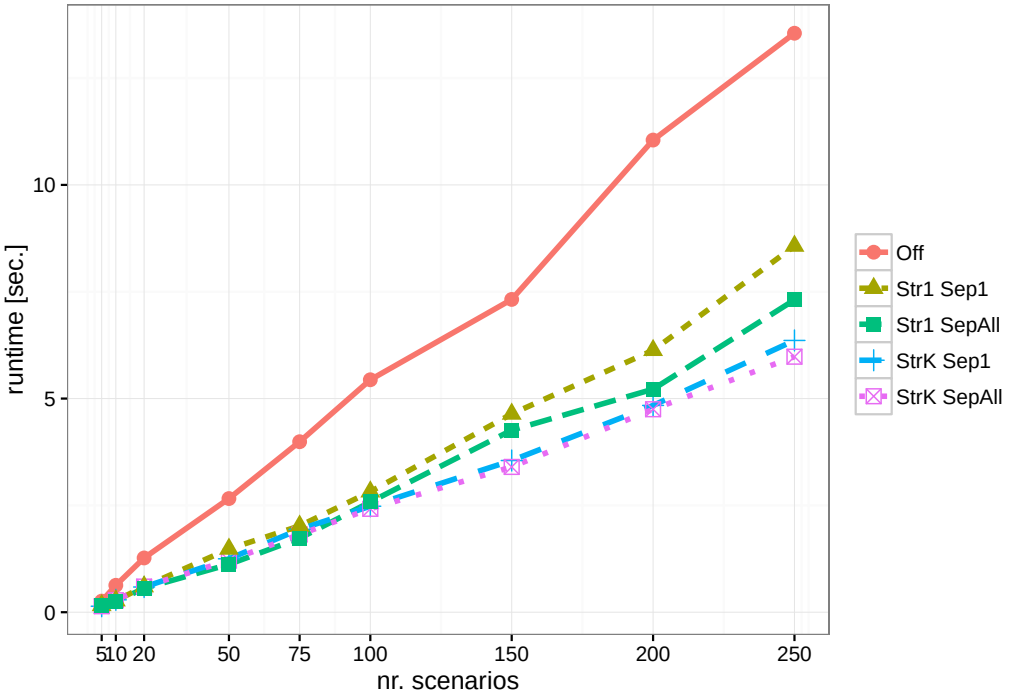


Figure 8.1: The median running time of the decomposed model DC1 on the BASIC instances grouped by the number of scenarios and with different cutpool strategies.

SEPALL and SEP1 separate cuts first from the pool and afterwards, cuts are separated via the minimum cut computation. It is possible to combine each basic pool strategy with each separation strategy leading to 5 combinations: STR1×SEPALL, STR1×SEP1, STRK×SEPALL, STRK×SEP1, and OFF.

To control the size of the cutpool(s) we implemented a pool cleaning procedure. For this purpose each directed cut is assigned an integer value which is 0 if the cut was active in the last LP of a solved subproblem, and in case the cut is inactive this value is incremented by one. Then, depending on a parameter CLEANIT, the pool gets cleaned every CLEANIT iterations by removing all constraints being inactive the last CLEANIT iterations. In our experiments, we use CLEANIT=5 but we also try CLEANIT=10 and CLEANIT=∞, i.e., no pool cleaning.

Experiments. Figure 8.1 shows the median running time (grouped by the number of scenarios) of the directed model DC1 for the different cutpool strategies on the BASIC instances. Although there are not huge differences strategy STRK (each scenario has its own cutpool, no shared cutpool) seems to perform better. We report that the results (i.e., relative performance of the strategies) for the semi-directed model SDC2 are almost identical. Hence, we decided to use STRK for the decompositions. Diagram 8.1 also indicates that STRK×SEPALL is slightly faster than STRK×SEP1. But this decision does not have a big impact; for ($\text{rSSTP}_{\text{dc1}}$) the average difference of the median running time is only 1%.

We also compare different pool cleaning iteration settings. However, for the BASIC instances, the differences are negligible. This also holds when considering all combinations

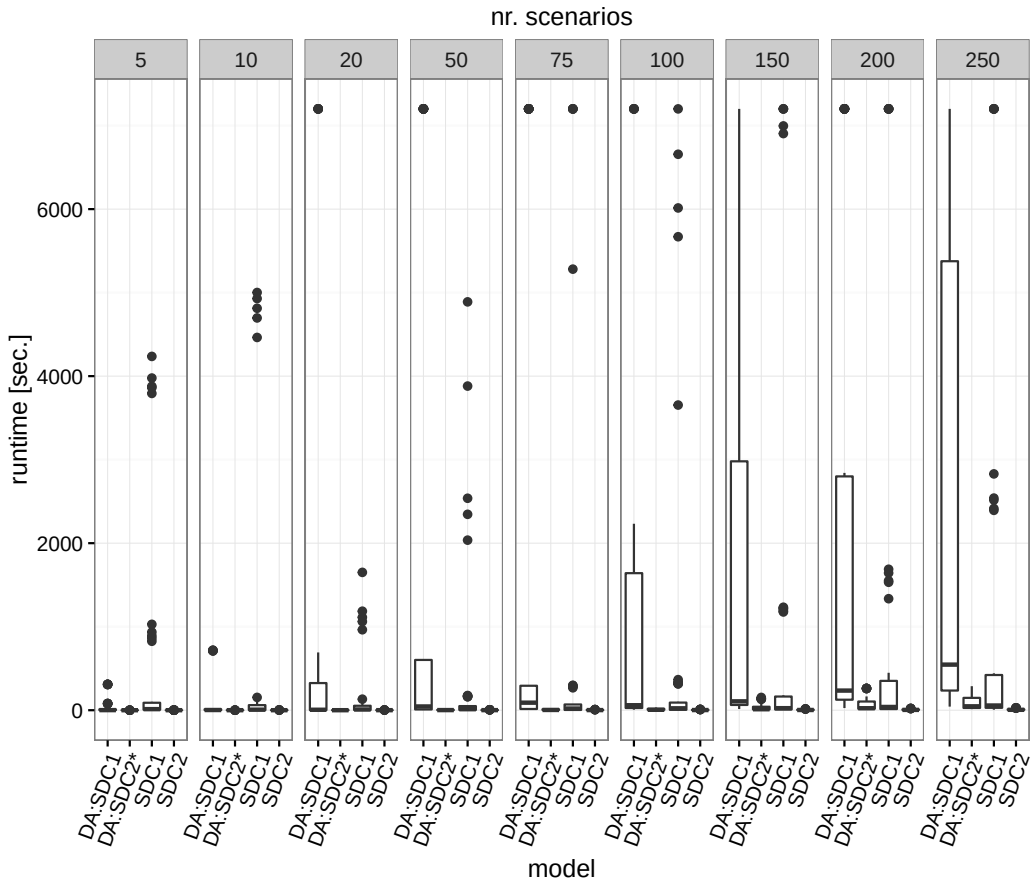


Figure 8.2: Running times of direct approaches DA:SDC1 and DA:SDC2* and decompositions SDC1 and SDC2 for K100 instances with 5–250 scenarios.

of (CLEANIT=5, CLEANIT=10, CLEANIT=∞) and (STRK×SEPALL, STRK×SEP1). In the end, we decided to use a conservative setting with CLEANIT=5. The intention is to restrict the size of the cutpools and the used internal memory.

8.3.2 Semi-directed formulations for the SSTP

Performance of SDC1. Section (6.4.2) shows that ($SSTP_{sdc1}$) is weaker than ($SSTP_{sdc2}$). In this part, we evaluate the computational performance of SDC1.

First, let us focus on the direct approaches DA:SDC1 and DA:SDC2*. Figure 8.2 contains a boxplot with the running time of the two direct approaches DA:SDC1 and DA:SDC2*, and the two decompositions of SDC1 and SDC2 for the K100 instances with 5–250 scenarios. The diagram shows that DA:SDC1 obviously does not scale well with an increasing number of scenarios. It is much slower and already for 20 scenarios it does not solve all instances. Overall, DA:SDC1 is not able to solve 12 out of 99 K100 instances within the time limit; DA:SDC2* solves all instances in at most 288 sec. Table 8.3 gives some more details which support the previous observations. In particular, the number of b&b nodes is very large for DA:SDC1 compared to DA:SDC2*. For all BASIC instances the speedup is 16.77 (median) and 301.5 (average) with a standard deviation of 1370.84.

	avg. runtime				avg. nr. b&b nodes			
K	DA:SDC1	DA:SDC2*	SDC1	SDC2	DA:SDC1	DA:SDC2*	SDC1	SDC2
5	37.25	**0.09**	459.56	0.36	1 378.64	1.73	1 284.85	**1.55**
10	70.62	**0.24**	462.51	0.57	725.36	**1.18**	616.02	**1.18**
20	751.53	**0.74**	133.21	0.77	6 195.98	**1.36**	184.96	**1.36**
50	1 388.12	2.93	313.66	**1.63**	1 630.05	**1.00**	128.13	**1.00**
75	1 388.72	7.05	666.20	**2.41**	687.80	**1.18**	133.80	**1.18**
100	1 082.43	10.50	587.44	**3.06**	227.00	**1.00**	106.42	**1.00**
150	1 829.90	33.77	800.23	**4.79**	216.75	**1.18**	110.38	**1.18**
200	1 890.40	67.10	886.67	**6.70**	90.82	**1.18**	90.93	**1.18**
250	2 411.19	86.95	1 003.20	**8.30**	59.87	**1.00**	77.25	**1.00**

Table 8.3: Average running times and average number of b&b nodes (of the master problem) for the direct approaches DA:SDC1 and DA:SDC2* and decompositions SDC1 and SDC2 on the K100 instances with 5–250 scenarios.

Now, we compare the decompositions of SDC1 and SDC2. Again, consider Figure 8.2 and Table 8.3 for the K100 instances. It can bee seen that SDC1 is slower than SDC2 (due to many more b&b nodes) but also slower than the direct approach DA:SDC2*, it produces many more outlier points, and there are always unsolved instances when the number of scenarios exceeds 75. Out of the 99 K100 instances SDC1 does not solve 5 instances and SDC2 solves all instances in at most 29 seconds. The overall speedup is 18.75 (median), 132.8 (average), with standard deviation of 361.41.

For the LIN01–03 instances with 5–250 scenarios the picture is similar although these instances are easier and can be solved by all models within the time limit. The maximum running times are 2198.48 sec. (DA:SDC1), 340.93 sec. (SDC1), 50.93 sec. (DA:SDC2*), and 7.23 sec. (SDC2). Here, the speedup from SDC1 to SDC2 is 5.34 (median), 73.1 (average), with standard deviation 184.43.

The instances from the WRP-A set are more difficult; SDC1 solves only 6 out of 36 instances. On the other hand, SDC2 is able to solve all instances—although there are three instances where not all runs are successful due to numerical issues and tailing off effects. The median speedup is 103.3, on average it is 188.1, with a standard deviation of 265.12; but the high number of unsolved instances needs to be kept in mind.

The overall running time for the BASIC instances for SDC1 is approximately 16.67 days—SDC2 only needs 32.6 hours. Moreover, SDC1 produces many more b&b nodes: for all BASIC instances it is 209 320 and 1 828 for SDC2.

Cut Strengthening. These experiments investigate the effects of the method for strengthening the L-shaped cuts as described in Section 7.2.1. We mainly compare SDC2 with and without applying the method; the latter approach is denoted by NoCS:SDC2. We remark that the results and differences for SDC2* and NoCS:SDC2* are similar.

Figure 8.3 shows the running time for the BASIC instances. Since there are many outlier points, the plot is restricted to 1 500 seconds to allow for focusing on the important parts. Clearly, the strengthening method significantly reduces the running time. Table 8.4 gives some more details for SDC2. SDC2 is significantly faster than NoCS:SDC2 and the speedup is very good. Over all instances, the speedup from NoCS:SDC2 to SDC2 is 21.31 (median) and 73.56 (average) with standard deviation 208.89.

The required running time for SDC2 for the instance set is 32.6 hours while the runs

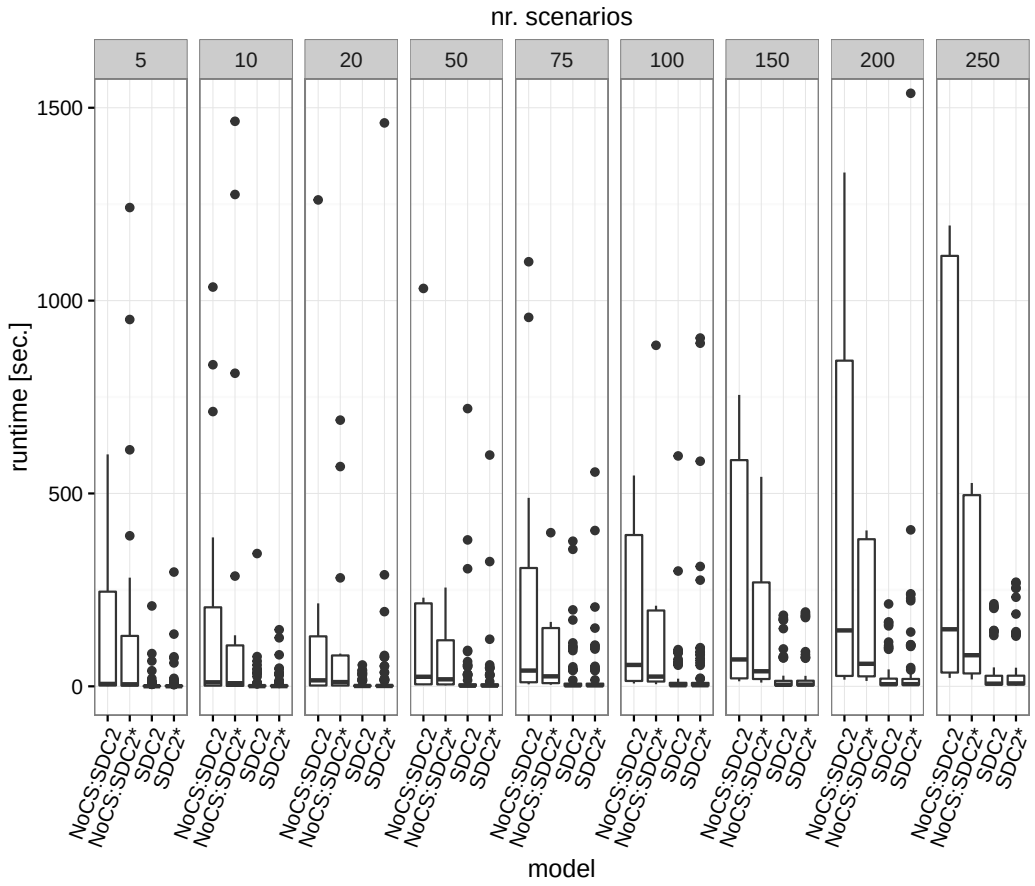


Figure 8.3: Running times of the semi-directed formulations with (SDC2, SDC2*) and without cut strengthening (NoCS:SDC2, NoCS:SDC2*) on the BASIC instances. The plot is restricted to 1 500 seconds such that many outlier points are not shown.

for NoCS:SDC2 take 330.06 hours. The overall number of b&b nodes is 1 828 (SDC2) and 13 864 (NoCS:SDC2). Over all instances, the median number of b&b nodes is 1 for both approaches, but the average increases from 2.26 to 17.12 with standard deviations 51.0 and 795.0, respectively.

SDC2 solves all instances at least once while 8 runs reach the time limit (0.9% of the runs). On the other hand, NoCS:SDC2 has 145 unfinished runs (17.9%) and 24 instances are always unsolved (at least one instance for each number of scenarios).

SDC2 vs. SDC2*. Comparing SDC2 and SDC2* does not show a clear winner and surprisingly, the ranking is different and depends on the cut strengthening method. If it is not used, Figure 8.3 gives a hint that NoCS:SDC2* is a bit faster than NoCS:SDC2. Overall, NoCS:SDC2* takes 319.8 hours (144 runs to time limit) and NoCS:SDC2 needs 330 hours (145 runs to time limit) for the BASIC instances.

But, if cut strengthening is used, SDC2 and SDC2* show the opposite behavior. The overall running time of SDC2 is 32.6 hours and for SDC2* it is 38.5 hours. The speedup is 1.04 (median) and 1.2 (average) with standard deviation 1.12. Moreover, the number of b&b nodes is 1 828 (SDC2) and 1 890 (SDC2*).

	avg. runtime		median runtime		std. dev. runtime		speedup	
K	NoCS	SDC2	NoCS	SDC2	NoCS	SDC2	avg.	median
5	1 166.70	**5.92**	6.25	**0.24**	2 544.77	**24.78**	197.08	26.04
10	1 041.37	**9.57**	10.41	**0.50**	2 441.23	**38.53**	108.82	20.82
20	1 365.14	**5.67**	15.54	**0.89**	2 722.65	**11.61**	**240.77**	17.46
50	1 520.95	**24.27**	24.77	**1.58**	2 876.95	**91.21**	62.67	15.68
75	1 411.54	**49.10**	40.78	**2.37**	2 743.05	**254.13**	28.75	17.21
100	1 630.20	**190.99**	55.45	**2.80**	2 941.22	**959.44**	8.54	19.80
150	1 637.04	**361.50**	69.64	**4.03**	2 905.88	**1498.91**	4.53	17.28
200	1 691.92	**280.93**	144.95	**5.53**	2 887.55	**1162.74**	6.02	**26.21**
250	1 737.87	**366.42**	147.89	**6.90**	2 887.54	**1460.78**	4.74	21.43

Table 8.4: Comparison of SDC2 with the approach without strengthened L-shaped cuts (NoCS) for the BASIC instances.

This behavior can be explained by considering and comparing the L-shaped optimality cuts of both formulations. In SDC2 the coefficient for an edge e is $c_e^k - \tilde{\beta}_e^k$. Without strengthened cuts $\tilde{\beta}_e$ is often 0 and in this case, the coefficient is c_e^k. For SDC2* this is different; here, the coefficient is $-\tilde{\beta}_e^k$ and hence, often 0. Although both L-shaped cuts are equally strong (cf. Lemma 7.23) the L-shaped cut of SDC2* is sparser and numerically more stable.

If the cut strengthening method is used this behavior is reversed. For SDC2 the coefficient decreases in the interval $[0, c_e^k]$ with 0 being the best possible value. And for SDC2* the interval is $[-c_e^k, 0]$ with best possible value $-c_e^k$. Hence, the cut strengthening method favors the stability and sparsity of SDC2-cuts over SDC2*-cuts.

The experiments support this explanation. For SDC2 the average number of zeros and non-zeros in the L-shaped optimality cuts is 31.84 and 100.9, respectively, and for SDC2* the average number of zeros and non-zeros is 39.56 and 93.22, respectively. Moreover, the different L-shaped cuts lead to an average number of master iterations of 24.68 for SDC2 and 25.18 for SDC2*.

Direct approach vs. decomposition. The experiments evaluating the performance of SDC1 already indicated that the decompositions outperform the direct approaches, in particular when the number of scenarios increases. Here, we take a closer look and compare SDC2 and DA:SDC2*. Figure 8.2 and Table 8.3 show running times for the K100 instances. Both models are able to solve all instances, DA:SDC2* takes at most 288 sec. and SDC2 takes at most 29 sec. The overall running time is 26.2 min. (SDC2) and 191.92 min. (DA:SDC2*); the number of b&b nodes is very similar. At around 20–50 scenarios, the decomposition becomes on average faster than the direct approach. The speedup is 2.35 (median), 3.65 (average), 23.67 (maximum), with standard deviation 4.0.

Instances from LIN01–03 are even simpler: there are no extreme outliers, SDC2 takes at most 7.23 sec. and DA:SDC2* takes at most 50.93 sec. for solving one instance. Moreover, the overall running time is 282.71 sec. (SDC2) and 1 228.92 sec. (DA:SDC2*). The speedup is 2.16 (median), 2.69 (average), 7.03 (maximum), with standard deviation 2.13. The turning point, where the decomposition gets on average faster, is again between 20 and 50 scenarios (on median 10–20 scenarios).

The WRP3-A instances are more difficult to solve, both for the direct approach and the decomposition. Figure 8.4 shows a boxplot with the running time, and Table 8.5 gives

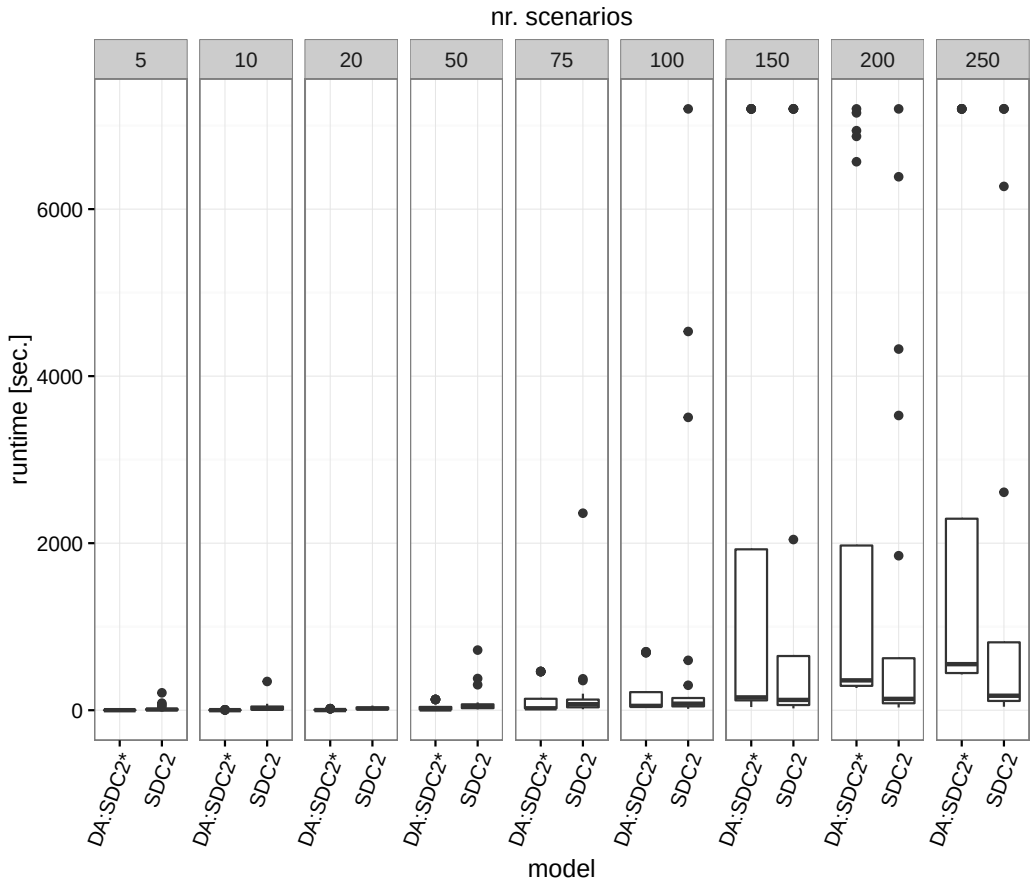


Figure 8.4: Running times of DA:SDC2* and SDC2 on WRP3-A instances with 5–250 scenarios.

details on the running time.

DA:SDC2* is not able to solve 2 instances and SDC2 solves all instances (but invokes a time limit at 8 runs). Thereby, the overall running time is 31.84 hours for SDC2 and 35.79 for DA:SDC2*. At around 100–150 scenarios the decomposition is able to outperform the direct approach. The overall speedup is 0.4 (median), 4.85 (average), 42.11 (maximum), with standard deviation 10.97; the speedup for each number of scenarios can be found in Table 8.5. The number of b&b nodes is larger than for the K100 and LIN01–03 instances. For SDC2 the median number of b&b nodes is 3.0 and 6.0 on average with standard deviation 9.13. For DA:SDC2* these values are 1.0 (median), 5.73 (average), and 11.04 (standard deviation).

The reason for the "weaker" performance of SDC2 compared to DA:SDC2* is twofold. First, it is the larger number of b&b nodes. Second, there are significantly more numerical problems leading to tailing off effects. For the unsuccessful runs, the number of tailing offs (notice, that these are tailing offs of the master problem) is ≥ 10. Moreover, there are more overall tailing offs: although the median number of tailing offs is 0 the average number is 1.9, the maximum is 25, and the standard deviation is 4.7. For the LIN01–03 instances there is no tailing off, and for the K100 instances the average is 0.002 and the maximum is 1.

However, as the number of scenarios increases the decomposition gets more competitive. When the number of scenarios increases to 300–1000 scenarios, or other instances are used,

	avg. runtime			median runtime			std. dev. runtime	
K	DA:SDC2*	SDC2	sUp	DA:SDC2*	SDC2	sUp	DA:SDC2*	SDC2
5	**0.37**	25.49	0.01	**0.28**	8.44	0.03	**0.31**	48.56
10	**1.35**	41.28	0.03	**0.52**	17.23	0.03	**1.70**	74.79
20	**5.68**	22.90	0.25	**1.67**	19.44	0.09	**7.40**	15.11
50	**38.68**	103.84	0.37	**9.63**	41.61	0.23	**53.25**	174.34
75	**133.08**	213.06	0.62	**23.09**	71.56	0.32	**195.43**	515.87
100	**206.73**	849.42	0.24	**51.95**	78.23	0.66	**288.64**	1 930.07
150	1 887.94	**1 611.05**	1.17	153.44	**123.28**	1.24	3 146.93	**2 900.17**
200	1 986.47	**1 242.68**	1.60	357.53	**135.73**	2.63	2 940.62	**2 254.10**
250	2 181.89	**1 622.18**	1.35	551.35	**173.53**	3.18	2 973.64	**2 803.77**

Table 8.5: The running time (average, median, speedup (sUp), and standard deviation) of DA:SDC2* and SDC2 on WRP3-A instances.

	avg. runtime			median runtime			std. dev. runtime	
K	DA:SDC2*	SDC2	sUp	DA:SDC2*	SDC2	sUp	DA:SDC2*	SDC2
300	683.26	**400.19**	1.71	112.25	**9.89**	11.35	1 654.93	**1 517.97**
400	942.69	**453.66**	2.08	279.65	**12.84**	21.78	1 760.53	**1 648.89**
500	1 078.22	**494.09**	2.18	349.09	**16.18**	21.58	1 698.42	**1 654.21**
750	2 113.50	**527.78**	4.00	868.26	**24.05**	36.10	2 435.89	**1 651.67**
1000	2 866.62	**614.51**	4.66	1 610.26	**35.14**	45.82	2 666.99	**1 698.28**

Table 8.6: Comparison of the running time of DA:SDC2* and SDC2 on the BASIC instances with 300–1000 scenarios.

the difference is clearer (as shown by the following experiments).

Higher number of scenarios for BASIC instances. The preceding experiments show a good performance of the decompositions SDC2 and SDC2* and—although worse—a decent performance of the direct approach DA:SDC2*. In particular, most of the instances of the BASIC set can be solved to optimality within a time limit of 2 hours. Now, we evaluate the performance when the number of scenarios increases to 300–1000.

Figure 8.5 and Table 8.6 show the results for SDC2 and DA:SDC2*; Figure 8.8 on page 149 gives a closer look at SDC2. As one can see, the decomposition clearly outperforms the direct approach and as the number of scenarios increases, the gap gets larger. Overall, SDC2 is not able to solve 4 instances and DA:SDC2* cannot solve 9 instances. The overall running time is 62.25 hours for SDC2 and 192.1 hours for DA:SDC2*. Thereby, the speedup is 20.21 (median), 22.41 (average), 61.57 (maximum), with standard deviation 14.55. The number of b&b nodes is still low: both models have as median 1 b&b node, on average 1.91 (SDC2) and 1.63 (DA:SDC2*), with at most 21 (SDC2) and 17 (DA:SDC2*) b&b nodes, and the standard deviation for both approaches is 2.7.

Again, the most difficult instances are WRP3-A instances. All unsolved runs for SDC2 as well as for DA:SDC2* occur on these instances.

Aggregated L-shaped cuts. Our standard setup of SDC2 uses disaggregated L-shaped cuts. To evaluate the difference to aggregated cuts, we also solve the BASIC instances with 5–250 scenarios with aggregated L-shaped cuts, i.e., after each iteration only one L-shaped cut is inserted instead of K disaggregated cuts. We denote this approach by AGL:SDC2. We

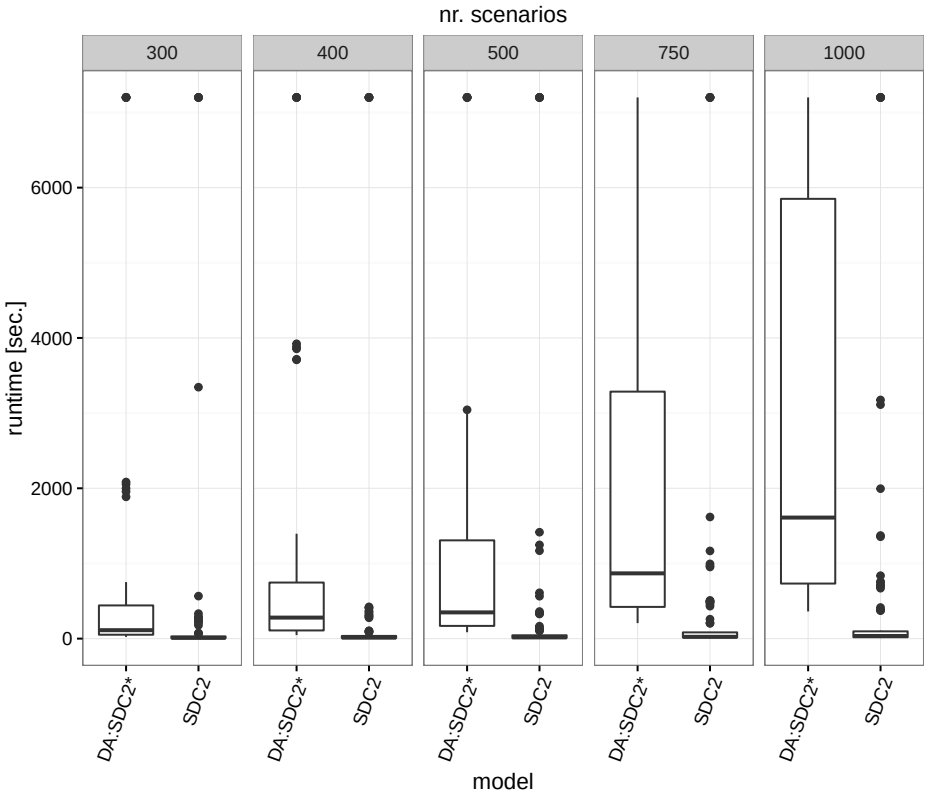


Figure 8.5: Boxplot showing the running time of DA:SDC2* and SDC2 on the BASIC instances with 300–1000 scenarios.

remark that the aggregated L-shaped cuts are also strengthened since the dual solution of each scenario is strengthened (the approach with aggregated and non-strengthened cuts is not evaluated in this thesis).

Overall, AGL:SDC2 takes 120.31 hours to solve all instances of the BASIC set; SDC2 takes 32.6 hours. Thereby, AGL:SDC2 never solves 6 instances whereas SDC2 is able to solve all instances at least once. Moreover, the number of b&b nodes and the number of tailing off effects is higher for AGL:SDC2: the number of b&b nodes increases from 344 to 2 390 and the number of tailing offs is now 6 856 instead of 1 828. The speedup is on average 5.83 with standard deviation 12.81. Surprisingly, the median speedup is 1. Hence, there is not such a big overall difference between AGL:SDC2 and SDC2. However, the version with disaggregated L-shaped cuts is more stable and in general, it is faster.

Further constraints. Section 8.1.4 introduces additional constraints for the second-stage subproblems. In general, these constraints improve the running time for solving the subproblems. On the other hand, the constraints influence the generated L-shaped cuts (and the cut strengthening procedure) due to additional variables in the dual.

In general, adding only (SEC2) (denoted by SEC), only the "degree-constraints" (8.1)–(8.4) (denoted by DEG), or all constraints (SEC+DEG) does not make a big difference. Experiments on the BASIC instances with 5–250 scenarios show that the constraints do not have a significant effect on the running time. The median running time for SDC2 is 2.54 sec.

	avg. runtime			median runtime			std. dev. runtime		
K	DC1	DC2	DC2*	DC1	DC2	DC2*	DC1	DC2	DC2*
5	**0.41**	0.53	0.72	**0.10**	0.16	0.16	**0.72**	0.97	1.44
10	**0.54**	0.78	0.84	**0.22**	0.28	0.33	**0.80**	1.23	1.25
20	**0.84**	1.21	1.44	**0.46**	0.66	0.70	**1.26**	1.57	2.16
50	**1.90**	3.66	13.19	**0.94**	1.35	1.40	**3.12**	8.93	87.49
75	10.20	**5.42**	8.40	**1.39**	2.04	2.08	55.08	**13.83**	37.68
100	8.27	**5.57**	34.25	**1.83**	2.58	2.62	34.05	**8.12**	224.37
150	13.20	**8.50**	39.68	**2.54**	3.59	3.78	51.33	**13.28**	221.29
200	**7.87**	11.26	11.92	**3.42**	4.71	5.22	**12.15**	14.65	15.08
250	22.74	**13.67**	47.56	**4.56**	6.87	7.27	128.48	**15.84**	178.16
300	**10.46**	20.31	20.49	**5.34**	7.78	8.55	**13.68**	56.14	44.99
400	**16.64**	22.96	23.31	**7.75**	12.00	11.50	35.61	33.53	**27.40**
500	**18.60**	30.19	57.49	**9.50**	14.62	14.29	**26.51**	52.64	206.43
750	**33.69**	45.53	150.61	**14.79**	21.29	23.57	**60.95**	65.62	769.76
1000	**42.37**	52.69	93.43	**20.25**	28.08	32.23	69.70	**61.46**	239.00

Table 8.7: Comparison of the running times of the rSSTP models DC1, DC2, and DC2* on the BASIC instances with 5–1000 scenarios.

and on average it is 143.8 sec. For SEC the running time is 3.56 sec. (median) and 126.4 sec. (avg.), for DEG it is 3.14 sec. and 136.1 sec., and for SECDEG it is 3.86 sec. and 124.5 sec. On the other hand, the number of master iterations increases on average from 16.27 (SDC2) to more than 24 (with additional constraints). Hence, it seems that the additional constraints decrease the running time for solving the subproblems but the L-shaped cuts are influenced negatively such that more master iterations are required.

8.3.3 Directed formulations for the rSSTP

Comparison of formulations. We start by comparing the decompositions DC1, DC2, and DC2* on the BASIC instances with 5–1000 scenarios. These approaches are very fast and require for all instances only 4.69 hours (DC1), 5.56 hours (DC2), and 12.58 hours (DC2*). Thereby, every approach is able to solve all instances in the time limit of 2 hours, the longest run for DC1 takes 1 222.41 sec., for DC2 118.27 sec., and for DC2* 2 108.53 sec.

Figure 8.6 and Table 8.7 give the running times of the three decompositions. In general, DC1 is the fastest approach followed by DC2 and the slowest model is DC2*. Since average running times are influenced stronger by outliers this can be seen best by comparing the median running time. The overall speedup from DC2 to DC1 is 1.63 (median) and 1.88 (average) with maximum 33.52 and standard deviation 2.14.

However, we report some more clear outlier points for DC1 than for DC2, in particular for 75, 100, 150, and 250 scenarios (can be seen best in Figure 8.7). This explains the larger standard deviation and the worse average running time for these scenario sizes. As for the semi-directed models the number of b&b nodes is low: the median number is 1 and the average number at most 1.27 for all models. Overall, DC1 generates 1 514 b&b nodes and DC2 1 418. Model DC1 seems to be a bit more vulnerable to numerical issues, as can be seen by the number of tailing offs, i.e., DC1 produces 67 tailing off effects and DC2 35.

The speedup from DC2 to DC2* is 1.1 (median) and 1.4 (average) with maximum 24.35 and standard deviation 2.14. DC2* has more outliers, the overall number of tailing offs is higher (95), and the overall number of b&b nodes is 1 440 which is similar to DC2 and

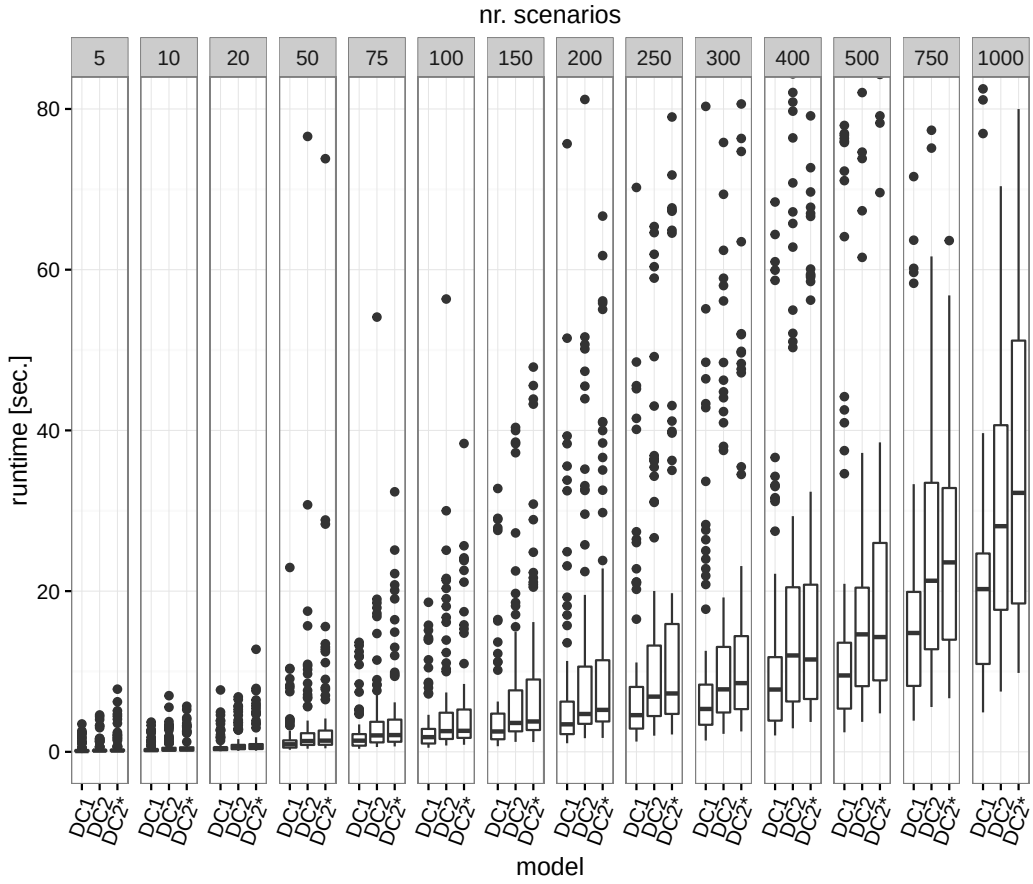


Figure 8.6: Running times of rSSTP models DC1, DC2, and DC2* on the BASIC instances with 5–1000 scenarios. The plot is restricted to 80 sec. such that some outlier points are cut off.

DC1. We observe that DC2 is in general faster than DC2*; but, when cut strengthening is not applied we report that DC2* is slightly better than DC2—similar to the semi-directed models.

Cut strengthening. To evaluate the influence of the cut strengthening method we compare DC2 and NoCS:DC2 (DC2 without cut strengthening) on the BASIC instances with 5–250 scenarios; we remark that DC2* and NoCS:DC2* show a similar behavior.

The overall running times for solving these instances are 315.89 hours (NoCS:DC2) and 1.27 hours (DC2). The average and median running times and the speedup grouped by the number of scenarios can be found in Table 8.8. Clearly, the strengthened cuts significantly reduce the required running time. Over all instances, the median speedup is 16.92 and the average is 548.10. This enhancement is even more extreme as for the semi-directed models.

The number of b&b nodes increases, too, when the strengthened cuts are not used. The median number of b&b nodes is not influenced (still 1.0), but the average number increases from 1.15 to 19.94 with standard deviation 1.14 and 66.3, respectively.

Although DC2 solves all instances within the time limit, NoCS:DC2 is not able to solve 25 instances with some additional runs reaching the time limit. Moreover, there are unsolved instances for each number of scenarios. One reason is the higher vulnerability for numerical problems: the overall number of tailing off effects increases from 35 to 7 756.

	avg. runtime		median runtime		speedup	
K	DC2	NoCS:DC2	DC2	NoCS:DC2	avg.	median
5	**0.53**	1 124.05	**0.16**	5.40	**2 120.85**	**33.75**
10	**0.78**	1 050.40	**0.28**	6.71	1 346.67	23.96
20	**1.21**	1 223.30	**0.66**	9.79	1 010.99	14.83
50	**3.66**	1 274.00	**1.35**	16.99	348.09	12.59
75	**5.42**	1 560.76	**2.04**	20.41	287.96	10.00
100	**5.57**	1 559.08	**2.58**	21.29	279.91	8.25
150	**8.50**	1 416.35	**3.59**	42.23	166.63	11.76
200	**11.26**	1 717.60	**4.71**	59.16	152.54	12.56
250	**13.67**	1 710.17	**6.87**	85.80	125.10	12.49

Table 8.8: Average and median running time of DC2 and NoCS:DC2 and the average and median speedup on the BASIC instances with 5–250 scenarios.

	avg. runtime		median runtime		sUp avgerage		sUp median	
K	DA:DC1	DA:DC2*	DA:DC1	DA:DC2*	DC1	DC2	DC1	DC2
5	**0.54**	0.83	**0.26**	0.42	1.32	1.57	2.60	2.63
10	**2.69**	2.77	**0.76**	1.36	4.98	3.55	3.45	4.86
20	**6.05**	11.23	**2.37**	4.97	7.20	9.28	5.15	7.53
50	**29.80**	74.92	**12.70**	27.77	15.68	20.47	13.51	20.57
75	**70.02**	241.13	**29.26**	77.89	6.86	44.49	21.05	38.18
100	**143.93**	491.63	**49.57**	148.31	17.40	88.26	27.09	57.48
150	**401.03**	1 176.71	**116.99**	418.35	30.38	138.44	46.06	116.53
200	**610.17**	1 627.29	**244.69**	1 152.53	**77.53**	**144.52**	71.55	244.70
250	**838.20**	1 973.90	**401.01**	2 010.17	36.86	144.40	**87.94**	**292.60**

Table 8.9: Average and median running times of DA:DC1 and DA:DC2* on the BASIC instances and the average and median speedup (sUp) for DA:DC1 compared to DC1 and DA:DC2* compared to DC2, respectively.

Direct approach vs. decomposition. Similar to the semi-directed models the decomposition outperforms the direct approach. For the directed models the difference is even more impressive. Figure 8.7 shows a boxplot with the running times of DA:DC1, DA:DC2*, and DC1 on the BASIC instances with 5–250 scenarios. Since the decompositions are very fast we use a smaller time limit of 1 hour (3 600 sec.) for the direct approaches in these experiments. Detailed running times for the direct approaches are given by Table 8.9 and for the decomposition by Table 8.7. As it turns out the decompositions already outperform the direct approaches for a very low number of scenarios. DC1 is faster (average and median) for all numbers of scenarios and DC2 and DC2* already beat the direct approaches for 5–10 scenarios.

DC1 solves all BASIC instances in 98.93 min., DC2 requires 75.91 min., DC2* runs 236.96 min. = 3.94 hours, DA:DC1 takes 52.56 hours, and DA:DC2* requires 140.01 hours. Thereby, the direct approaches cannot solve all instances: DA:DC1 fails on 7 instances and DA:DC2* on 22 instances (out of 162 instances). The median number of b&b nodes is 1 for all approaches and the average is at most 1.27.

Table 8.9 gives the speedup from the direct approaches to the decompositions (average over instances with the same number of scenarios); clearly, the speedup is huge and increases with the number of scenarios. We report that the maximum speedup for one single instance is 149.10 (DC1) and 463.70 (DC2).

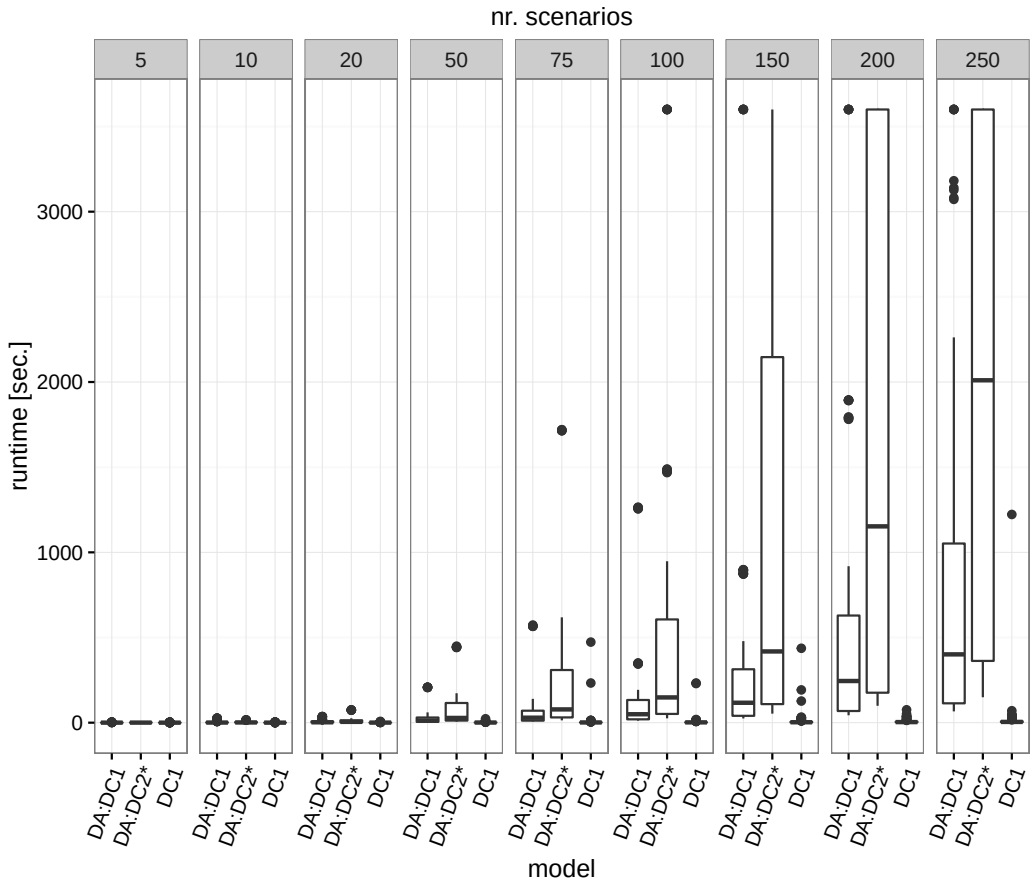


Figure 8.7: Running times of the directed models DA:DC1, DA:DC2*, and DC1 on the BASIC instances with 5–250 scenarios. The time limit is 1 hour.

We also let the direct approaches solve the instances with more than 300 scenarios. But with an increasing number of scenarios the number of unsolved instances increases drastically. For DA:DC1 the number of unsolved instances is 2, 4, 5, and 9 (each out of 18 instances) for 300, 400, 500, and 750 scenarios, respectively, and DA:DC2* cannot solve 9, 11, 13, and 16 instances.

8.3.4 Comparison of SSTP and rSSTP

Since rSSTP instances can be interpreted as SSTP instances, and all instances in our testset are rSSTP instances, we are able to compare the semi-directed and the directed models. In the following, we compare the performance and the constructed solutions—the solution values and the topologies. For these experiments, we use the models SDC2 and DC1.

Running time. Overall, the running time for the directed model DC1 for solving the BASIC instances with 5–1000 scenarios is 4.69 hours while the semi-directed model SDC2 takes 94.61 hours. DC1 solves all instances in at most 1 222.41 sec. and SDC2 does not solve 4 instances within the time limit of 2 hours and 12 additional runs reach the time limit. The overall speedup is 1.92 (median) and 5.9 (average) with maximum 177.9 and standard

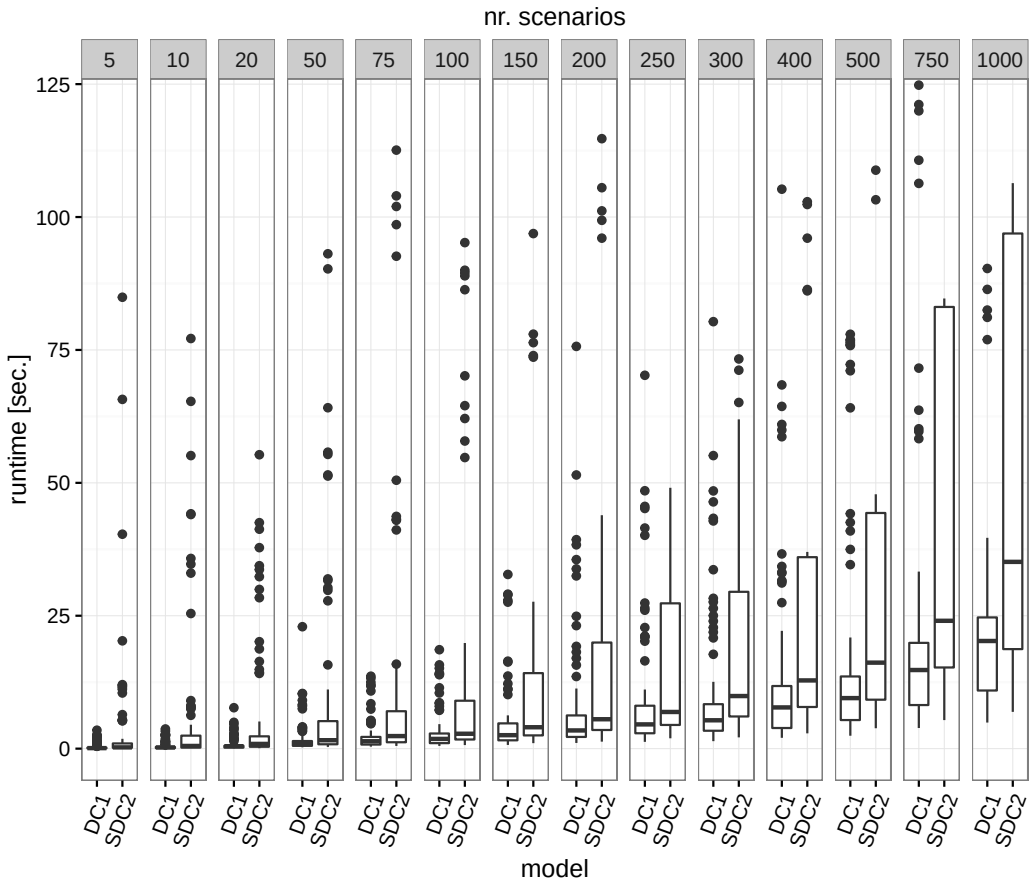


Figure 8.8: Running times of the directed model DC1 and the semi-directed model SDC2 on the BASIC instances with 5–1000 scenarios. The plot is restricted to 125 sec.

deviation 16.17. The boxplot in Figure 8.8 directly compares the running times; to show the important parts the boxplot is restricted to 125 sec.

Tree in first stage. In this experiment, we evaluate how much costlier solutions are when the first stage needs to be a tree. Figure 8.9 shows one example instance from the BASIC instances where the optimum SSTP solution is not a tree.

However, this instance is more of an exception. On the BASIC instances with 5–1000 scenarios there are almost no differences in the optimum solution value. Out of the 252 instances we evaluate 244 instances (instances where SDC2 is successful in all 5 runs). Thereof, 220 instances (90.16%) have the same solution value and 24 have not (9.84%). The instances with same solution value also have the same number of first-stage edges and the same first-stage and second-stage cost.

Out of the 24 instances with different solutions the difference is at most 0.06% (of course, the rSSTP solution value is larger). Thereof, the number of first-stage edges differs at most by 2 (on average 1.28) and the most extreme difference is for an instance with 24 edges installed for the rSSTP and 26 for the SSTP. There are only 3 instances where the SSTP first-stage solution contains more edges than the rSSTP solution. Finally, we remark that out of the 24 instances, 23 instances are WRP3-A instances and one is a K100 instance.

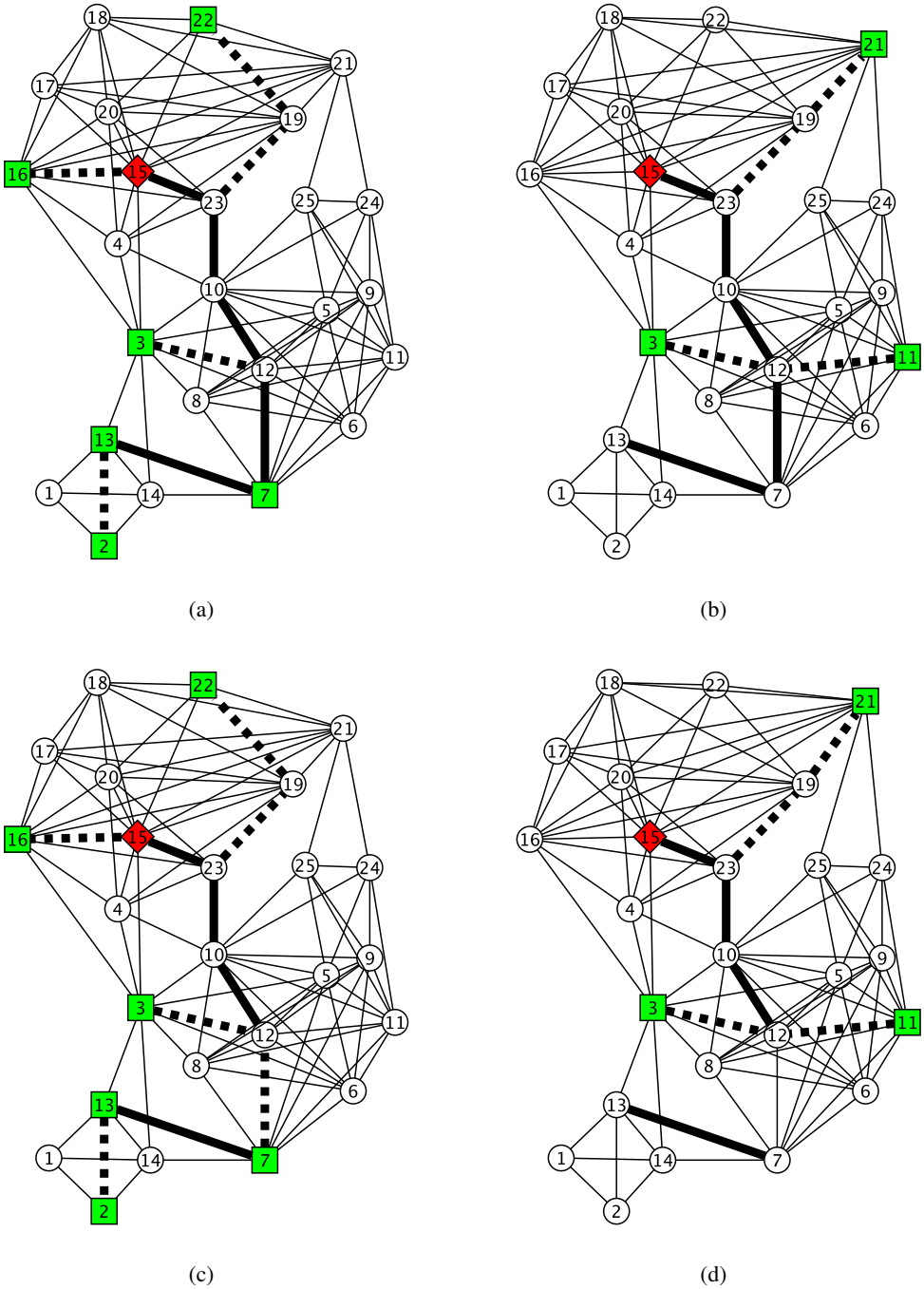


Figure 8.9: Optimum solutions to instance `K100.7.red-33-5-10-30-5s.sstp` with 5 scenarios. Thick edges are solution edges; first-stage edges are solid and second-stage edges are dotted. The red diamond is the root node, green vertices are terminals in the scenario, and circles are non-terminals. (a) and (b) show the optimum solutions to the rSSTP for scenarios 1 and 4. (c) and (d) show the optimum solutions to the SSTP for scenarios 1 and 4. The first-stage solutions differ in one edge: the rSSTP solution uses edge $\{7, 12\}$ which is not contained in the optimum SSTP first-stage solution. For both problems scenario 4 ((b) and (d)) contains irrelevant parts which are not necessary for connecting the terminals.

8.3.5 Further instances

In the preceding experiments we focus on the BASIC instance set, mostly with 5–250 scenarios. Here, we consider the other instances and solve them with the semi-directed model SDC2 and the directed model DC1. Depending on the size of the underlying graph and the complexity of the related deterministic instance we use a restricted set of scenario sizes.

More tables containing detailed results for each instance can be found in Section 8.3.7.

COPENHAGEN. In general, the instances with 5–250 scenarios are not difficult. DC1 solves each instance in at most 985 sec. and SDC2 solves almost all instances; only the largest RC instance remains unsolved for 100, 150, 200, and 250 scenarios. The median running time is 11.18 sec. (SDC2) and 1.26 sec. (DC1), and the average running time is 605.0 sec. (SDC2) and 37.72 sec. (DC1).

LIN04–06. The LIN01–03 instances are the easiest instances in our BASIC set. The LIN04–06 instances contain roughly three times as many vertices and edges but these instances are also not difficult: SDC2 and DC1 solve all instances with 5–250 scenarios within at most 12.56 min. and 1.87 min., respectively. DC1 solves each instance in the b&b root node and SDC2 generates at most 11 nodes (1.96 on average). Moreover, these instances do not lead to numerical problems, i.e., there is not a single tailing off effect.

P100. These instances behave similar to the K100 instances from the PCSTP group. For 5–250 scenarios, SDC2 and DC1 are able to solve all instances to optimality. The maximum required running time is 139.3 sec. (SDC2) and 31.71 sec. (DC1). These instances invoke only one tailing off effect and also very few b&b nodes: each b&b tree contains at most 3 nodes.

PUCN. The PUCN instances are very difficult: SDC2 is not able to solve any instance (with 5, 10, and 20 scenarios) within the time limit of 2 hours. On the other hand, DC1 solves all instances with 5–20 scenarios with a median running time of 20.74 sec. and on average in 566.9 sec. The running time increases drastically with an increasing number of scenarios: for 5 scenarios DC1 takes on average 5.19 sec. and for 20 scenarios it already requires 1 556.85 sec. on average.

VIENNA. Since these instances are relatively large we consider instances with 5–100 scenarios here. For these sizes both SDC2 and DC1 solve all instances to optimality within at most 30.53 min. (SDC2) and 3.28 min. (DC1). The number of generated b&b nodes is also moderate: SDC2 generates at most 21 nodes (3.67 on average) and DC1 generates at most 13 nodes (1.83 on average).

WRP3-B. Since the WRP3-A group contains the most difficult instances in the BASIC set we consider instances from the WRP3-B group with 5–100 scenarios. Again, SDC2 has difficulties with this type of instance and cannot solve two instances within two hours. The median running time is 6.21 min. and on average it is 27.68 min. The number of required b&b nodes is on average 11.27 which is higher than for most of the other instances. On the

other hand, DC1 solves all instances in at most 3.45 min. and generates at most 9 b&b nodes (on average 1.5).

8.3.6 Further experiments and discussion

Comparison to [27]. Here, we compare our two-stage b&c algorithm with the preliminary version presented by Bomze, Chimani, Jünger, Ljubić, Mutzel, and Zey [27]. The most influencing changes are the cut strengthening procedure, the addition of no-good cuts, and the pool separation. For these experiments we use the generated LIN instances from [27] and compare the results directly.

In [27], the two-stage b&c was not able to solve all instances to optimality. Now, SDC2 solves all instances in at most 46.15 min. and on average in 378.1 sec.; the overall average running time of [27] was 1 482.0 sec. (although on a different machine). As always, DC1 is faster than SDC2 and takes at most 189.41 sec. and on average 13.62 sec.

More detailed results can be found in Section 8.3.7, Table 8.18.

Comparison to [95]. Hokama, San Felice, Bracht, and Usberti [95] described a heuristic called BRKGA (short for biased random-key genetic algorithm) for the SSTP which is based on the minimum spanning tree heuristic and a genetic algorithm (the main ideas are described in Section 4.3). In this work, detailed results were given for the LIN instances from [27]. Here, we compare this heuristic to our exact method SDC2.

The solution value computed by BRKGA is very good. Over all 24 instances, the average gap of the heuristic solution value to the optimum solution is 0.45% (maximum 3.57%). On the other hand, the running time is very diverse and more unpredictable. For example, instance LIN01_53_80 takes 2.2 sec. for 20 scenarios but for 10 scenarios BRKGA takes 1027.4 sec.; for comparison, SDC2 takes 0.66 sec. and 0.23 sec., respectively. When comparing the running times directly it has to be kept in mind that the used computer is different ([95] use an Intel XEON CPU E3-1230 V2 with 3.30GHz and 32 GB RAM). Grouped by the number of scenarios the average running times are as follows: for $K = 5$: 80.74 sec. (SDC2) and 111.73 sec. (BRKGA), $K = 10$: 123.32 sec. and 884.38 sec., $K = 20$: 671.40 sec. and 1 050.52 sec., and $K = 50$: 636.94 sec. and 1 724.40 sec. Detailed statistics are given by Table 8.19 in Section 8.3.7.

Heuristics: BUYNONE and BUYALL. Here, we compare the two heuristics BUYNONE and BUYALL to the two-stage branch&cut approach with model SDC2.

BUYNONE fixes the first-stage variables to **0** and connects all scenario terminal sets optimally by second-stage edges. The optimum solution to BUYNONE can be obtained by solving each scenario independently as deterministic STP. We implemented this heuristic twice: for the decomposition and for the direct approach.

Contrarily, BUYALL connects each terminal set already in the first stage. Hence, this problem is a deterministic STP with the terminal set being the union of all scenario terminal sets and with edge costs from the first stage.

Of course, the running time is much less for the heuristics than for the decomposition since only one or K deterministic STPs need to be solved, respectively. Solving all instances from the BASIC set with 5–250 scenarios takes BUYALL 29.97 sec. (again 5 independent runs per instance), the decomposition implementation of BUYNONE requires 17.53 min.,

the direct approach of BuyNone takes 23.11 hours (and does not solve 2 instances within the time limit of 2 hours), and, as mentioned before, sdc2 runs in 32.6 hours.

The solution quality for BuyNone is pretty good whereas BuyAll gives much worse solutions. Overall, the percentage difference to the optimum solution for BuyNone is 1.74% (median), 1.89% (average), 7.1% (maximum), and the standard deviation is 1.4. Actually, for 15 instances (out of 162) the optimum solution is equal to the BuyNone solution. For BuyAll the median percentage difference is 99.5%, on average it is 107.9%, the maximum is 250.7%, and the standard deviation is 51.68.

Hence, BuyAll is very fast since only a single STP needs to be solved, but this approach gives a bad solution quality. BuyNone is fast and gives good solutions within 7.1% of the optimum.

Elapsed times. In this part the percentage running times of the different parts of the two-stage b&c algorithm are evaluated on the Basic instances with 5–250 scenarios. We always report the median percentage first and the average is following in round brackets.

First, we consider sdc2 with the default settings. Comparing the master and the subproblem the evaluation showed that 97.29% (92.96%) of the running time is spent in the second stage. In particular, most of this time L-shaped cuts are separated and less than 0.001% of the time is required for solving the second stage to integer optimality. Moreover, solving the master LP takes only about 2.4% (6.38%) of the time. The remaining running times for loading the instance, initialization, and the primal heuristic are marginal.

In the second stage most of the time is spent in the LP solver with 69.74% (68.89%), the separation of directed cuts with 17.54% (17.58%), and the pool separation with 5.49% (10.11%). More noticeable parts are the initialization with 1.35% (1.75%) and the method for strengthening the L-shaped cuts with 1.39% (1.53%). Of the time for separating directed cuts, most of the time goes to the max flow/min cut algorithm which takes about 80%.

Summarizing and related to the overall running time, the most time consuming parts are as follows: solving the second stage LP 63.81% (60.95%), separation of directed cuts in the second stage 15.91% (15.79%), pool separation 4.93% (8.28%), solving the master LP 2.4% (6.38%), initialization of the scenarios 1.2% (1.56%), and L-shaped cut strengthening 1.15% (1.33%). We remark, that sdc2* shows a very similar distribution.

If sdc2 is used with option Sep1 (separate cuts from the pool only in the first iteration, compare Section 8.3.1), the times spent in the pool separation is reduced to 0.91% (1.09%) while the time spent for separating directed cuts increases to 20.87% (21.05%). The remaining parts behave similar.

For the rSSTP and the directed models the running times are distributed slightly different since the master problem is expanded by directed cuts. This also affects the second stage which is in general easier to solve. The most time consuming parts for dc1 are as follows: solving LPs of the subproblems 49.19% (47.35%), separation of directed cuts in the subproblems 25.62% (25.69%), initializations of the subproblems 4.3% (4.3%), pool separation 3.04% (3.82%), strengthening of L-shaped cuts 2.9% (3.17%), solving the master LP 2.81% (5.63%), and separation of directed cuts in the master <0.01% (3.99%).

The most noticeable difference to the timing statistics of sdc2 is the ratio of the two most time consuming parts, i.e., solving the LPs in the second stage and the separation of the directed cuts in the second stage. The sum of the running times of these two parts is similar but the ratio is different.

The percentage running times of the directed models dc2 and dc2* behave similar to

SDC2. Here, for DC2, the subproblems LPs take 67.11% (65.08%) and the separation of directed cuts in the subproblems 15.72% (15.94%) of the time.

This indicates that adding the capacity constraints increases the running time for solving the second stage LPs slightly; but on the other hand, computing the max flow/min cut seems to be faster for (s)DC2.

Integer optimality cuts and number of b&b nodes. The experiments show that the number of b&b nodes of all models (except SDC1) is quite low on the generated instances. Moreover, a previously not mentioned aspect is the number of inserted integer optimality cuts in the two-stage b&c algorithm which is also very low: in all runs on the BASIC instances with 5–1000 scenarios DC1 and SDC2 each insert only a total of 55 integer optimality cuts and no-good cuts.

The reason for the low number of b&b nodes and integer optimality cuts is the strength of the directed cut formulation (STP_{dc}) for the Steiner tree problem and the used STP instances for generating the stochastic instances. Consider the integer optimality cuts: these cuts are inserted when the first-stage solution is integer, the second stage is solved to integer optimality, and the bounds on the θ variables are not correct. But if the second stage is solved as relaxed problem and the solution is integer anyway the L-shaped optimality cuts already imply the correct bound. Since the used STP instances are quite simple for (STP_{dc}), there is mostly no integrality gap and the relaxed problem often has an integer optimum solution. Moreover, (STP_{dc}) does not need many b&b nodes for solving these instances. These aspects translate directly to the stochastic approaches and all models; only model SDC1 is weaker and moreover, as discussed in the experiments on the cut strengthening method, not using this method for (s)DC2$^{(*)}$ highly increases the number of b&b nodes, too.

Discussion on further experiments. In the following we describe some experiments which are interesting for future work; some of the experiments can be found in the computational study for the stochastic survivable network design problem (SSNDP) in Section 10.2.

- Disaggregated integer optimality cuts, as described in Section 7.2.4, are not considered here. We skip the evaluation since the models do not need many integer optimality cuts on our generated instances; we refer to the SSNDP experiments.
- Pareto optimal L-shaped optimality cuts by the method of Magnanti and Wong [131], as described in Section 7.2.1, may influence the number of master iterations. We do not make experiments for the SSTP but we make experiments for the SSNDP.
- Method *Laminarize* heuristically improves the L-shaped cuts, cf. Section 7.2.1. However, preliminary experiments show no positive impact on the running time.

  Open problem 8.4. Does heuristic *Laminarize* improve the performance of the two-stage b&c algorithm?

- Section 7.2.3 introduces cut-based constraints (Cc) which are the only constraints that can be added to the first-stage master problem. We do not evaluate these constraints experimentally.

  Open problem 8.5. Do cut-based constraints improve the performance of the two-stage b&c algorithm?

- Lemma 6.3 and Theorem 6.9 state that the first-stage variables of models ($SSTP_{sdc2^*}$) and ($rSSTP_{dc2^*}$) can be relaxed without losing integrality. We do not evaluate the impact on the running time since (a) this only influences the direct approaches, and (b) the models do not require many b&b nodes on the instances anyway.
- We do not investigate the impact (benefit, bounds, running time) of the primal heuristic. The reason is again the low number of b&b nodes.
- We do not implement the improvement of the lower bound L as described in Section 7.2.5; maybe this procedure has a positive effect on the algorithm, in particular on instances causing many b&b nodes.
- We introduce new integer optimality cuts in Section 7.2.2. But since the number of introduced integer optimality cuts is so low on the generated instances we do not evaluate these cuts further. However, we make experiments for the SSNDP.
- In the default setup we always add no-good cuts. Due to the low number of b&b nodes we do not investigate the effects when these cuts are not added. Again, we make experiments for the SSNDP.
- Reducing numerical problems is an important aspect. Moreover, the number of iterations until a tailing off effect is triggered is important. Our standard setup for this threshold is 50 iterations. This value implies a tradeoff; a lower number leads to a higher number of b&b nodes and an increased bound may result in unnecessary optimizations of all scenarios and additional unnecessary L-shaped cuts.
- We do not run the algorithm on all instances from our instance set. In particular, we do not solve the large instances from the Lin07–10 and wrp3-C sets. It would be interesting to further investigate (and possibly improve) the performance on the larger instances.
- We do not implement the flow-based models since it is known from many network design problems that cut-based models in general perform better.
- It would be interesting to evaluate the performance of the SSTP models on instances with no global terminal, i.e, without a special root node.
- Chapter 10 contains the computational study on the SSNDP. Here, we evaluate further interesting aspects concerning the two-stage stochastic survivable network design problems: the *value of the stochastic solution*, *in-sample* and *out-of-sample stability*, an expension of the two-stage b&c algorithm by *scenario generation*, the performance on denser graphs, and the impact of a costlier second stage.

8.3.7 Detailed results

This section contains further detailed statistics on the generated stochastic instances and more detailed results from our computational study.

Table 8.10 gives detailed statistics of the generated (r)SSTP instances. We use $t^* := \text{avg}_{k \in \mathcal{K}} |T^k|$ and the average (avg) is taken over all stochastic instances (implied by number of scenarios) of one base instance.

class	inst.	$\|V\|$	$\|E\|$	$\|T\|$	$\frac{\|E\|}{\|V\|}$	$\frac{\|T\|}{\|V\|}$	avg t^*	avg $\frac{t^*}{\|T\|}$	avg $\frac{t^*}{\|V\|}$
COPENHAGEN	IND1	18	31	10	1.72	0.56	4.33	0.43	0.24
	IND2	31	57	10	1.84	0.32	5.02	0.50	0.16
	IND3	16	23	10	1.44	0.63	4.47	0.45	0.28
	IND4	74	146	25	1.97	0.34	11.31	0.45	0.15
	IND5	114	228	33	2.00	0.29	15.50	0.47	0.14
	RC01	21	35	10	1.67	0.48	4.70	0.47	0.22
	RC02	87	176	30	2.02	0.34	13.57	0.45	0.16
	RC03	109	202	50	1.85	0.46	20.17	0.40	0.19
	RC04	121	197	70	1.63	0.58	26.29	0.38	0.22
	RC05	247	486	100	1.97	0.40	40.92	0.41	0.17
PCSTP	K100	45	191	12	4.24	0.27	6.37	0.53	0.14
	K100.1	42	185	9	4.40	0.21	5.44	0.60	0.13
	K100.2	24	83	8	3.46	0.33	4.08	0.51	0.17
	K100.3	26	123	8	4.73	0.31	4.38	0.55	0.17
	K100.4	29	113	8	3.90	0.28	4.51	0.56	0.16
	K100.5	31	120	13	3.87	0.42	6.11	0.47	0.20
	K100.6	22	64	9	2.91	0.41	4.25	0.47	0.19
	K100.7	25	93	8	3.72	0.32	4.13	0.52	0.17
	K100.8	43	144	11	3.35	0.26	5.86	0.53	0.14
	K100.9	22	70	10	3.18	0.45	4.55	0.45	0.21
	K100.10	27	78	12	2.89	0.44	5.22	0.44	0.19
	P100	66	163	20	2.47	0.30	9.53	0.48	0.14
	P100.1	84	196	28	2.33	0.33	12.74	0.45	0.15
	P100.2	75	187	19	2.49	0.25	9.82	0.52	0.13
	P100.3	91	237	21	2.60	0.23	10.92	0.52	0.12
	P100.4	69	186	24	2.70	0.35	10.95	0.46	0.16
LIN	LIN01	53	80	4	1.51	0.08	4.49	1.12	0.08
	LIN02	55	82	6	1.49	0.11	5.38	0.90	0.10
	LIN03	57	84	8	1.47	0.14	6.21	0.78	0.11
	LIN04	157	266	6	1.69	0.04	10.13	1.69	0.06
	LIN05	160	269	9	1.68	0.06	11.34	1.26	0.07
	LIN06	165	274	14	1.66	0.08	12.80	0.91	0.08
	LIN07	307	526	6	1.71	0.02	18.12	3.02	0.06
	LIN08	311	530	10	1.70	0.03	18.49	1.85	0.06
	LIN09	313	532	12	1.70	0.04	19.57	1.63	0.06
	LIN10	321	540	20	1.68	0.06	22.13	1.11	0.07
PUCN	CC3-4N	64	288	8	4.50	0.13	6.27	0.78	0.10
	CC3-5N	125	750	13	6.00	0.10	10.32	0.79	0.08
	CC6-2N	64	192	12	3.00	0.19	7.42	0.62	0.12
VIENNA	I052A	160	237	23	1.48	0.14	15.29	0.66	0.10
	I056A	290	439	34	1.51	0.12	24.29	0.71	0.08
WRP3	WRP3-11	128	227	11	1.77	0.09	9.92	0.90	0.08
	WRP3-12	84	149	12	1.77	0.14	7.94	0.66	0.09
	WRP3-14	128	247	14	1.93	0.11	10.98	0.78	0.09
	WRP3-15	138	257	15	1.86	0.11	11.79	0.79	0.09
	WRP3-16	204	374	16	1.83	0.08	15.17	0.95	0.07
	WRP3-17	177	354	17	2.00	0.10	14.47	0.85	0.08
	WRP3-19	189	353	19	1.87	0.10	15.17	0.80	0.08
	WRP3-23	132	230	23	1.74	0.17	13.69	0.60	0.10
	WRP3-13	311	613	13	1.97	0.04	20.01	1.54	0.06
	WRP3-20	245	454	20	1.85	0.08	18.26	0.91	0.07
	WRP3-21	237	444	21	1.87	0.09	19.07	0.91	0.08
	WRP3-22	233	431	22	1.85	0.09	18.72	0.85	0.08
	WRP3-24	262	487	24	1.86	0.09	20.77	0.87	0.08
	WRP3-25	246	468	25	1.90	0.10	19.89	0.80	0.08

Table 8.10: Details on SSTP instances.

The following tables (which can be found on our homepage [178]) contain detailed results of the experiments for the semi-directed model SDC2 and the directed model DC1. For each instance (inst.) we report the optimum solution or the best found solution (opt), the average running time in seconds (t[s]), the average number of b&b nodes (b&b), and the status (st). The status is a value from the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ which gives the success rate, i.e., the percentage of successful runs on the instance (out of 5 runs); hence, 1 means that the algorithm is always successful and 0 means that the algorithm always reaches the time limit.

LIN instances with 5–250 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
LIN01	53	80	5	1	637.22	0.10	1	1	637.22	0.05	1
LIN01	53	80	10	1	628.21	0.19	1	1	628.21	0.09	1
LIN01	53	80	20	1	664.61	0.40	3	1	664.61	0.21	1
LIN01	53	80	50	1	637.66	0.82	1	1	637.66	0.60	1
LIN01	53	80	75	1	666.10	1.44	1	1	666.10	1.02	1
LIN01	53	80	100	1	683.41	1.99	1	1	683.41	1.22	1
LIN01	53	80	150	1	677.29	3.55	3	1	677.29	2.00	1
LIN01	53	80	200	1	690.01	3.62	1	1	690.01	2.29	1
LIN01	53	80	250	1	696.96	4.61	1	1	696.96	3.01	1
LIN02	55	82	5	1	988.80	0.26	1	1	988.80	0.08	1
LIN02	55	82	10	1	745.44	0.11	1	1	745.44	0.10	1
LIN02	55	82	20	1	702.52	0.25	1	1	702.52	0.20	1
LIN02	55	82	50	1	696.30	0.70	1	1	696.30	0.51	1
LIN02	55	82	75	1	679.12	1.09	1	1	679.12	0.80	1
LIN02	55	82	100	1	664.72	1.38	1	1	664.72	0.98	1
LIN02	55	82	150	1	673.40	2.30	1	1	673.40	1.52	1
LIN02	55	82	200	1	675.13	3.19	1	1	675.13	2.00	1
LIN02	55	82	250	1	678.51	4.12	1	1	678.51	2.82	1
LIN03	57	84	5	1	1095.65	0.42	1	1	1095.65	0.16	1
LIN03	57	84	10	1	1028.29	0.66	1	1	1028.29	0.25	1
LIN03	57	84	20	1	997.52	1.38	1	1	997.52	0.53	1
LIN03	57	84	50	1	927.09	1.90	1	1	927.09	1.04	1
LIN03	57	84	75	1	876.77	2.48	1	1	876.77	1.37	1
LIN03	57	84	100	1	862.47	3.06	1	1	862.47	1.96	1
LIN03	57	84	150	1	860.77	4.23	1	1	860.77	2.55	1
LIN03	57	84	200	1	842.79	5.41	1	1	842.79	3.32	1
LIN03	57	84	250	1	842.09	6.89	1	1	842.09	4.41	1
LIN04	157	266	5	1	1722.94	6.36	1	1	1722.94	1.69	1
LIN04	157	266	10	1	1891.36	25.19	1	1	1891.36	3.87	1
LIN04	157	266	20	1	1783.85	33.87	1	1	1783.85	6.10	1
LIN04	157	266	50	1	1907.47	159.08	1	1	1907.47	20.62	1
LIN04	157	266	75	1	1939.45	248.39	1	1	1939.45	30.04	1
LIN04	157	266	100	1	1899.06	254.71	3	1	1899.06	36.85	1
LIN04	157	266	150	1	1920.20	420.96	1	1	1920.20	53.54	1
LIN04	157	266	200	1	1900.51	492.54	1	1	1900.51	71.82	1
LIN04	157	266	250	1	1891.97	630.09	1	1	1891.97	85.38	1
LIN05	160	269	5	1	2197.57	72.63	3	1	2197.57	4.56	1
LIN05	160	269	10	1	2138.60	84.91	5	1	2138.75	7.10	1
LIN05	160	269	20	1	2125.00	133.77	1	1	2125.00	12.86	1
LIN05	160	269	50	1	2067.80	174.17	3	1	2067.80	23.84	3
LIN05	160	269	75	1	2071.93	217.19	1	1	2071.93	31.09	1
LIN05	160	269	100	1	2058.40	266.97	1	1	2058.40	39.74	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
LIN05	160	269	150	1	2038.04	342.38	1	1	2038.04	57.74	1
LIN05	160	269	200	1	1993.54	409.33	1	1	1993.54	75.00	1
LIN05	160	269	250	1	2016.18	540.71	1	1	2016.18	92.06	1
LIN06	165	274	5	1	1791.28	25.62	1	1	1791.28	2.01	1
LIN06	165	274	10	1	1927.75	47.84	1	1	1927.75	4.77	1
LIN06	165	274	20	1	1865.48	58.31	1	1	1865.48	8.48	1
LIN06	165	274	50	1	1971.19	183.89	11	1	1971.19	25.51	1
LIN06	165	274	75	1	1976.71	221.83	1	1	1976.71	35.54	1
LIN06	165	274	100	1	1999.09	317.40	1	1	1999.09	45.83	1
LIN06	165	274	150	1	2002.55	462.72	3	1	2002.55	66.83	1
LIN06	165	274	200	1	2004.06	562.49	3	1	2004.06	88.26	1
LIN06	165	274	250	1	2021.32	720.60	3	1	2021.32	108.12	1

Table 8.11: Details on LIN01–LIN06 instances

WRP3-A instances with 5–250 scenarios and WRP3-B instances with 5–100 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
WRP3-11	128	227	5	1	4092.08	11.42	3	1	4092.10	2.10	4.2
WRP3-11	128	227	10	1	4236.37	34.59	9	1	4236.47	2.53	1
WRP3-11	128	227	20	1	4701.09	37.14	3	1	4701.16	4.15	1
WRP3-11	128	227	50	1	4794.32	55.60	5	1	4794.41	11.58	1.4
WRP3-11	128	227	75	1	4827.22	101.95	9	1	4827.22	12.61	1
WRP3-11	128	227	100	1	4990.24	89.99	3	1	4990.29	14.60	1
WRP3-11	128	227	150	1	5091.63	170.99	7	1	5091.70	29.24	1
WRP3-11	128	227	200	1	5054.05	172.65	3	1	5054.05	35.90	1
WRP3-11	128	227	250	1	5051.40	207.10	3	1	5051.42	44.18	1
WRP3-12	84	149	5	1	4120.29	1.08	1	1	4120.29	0.13	1
WRP3-12	84	149	10	1	5097.18	5.30	1.4	1	5097.18	0.24	1
WRP3-12	84	149	20	1	5692.56	4.75	3	1	5692.56	0.45	1
WRP3-12	84	149	50	1	5381.84	11.92	3	1	5381.84	1.31	1
WRP3-12	84	149	75	1	5465.06	13.93	3	1	5465.06	1.90	1
WRP3-12	84	149	100	1	5488.14	18.28	3	1	5488.14	2.66	1
WRP3-12	84	149	150	1	5450.69	24.75	1	1	5450.69	4.70	1
WRP3-12	84	149	200	1	5643.96	36.62	3	1	5643.96	5.11	1
WRP3-12	84	149	250	1	5567.71	44.74	3	1	5567.71	7.37	1
WRP3-14	128	247	5	1	5972.91	5.54	1	1	5973.47	1.17	1
WRP3-14	128	247	10	1	5955.82	8.00	1	1	5956.12	1.28	1
WRP3-14	128	247	20	1	6057.21	15.71	1	1	6057.73	2.30	1
WRP3-14	128	247	50	1	6259.91	30.29	1	1	6260.34	3.66	1
WRP3-14	128	247	75	1	6202.35	44.29	1	1	6202.64	5.03	1
WRP3-14	128	247	100	1	6211.23	61.88	1	1	6211.63	7.92	1
WRP3-14	128	247	150	1	6191.97	79.76	1	1	6192.26	11.69	1
WRP3-14	128	247	200	1	6205.75	103.37	1	1	6206.04	16.74	1
WRP3-14	128	247	250	1	6129.14	140.54	1	1	6129.39	24.58	1
WRP3-15	138	257	5	1	7621.03	83.92	16.2	1	7621.03	2.32	1.8
WRP3-15	138	257	10	1	7386.36	117.21	15	1	7386.36	2.60	2.6
WRP3-15	138	257	20	1	6122.37	34.02	2.2	1	6122.37	3.18	1.8
WRP3-15	138	257	50	1	6467.10	317.56	7.4	1	6467.10	5.77	2.2
WRP3-15	138	257	75	1	6515.93	692.09	9.4	1	6515.93	146.54	10.2
WRP3-15	138	257	100	0.8	6546.29	3227.53	23	1	6546.29	100.65	9
WRP3-15	138	257	150	0.2	6622.71	6168.72	30.6	1	6622.71	157.81	13.8
WRP3-15	138	257	200	1	6682.51	4658.07	18.6	1	6682.51	37.30	3
WRP3-15	138	257	250	0.4	6693.66	6096.34	19.8	1	6693.66	271.17	4.6

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
WRP3-16	204	374	5	1	6407.82	679.07	2.2	1	6408.55	13.85	1.8
WRP3-16	204	374	10	1	6404.39	2993.07	13	1	6404.78	29.12	1.4
WRP3-16	204	374	20	0	6709.97	7200.00	15.4	1	6705.98	66.07	1.4
WRP3-16	204	374	50	1	6909.23	5725.49	11.4	1	6909.23	101.86	1
WRP3-16	204	374	75	0.8	7085.08	6456.27	14.6	1	7085.19	132.75	1
WRP3-16	204	374	100	0	7113.85	7200.00	11.4	1	7113.79	192.09	1
WRP3-17	177	354	5	1	7106.72	325.13	47	1	7107.03	5.28	1.4
WRP3-17	177	354	10	1	6908.86	190.16	5.8	1	6909.24	7.21	1.8
WRP3-17	177	354	20	1	6956.2	95.86	1	1	6956.35	6.81	1
WRP3-17	177	354	50	1	7166.13	322.68	2.2	1	7166.24	14.67	1.4
WRP3-17	177	354	75	1	7205.86	291.07	6.2	1	7206.50	22.63	1.4
WRP3-17	177	354	100	1	7232.26	240.91	1	1	7232.95	27.13	1
WRP3-19	189	353	5	1	6303.67	56.69	3	1	6304.59	4.01	1
WRP3-19	189	353	10	1	6995.39	137.09	3	1	6996.22	10.60	1.4
WRP3-19	189	353	20	1	7881.34	322.92	7	1	7881.85	19.11	1.4
WRP3-19	189	353	50	1	7905.21	502.56	6.6	1	7905.65	39.86	1
WRP3-19	189	353	75	1	7986.87	1392.51	19	1	7987.09	79.10	3
WRP3-19	189	353	100	1	8199.96	1059.29	3	1	8200.11	83.62	1
WRP3-23	132	230	5	1	10954.6	70.93	5	1	10954.60	2.76	3.4
WRP3-23	132	230	10	1	9465.78	100.69	3.4	1	9465.86	3.77	1
WRP3-23	132	230	20	1	9108.81	283.40	9.8	1	9109.05	9.15	3.8
WRP3-23	132	230	50	1	9245.02	399.34	13.4	1	9245.18	12.19	1.4
WRP3-23	132	230	75	1	9831.33	1863.46	33.8	1	9831.33	17.66	1
WRP3-23	132	230	100	1	9534.51	1957.49	32.2	1	9534.51	23.81	1

Table 8.12: Details on WRP3-A and WRP3-B instances

PUCN instances with 5–20 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
cc3-4n	64	288	5	0	15.2652	7200	298.6	1	15.2652	1.20	1
cc3-4n	64	288	10	0	15.6408	7200	95.8	1	15.6408	3.55	7
cc3-4n	64	288	20	0	15.6458	7200	22.6	1	15.562	453.36	39.8
cc3-5n	125	750	5	0	20.5002	7200	3.8	1	18.4092	11.15	1
cc3-5n	125	750	10	0	22.8032	7200	1	1	21.7196	392.19	12.2
cc3-5n	125	750	20	0	24.4458	7200	1	1	23.7856	4066.24	19.4
cc6-2n	64	192	5	0	20.526	7200	50.2	1	19.0874	3.22	1
cc6-2n	64	192	10	0	20.9902	7200	26.2	1	19.4244	20.12	13
cc6-2n	64	192	20	0	21.487	7200	12.6	1	19.7262	150.95	19

Table 8.13: Details on PUCN instances

COPENHAGEN instances with 5–250 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
IND1	18	31	5	1	333.257	0.01	1	1	333.257	0.01	1
IND1	18	31	10	1	321.261	0.04	1	1	321.261	0.03	1
IND1	18	31	20	1	272.636	0.05	1	1	272.636	0.03	1
IND1	18	31	50	1	340.047	0.17	1	1	340.047	0.13	1
IND1	18	31	75	1	312.575	0.22	1	1	312.575	0.19	1
IND1	18	31	100	1	309.425	0.33	1	1	309.425	0.22	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
IND1	18	31	150	1	291.673	0.52	1	1	291.673	0.34	1
IND1	18	31	200	1	289.228	0.66	1	1	289.228	0.44	1
IND1	18	31	250	1	290.904	0.81	1	1	290.904	0.60	1
IND2	31	57	5	1	5824.01	0.06	1	1	5824.01	0.02	1
IND2	31	57	10	1	6064.49	0.14	1	1	6064.49	0.05	1
IND2	31	57	20	1	5099.93	0.14	1	1	5099.93	0.08	1
IND2	31	57	50	1	6245.66	0.38	1	1	6245.66	0.23	1
IND2	31	57	75	1	5935.01	0.60	1	1	5935.01	0.34	1
IND2	31	57	100	1	5769.38	0.79	1	1	5769.38	0.39	1
IND2	31	57	150	1	5734.94	0.82	1	1	5734.94	0.57	1
IND2	31	57	200	1	5853.90	1.14	1	1	5853.90	0.69	1
IND2	31	57	250	1	5979.01	1.54	1	1	5979.01	0.99	1
IND3	16	23	5	1	461.367	0.01	1	1	461.367	0.01	1
IND3	16	23	10	1	441.80	0.04	1	1	441.80	0.01	1
IND3	16	23	20	1	371.231	0.05	1	1	371.231	0.02	1
IND3	16	23	50	1	378.486	0.12	1	1	378.486	0.06	1
IND3	16	23	75	1	380.197	0.21	1	1	380.197	0.09	1
IND3	16	23	100	1	381.665	0.25	1	1	381.665	0.12	1
IND3	16	23	150	1	375.471	0.37	1	1	375.471	0.18	1
IND3	16	23	200	1	359.931	0.48	1	1	359.931	0.23	1
IND3	16	23	250	1	363.069	0.60	1	1	363.069	0.30	1
IND4	74	146	5	1	589.509	4.25	3	1	592.74	0.58	1
IND4	74	146	10	1	671.807	5.03	1	1	671.807	0.75	1
IND4	74	146	20	1	708.439	12.81	1	1	708.439	1.82	1
IND4	74	146	50	1	677.684	16.46	1	1	677.684	3.54	1
IND4	74	146	75	1	681.868	24.69	1	1	681.868	5.23	1
IND4	74	146	100	1	667.225	40.30	4.2	1	667.225	7.12	1
IND4	74	146	150	1	673.259	64.67	5	1	673.259	11.28	1
IND4	74	146	200	1	666.797	87.56	5	1	666.797	14.43	1
IND4	74	146	250	1	663.889	103.93	5	1	663.889	18.04	1
IND5	114	228	5	1	901.039	9.33	1	1	901.134	0.97	1
IND5	114	228	10	1	836.947	9.24	1	1	836.947	1.39	1
IND5	114	228	20	1	857.567	20.99	1	1	857.567	2.89	1
IND5	114	228	50	1	876.561	41.53	1	1	876.561	7.31	1
IND5	114	228	75	1	888.897	63.26	1	1	888.897	10.46	1
IND5	114	228	100	1	902.168	80.98	1	1	902.168	14.08	1
IND5	114	228	150	1	913.327	112.13	1	1	913.327	20.2	1
IND5	114	228	200	1	908.848	147.87	1	1	908.848	28.28	1
IND5	114	228	250	1	912.536	179.88	1	1	912.536	35.89	1
RC01	21	35	5	1	19086.80	0.02	1	1	19086.80	0.01	1
RC01	21	35	10	1	19581.10	0.03	1	1	19581.10	0.02	1
RC01	21	35	20	1	18837.60	0.09	1	1	18837.60	0.03	1
RC01	21	35	50	1	19197.60	0.24	1	1	19197.60	0.09	1
RC01	21	35	75	1	18286.00	0.24	1	1	18286.00	0.12	1
RC01	21	35	100	1	18275.70	0.37	1	1	18275.70	0.17	1
RC01	21	35	150	1	17944.00	0.48	1	1	17944.00	0.25	1
RC01	21	35	200	1	17923.40	0.62	1	1	17923.40	0.30	1
RC01	21	35	250	1	17886.00	0.82	1	1	17886.00	0.37	1
RC02	87	176	5	1	30486.00	5.50	1	1	30486.00	0.78	1
RC02	87	176	10	1	26552.70	5.29	1	1	26552.70	0.77	1
RC02	87	176	20	1	30328.90	13.07	1	1	30328.90	1.96	1
RC02	87	176	50	1	31660.60	39.44	1	1	31660.60	5.51	1
RC02	87	176	75	1	31783.80	56.52	1	1	31783.80	6.99	1
RC02	87	176	100	1	31075.60	59.39	1	1	31075.60	9.45	1
RC02	87	176	150	1	31180.60	100.85	1	1	31180.60	14.61	1
RC02	87	176	200	1	30882.80	129.96	1	1	30882.80	20.19	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
RC02	87	176	250	1	30817.20	157.74	1	1	30817.20	24.65	1
RC03	109	202	5	1	42385.10	6.44	1.4	1	42385.10	1.11	1
RC03	109	202	10	1	42152.10	20.26	3.8	1	42301.60	3.3	1.4
RC03	109	202	20	1	41764.70	30.83	3	1	41764.70	3.09	1
RC03	109	202	50	1	41215.60	74.60	10.2	1	41215.60	7.25	1
RC03	109	202	75	1	41268.10	83.29	3	1	41268.10	10.60	1
RC03	109	202	100	1	40951.60	150.06	3	1	40951.60	14.08	1
RC03	109	202	150	1	41245.50	117.16	1	1	41245.50	21.01	1
RC03	109	202	200	1	41069.10	128.80	1	1	41069.10	26.01	1
RC03	109	202	250	1	41083.90	148.15	1	1	41083.90	31.61	1
RC04	121	197	5	1	45957.20	6.18	4.6	1	45957.20	0.48	1
RC04	121	197	10	1	44933.30	10.87	1	1	44934.40	0.78	1
RC04	121	197	20	1	46006.80	16.13	1	1	46006.80	1.22	1
RC04	121	197	50	1	45796.80	45.18	3	1	45796.80	2.63	1
RC04	121	197	75	1	45962.00	64.39	3	1	45962.00	3.64	1
RC04	121	197	100	1	46228.20	167.26	4.6	1	46228.20	4.22	1
RC04	121	197	150	1	46093.70	732.29	11.4	1	46093.70	6.44	1
RC04	121	197	200	1	46119.10	584.70	9	1	46119.10	9.13	1
RC04	121	197	250	1	46323.20	354.38	4.2	1	46323.20	12.68	1
RC05	247	486	5	1	57554.40	2932.21	33.4	1	57554.40	49.57	3.4
RC05	247	486	10	1	57366.30	3632.36	19.4	1	57368.00	58.80	2.2
RC05	247	486	20	1	57911.70	3317.22	7	1	57911.70	80.03	1.4
RC05	247	486	50	1	58649.90	4762.29	4.2	1	58797.10	196.45	3
RC05	247	486	75	0.8	58390.20	6662.51	5	1	58460.00	274.81	1.8
RC05	247	486	100	0	58249.20	7200	2.2	1	58335.30	469.19	2.6
RC05	247	486	150	0	58764.30	7200	1	1	58451.30	513.26	1.8
RC05	247	486	200	0	59855.10	7200	1	1	58602.50	611.06	1.4
RC05	247	486	250	0	61237.70	7200	1	1	58864.30	704.74	1

Table 8.14: Details on COPENHAGEN instances

PCSTP instances (K100 and P100) with 5–250 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
K100	45	191	5	1	177493.00	1.67	3	1	177493.00	0.45	1
K100	45	191	10	1	172609.00	2.45	1	1	172609.00	0.77	1
K100	45	191	20	1	155546.00	2.38	1	1	155546.00	0.94	1
K100	45	191	50	1	155391.00	5.21	1	1	155391.00	2.24	1
K100	45	191	75	1	152402.00	7.10	1	1	152402.00	3.23	1
K100	45	191	100	1	152059.00	9.01	1	1	152059.00	4.35	1
K100	45	191	150	1	152968.00	14.33	1	1	152968.00	5.88	1
K100	45	191	200	1	153676.00	19.97	1	1	153676.00	8.08	1
K100	45	191	250	1	156196.00	27.14	1	1	156196.00	10.51	1
K100.1	42	185	5	1	156602.00	0.18	1	1	156602.00	0.10	1
K100.1	42	185	10	1	179395.00	0.73	1	1	179395.00	0.33	1
K100.1	42	185	20	1	150544.00	0.87	1	1	150544.00	0.49	1
K100.1	42	185	50	1	144432.00	2.35	1	1	144432.00	1.41	1
K100.1	42	185	75	1	144184.00	3.93	3	1	144184.00	2.13	1
K100.1	42	185	100	1	147976.00	4.55	1	1	147976.00	2.67	1
K100.1	42	185	150	1	149961.00	7.81	3	1	149961.00	4.42	1
K100.1	42	185	200	1	150957.00	11.01	3	1	150957.00	6.10	1
K100.1	42	185	250	1	152899.00	11.07	1	1	152899.00	7.03	1
K100.2	24	83	5	1	116034.00	0.16	1	1	116034.00	0.08	1
K100.2	24	83	10	1	125139.00	0.53	3	1	125139.00	0.18	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
K100.2	24	83	20	1	120491.00	0.37	1	1	120491.00	0.21	1
K100.2	24	83	50	1	114203.00	0.81	1	1	114203.00	0.59	1
K100.2	24	83	75	1	114721.00	1.36	1	1	114721.00	0.92	1
K100.2	24	83	100	1	113183.00	1.85	1	1	113183.00	1.21	1
K100.2	24	83	150	1	114057.00	2.46	1	1	114057.00	1.79	1
K100.2	24	83	200	1	111991.00	3.38	1	1	111991.00	2.64	1
K100.2	24	83	250	1	111190.00	4.19	1	1	111190.00	5.74	1.4
K100.3	26	123	5	1	113056.00	0.22	1	1	113056.00	0.13	1
K100.3	26	123	10	1	116570.00	0.36	1	1	116570.00	0.22	1
K100.3	26	123	20	1	119037.00	1.02	3	1	119037.00	0.49	1
K100.3	26	123	50	1	114973.00	1.40	1	1	114973.00	0.89	1
K100.3	26	123	75	1	118298.00	2.25	1	1	118298.00	1.44	1
K100.3	26	123	100	1	113969.00	2.45	1	1	113969.00	1.83	1
K100.3	26	123	150	1	109087.00	3.65	1	1	109087.00	2.44	1
K100.3	26	123	200	1	109999.00	5.51	1	1	109999.00	3.53	1
K100.3	26	123	250	1	109973.00	6.93	1	1	109973.00	4.49	1
K100.4	29	113	5	1	109270.00	0.18	1	1	109270.00	0.09	1
K100.4	29	113	10	1	98672.90	0.28	1	1	98672.90	0.14	1
K100.4	29	113	20	1	95027.30	0.39	1	1	95027.30	0.23	1
K100.4	29	113	50	1	89067.30	0.87	1	1	89067.30	0.48	1
K100.4	29	113	75	1	88552.80	1.18	1	1	88552.80	0.66	1
K100.4	29	113	100	1	91394.70	1.75	1	1	91394.70	0.93	1
K100.4	29	113	150	1	90219.80	2.48	1	1	90219.80	1.42	1
K100.4	29	113	200	1	91033.30	3.82	1	1	91033.30	2.07	1
K100.4	29	113	250	1	91684.30	4.66	1	1	91684.30	2.61	1
K100.5	31	120	5	1	207291.00	0.61	1	1	207291.00	0.19	1
K100.5	31	120	10	1	178010.00	0.71	1	1	178010.00	0.27	1
K100.5	31	120	20	1	159272.00	1.43	3	1	159272.00	0.53	1
K100.5	31	120	50	1	152412.00	2.24	1	1	152412.00	1.16	1
K100.5	31	120	75	1	155357.00	3.06	1	1	155357.00	1.51	1
K100.5	31	120	100	1	154027.00	4.03	1	1	154027.00	1.95	1
K100.5	31	120	150	1	153676.00	6.33	1	1	153676.00	3.30	1
K100.5	31	120	200	1	152382.00	8.91	1	1	152382.00	3.94	1
K100.5	31	120	250	1	152998.00	11.00	1	1	152998.00	5.50	1
K100.6	22	64	5	1	163939.00	0.11	1	1	163939.00	0.06	1
K100.6	22	64	10	1	145688.00	0.15	1	1	145688.00	0.09	1
K100.6	22	64	20	1	137463.00	0.22	1	1	137463.00	0.18	1
K100.6	22	64	50	1	145298.00	0.68	1	1	145298.00	0.48	1
K100.6	22	64	75	1	140969.00	0.99	1	1	140969.00	0.72	1
K100.6	22	64	100	1	142446.00	1.33	1	1	142446.00	0.85	1
K100.6	22	64	150	1	144294.00	1.96	1	1	144294.00	1.27	1
K100.6	22	64	200	1	143855.00	2.58	1	1	143855.00	1.83	1
K100.6	22	64	250	1	141002.00	3.00	1	1	141002.00	2.04	1
K100.7	25	93	5	1	153754.00	0.23	3	1	153836.00	0.06	1
K100.7	25	93	10	1	159474.00	0.27	1	1	159474.00	0.14	1
K100.7	25	93	20	1	152902.00	0.48	1	1	152902.00	0.27	1
K100.7	25	93	50	1	138843.00	0.88	1	1	138843.00	0.53	1
K100.7	25	93	75	1	136692.00	1.09	1	1	136692.00	0.73	1
K100.7	25	93	100	1	137551.00	1.60	1	1	137551.00	1.01	1
K100.7	25	93	150	1	138191.00	2.47	1	1	138191.00	1.52	1
K100.7	25	93	200	1	137639.00	3.42	1	1	137639.00	2.27	1
K100.7	25	93	250	1	140635.00	4.56	1	1	140635.00	2.83	1
K100.8	43	144	5	1	142996.00	0.24	1	1	142996.00	0.09	1
K100.8	43	144	10	1	162983.00	0.48	1	1	162983.00	0.25	1
K100.8	43	144	20	1	173768.00	0.92	1	1	173768.00	0.48	1
K100.8	43	144	50	1	166426.00	2.35	1	1	166426.00	1.17	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
K100.8	43	144	75	1	166197.00	3.57	1	1	166197.00	1.78	1
K100.8	43	144	100	1	166375.00	4.45	1	1	166375.00	2.34	1
K100.8	43	144	150	1	170124.00	7.28	1	1	170124.00	3.57	1
K100.8	43	144	200	1	173631.00	9.73	1	1	173631.00	5.02	1
K100.8	43	144	250	1	174016.00	11.71	1	1	174016.00	6.28	1
K100.9	22	70	5	1	122937.00	0.15	3	1	122937.00	0.05	1
K100.9	22	70	10	1	119444.00	0.13	1	1	119444.00	0.07	1
K100.9	22	70	20	1	106355.00	0.09	1	1	106355.00	0.08	1
K100.9	22	70	50	1	109768.00	0.36	1	1	109768.00	0.27	1
K100.9	22	70	75	1	107906.00	0.51	1	1	107906.00	0.39	1
K100.9	22	70	100	1	107147.00	0.69	1	1	107147.00	0.52	1
K100.9	22	70	150	1	109671.00	1.06	1	1	109671.00	0.78	1
K100.9	22	70	200	1	110904.00	1.41	1	1	110904.00	1.10	1
K100.9	22	70	250	1	112679.00	1.96	1	1	112679.00	1.38	1
K100.10	27	78	5	1	165573.00	0.15	1	1	165573.00	0.06	1
K100.10	27	78	10	1	150002.00	0.13	1	1	150002.00	0.09	1
K100.10	27	78	20	1	150408.00	0.32	1	1	150408.00	0.17	1
K100.10	27	78	50	1	168981.00	0.83	1	1	168981.00	0.53	1
K100.10	27	78	75	1	181791.00	1.47	1	1	181791.00	0.80	1
K100.10	27	78	100	1	181178.00	1.98	1	1	181178.00	1.14	1
K100.10	27	78	150	1	183114.00	2.91	1	1	183114.00	1.73	1
K100.10	27	78	200	1	185982.00	4.00	1	1	185982.00	2.48	1
K100.10	27	78	250	1	185437.00	5.09	1	1	185437.00	3.31	1

Table 8.15: Details on K100 instances (PCSTP group)

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
P100	66	163	5	1	314963	1.43	1	1	314963	0.21	1
P100	66	163	10	1	349544	2.27	1	1	349544	0.36	1
P100	66	163	20	1	360151	4.20	3	1	360151	0.67	1
P100	66	163	50	1	375177	7.65	1	1	375177	1.41	1
P100	66	163	75	1	389491	9.90	1	1	389491	2.01	1
P100	66	163	100	1	386804	15.58	1	1	386804	3.09	1
P100	66	163	150	1	389343	21.56	1	1	389343	4.69	1
P100	66	163	200	1	389738	26.93	1	1	389738	7.07	1
P100	66	163	250	1	390390	34.19	1	1	390390	8.21	1
P100.1	84	196	5	1	582398	4.97	1	1	582398	0.44	1
P100.1	84	196	10	1	572280	7.94	3	1	572280	0.84	1
P100.1	84	196	20	1	557082	9.29	1	1	557082	1.47	1
P100.1	84	196	50	1	557269	14.92	1	1	557269	3.13	1
P100.1	84	196	75	1	545779	18.83	1	1	545779	4.63	1
P100.1	84	196	100	1	540791	23.12	1	1	540791	5.72	1
P100.1	84	196	150	1	533899	35.93	1	1	533899	8.82	1
P100.1	84	196	200	1	534023	44.94	1	1	534023	12.77	1
P100.1	84	196	250	1	539810	53.70	1	1	539810	15.72	1
P100.2	75	187	5	1	412061	3.72	1	1	413341	0.38	1
P100.2	75	187	10	1	419222	4.95	1	1	421373	0.65	1
P100.2	75	187	20	1	390133	6.60	1	1	390133	1.09	1
P100.2	75	187	50	1	401020	18.17	1	1	401020	3.17	1
P100.2	75	187	75	1	410826	25.83	1	1	410826	4.91	1
P100.2	75	187	100	1	416608	33.36	1	1	416608	5.95	1
P100.2	75	187	150	1	419230	57.33	1	1	419230	9.63	1
P100.2	75	187	200	1	417742	70.98	1	1	417742	12.83	1
P100.2	75	187	250	1	420325	87.25	1	1	420325	16.61	1

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
P100.3	91	237	5	1	568266	7.81	1	1	568266	0.63	1
P100.3	91	237	10	1	487316	4.38	1	1	487316	0.91	1
P100.3	91	237	20	1	532679	13.69	1	1	532679	1.94	1
P100.3	91	237	50	1	523121	26.73	1	1	523121	4.92	1
P100.3	91	237	75	1	521291	39.16	1	1	521291	7.11	1
P100.3	91	237	100	1	516784	52.84	1	1	516784	10.27	1
P100.3	91	237	150	1	526432	74.32	1	1	526432	17.26	1
P100.3	91	237	200	1	521034	103.64	3	1	521034	24.05	3
P100.3	91	237	250	1	518514	131.30	3	1	518514	27.92	1
P100.4	69	186	5	1	515191	0.44	1	1	515191	0.09	1
P100.4	69	186	10	1	516098	0.79	1	1	516098	0.24	1
P100.4	69	186	20	1	542476	1.44	1	1	542476	0.83	1.4
P100.4	69	186	50	1	523388	3.53	1	1	523388	1.35	1
P100.4	69	186	75	1	522994	5.15	1	1	522994	1.84	1
P100.4	69	186	100	1	516570	6.90	1	1	516570	2.48	1
P100.4	69	186	150	1	496636	9.16	1	1	496636	3.92	1
P100.4	69	186	200	1	496437	12.52	1	1	496437	5.02	1
P100.4	69	186	250	1	488401	15.62	1	1	488401	6.49	1

Table 8.16: Details on P100 instances (PCSTP group)

VIENNA instances with 5–100 scenarios

				SDC2				DC1			
inst.	$\|V\|$	$\|E\|$	K	st	opt	t[s]	b&b	st	opt	t[s]	b&b
I052A	160	237	5	1	114146	9.53	3	1	114146	0.77	1.8
I052A	160	237	10	1	126952	27.32	3.4	1	126952	1.86	1
I052A	160	237	20	1	123713	31.75	3.4	1	123713	2.89	3
I052A	160	237	50	1	124967	72.17	3	1	124967	6.47	1
I052A	160	237	75	1	124812	82.84	1	1	124812	9.96	1
I052A	160	237	100	1	125279	182.77	3	1	125279	13.45	1
I056A	290	439	5	1	358435	210.00	1.8	1	358435	14.32	1.8
I056A	290	439	10	1	376070	414.86	9.8	1	376070	33.72	4.6
I056A	290	439	20	1	366687	440.32	3	1	366687	40.85	2.2
I056A	290	439	50	1	359336	695.30	3.8	1	359336	80.22	1.8
I056A	290	439	75	1	365620	1288.95	3.8	1	365620	126.78	1.4
I056A	290	439	100	1	366434	1451.17	1.4	1	366434	163.32	1.4

Table 8.17: Details on VIENNA instances

LIN instances from [27]

Here, we compare our two-stage b&c algorithm with the preliminary version by Bomze, Chimani, Jünger, Ljubić, Mutzel, and Zey [27]. For this purpose we use the same instances from [27]. Notice that these instances are based on the same deterministic STP instances (LIN01–06 group) but the stochastic instances are different since the set of scenarios was generated differently. Contrarily to our instance set the scenarios of the stochastic instances from [27] were generated independently. This means that two stochastic instances, which are based on the same deterministic instance, have independently generated scenario sets and, e.g., the scenario set of a 5-scenario instance is not a subset of the related 10-scenario instance (as in our instance set).

The following table shows the main results when comparing SDC2, DC1, and SDC2

from [27]. For each instance (inst.) we report the optimum solution (opt), the average running time in seconds (t[s]), and the average number of b&b nodes (b&b). Notice that for SDC2 from [27] the values were not averages since [27] did perform only one run per instance. Moreover, we do not report the optimum solution to DC1 since it is equal to SDC2 and we further do not report the status for SDC2 and DC1 since all runs are successful. On the other hand, SDC2 from [27] was not able to solve all instances: LIN05_160_269 with 20 scenarios was unsolved (the gap reported by [27] was 4.7% and the actual gap is 4.6%). Moreover, our new experiments reveal that the solution to instance LIN06_165_274 with 50 scenarios was not correct due to the missing no-good cuts. The correct value (also confirmed by solving this instance with the direct approach) of the optimum solution is 2 187.55 instead of 2 186.8. All other solution values are identical.

There are some further important remarks. First, the reported numbers by [27] were given by one digit after the radix point; since we have the original data we report two digits here. Second, we use the notation of [27] and encode the number of vertices and edges of an instance into the instance name (separated by underscores). Third, the runs of [27] were performed on a weaker computer (Core-i7 2.67GHz, 12 GB RAM), with an older version of CPLEX (10.1), and the reported running times are the ones from [27].

		SDC2			DC1		SDC2 [27]		
instance	K	opt	t[s]	b&b	t[s]	b&b	opt	t[s]	b&b
LIN01_53_80	5	797.03	0.28	1	0.06	1	797.03	2.20	1
LIN01_53_80	10	633.17	0.23	3	0.10	1	633.17	2.50	3
LIN01_53_80	20	753.93	0.66	3	0.23	1	753.93	6.90	3
LIN01_53_80	50	768.91	1.61	3	0.47	1	768.91	10.40	3
LIN02_55_82	5	476.20	0.08	1	0.03	1	476.20	1.10	1
LIN02_55_82	10	739.07	0.31	1	0.12	1	739.07	3.00	1
LIN02_55_82	20	752.18	0.32	1	0.18	1	752.18	4.30	1
LIN02_55_82	50	732.56	1.02	1	0.53	1	732.56	10.70	1
LIN03_57_84	5	652.99	0.22	1	0.06	1	652.99	1.90	1
LIN03_57_84	10	834.73	1.63	7	0.21	1	834.73	8.70	7
LIN03_57_84	20	854.86	0.95	1	0.31	1	854.86	7.30	1
LIN03_57_84	50	895.69	4.18	3	0.73	1	895.69	21.30	3
LIN04_157_266	5	1922.10	141.92	3	3.86	1	1922.10	959.20	47
LIN04_157_266	10	1959.10	125.63	6.6	4.54	1	1959.10	989.20	7
LIN04_157_266	20	1954.90	410.28	11	11.07	1	1954.90	3016.70	13
LIN04_157_266	50	2097.71	1494.99	16.6	31.78	1	2097.71	5330.20	11
LIN05_160_269	5	2215.52	300.64	7	4.11	1	2215.52	2681.20	35
LIN05_160_269	10	2210.19	535.70	43	5.65	1	2210.19	4096.00	35
LIN05_160_269	20	2305.38	2791.05	25.8	21.01	1	*2412.23*	*7200.00*	17
LIN05_160_269	50	2297.04	989.62	1	30.27	1	2297.04	3627.40	1
LIN06_165_274	5	1975.76	41.28	27	1.87	1	1975.76	760.90	19
LIN06_165_274	10	1918.65	76.42	3	4.78	1	1918.65	808.40	3
LIN06_165_274	20	2457.56	825.13	11	15.41	1	2457.56	3222.90	11
LIN06_165_274	50	2187.55	1330.24	28.6	189.41	21.4	*2186.80*	2795.50	11

Table 8.18: Details on LIN instances from [27] and comparison to [27]

Comparison to BRGKA heuristic by [95]

Finally, we compare SDC2 to the biased random-key genetic algorithm (BRGKA) by Hokama, San Felice, Bracht, and Usberti [95] on the LIN instances from [27].

We report the solution value computed by the heuristic BRKGA (sol), the running time

in seconds (t[s]), the gap (in %) to the optimum solution, the quotient of the BRKGA running time to the SDC2 running time (quot), and the inverse quotient (1/quot). For the latter two values only the entry ≥ 1 is given and the entry in the other column is empty. Notice that BRKGA was performed on a different computer (Intel XEON CPU E3-1230 V2 with 3.30GHz and 32 GB RAM).

		SDC2		BRKGA [95]		comparison		
instance	K	opt	t[s]	sol	t[s]	gap	quot	1/quot
LIN01_53_80	5	797.03	0.28	797.00	193.70	0.00	691.79	
LIN01_53_80	10	633.17	0.23	636.00	1027.40	0.45	4466.96	
LIN01_53_80	20	753.93	0.66	754.40	2.20	0.06	3.33	
LIN01_53_80	50	768.91	1.61	769.00	1456.30	0.01	904.53	
LIN02_55_82	5	476.20	0.08	476.20	0.30	0.00	3.75	
LIN02_55_82	10	739.07	0.31	739.10	2529.00	0.00	8158.06	
LIN02_55_82	20	752.18	0.32	752.20	4.60	0.00	14.38	
LIN02_55_82	50	732.56	1.02	732.80	803.00	0.03	787.25	
LIN03_57_84	5	652.99	0.22	655.20	0.30	0.34	1.36	
LIN03_57_84	10	834.73	1.63	840.60	38.40	0.70	23.56	
LIN03_57_84	20	854.86	0.95	855.50	583.40	0.07	614.11	
LIN03_57_84	50	895.69	4.18	896.00	502.00	0.03	120.10	
LIN04_157_266	5	1922.10	141.92	1923.00	44.00	0.05		3.23
LIN04_157_266	10	1959.10	125.63	1972.40	1161.60	0.68	9.25	
LIN04_157_266	20	1954.90	410.28	1981.20	3125.30	1.35	7.62	
LIN04_157_266	50	2097.71	1494.99	2172.50	2816.80	3.57	1.88	
LIN05_160_269	5	2215.52	300.64	2224.90	20.20	0.42		14.88
LIN05_160_269	10	2210.19	535.70	2224.30	160.50	0.64		3.34
LIN05_160_269	20	2305.38	2791.05	2307.00	1884.70	0.07		1.48
LIN05_160_269	50	2297.04	989.62	2301.60	1645.40	0.20	1.66	
LIN06_165_274	5	1975.76	41.28	1986.90	411.90	0.56	9.98	
LIN06_165_274	10	1918.65	76.42	1935.10	389.40	0.86	5.10	
LIN06_165_274	20	2457.56	825.13	2463.90	702.90	0.26		1.17
LIN06_165_274	50	2187.55	1330.24	2196.00	3122.90	0.39	2.35	

Table 8.19: Details on comparison to BRKGA heuristic

Part III

The two-stage stochastic survivable network design problem

Chapter 9

Introduction, IP formulations, and decomposition

Section 9.1 gives the definition of the *two-stage stochastic survivable network design problem* and an overview of the related work. Afterwards, in Section 9.2, we recall the undirected cut-based model and discuss some polyhedral properties for this model. Moreover, we use orientation properties from the deterministic problem, introduce stronger semi-directed cut-based formulations, and compare the strength of the models. In Section 9.3 we describe the decomposition, the generated L-shaped optimality cuts as well as the method for strengthening the optimality cuts, and new and stronger integer optimality cuts.

9.1 Introduction

Problem definition. The *stochastic survivable network design problem (SSNDP)* is a straight-forward extension of the deterministic SNDP (cf. Section 3.1) to a two-stage stochastic optimization problem and, on the other hand, an expansion of the stochastic Steiner tree problem to higher connectivity requirements. As for the SSTP, we assume that the actual connectivity requirements as well as the future edge costs are subject to uncertainty and only known in the second stage. These values together form a random vector for which we assume that it has finite support. It can therefore be modeled by using a finite set of scenarios $\mathcal{K} = \{1, \dots, K\}$, $K \geq 1$. Similar to the SNDP we assume that the connectivity requirement matrix ρ^k of each scenario $k \in \mathcal{K}$ is unitary.

Problem 9.1 (Stochastic survivable network design problem (SSNDP)):

Given: *undirected graph $G = (V, E)$, first-stage edge cost $c_e^0 \in \mathbb{R}^{\geq 0}, \forall e \in E$, and a set of $K \geq 1$ scenarios. Each scenario $k \in \mathcal{K}$ is defined by its probability $p^k \in (0; 1]$, second-stage edge cost $c_e^k \in \mathbb{R}^{\geq 0}, \forall e \in E$, and a symmetric and unitary connectivity requirement matrix $\rho^k \in \mathbb{N}^{|V|\times|V|}$. Moreover, it holds $\sum_{k\in\mathcal{K}} p^k = 1$.*

Solution: *$K+1$ edge sets $E^0, \dots, E^K \subseteq E$ such that $G[E^0 \cup E^k]$ contains ρ_{uv}^k edge-disjoint paths between vertices $u, v \in V, u \neq v, \forall k \in \mathcal{K}$*

Objective: *minimize the expected cost $\sum_{e\in E^0} c_e^0 + \sum_{k\in\mathcal{K}} p^k \sum_{e\in E^k} c_e^k$*

Since the SSTP is a special case of the SSNDP, it can be observed that the optimal first-stage solution E^0 of the SSNDP is not necessarily connected.

Complexity and related work. There exists a large body of work on different variants of the deterministic survivable network design problem. We refer to Kandyba-Chimani [105] and Kerivin and Mahjoub [108] for a comprehensive literature overview on the SNDP. Many polyhedral studies were done in the 90s, see, e.g., Grötschel, Monma, and Stoer [80] and [105, 108]. A decade later the question of deriving stronger mixed integer programming formulations by orienting the k-connected subgraphs has been considered by e.g. Balakrishnan, Magnanti, and Mirchandani [9], Magnanti and Raghavan [130], and Raghavan [146]. Among the approximation algorithms for the SNDP, we point to the work of Jain [99] whose approximation factor of two remains the best one up to date.

Regarding the stochastic variants there are significantly less results published so far and to the best of our knowledge all publications consider special cases of the SSNDP. One of the investigated cases is the stochastic Steiner tree problem (connectivity requirement 0 or 1); for the related work on the SSTP see Section 4.3. For the SSNDP involving connectivity requirements ≥ 2, up to our knowledge, there only exists an $O(1)$ approximation algorithm by Gupta, Krishnaswamy, and Ravi [88] for the following special case of the $\{0, \kappa\}$-SSNDP: For each pair of distinct nodes i and j, a single scenario, whose probability is p_{ij}, is given in which nodes i and j need to be κ-edge-connected. But in general, however, it follows by Ravi and Sinha [150] that the SSNDP is as hard to approximate as label cover—which is $\Omega(\log^{2-\epsilon} n)$ hard. In fact, the hardness-proof already works for the stochastic shortest path problem.

Besides the design of survivable networks a lot of research has been done concerning the design of reliable networks; recent articles can be found in the special issue by Rak and Sterbenz [149]. Design of reliable networks under network uncertainty using the approach of *chance-constrained programming* (see, e.g., Prékopa [143]) has been considered in Song and Luedtke [164] and Song and Zhang [165]. In chance-constrained programming, there is usually one decision horizon, i.e., no recourse, and a feasible solution has to satisfy the constraints with a given probability. [164, 165] considered s-t-paths and the Steiner tree problem under possible network failure scenarios. In contrast to the SSNDP studied in this thesis, these problems assume that the set of customers remains the same across all scenarios but a whole subnetwork can be subject to failure. Each failure scenario happens with a certain probability and the goal is to find a reliable network that ensures given connectivity requirements with a certain probability. The authors introduced several (M)IP formulations, facet-defining inequalities, and provided computational studies.

9.2 IP formulations

In Section 9.2.1 we first present the undirected cut-based formulation for the SSNDP. In Section 9.2.2 we show how to derive stronger formulations using the orientation properties presented in Section 3.4.2 by assigning a unique direction to each edge of a feasible second-stage solution.

9.2.1 Undirected formulation

Let the binary variable x_e^0 indicate whether edge $e \in E$ belongs to the first-stage solution and binary second-stage variable x_e^k indicate whether e belongs to the solution edges of scenario k, for all scenarios $k \in \mathcal{K}$. Moreover, we expand the connectivity function f from

the deterministic SNDP to f^k, for each scenario $k \in \mathcal{K}$ and $S \subset V$:

$$f^k(S) := \max \left\{ \rho^k_{uv} \,\middle|\, u \notin S, v \in S \right\}$$

Then, the undirected cut formulation for the SSNDP reads as follows:

$$\begin{aligned}
(\text{SSNDP}_{\text{uc}}) \ \min & \sum_{e \in E} c^0_e x^0_e + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} c^k_e x^k_e \\
\text{s.t. } & (x^0 + x^k)(\delta(S)) \geq f^k(S) && \forall k \in \mathcal{K}, \forall \emptyset \neq S \subset V && (9.1) \\
& x^0_e + x^k_e \leq 1 && \forall k \in \mathcal{K}, \forall e \in E && (9.2) \\
& x^{0\ldots K} \in \{0,1\}^{|E| \cdot (K+1)} && && (9.3)
\end{aligned}$$

This model is a direct extension of the model from Section 3.2. Constraints (9.1) ensure edge-connectivity between each pair of nodes in each scenario while first- and second-stage edges can be used. The additional constraints (9.2) forbid the installation of the same edge in the first stage and in any scenario.

Observation 9.2. *Formulation ($SSNDP_{uc}$) models the stochastic survivable network design problem correctly.*

Polyhedral properties. Let $\mathcal{S}^k$ be the convex hull of all integer points that define feasible SNDP solutions w.r.t. connectivity requirements $\rho^k, \forall k \in \mathcal{K}$, i.e.,

$$\mathcal{S}^k = \text{conv} \left\{ x^k \in \{0,1\}^{|E|} \,\middle|\, x^k(\delta(S)) \geq f^k(S), \forall \emptyset \neq S \subset V \right\}$$

Similarly, let $\mathcal{S}$ be the convex hull of all integer points that define feasible SSNDP solutions, i.e.,

$$\mathcal{S} = \text{conv} \left\{ x^{0\ldots K} \in \{0,1\}^{|E| \cdot (K+1)} \,\middle|\, x^{0\ldots K} \text{ satisfies (9.1) and (9.2)} \right\}$$

In the following, we study some properties of the polytope $\mathcal{S}$.

Lemma 9.3. *If the polytope $\mathcal{S}^k$ is full-dimensional for each $k \in \mathcal{K}$, then the polytope $\mathcal{S}$ is full-dimensional as well.*

Proof. Let $m := |E|$. We need to show that $\dim(\mathcal{S}) = m(K+1)$. To this end, we construct a matrix M that contains $m(K+1)$ linearly independent feasible solutions to the SSNDP. In the last step we extend it by one additional solution with the whole collection of solutions being affinely independent.

The matrix M contains $(K+1) \cdot (K+1)$ blocks of size $m \times m$ and each row of the matrix represents one feasible solution in $\mathcal{S}$. Each block column corresponds to a binary variable of the vector $x^{0\ldots K}$, the first m rows represent feasible independent solutions involving the x^0 variables, and the next Km solutions are linearly independent with respect to the x^k variables, for all $k \in \mathcal{K}$.

For each edge $e \in E$ let the solution s_e contain all edges except e. Then, we observe that for each scenario $k \in \mathcal{K}$ the collection of the m solutions $\mathcal{E} = \cup_{e \in E} s_e$ represents a set of m linearly independent points of the polytope $\mathcal{S}^k$. Let A_0 denote the $m \times m$ matrix obtained by row-wise concatenation of the characteristic vectors of these solutions, i.e., $A_0 = \mathbf{1} - I_m$ with the $m \times m$ identity matrix I_m. The matrix M is constructed as follows.

$$\begin{bmatrix} A_0 & I_m & I_m & I_m & I_m \\ \mathbf{0} & A_0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & A_0 & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & A_0 & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & A_0 \end{bmatrix} \quad \begin{bmatrix} A^\ell & \mathbf{1} - A^\ell & \mathbf{1} - A^\ell & \mathbf{0} & \mathbf{1} - A^\ell \\ \mathbf{0} & A_0 & \mathbf{1} & A^\ell & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & A_0 & A^\ell & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & A^\ell & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & A^\ell & A_0 \end{bmatrix}$$

(a) (b)

Figure 9.1: Structures of the constructed matrices with $K = 4$. (a) Lemma 9.3 and (b) Theorem 9.4; for the latter we have $\ell = 3$.

1. Initialize the first $m \times m$ block with A_0. Fill out the remaining K blocks at position $[0, k]$, $k \in \mathcal{K}$, with I_m.

2. For $k \in \mathcal{K}$: set up the block at the position $[k, 0]$ to $\mathbf{0}$, and the block at the position $[k, k]$ to A_0. The remaining blocks at positions $[\ell, k]$ are set to $\mathbf{1}$, for all $\ell \in \mathcal{K}, \ell \neq k$.

It is not difficult to see that the obtained matrix (cf. Figure 9.1) has full rank $m(K + 1)$. In the last step, we add the vector that is obtained by concatenating the $\mathbf{0}$ vector solution for x^0 and $\mathbf{1}$ for the remaining coordinates x^1 to x^K. Subtracting all solutions contained in the matrix from the latter solution gives a new matrix—with full rank, too. Hence, all solutions are affinely independent. □

Theorem 9.4. *If for all $k \in \mathcal{K}$, the polytopes $\mathcal{S}^k$ are full-dimensional and the inequality $\pi x^\ell \geq \pi_0$, with coefficients $\pi_e \in \mathbb{N}, \forall e \in E$, and $\pi_0 \geq 1$, defines a facet of the polytope $\mathcal{S}^\ell$ for some $\ell \in \mathcal{K}$, then the inequality $\pi x^0 + \pi x^\ell \geq \pi_0$ is facet defining for the polytope $\mathcal{S}$.*

Proof. We denote the affine independent solutions of the polytope $\mathcal{S}^\ell$ that satisfy $\pi x^\ell = \pi_0$ by $T_1^\ell, \ldots, T_m^\ell$. Since $\pi_0 \geq 1$, these points are also linearly independent (the origin does not belong to the set of feasible points). Let A^ℓ be the matrix obtained by the row-wise insertion of these solutions, and let $\mathbf{1} - A^\ell$ be the complementary matrix of A^ℓ. Construction is done by a row-wise insertion of blocks, similarly to above.

1. Initialize the first $m \times m$ block with A^ℓ (meaning, set first-stage solutions to be equal to the solutions of $\mathcal{S}^\ell$). Fill out the remaining blocks at the position $[0, k]$ with $\mathbf{1} - A^\ell$, for all $k \in \mathcal{K}$, $k \neq \ell$. The block at the position $[0, \ell]$ is set to $\mathbf{0}$.

2. For $k \in \mathcal{K}$, $k \neq \ell$: set up the block at the position $[k, 0]$ to $\mathbf{0}$, at the position $[k, \ell]$ to A^ℓ and the block at the position $[k, k]$ to A_0 (defined above). The remaining blocks at $[k, i]$ are set to $\mathbf{1}$, for all $i \in \mathcal{K}, i \neq k, \ell$.

3. Set up the block at the position $[\ell, 0]$ to $\mathbf{0}$, and the block at $[\ell, \ell]$ to A^ℓ. The remaining blocks at $[\ell, i]$ are set to $\mathbf{1}$, for all $i \in \mathcal{K}, i \neq \ell$.

It is not difficult to see that the obtained matrix (cf. Figure 9.1) has full rank, i.e., $m(K + 1)$, and each row satisfies $\pi x^0 + \pi x^\ell = \pi_0$ which concludes the proof. □

Hence, under some special conditions (given in Stoer [166]), the undirected cuts (9.1) are facet-defining. Moreover, many facet-defining inequalities known for the deterministic

case, e.g., partition inequalities (again under some conditions given by [166]), can therefore be easily translated into facets of (SSNDP_{uc}).

In the following section we introduce stronger semi-directed models for which no similar results are known. This leads to the following open problem.

Open problem 9.1. Is it possible to derive properties of the polytopes of the semi-directed models for the SSNDP from the deterministic models, similar to the undirected model?

9.2.2 Semi-directed formulations

In the following, we will introduce two semi-directed models that are stronger than the undirected model (SSNDP_{uc}). As for the SSTP, the main difficulty with the SSNDP arises from the fact that the first-stage solution may be disconnected. Hence, the first-stage decision variables remain associated with undirected edges. Once the solution gets completed in the second stage, one can provide a valid orientation, i.e., one can orient the edges of $E^0 \cup E^k$ independently for each scenario. For each scenario $k \in \mathcal{K}$ we set the root r^k to be one of the vertices with the highest connectivity requirement and search for individual orientations of the combined solution $E^0 \cup E^k$.

By borrowing the notation from Balakrishnan, Magnanti, and Mirchandani [9], let

$$\mathcal{S}_1^k := \left\{ S \subset V \,\middle|\, f^k(S) = 1, r^k \notin S \right\}$$
$$\mathcal{S}_{\geq 2}^k := \left\{ S \subset V \,\middle|\, f^k(S) \geq 2 \right\}$$

be the set of *regular cutsets* and *critical cutsets*, respectively.

Given the installation of undirected edges from the first stage, the following model constructs oriented second-stage solutions. As before, we use variables $x^{0\ldots K}$ to model the solution edges. In addition, we introduce a continuous variable z_{ij}^k associated to a directed arc $(i, j) \in A$ and a scenario $k \in \mathcal{K}$ to "orient" the second-stage solutions. The first semi-directed model is called ($\text{SSNDP}_{\text{sdc1}}$):

$$(\text{SSNDP}_{\text{sdc1}}) \ \min \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} c_e^k x_e^k$$

$$\text{s.t. } (x^0 + x^k)(\delta(S)) \geq f^k(S) \qquad \forall k \in \mathcal{K}, \forall S \in \mathcal{S}_{\geq 2}^k \tag{9.4}$$

$$(x^0 + z^k)(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \in \mathcal{S}_1^k \tag{9.5}$$

$$x_e^k \geq z_{ij}^k + z_{ji}^k \qquad \forall k \in \mathcal{K}, \forall e = \{i, j\} \in E \tag{9.6}$$

$$x_e^0 + x_e^k \leq 1 \qquad \forall k \in \mathcal{K}, \forall e \in E \tag{9.7}$$

$$z^{1\ldots K} \geq \mathbf{0} \tag{9.8}$$

$$x^{0\ldots K} \in \{0, 1\}^{|E| \cdot (K+1)} \tag{9.9}$$

Constraints (9.4) ensure that in each scenario k there are at least ρ_{uv}^k edge-disjoint paths between u and v, with $u \in S, v \notin S$, consisting of first- and second-stage edges. Due to constraints (9.5) there is at least one semi-directed path from a root node r to a vertex u whenever $\rho_{ru}^k = 1$. If an edge is purchased in the second stage, then constraints (9.5) associated to bridges will force the orientation of those bridges away from the root node r^k. Furthermore, since variables z_{ij}^k are fractional, by using the same arguments as in Lemma 3.11 the model is valid for any $\rho_{uv}^k \in \mathbb{N}$. Hence, we have the following lemma.

Lemma 9.5. *Formulation ($SSNDP_{sdc1}$) models the stochastic survivable network design problem correctly.*

Proof. Consider an optimal solution to an SSNDP instance with its characteristic binary vector being described by $\tilde{x}^{0\dots K}$. Using this $\tilde{x}^{0\dots K}$ as solution to ($\text{SSNDP}_{\text{sdc1}}$), all undirected cuts (9.4) are obviously satisfied. Moreover, due to Theorem 3.10 and 3.11 it is possible to find an orientation in each scenario such that constraints (9.5) are satisfied. Following this orientation the values for the directed variables z^k can be set accordingly. Non-negativity and all other constraints follow directly. Hence, there exists a feasible solution to ($\text{SSNDP}_{\text{sdc1}}$) with the same objective value.

On the other hand, each optimum solution to ($\text{SSNDP}_{\text{sdc1}}$) obviously satisfies all connectivity requirements and induces a solution to the SSNDP with the same cost. □

Remark 9.6. Note that constraints (9.4) can also be expressed as:

$$x^0(\delta(W)) + z^k(\delta^-(W)) + z^k(\delta^+(W)) \geq f^k(W)$$

Maybe these constraints better explain the original intention of this model. However, one easily observes that this is just an equivalent way of rewriting (9.4), without any influence on the lower bounds of the given model.

Let the polytopes of the undirected and semi-directed formulations be denoted by:

$$\mathcal{P}_{\text{uc}}^{\text{SSNDP}} = \left\{ x^{0\dots K} \in [0,1]^{|E|\cdot(K+1)} \,\middle|\, x^{0\dots K} \text{ satisfies (9.1) and (9.2)} \right\}$$

$$\mathcal{P}_{\text{sdc1}}^{\text{SSNDP}} = \left\{ (x^{0\dots K}, z^{1\dots K}) \in [0,1]^{|E|\cdot(K+1)} \times [0,1]^{|A|\cdot K} \,\middle|\, (x^{0\dots K}, z^{1\dots K}) \text{ satisfies (9.4)–(9.7)} \right\}$$

We consider the linear projection onto the space of undirected edge variables $x^{0\dots K}$:

$$\text{Proj}_{x^{0\dots K}}\left(\mathcal{P}_{\text{sdc1}}^{\text{SSNDP}}\right) = \left\{ x^{0\dots K} \,\middle|\, \exists z^{1\dots K} \colon (x^{0\dots K}, z^{1\dots K}) \in \mathcal{P}_{\text{sdc1}}^{\text{SSNDP}} \right\}$$

Lemma 9.7. *$\mathcal{P}_{uc}^{SSNDP} \supsetneq Proj_{x^{0\dots K}}\left(\mathcal{P}_{sdc1}^{SSNDP}\right)$, i.e., the semi-directed cut-based formulation ($SSNDP_{sdc1}$) is stronger than the undirected cut-based formulation ($SSNDP_{uc}$).*

Proof. It is easy to see that any solution $\tilde{x}^{0\dots K} \in \text{Proj}_{x^{0\dots K}}\left(\mathcal{P}_{\text{sdc1}}^{\text{SSNDP}}\right)$ is a valid solution to (SSNDP_{uc}) with the same objective value. The strict inequality follows from the example given by Figure 6.4 for the SSTP and models ($\text{SSTP}_{\text{sdc1}}$) and ($\text{SSTP}_{\text{uc}}$). □

Stronger semi-directed formulation. The following model represents an alternative model to ($\text{SSNDP}_{\text{sdc1}}$) and follows the ideas of model ($\text{SSTP}_{\text{sdc2}}$) for the SSTP. Thereby, binary edge variables y^k, $\forall k \in \mathcal{K}$, are used to model the second-stage solution; notice that contrarily to the SSTP these variables are undirected edge variables. These variables additionally include the edges that are already bought in the first stage, i.e., we have $y_e^k = 1$

if $e \in E^0 \cup E^k$, and $y_e^k = 0$, otherwise. Moreover, continuous variables z_{ij}^k are used to orient the edges from $E^0 \cup E^k$, $\forall k \in \mathcal{K}, \forall (i,j) \in A$. The model is called (SSNDP$_{\text{sdc2}}$):

$$\text{(SSNDP}_{\text{sdc2}}\text{)} \quad \min \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} c_e^k (y_e^k - x_e^0)$$

$$\text{s.t. } z^k(\delta^-(S)) \geq f^k(S)/2 \qquad \forall k \in \mathcal{K}, \forall S \in \mathcal{S}_{\geq 2}^k \tag{9.10}$$

$$z^k(\delta^-(S)) \geq 1 \qquad \forall k \in \mathcal{K}, \forall S \in \mathcal{S}_1^k \tag{9.11}$$

$$z_{ij}^k + z_{ji}^k \geq x_e^0 \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E \tag{9.12}$$

$$y_e^k \geq z_{ij}^k + z_{ji}^k \qquad \forall k \in \mathcal{K}, \forall e = \{i,j\} \in E \tag{9.13}$$

$$z^{1\ldots K} \geq \mathbf{0} \tag{9.14}$$

$$x^0 \in \{0,1\}^{|E|} \tag{9.15}$$

$$y^{1\ldots K} \in \{0,1\}^{|E| \cdot K} \tag{9.16}$$

The directed cuts (9.10) and (9.11) model the orientation of the solution and ensure the required connectivities independently for each scenario. Notice that due to the symmetry, if $S \in \mathcal{S}_{\geq 2}^k$, it follows that $V \setminus S \in \mathcal{S}_{\geq 2}^k$, too. Hence, for each $S \in \mathcal{S}_{\geq 2}^k$ the ingoing and outgoing cut, i.e., $z^k(\delta^-(S)) \geq f^k(S)/2$ and $z^k(\delta^+(S)) \geq f^k(S)/2$, is contained in (SSNDP$_{\text{sdc2}}$).

Constraints (9.12) and (9.13) ensure that variables z_{ij}^k can be used only along the edges that are either purchased in the first stage or added in the second stage. In particular, (9.12) forces the orientation of selected first-stage edges in each scenario. Therefore, these constraints strengthen the model as they impose restrictions on the first-stage solutions: only first-stage solutions allowing for feasible orientations are valid. Moreover, it holds $y_e^k \geq x_e^0$ which explains the corrective term in the objective function. Since the variables $z^{1\ldots K}$ are fractional, the model is valid.

Lemma 9.8. *Formulation* (*SSNDP$_{sdc2}$*) *models the stochastic survivable network design problem correctly.*

Proof. Let $\tilde{x}^{0\ldots K}$ describe an optimal solution to an SSNDP instance. Now, consider $(\bar{x}^0 := \tilde{x}^0, (\bar{y}^k := \tilde{x}^0 + \tilde{x}^k)_{k=1,\ldots,K})$. It is easy to find an orientation for the directed arc variables z^k using only edges $\{e \in E \mid \tilde{x}_e^0 + \tilde{x}_e^k = 1\} = \{e \in E \mid \bar{y}_e^k = 1\}$ for each scenario k and hence create a valid solution to (SSNDP$_{\text{sdc2}}$) with the same objective value.

Conversely, an optimal solution $(\bar{x}^0, \bar{y}^{1\ldots K}, \bar{z}^{1\ldots K})$ to (SSNDP$_{\text{sdc2}}$) satisfies all edge connectivity requirements in each scenario due to the correctness of the directed formulation (SNDP$_{\text{dc}}$). Thereby, capacity constraints (9.12) and (9.13) imply that $(\tilde{x}^0 := \bar{x}^0, (\tilde{x}^k := \bar{x}^0 - \bar{y}^k)_{k=1,\ldots,K})$ is a feasible solution to the SSNDP with the same cost. □

Similar to the stochastic Steiner tree model (SSTP$_{\text{sdc2}}$) the first-stage variables can be relaxed without losing overall integrality, compare Lemma 6.3. We denote the model with relaxed first-stage variables by (SSNDP$_{\text{sdc2}}^{\text{rel:x}^0}$).

Lemma 9.9. *The optimum solution to* (*SSNDP$_{sdc2}^{rel:x^0}$*) *is integer.*

Proof. Assume there exists an optimum solution $(\tilde{x}^0, \tilde{y}^{1\ldots K}, \tilde{z}^{1\ldots K})$ to (SSNDP$_{\text{sdc2}}^{\text{rel:x}^0}$) such that a variable $\tilde{x}_e^0$ corresponding to edge $e = \{i,j\} \in E$ is fractional. The term in the objective function corresponding to edge e is $(c_e^0 - c_e^*)\tilde{x}_e^0 + \sum_{k \in \mathcal{K}} p^k c_e^k \tilde{y}_e^k$.

If $c_e^0 < c_e^*$ set $\tilde{x}_e^0 := 1$ and, if necessary, increase z_{ij}^k or z_{ji}^k (or both) for all scenarios $k \in \mathcal{K}$ such that $\tilde{z}_{ij}^k + \tilde{z}_{ji}^k = 1$. If $c_e^0 > c_e^*$ set $\tilde{x}_e^0 := 0$. Since $\tilde{y}_e^k = 1, \forall k \in \mathcal{K}$, all constraints are still satisfied and the new solution is feasible. Moreover, the objective value improves in both cases which is a contradiction.

In case $c_e^0 = c_e^*$ variable x_e^0 has coefficient 0 in the objective function and can be fixed to $\tilde{x}_e^0 := 0$. □

Let the polytope of the second semi-directed model and the projection onto the undirected variable space be denoted by:

$$\mathcal{P}_{\text{sdc2}}^{\text{SSNDP}} = \Big\{(x^0, y^{1\dots K}, z^{1\dots K}) \in [0,1]^{|E|} \times [0,1]^{|E|\cdot K} \times [0,1]^{|A|\cdot K} \Big| \\ (x^0, y^{1\dots K}, z^{1\dots K}) \text{ satisfies (9.10)–(9.13)}\Big\}$$

$$\text{Proj}_{x^{0\dots K}}\left(\mathcal{P}_{\text{sdc2}}^{\text{SSNDP}}\right) = \Big\{x^{0\dots K} \Big| \exists (y^{1\dots K}, z^{1\dots K})\colon (x^0, y^{1\dots K}, z^{1\dots K}) \in \mathcal{P}_{\text{sdc2}}^{\text{SSNDP}}, \\ x_e^k = y_e^k - x_e^0, \forall k \in \mathcal{K}, \forall e \in E\Big\}$$

Theorem 9.10. *$Proj_{x^{0\dots K}}\left(\mathcal{P}_{sdc1}^{SSNDP}\right) \supsetneq Proj_{x^{0\dots K}}\left(\mathcal{P}_{sdc2}^{SSNDP}\right)$, i.e., the semi-directed cut-based formulation ($SSNDP_{sdc2}$) is stronger than the semi-directed cut-based formulation ($SSNDP_{sdc1}$).*

Proof. Let $(\hat{x}^0, \hat{y}^{1\dots K}, \hat{z}^{1\dots K}) \in \mathcal{P}_{\text{sdc2}}^{\text{SSNDP}}$. For each $k \in \mathcal{K}$ and $(i,j) \in A$ set $\lambda_{ij}^k := 0$ if $\hat{z}_{ij}^k + \hat{z}_{ji}^k = 0$ and $\lambda_{ij}^k := \hat{z}_{ij}^k/(\hat{z}_{ij}^k + \hat{z}_{ji}^k)$, otherwise. Hence, $\lambda_{ij}^k + \lambda_{ji}^k = 1, \forall \{i,j\} \in E$ and $k \in \mathcal{K}$ with $\hat{z}_{ij}^k + \hat{z}_{ji}^k > 0$. Moreover, set $\bar{x}^0 := \hat{x}^0, \bar{x}^k := \hat{y}^k - \hat{x}^0$, and for all arcs $(i,j) \in A$ with $e = \{i,j\}$ set $\bar{z}_{ij}^k := \hat{z}_{ij}^k - \lambda_{ij}^k \hat{x}_e^0$, for all $k \in \mathcal{K}$.

Interpreting $(\bar{x}^{0\dots K}, \bar{z}^{1\dots K})$ as solution to ($\text{SSNDP}_{\text{sdc1}}$) obviously gives the same objective value. This solution is also feasible due to the following arguments.

The connectivity constraints (9.4) are satisfied since for each $S \in \mathcal{S}_{\geq 2}^k$, $k \in \mathcal{K}$, we have:

$$(\bar{x}^0 + \bar{x}^k)(\delta(S)) = \hat{x}^0(\delta(S)) + \hat{y}^k(\delta(S)) - \hat{x}^0(\delta(S)) \\ \overset{(9.13)}{\geq} \sum_{e=\{i,j\}\in\delta(S)} (\hat{z}_{ij}^k + \hat{z}_{ji}^k) = \hat{z}^k(\delta^-(S)) + \hat{z}^k(\delta^+(S)) \overset{(9.10)}{\geq} f^k(S)$$

The Steiner cuts (9.5) are also fulfilled since for each $S \in \mathcal{S}_1^k$, $k \in \mathcal{K}$, we have:

$$\bar{x}^0(\delta(S)) + \bar{z}^k(\delta^-(S)) = \hat{x}^0(\delta(S)) + \sum_{\substack{(i,j)\in\delta^-(S),\\ e=\{i,j\}}} (\hat{z}_{ij}^k - \lambda_{ij}^k \hat{x}_e^0) \\ \geq \hat{z}^k(\delta^-(S)) \overset{(9.11)}{\geq} 1$$

Constraints (9.6) are satisfied since $\bar{z}_{ij}^k + \bar{z}_{ji}^k = \hat{z}_{ij}^k + \hat{z}_{ji}^k - \hat{x}_e^0 \leq \hat{y}_e^k - \hat{x}_e^0 = \bar{x}_e^k$, and the same holds for constraints (9.7) since $\bar{x}_e^0 + \bar{x}_e^k = \hat{x}_e^0 + \hat{y}_e^k - \hat{x}_e^0 \leq 1$. Moreover, z^k variables are non-negative: $\bar{z}_{ij}^k = \hat{z}_{ij}^k - (\hat{z}_{ij}^k/(\hat{z}_{ij}^k + \hat{z}_{ji}^k))\hat{x}_e^0 \geq \hat{z}_{ij}^k - (\hat{z}_{ij}^k/(\hat{z}_{ij}^k + \hat{z}_{ji}^k))(\hat{z}_{ij}^k + \hat{z}_{ji}^k) = 0$.

Last but not least, it trivially holds $\bar{x}^0 \in [0,1]^{|E|}$ and $\bar{x}^k \in [0,1]^{|E|}, \forall k \in \mathcal{K}$, since $\bar{x}_e^k = \hat{y}_e^k - \hat{x}_e^0$ and $\hat{y}_e^k \geq \hat{x}_e^0$. Hence, $(\bar{x}^{0\dots K}, \bar{z}^{1\dots K})$ is a feasible solution to ($\text{SSNDP}_{\text{sdc1}}$) with the same objective value.

For an instance with strict inequality consider the example from Theorem 6.13 which shows that ($\text{SSTP}_{\text{sdc1}}$) $\supsetneq$ ($\text{SSTP}_{\text{sdc2}}$). □

Discussion. We close the part on the semi-directed models for the SSNDP by discussing some related (open) problems.

The first open problem concerns the strength of the semi-directed models, which are not stronger than the undirected model if the set of Steiner cuts is empty. In fact, this question already concerns the deterministic SNDP:

Open problem 9.2. Is it possible to model the SNDP such that this model is stronger than (NDP_{uc}) even when the set of Steiner cuts is empty?

Second, we briefly consider a problem closely related to the SSNDP, namely the *stochastic node-connectivity SNDP*. Thereby, a feasible solution has to contain ρ^k_{uv} node-disjoint paths between two vertices $u, v \in V, u \neq v, \forall k \in \mathcal{K}$. We remark that—by using orientation properties from Chimani, Kandyba, Ljubić, and Mutzel [38]—the node-connectivity version with vertex types $\{0, 1, 2\}$ can be formulated analogously to ($\text{SSNDP}_{\text{sdc2}}$) which directly leads to a stronger semi-directed model.

Moreover, the described decomposition (Section 9.3) and the L-shaped optimality cuts with their strengthening can be derived from this model in a similar way, to be used within the two-stage b&c algorithm.

Last but not least, we like to mention that several interesting problems are still open for the deterministic SNDP.

Open problem 9.3. Does there exist a stronger model for the node-connectivity SNDP for higher connectivity-requirements > 2?

Open problem 9.4. Does there exist a stronger model for the survivable network design problem with both edge- and node-connectivity requirements?

However, if such models will be formulated in the future, by using the same techniques presented in this thesis, any improvements in the deterministic context should be transferable to the related stochastic problems, formulations, and algorithms.

9.3 Decomposition

For the ease of presentation, we will demonstrate how to solve the SSNDP using the two-stage b&c applied to the strongest model, namely ($\text{SSNDP}_{\text{sdc2}}$). The decomposition, the LPs, and the optimality cuts are similar to the SSTP case and model ($\text{SSTP}_{\text{sdc2}}$), compare Section 7.1. We follow the preceding descriptions and use the same notations as before. In particular, model ($\text{SSNDP}_{\text{sdc2}}$) can be decomposed by introducing $K + 1$ new variables $\theta, \theta^1, \dots, \theta^K$ which represent and replace the K scenario costs as lower bounds. The structure of the relaxed master problem is identical to the one of model ($\text{SSTP}_{\text{sdc2}}$) and reads as follows.

$$(\text{RMP}_{\text{sdc2}}) \quad \min \sum_{e \in E} c^0_e x^0_e + \theta$$

$$\text{s.t. } \theta \geq \sum_{k \in \mathcal{K}} p^k \theta^k \tag{9.17}$$

$$\text{Optimality cuts} \tag{9.18}$$

$$x^0 \in [0, 1]^{|E|} \tag{9.19}$$

$$\theta, \theta^1, \dots, \theta^K \geq 0 \tag{9.20}$$

Again, constraint (9.17) relates the overall second-stage cost represented by θ and the K second-stage costs $\theta^1, \ldots, \theta^K$ for the scenarios. Constraints (9.18) are separated L-shaped optimality cuts and integer optimality cuts, which are described in the following.

For each fixed—and possibly fractional—first-stage solution $\tilde{x}^0$, the second-stage problem decomposes into K independent subproblems, each being a *restricted deterministic SNDP*. They are special cases of the deterministic SNDP due to the capacity constraints (9.12) and (9.22), respectively. To simplify the notation, we define $\mathcal{S}^k := \mathcal{S}^k_1 \cup \mathcal{S}^k_{\geq 2}$ and merge the constraints (9.10) and (9.11) into (9.21) by using functions $\Phi^k \colon 2^V \to \mathbb{N}$, for all $k \in \mathcal{K}$, which give the correct right-hand side of the directed cuts:

$$\Phi^k(S) := \begin{cases} \frac{1}{2} f^k(S), & S \in \mathcal{S}^k_{\geq 2} \\ 1, & S \in \mathcal{S}^k_1 \end{cases}$$

For a given first-stage solution $\tilde{x}^0$, and for each $k \in \mathcal{K}$, the relaxed subproblem—already transformed into standard form—is given as follows:

$$\begin{aligned}
(\mathrm{RSP}_{\mathrm{sdc2}})\ \min & \sum_{e \in E} c^k_e y^k_e - \sum_{e \in E} c^k_e \tilde{x}^0_e \\
\text{s.t. } & z^k(\delta^-(S)) \geq \Phi^k(S) && \forall S \in \mathcal{S}^k && (9.21)\\
& z^k_{ij} + z^k_{ji} \geq \tilde{x}^0_e && \forall e = \{i,j\} \in E && (9.22)\\
& y^k_e - z^k_{ij} - z^k_{ji} \geq 0 && \forall e = \{i,j\} \in E && (9.23)\\
& -y^k_e \geq -1 && \forall e \in E && (9.24)\\
& z^k \geq \mathbf{0} && && (9.25)\\
& y^k \geq \mathbf{0} && && (9.26)
\end{aligned}$$

Notice that, contrarily to the stochastic STP, cf. Section 7.1, the constraints on the upper bound (9.24) are necessary for the SSNDP. By using dual variables α^k_S, β^k_e, γ^k_e, and τ^k_e associated to constraints (9.21), (9.22), (9.23), and (9.24), we obtain the following dual problem, for each scenario $k \in \mathcal{K}$ and fixed first-stage solution $\tilde{x}^0 \in [0,1]^{|E|}$:

$$\begin{aligned}
(\mathrm{D{:}RSP}_{\mathrm{sdc2}})\ \max & \sum_{S \in \mathcal{S}^k} \Phi^k(S)\alpha^k_S + \sum_{e \in E} (\tilde{x}^0_e \beta^k_e - \tau^k_e) - \sum_{e \in E} c^k_e \tilde{x}^0_e \\
& \gamma^k_e - \tau^k_e \leq c^k_e && \forall e \in E && (9.27)\\
\sum_{\substack{S \in \mathcal{S}^k: \\ (i,j) \in \delta^-(S)}} & \alpha^k_S + \beta^k_e - \gamma^k_e \leq 0 && \forall (i,j) \in A, e = \{i,j\} && (9.28)\\
& \alpha^k, \beta^k, \gamma^k, \tau^k \geq \mathbf{0} && && (9.29)
\end{aligned}$$

Let $(\tilde{\alpha}^k, \tilde{\beta}^k, \tilde{\gamma}^k, \tilde{\tau}^k)$ describe an optimum solution to (D:RSP$_{\mathrm{sdc2}}$). A (disaggregated) *L-shaped optimality cut* is then defined as follows:

$$\theta^k + \sum_{e \in E} (c^k_e - \tilde{\beta}^k_e) x^0_e \geq \sum_{S \in \mathcal{S}^k} \Phi^k(S) \tilde{\alpha}^k_S - \sum_{e \in E} \tilde{\tau}^k_e \tag{9.30}$$

Deriving stronger L-shaped cuts. The idea of the method for strengthening the L-shaped optimality cuts for the SSTP and rSSTP, cf. Section 7.2.1, is also applicable here.

Again, if an edge $e \in E$ is not used in the current first-stage solution, i.e., $\tilde{x}^0_e = 0$, then the corresponding dual variable β^k_e does not appear in the objective function of the dual (D:RSP$_{\text{sdc2}}$). Furthermore, the variable γ^k_e does not appear in the objective function either, and therefore, LP-optimal solutions frequently have a positive slack in constraints (9.28). Therefore, the values of the dual variables β^k_e can be increased as follows:

Let $(\tilde{\alpha}^k, \tilde{\beta}^k, \tilde{\gamma}^k, \tilde{\tau}^k)$ be an optimal solution to (D:RSP$_{\text{sdc2}}$) as before. For all edges $e = \{i, j\} \in E$ set

$$\hat{\beta}^k_e := \begin{cases} \tilde{\gamma}^k_e - \max\limits_{a \in \{(i,j),(j,i)\}} \left\{ \sum\limits_{S \in \mathcal{S}^k : a \in \delta^-(S)} \tilde{\alpha}^k_S \right\} & \text{if } \tilde{x}^0_e = 0 \\ \tilde{\beta}^k_e & \text{otherwise.} \end{cases}$$

If $\hat{\beta}^k_e > \tilde{\beta}^k_e$ holds for at least one edge $e \in E$ the strengthened L-shaped cut is given as:

$$\theta^k + \sum_{e \in E} (c^k_e - \hat{\beta}^k_e) x^0_e \geq \sum_{S \in \mathcal{S}^k} \Phi^k(S) \tilde{\alpha}^k_S - \sum_{e \in E} \tilde{\tau}^k_e. \tag{9.31}$$

Theorem 9.11. *The strengthened L-shaped cuts* (9.31) *are valid and stronger than the standard L-shaped cuts* (9.30)*.*

Proof. Obviously, the new values of the dual variables constitute a feasible (and LP-optimal) solution to the dual subproblem (D:RSP$_{\text{sdc2}}$) since $\hat{\beta}^k$ is set without violating any dual constraints.

Furthermore, notice that $\hat{\beta}^k_e \geq \tilde{\beta}^k_e$, for all $e \in E$, and that the right-hand side of both cuts is identical. Since there exists $e_1 \in E$ such that $\hat{\beta}^k_{e_1} > \tilde{\beta}^k_{e_1}$, the coefficient of $x^0_{e_1}$ is smaller for the strengthened L-shaped cut than for the standard one. □

Integer optimality cuts. We use the same notation as in Section 7.2.2. Let $(\tilde{x}^0, \tilde{\theta})$ be a first-stage solution with $\tilde{x}^0$ being binary, and let $\tilde{S}$ denote the associated 1-index set and edge set. Moreover, let $\tilde{q}$ denote the value of the second-stage recourse function w.r.t. $\tilde{x}^0$ (and $\tilde{S}$, respectively), i.e., $\tilde{q} := \sum_{k \in \mathcal{K}} p^k \tilde{q}^k$ with $\tilde{q}^k = Q^k(\tilde{x}^0)$ being the optimum solution value of the subproblem (SP$(k, \tilde{x}^0)$). Last but not least, let L be a valid lower bound for the expected second-stage cost.

To explicitly cut off the solution $(\tilde{x}^0, \tilde{\theta})$ we use the general integer optimality cuts of the integer L-shaped scheme, cf. Section 7.2.2 and Section 2.3.3:

$$\theta + \sum_{e \in \tilde{S}} (L - \tilde{q}) x^0_e + \sum_{e \notin \tilde{S}} (\tilde{q} - L) x^0_e \geq (L - \tilde{q})(|\tilde{S}| - 1) + L \tag{Ic}$$

Moreover, we use the numerically more stable no-good cuts, see Section 7.2.3:

$$\sum_{i \in \tilde{S}} x^0_i - \sum_{i \notin \tilde{S}} x^0_i \leq |\tilde{S}| - 1 \tag{Ng}$$

Section 7.2.2 introduces new and stronger integer optimality cuts for the formulation (SSTP$_{\text{sdc2}}$). The constraints are valid for model (SSNDP$_{\text{sdc2}}$) as well, since the arguments do not depend on the Steiner tree structure of a solution:

1. Observation 7.7 is still valid, i.e., $\tilde{q}$ is a lower bound for the second-stage cost of a solution $\check{S}$ with $\check{S} \subseteq \tilde{S}$. This implies the feasibility of the integer optimality cuts (Ic^-) which are (under two conditions) stronger than the standard integer optimality cuts (Ic), compare Lemma 7.9.

2. For a first-stage solution $\hat{S} \supsetneq \tilde{S}$, with $E' = \hat{S} \setminus \tilde{S}$, and optimum solution $\hat{S}^k$ for scenario $k \in \mathcal{K}$, it holds that $\hat{S}^k \cup E'$ is a feasible solution to scenario k and first-stage solution $\tilde{S}$. Hence, Lemma 7.10 follows which leads to the validity of the integer optimality cuts (Ic^+):

$$\theta + \sum_{e \notin \tilde{S}} \min\{c_e^*, \tilde{q} - L\} x_e^0 \geq \tilde{q} \qquad (\text{Ic}^+)$$

Last but not least, the described disaggregation of the integer optimality cuts in Section 7.2.4 and the cut-based constraints, see Section 7.2.3, are valid for ($\text{SSNDP}_{\text{sdc2}}$), too.

On the other hand, the heuristic *Laminarize* for improving the L-shaped optimality cuts (described in Section 7.2.1) is not (directly) applicable due to the additional cuts with right-hand side ≥ 2. This leads to the following open problem.

Open problem 9.5. Is it possible to extend method *Laminarize* for the stochastic survivable network design problem?

Chapter 10

Computational study

This chapter presents the results of the computational study for the two-stage stochastic survivable network design problem. We start in Section 10.1 by describing the generated instances, the implemented algorithms, and the experimental setup. Section 10.2 describes the experiments and presents the results, e.g., on the value of the stochastic solution, the sample stability, the comparison of the decomposition and the direct approach, and the benefit of the cut strengthening method.

10.1 Computational setup

Considered problem. To evaluate the performance of the two-stage b&c algorithm we focus on the restricted version of the SSNDP where connectivity requirements in each scenario $k \in \mathcal{K}$ are defined by nodes of type two (subset $\mathcal{R}_2^k \subseteq V$), type one (subset $\mathcal{R}_1^k \subseteq V$), and type zero ($V \setminus (\mathcal{R}_2^k \cup \mathcal{R}_1^k)$). The main motivation for this choice is the application in the design of telecommunication networks where nodes of type two are important infrastructure nodes or business customers, nodes of type one are single households, and nodes of type zero are, e.g., street intersections. For two distinct nodes u and v and each scenario $k \in \mathcal{K}$, the connectivity requirement is $\rho_{uv}^k = 2$ if both u and v are in $\mathcal{R}_2^k$, $\rho_{uv}^k = 1$ if one of them is in $\mathcal{R}_1^k$ and the other in $\mathcal{R}_1^k \cup \mathcal{R}_2^k$, and $\rho_{uv}^k = 0$, otherwise.

Benchmark instances. We start by generating deterministic instances by adopting and customizing the idea of Johnson, Minkoff, and Phillips [100] (originally used for the prize-collecting Steiner tree problem), which is frequently used as instance generator in the network design community. After randomly distributing $n \in \{30, 50, 75\}$ points in the unit square, a minimum spanning tree is computed using the points as vertices and the Euclidean distances between all vertex pairs as edge costs. To generate only feasible instances we augment this MST by inserting edges (i.e., biconnectivity augmentation) between leaves which are adjacent in the planar embedding, see Figure 10.1 for an example.

The resulting biconnected graph is extended by adding all edges for which the Euclidean distance is less than or equal to $1.6\alpha/\sqrt{n}$. We have introduced α in order to control the density of the graph. In our experiments we use $\alpha = 0.9$ which leads to graphs with average density 2.07: for $n = 30$, $n = 50$, and $n = 75$ the density is 1.97, 2.11, and 2.12, respectively. For comparison, we remark that the street graphs from the 9th DIMACS challenge on the shortest path problem have an average density of 2.48 [58].

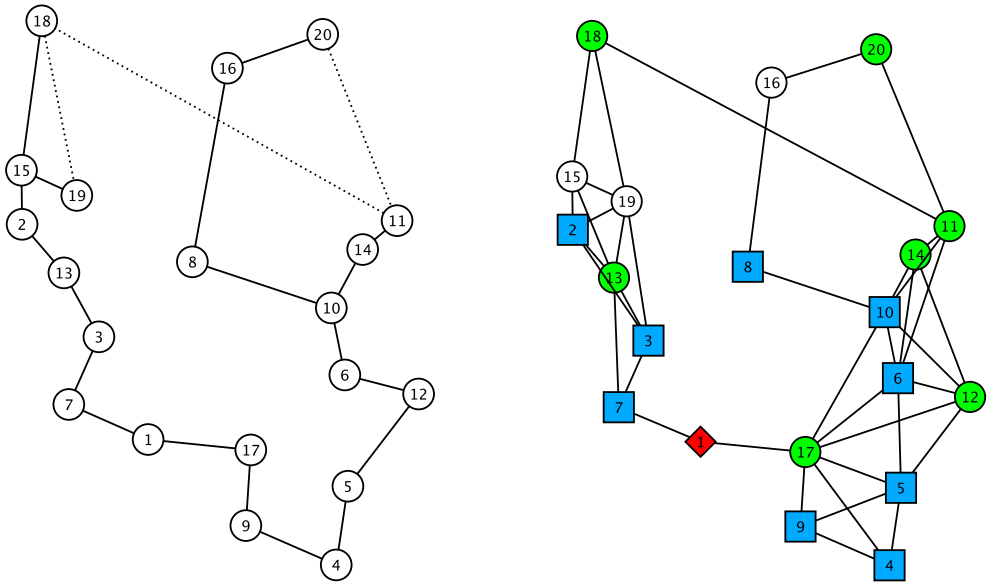


Figure 10.1: (a) A generated underlying graph with 20 vertices. Edges of the minimum spanning tree are drawn as solid lines and edges added through the biconnectivity augmentation as dotted lines. (b) The resulting graph with 40 edges after connecting nearby vertices and adding edge connectivity requirements. The red diamond is the root node, blue rectangles and green circles depict potential $\mathcal{R}_2$ and $\mathcal{R}_1$ nodes, respectively. All other white circles represent nodes without any connectivity requirements.

This method differs from the original method since the original generator by [100] does not compute the minimum spanning tree, the graph is not augmented to biconnectivity, and the original parameter used by [100] is 1.6, which corresponds to $\alpha = 1$ in our setting.

After generating the graph, edge-connectivity requirements are set as follows. We add each vertex with probability $\rho\%$ to the base set of $\mathcal{R}_2$ or $\mathcal{R}_1$ customers, respectively. Here, we use $\rho = 40$ and we additionally select an $\mathcal{R}_2$ vertex randomly as special root node. An example is given by Figure 10.1 (b).

To transform these instances into stochastic ones we use the same method as for the stochastic STP, cf. Section 8.2, by randomly and independently generating $\bar{K} = 1\,000$ scenarios. The probabilities are set by distributing 10 000 points over all $\bar{K}$ scenarios. We start by assigning 1 point to each scenario (1 point corresponds to a probability of 0.01%). Then, we distribute the remaining $10\,000 - \bar{K}$ points by selecting one of the $\bar{K}$ scenarios uniformly at random and increasing its number of points by 1. This procedure continues until all 10 000 points are distributed. Hence, at the end, each scenario has a probability ≥ 0.0001 and all probabilities sum up to 1.

Edge costs c^0 in the first stage are the Euclidean distances and in the second stage for each edge e and scenario $k \in \mathcal{K}$ independently and randomly drawn from the interval $[1.1c_e^0, 1.3c_e^0]$. Edge-connectivity requirements are generated independently for each scenario k by randomly drawing a vertex from the set $\mathcal{R}_1$ as $\mathcal{R}_1^k$ customer with probability $\rho^k\%$, and with the same probability a vertex from $\mathcal{R}_2$ is selected as $\mathcal{R}_2^k$ customer. Here, we use $\rho^k = 30$ for all scenarios. Moreover, the special root node is set to be an $\mathcal{R}_2^k$ node in each scenario.

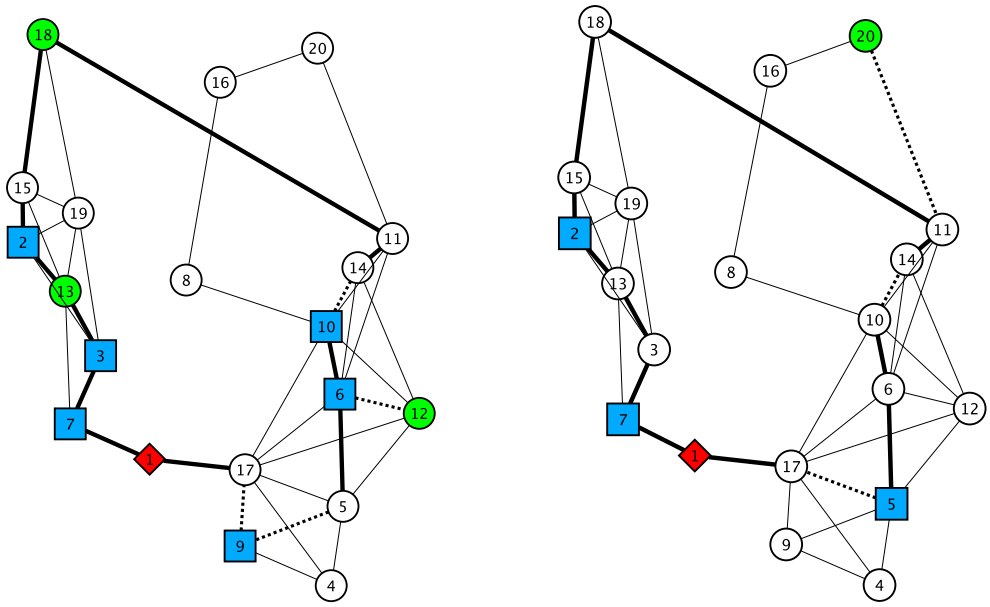


Figure 10.2: An example SSNDP instance with 20 nodes, 40 edges, and 5 scenarios generated from the base instance given in Figure 10.1. Blue rectangles imply connectivity requirement 2 (together with the root node which is the red diamond) and green circles imply connectivity 1. (a) and (b) show the optimal solution for the first and second scenario, respectively, with bold edges being selected first- and dotted edges being selected second-stage edges.

For each deterministic instance we generate $\bar{K} = 1\,000$ scenarios and take the first $K \in K^*$ to obtain an SSNDP instance with K scenarios. Again, we use 14 values for K: $K^* = \{5, 10, 20, 50, 75, 100, 150, 200, 250, 300, 400, 500, 750, 1\,000\}$. Probabilities for the scenarios of the instances with $K < \bar{K}$ are scaled appropriately. Overall, we generate 20 graphs for each $n \in \{30, 50, 75\}$ and $K \in K^*$ leading to 840 instances; these instances can be downloaded from our SSNDP homepage [177]. Due to the high computational effort we do not use all instances in every experiment; the used set will be stated for each experiment independently. Figure 10.2 illustrates an example SSNDP instance based on the instance from Figure 10.1.

Computational settings. We implemented the single-stage b&c and the two-stage b&c for the strongest of the three presented models, namely ($\text{SSNDP}_{\text{sdc2}}$), considering the following settings:

- DA: the direct approach, i.e., one single b&c algorithm applied to ($\text{SSNDP}_{\text{sdc2}}$).
- 2BC: the two-stage b&c algorithm with the separation and strengthening of L-shaped optimality cuts, integer optimality cuts, and no-good cuts.
- NoCS:2BC: the same as 2BC, but without strengthening the L-shaped optimality cuts.
- MW:NoCS:2BC: the same as NoCS:2BC, but with Pareto optimal L-shaped cuts added at each iteration by using the method by Magnanti and Wong [131] (i.e., after

solving the subproblem an additional LP is solved to obtain stronger dual multipliers corresponding to a Pareto optimal L-shaped cut).

- NoNG:2BC: by default 2BC introduces no-good cuts (Ng). NoNG:2BC is the same as 2BC but without using no-good cuts.

The computational setup is identical to the experiments on the stochastic STP. We use ABACUS 3.0 as a generic branch&cut framework with IBM ILOG CPLEX (version 12.2) as LP solver via the interface COIN-Osi. All experiments are performed on an Intel Xeon 2.5 GHz machine with six cores and 64 GB RAM under Ubuntu 12.04. Each run is performed on a single core and the time limit is set to 2 hours (i.e., 7 200 sec.). We perform 5 independent runs for each approach.

Please notice the remarks on the presentation of the results as described at the beginning of Section 8.3 for the experiments on the stochastic Steiner tree problem.

10.2 Experiments

Value of stochastic solution. Since this work presents the first experimental study concerning the SSNDP we start by analyzing the *value of the stochastic solution (VSS)* in order to assess the actual need for formulating the considered deterministic problem as a stochastic problem (for a detailed definition of the VSS and related values we refer the reader to, e.g., Birge and Louveaux [20], Maggioni and Wallace [129]).

To calculate the VSS we first need to find the optimum solution to a deterministic problem in which all random variables are replaced by their expected values (also known as the *expected value problem*, EV). Let $\bar{x}^0$ denote an optimal first-stage solution to this problem. We then evaluate this first-stage solution by considering the EEV (*expected result of the EV solution*): this is the optimal solution value to the original stochastic problem in which the first-stage variables are fixed to $\bar{x}^0$. Finally, the VSS is obtained as VSS = EEV − opt, where opt denotes the optimal SSNDP solution. Hence, the VSS measures the quality of the stochastic solution compared to the solution of the problem using the expected values—which is obviously much easier to compute. The larger the gap between the VSS and the EEV, the more risky and costly it is to replace the uncertain input parameters with their expected values.

For the graphs with 50 vertices, Table 10.1 shows the VSS results, grouped by the number of scenarios K, with $K \in \{5, 10, 20, 50\}$. We report the relative cost increase of the EEV solution compared to opt and the number of edges installed in the first stage (for opt and EEV, respectively). Not surprisingly, the solution costs increase drastically when the EEV is used: on average, EEV solutions are between 20% and 27% costlier than the optimal SSNDP solutions. The average gap between opt and EEV also increases with an increasing number of scenarios. Looking at the structure of optimal EEV solutions we observe that they typically consist of significantly more edges installed in the first stage: on average, between 76%–90% of the overall solution cost is induced by the first-stage solution, whereas for opt the corresponding values range between 35%–50%. Finally, for both opt and EEV we notice that with increasing number of scenarios the number of edges installed in the first stage decreases.

Sample stability. Most stochastic programs cannot be solved (directly) because of, e.g., a continuous distribution of the random variables vector ξ or due to a huge number of

K	value	sol. value increase (%)	nr. edges 1st stage opt	nr. edges 1st stage EV sol.	% 1st stage costs opt	% 1st stage costs EV sol.
5	avg.	20.25	15.95	40.70	50.13	89.08
	min.	7.50	6.00	36.00	26.26	73.54
	max.	35.71	29.00	44.00	77.61	98.69
	std. dev.	7.29	5.01	2.18	12.09	6.80
10	avg.	24.06	14.50	43.65	42.67	87.96
	min.	8.75	7.00	38.00	18.12	71.66
	max.	39.04	29.00	46.00	72.02	99.04
	std. dev.	7.73	6.09	2.21	15.82	6.68
20	avg.	24.96	12.40	43.15	38.04	79.79
	min.	15.39	3.00	39.00	9.98	67.37
	max.	38.16	28.00	48.00	72.69	91.73
	std. dev.	6.16	6.60	2.37	17.00	7.30
50	avg.	26.95	11.63	42.44	35.44	76.60
	min.	19.39	2.00	39.00	10.97	67.29
	max.	34.88	29.00	47.00	73.50	87.90
	std. dev.	5.16	6.88	2.31	17.70	5.80

Table 10.1: Results concerning the VSS for instances with 50 vertices and $K \in \{5, 10, 20, 50\}$ scenarios, cf. text.

possible scenarios K. To create a deterministic equivalent of a reasonable size one typically samples a set of scenarios which then can be solved to optimality. It is therefore important to evaluate the underlying scenario generation procedure and to estimate the required number of scenarios needed to achieve good and stable solutions. Two related quality measures are *in-sample stability* and *out-of-sample stability* whose definitions we briefly recall here (cf. Kaut and Wallace [107], King and Wallace [109] for in-depth discussions).

Consider a two-stage stochastic program in a simplified notation: $\min_{x^0 \in X} f(x^0, \xi)$ where x^0 are the first-stage variables, X is the feasible set, f is the objective function, and ξ the random variables vector (here, we use a simplified notation for a stochastic (mixed-integer) linear program with fixed recourse, i.e., $\min_{x^0 \in X} f(x^0, \xi)$ is $\min_{x^0 \in X} c^T x^0 + Q(x^0, \xi)$, where $Q(x^0, \xi) = \min\{q(\xi)^T y \mid Wy = h(\xi) - T(\xi)x^0,\ y \in Y\}$ with second-stage variables y and feasible set Y, cf. e.g., Birge and Louveaux [20]). Moreover, let $\min_{x^0 \in X} f(x^0, s)$ denote the stochastic program restricted to a (sampled) scenario set s. Now, let $\bar{s}, \hat{s}$ denote two scenario sets of the same size and let $\bar{x}^0, \hat{x}^0$ denote optimal solutions to $\min_{x^0 \in X} f(x^0, \bar{s})$ and $\min_{x^0 \in X} f(x^0, \hat{s})$, respectively. A scenario generation method is called *in-sample stable* if the optimal solution values of two independently sampled scenario sets are similar, i.e., if $f(\bar{x}^0, \bar{s}) \approx f(\hat{x}^0, \hat{s})$. If $f(\bar{x}^0, \xi) \approx f(\hat{x}^0, \xi)$ holds, the method is called *out-of-sample stable*.

For this analysis we consider instances with 50 vertices that can be solved to optimality for 1 000 scenarios (by 2BC). We then compute the in- and out-of-sample stability values by sampling K out of these 1 000 scenarios; we let the sample size K vary between 5 and 500. For each fixed value of K we create 20 instances by sampling K scenarios out of the 1 000. Figure 10.3 shows the results for one representative instance from this set with $|V| = 50$ and $|E| = 100$; the results for the other instances look very similar. Given a fixed value of K the blue crosses and red circles show the distribution of 20 solution values concerning the in-sample and out-of-sample stability, respectively. The solid horizontal line shows the value of the optimal solution i.e., the solution obtained by taking all 1 000 scenarios into account. As one can see, already for $K \geq 20$ scenarios out-of-sample stability can be

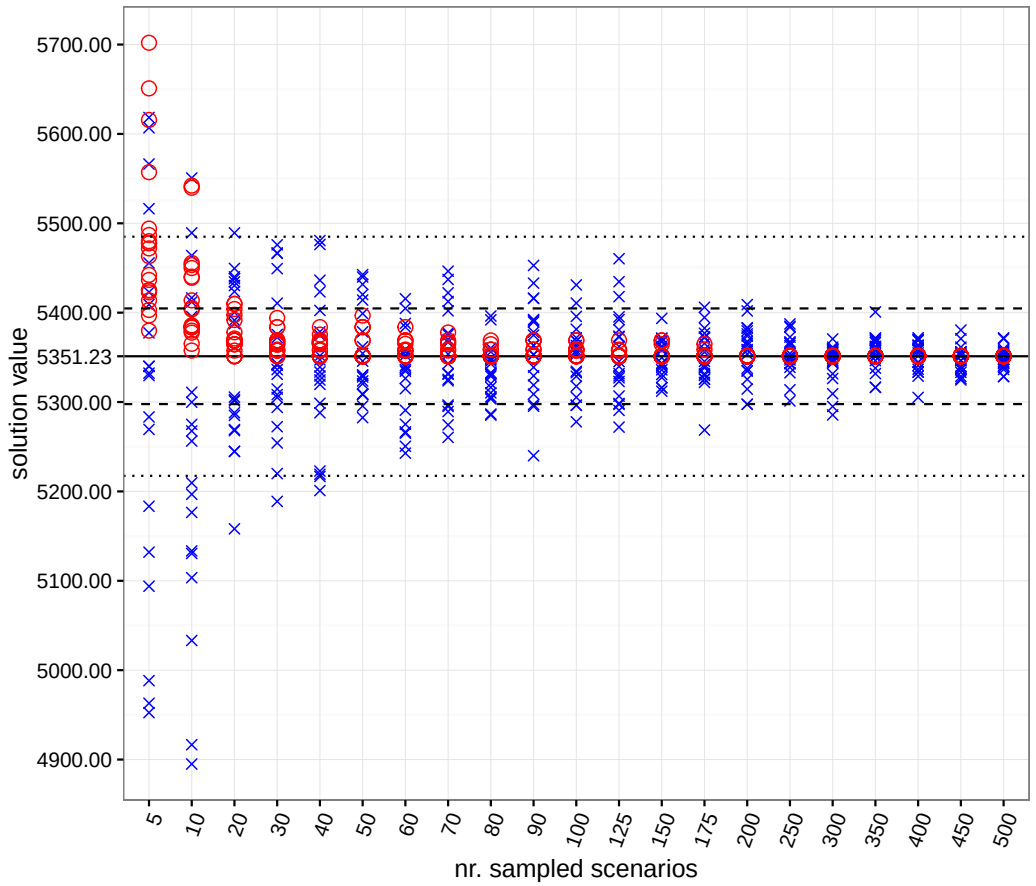


Figure 10.3: In- and out-of-sample stability for an instance with 50 vertices and 100 edges. The horizontal axis gives the sample size and the vertical axis the objective value of in-sample (blue crosses) and out-of-sample stability (red circles), respectively. Each data point represents one sample. The solid horizontal line is the true optimum solution value opt = 5 351.23. The dashed horizontal lines indicate the interval [0.99opt, 1.01opt] and the dotted lines the interval [0.975opt, 1.025opt].

reached. Here, optimal first-stage solutions, evaluated on the whole set of 1 000 scenarios, fall within a 2.5% confidence interval. For $K \geq 30$ they already fall within a 1% confidence interval. Moreover, we report that the out-of-sample stability values equal the optimum for $K \geq 200$. Similarly, in-sample stability can be reached for $K \geq 50$: here, optimal solution values for K-scenario solutions lie within a 5% confidence interval. The 2% confidence interval is more difficult to reach and for this instance $K \geq 350$ scenarios are necessary.

Direct approach vs. decomposition. Figure 10.4 shows a boxplot of the running time for the direct approach DA and the decomposition 2BC on the instances with 30 vertices. We report that all instances with $n = 30$ can be solved to optimality by 2BC (no run reaches the time limit), but there are 115 runs (23 instances) that cannot be solved by DA within the time limit of 2 hours (out of 1 400 runs for 280 instances). Moreover, we observe that for up to 10–20 scenarios DA is mostly faster, but for instances with 20–50 or more scenarios the decomposition clearly outperforms DA; detailed running times can be found in Table 10.2.

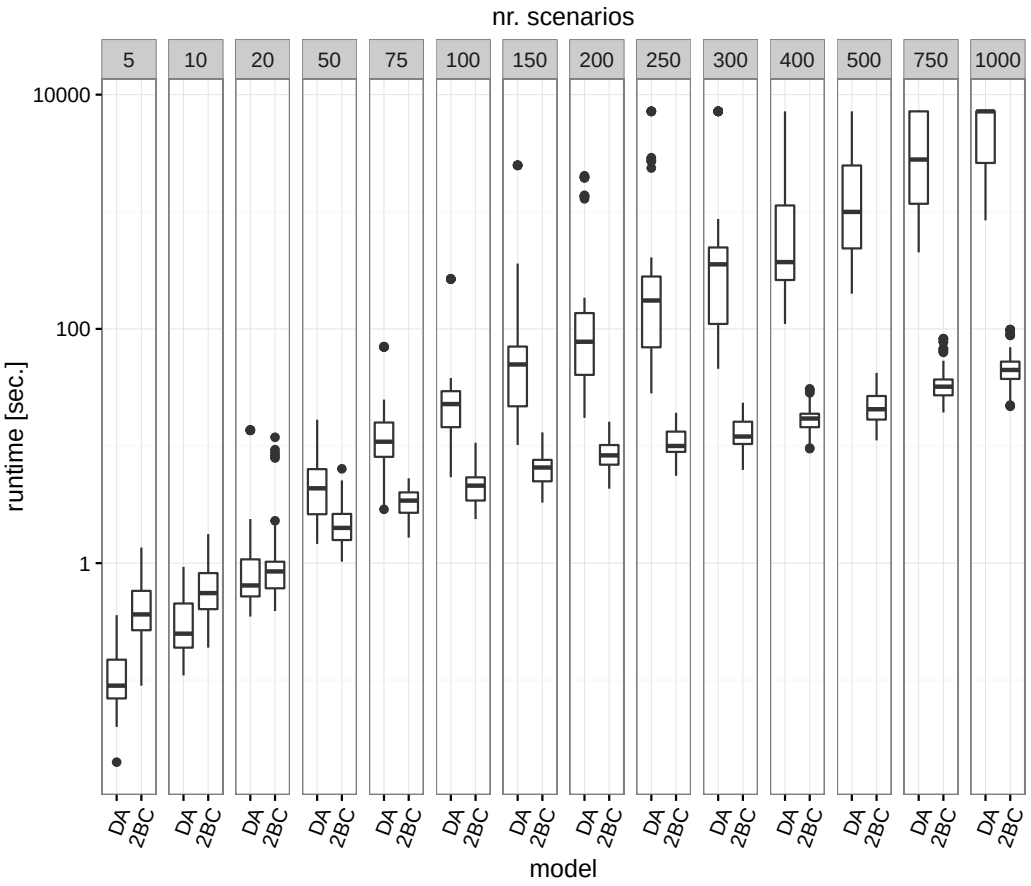


Figure 10.4: Running times of the direct approach DA and the decomposition 2BC on instances with 30 vertices and 5–1000 scenarios. Notice that the vertical axis has logarithmic scale.

Overall, DA takes 383.92 hours on all instances with $n = 30$ and 2BC takes only 4.86 hours. The median speedup is 7.04, on average it is 38.39, and the standard deviation is 80.02. The number of b&b nodes is more than 5 times higher for DA which generates overall 15 100 nodes—2BC generates only 2 690 b&b nodes. However, for most instances this number is low: the median number of b&b nodes is 1 for both approaches, on average DA generates 10.79 nodes and 2BC 1.92 nodes.

For a lower number of scenarios DA is superior due to the set-up overhead needed for the decomposition. With an increasing number of scenarios the two-stage b&c pays off and significantly outperforms the DA approach.

A similar behavior can also be observed on the sets of larger instances: the boxplots in Figure 10.5 and 10.6 show the distribution of the running times for the instances with 50 and 75 vertices, respectively.

First, consider the instances with $n = 50$. Due to the increasing difficulty we consider only instances with 5–250 scenarios (180 instances). Here, 2BC reaches the time limit in 38 runs and 7 instances are unsolved; DA cannot solve 53 instances (267 runs to time limit). Overall, DA takes 662.1 hours for the runs and 2BC takes 102.86 hours. The speedup from DA to 2BC is 8.27 (median) and 36.83 (average) with standard deviation 50.58. The number of b&b nodes is higher for the direct approach where the median number is 27 and on

	avg. runtime		median runtime		std. dev. runtime		avg. nr. b&b nodes	
K	DA	2BC	DA	2BC	DA	2BC	DA	2BC
5	**0.12**	0.45	**0.09**	0.36	**0.08**	0.28	**2.70**	**2.70**
10	**0.35**	0.67	**0.25**	0.56	**0.22**	0.40	3.30	**2.70**
20	1.49	**1.36**	**0.64**	0.85	2.84	**1.91**	4.90	**3.80**
50	5.04	**2.16**	4.36	**2.00**	3.32	**0.88**	2.40	**1.40**
75	14.72	**3.42**	10.89	**3.42**	14.01	**1.04**	4.50	**2.10**
100	33.17	**4.70**	22.81	**4.59**	54.60	**1.67**	6.90	**2.10**
150	182.10	**6.77**	49.72	**6.55**	537.61	**2.41**	22.00	**1.60**
200	238.94	**8.65**	77.73	**8.32**	489.66	**2.68**	15.70	**1.50**
250	640.70	**10.80**	174.84	**10.01**	1614.81	**3.24**	24.54	**1.30**
300	990.83	**12.87**	355.89	**12.05**	2090.20	**3.85**	26.22	**1.60**
400	1244.98	**17.29**	372.14	**17.17**	2038.46	**4.93**	14.58	**1.20**
500	1805.79	**22.23**	997.68	**20.61**	2036.42	**7.60**	11.12	**1.60**
750	3591.45	**35.33**	2792.60	**32.12**	2584.96	**14.90**	8.08	**1.70**
1000	5071.70	**48.25**	7200.00	**44.62**	2409.38	**18.24**	4.06	**1.60**

Table 10.2: Comparison of DA and 2BC for the instances with 30 vertices and 5–1 000 scenarios.

	avg. runtime		median runtime		std. dev. runtime		avg. nr. b&b nodes	
K	DA	2BC	DA	2BC	DA	2BC	DA	2BC
5	**3.23**	366.03	**0.36**	2.31	**10.79**	1 566.69	**70.10**	945.68
10	**6.45**	166.60	**1.88**	5.42	**12.72**	685.98	**40.80**	406.50
20	25.07	**18.95**	10.30	**9.25**	**34.10**	37.53	31.40	**26.50**
50	1 635.89	**388.74**	169.90	**21.46**	2 555.17	**1 570.77**	818.66	**55.66**
75	2 023.34	**427.59**	628.21	**24.00**	2 715.91	**1 570.75**	458.96	**59.92**
100	3 504.95	**417.15**	3 005.85	**32.86**	3 027.41	**1 567.30**	420.64	**46.62**
150	4 954.56	**520.50**	7 200.00	**50.06**	3 082.17	**1 607.90**	215.70	**46.04**
200	5 563.47	**593.47**	7 200.00	**68.48**	2 605.52	**1 674.68**	87.92	**45.46**
250	6 118.87	**804.16**	7 200.00	**87.70**	2 174.36	**2 143.16**	**46.38**	50.22

Table 10.3: Comparison of DA and 2BC for the instances with 50 vertices and 5–250 scenarios.

average it is 243.4 while 2BC generates only 7.0 (median) and 187 (average) b&b nodes.

Although not visible in Figure 10.5, we report that already for 20 scenarios 2BC outperforms DA in general, cf. Table 10.3. Furthermore, the performance of 2BC remains relatively stable, whereas high dispersion and skewness of the running times for DA can be observed. More precisely, 18 out of 20 instances with 50 nodes and 250 scenarios are solved within the time limit of 2 hours using 2BC (the average running time of 2BC is 13 min.), whereas only 5 of them can be solved with the DA approach (with an average running time of 101 min.).

However, there are some few extreme outlier points in the boxplots for the decomposition which highly influence the average running time reported in Table 10.3; in particular for 5 and 10 scenarios the average running time of the decomposition is much higher than the average running time of the direct approach. Moreover, for all scenario numbers—except for 10 and 20—there exist instances where 2BC is failing. The reason for the outlier points of the decomposition are mainly numerical issues; for all of these instances the optimum solution is known early but needs to be verified (many b&b nodes and many tailing off effects). For the unsolved runs of 2BC the median and average number of b&b nodes is 613.0 and 1 889.0, respectively, whereas for the successful runs these values are 5.0 and

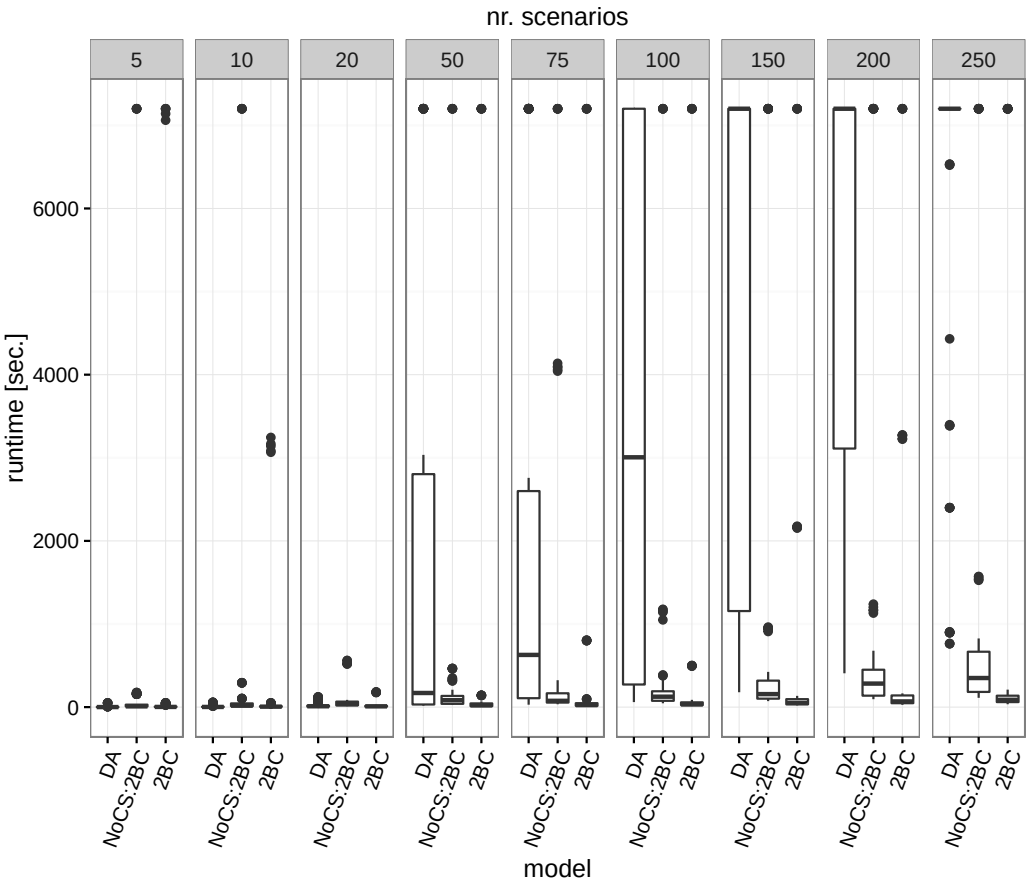


Figure 10.5: Running times of DA, 2BC, and NoCS:2BC (2BC without cut strengthening) on instances with 50 vertices and 5–250 scenarios.

111.9, respectively. The same holds for the tailing off effects: the unsolved runs cause 66.0 (median) and 113.1 (average) tailing off effects and for the successful runs the values are 0 (median) and 2.44 (average).

The picture for the instances with $n = 75$ is similar although the difficulty increases a lot, cf. Figure 10.6 and Table 10.4. Here, we consider only instances with 5–100 scenarios. The decomposition cannot solve 9 (out of 120) instances and the direct approach fails on 45 instances. For the 100 scenario instances 2BC solves 15 out of 20 to optimality and DA solves only 3 within the same time limit. Over all instances with $n = 75$ DA runs take 524.37 hours and 2BC takes 189.42 hours. But, comparing the running times and, in particular, the speedup, is not very interesting since there are so many runs reaching the time limit. In particular, the median running time of the direct approach for 50, 75, and 100 scenarios is defined by the time limit.

The decomposition shows the same problems as for instances with $n = 50$. Due to numerical issues there are some outlier points; for all of these instances the optimum solution is known early but needs to be verified (many b&b nodes and tailing off effects). When comparing the successful and unsuccessful runs the number of tailing off effects increases from 1.0 (median) and 10.87 (average) to 36.0 (median) and 59.02 (average). The same holds for the number of b&b nodes where the values increase from 15.0 (median) and

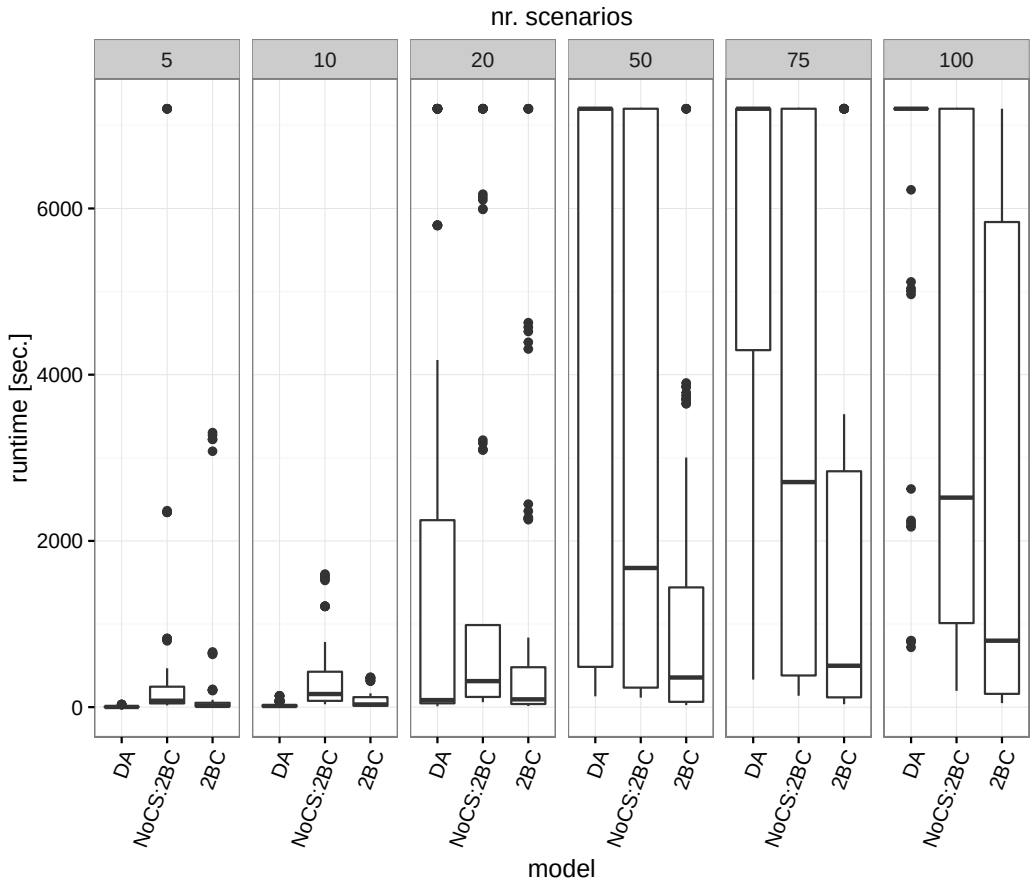


Figure 10.6: Running times of DA, 2BC, and NoCS:2BC on instances with 75 vertices and 5–100 scenarios.

104.1 (average) to 353.0 (median) and 493.7 (median).

Despite the fact that sometimes the decomposition has to deal with numerical problems, we report that the solution values are identical for all instances which can be solved both by DA and 2BC.

Cut strengthening. Figures 10.5 and 10.6 also highlight the benefits of our strengthening procedure. The running times of 2BC are always smaller and less dispersed when compared to the running times of NoCS:2BC. The median and average speedup for the instances with 30 vertices is 3.12 and 3.18, respectively, the maximum is 6.71. Moreover, the speedup increases with an increasing graph size. For instances with 50 vertices it is 3.47 and 4.16, respectively, with maximum 10.15, and the most significant speedup of 16.31 is achieved when solving the largest instances, i.e., graphs with 75 vertices, where the median speedup is 3.98 and the average is 4.1.

The impact of the strengthened L-shaped optimality cuts can be seen best by comparing the number of master iterations in the b&b root node. For 2BC the number of master iterations on the instances with 30 vertices is 11.0 (median), the average is 12.34, and the standard deviation is 4.9. For NoCS:2BC the median is 24.0, the average is 26.36, and the standard deviation is 10.26. Moreover, the number of unsolved instances increases from 7

	avg. runtime		median runtime		std. dev. runtime		avg. nr. b&b nodes	
K	DA	2BC	DA	2BC	DA	2BC	DA	2BC
5	**4.98**	224.12	**2.25**	15.46	**6.96**	705.37	**23.20**	120.80
10	**27.15**	78.82	**14.23**	31.97	**33.72**	98.75	44.30	**40.40**
20	1 760.30	**846.59**	**84.44**	93.71	2 734.20	**1 804.11**	1 636.94	**266.90**
50	5 010.47	**1 254.18**	7 200.00	**356.65**	3 142.74	**1 820.88**	797.98	**125.10**
75	5 537.54	**1 811.28**	7 200.00	**498.04**	2 605.31	**2 505.09**	336.56	**130.42**
100	6 530.23	**2 629.51**	7 200.00	**940.11**	**1 756.17**	2 947.06	170.82	**117.26**

Table 10.4: Comparison of DA and 2BC on the instances with 75 vertices and 5–100 scenarios.

to 11 (for $n = 50$) and from 9 to 25 ($n = 75$); the instances with 30 vertices are easy and each instance can be solved.

Optimality cuts and b&b nodes. Here, we analyze the number of L-shaped and integer optimality cuts and the number of b&b nodes. Notice that since 2BC also inserts no-good cuts the number of integer optimality cuts includes both integer optimality cuts (Ic) and no-good cuts (Ng).

For instances with 30 vertices very few integer optimality cuts are necessary. The median number of integer optimality cuts is 0, the average number is 0.7, the maximum 16, and the standard deviation is 1.58. Hence, these instances are easy and, as for the SSTP, only few integer optimality cuts are required. The number of b&b nodes behaves similar. Here, the median number of b&b nodes is 1.0, the average is 1.9, the maximum 41, and the standard deviation is 2.95.

Not surprisingly, the number of L-shaped cuts increases with an increasing number of scenarios: for 5 scenarios the median is 114.50 and the average is 120.95, and for 1 000 scenarios the median is 9 395 and the average is 9 162.30.

The instances with 50 vertices are more difficult. Here, the median number of integer optimality cuts is 2.0, on average it is 98.22, and the standard deviation is 796.01; detailed values are given by Table 10.5. In general, the number of integer optimality cuts is low and moreover, it is not predictable and there are instances where many integer optimality cuts are required. Over all instances and all runs 88 398 integer optimality cuts are inserted. But there are two instances that together generate 68 930 integer optimality cuts (77.98% of all cuts). These two instances highly influence the average value: without these runs the average is 22.09 and the standard deviation is 458.0.

In general, the number of L-shaped cuts increases with the number of scenarios and the number of b&b nodes and integer optimality cuts remains quite low. However, as mentioned before and as one can see in Figure 10.5, there are outlier points that highly influence the average values and the standard deviation.

For instances with 75 vertices the median number of integer optimality cuts is 8.0, the average is 60.16, and the standard deviation is 140.18. The values are smaller for $n = 50$ since only instances with 5–100 scenarios are solved (instead of 5–250 scenarios) and there are more unsolved instances. Again, the number of b&b nodes is similar: the median is 21.0, on average there are 133.3 nodes, and the standard deviation is 313.0.

Last but not least, we report the number of L-shaped optimality cuts. For 5 scenarios the median number of separated cuts is 602.5 and on average there are 4 571.90 cuts. For 100 scenarios the median is 11 826.5 and the average is 30 658.34.

	nr. int. opt. cuts			nr. L-shaped cuts			nr. b&b nodes		
K	avg.	med.	std. dev.	avg.	med.	std. dev.	avg.	med.	std. dev.
5	520.86	0	2 199.73	7 012.41	242.50	28 426.94	945.68	3	3 986.69
10	193.10	0	806.02	4 553.05	430.00	17 001.77	406.50	5	1 678.75
20	9.60	2	28.45	1 112.45	731.50	1 566.20	26.50	10	75.24
50	24.66	4	71.52	8 955.32	1 188.50	32 562.01	55.66	11	161.90
75	30.96	4	91.95	12 421.19	1 621.00	42 500.32	59.92	7	157.72
100	29.12	4	95.64	10 416.56	2 054.00	33 658.16	46.62	5	138.05
150	26.20	4	67.96	12 486.56	2 849.50	33 197.29	46.04	6	121.47
200	22.54	3	56.77	13 188.84	3 700.00	30 695.46	45.46	7	121.14
250	26.94	3	74.76	15 921.07	5 015.50	33 907.50	50.22	6	152.83

Table 10.5: The number of integer optimality cuts, L-shaped optimality cuts, and b&b nodes for 2BC on instances with $n = 50$ vertices and 5–250 scenarios.

New integer optimality cuts. Here, we evaluate the effect of adding the new integer optimality cuts (Ic^-) and (Ic^+), respectively. For this experiment we consider instances with 50 vertices which are solved by 2BC within the time limit and where 2BC generates ≥ 2 integer optimality cuts; selecting the instances with these properties results in a set with 77 instances which we refer to as N50IC.

We re-run 2BC on N50IC and moreover, we run 2BC- and 2BC+ which is 2BC with additionally generated cuts (Ic^-) or (Ic^+), respectively. First of all, 2BC solves all instances to optimality (one run reaches the time limit), 2BC- is always successful, and 2BC+ cannot solve two instances (i.e., 10 runs to time limit). Overall, 2BC requires 29.03 hours to solve this instance set, 2BC- takes 28.76 hours, and 2BC+ runs 39.98 hours. The median running time for 2BC, 2BC-, and 2BC+ is 47.43 sec., 46.66 sec., and 52.60 sec., respectively, and the average running times are 271.40 sec., 269.00 sec., and 364.50 sec. The standard deviation is as follows: 962.40 (2BC), 953.49 (2BC-), and 1 265.82 (2BC+).

The main reason for the worse performance of 2BC+ is the increased numerical instability due to the coefficients (which are mostly c_e^*). First, this can be seen by comparing the number of b&b nodes: although the median number is identical for all approaches, 2BC and 2BC- generate on average 388.5 and 384.4 nodes, respectively, while 2BC+ requires 516.3—this is an increase by over 32%. Second, the number of tailing off effects is more than doubled from 3 485 (2BC) and 3 575 (2BC-) to 8 643 (2BC+).

On the other hand, in general 2BC and 2BC- perform very similar. The number of b&b nodes decreases slightly, but the number of tailing off effects increases. The overall speedup from 2BC to 2BC- is 1.007 (median) and 1.001 (average) with standard deviation 0.04.

Hence, (Ic^-) has no significant impact on the running time. Moreover, (Ic^-) seems to increase the numerical instability—this holds even more for the theoretically stronger cuts (Ic^+). We close the discussion by mentioning that—despite the increased numerical problems—the computed solution values are identical for all instances.

Disaggregated integer optimality cuts. Disaggregating the integer optimality cuts has a minor positive effect on the approach 2BC. The overall running time on the instance set N50IC is 28.83 hours (11.7 minutes less than 2BC). In particular, the median running time on this set is 47.08 sec. (0.35 sec. less), on average it is 269.6 sec. (1.8 sec. less), and the standard deviation is 959.93 (2.47 less). Moreover, the number of b&b nodes is 14 9475 (90 less). The only aspect where 2BC is slightly better is the number of tailing off effects;

the approach with disaggregated cuts generates 3 510 nodes (25 more).

Overall, using disaggregated or aggregated integer optimality cuts does not make a significant difference.

No-good cuts. No-good cuts (Ng) are introduced in Section 7.2.3 as constraints for cutting off the current first-stage solution. Since the coefficients of the cuts are all binary these cuts are numerically more stable than the standard integer optimality cuts. Moreover, they seem to be important for reducing numerical problems and ensuring the correct optimum solution (value).

We run NoNG:2BC on the same instances as before, i.e., for $n = 30$ instances with 5–1000 scenarios, for $n = 50$ with 5–250 scenarios, and for $n = 75$ with 5–100 scenarios. Overall, there are 80 instances where both NoNG:2BC and 2BC succeed but the computed optimum solution value of NoNG:2BC is smaller than the solution value computed by 2BC; and as already mentioned, whenever DA is successful the solution values of DA and 2BC are identical. For $n = 30$ there are 28 instances with different values, 34 instances for $n = 50$, and 18 instances for $n = 75$. For example, for the instances with 30 vertices where the solution differs, the maximal difference is 2.59, the median is 0.04 and the average difference is 0.59.

We did not evaluate all runs in detail, but the logfiles that we looked into show that the inserted integer optimality cuts are (numerically) not sufficient to cut off the current first-stage solution. Then, NoNG:2BC does not find violated integer optimality cuts and falsely assumes that the current first-stage solution $(\tilde{x}^0, \tilde{\theta})$ is correct.

Pareto optimal L-shaped cuts. A well-known and frequently used approach for strengthening L-shaped cuts is the method for finding Pareto optimal cuts by Magnanti and Wong [131]; we discuss the method in Section 7.2.1. In the following, we compare the performance of three decomposition approaches: 2BC and NoCS:2BC as described above, and MW:NoCS:2BC, which is NoCS:2BC with Pareto optimal L-shaped cuts added at each iteration, i.e., after solving the subproblem an additional LP is solved to obtain stronger dual multipliers corresponding to a Pareto optimal L-shaped cut. We compare the three approaches by considering the running times and the number of master iterations in the b&b root node.

Figure 10.7 reports the running times for graphs with 50 vertices—a similar behavior can be observed for the remaining instances. The plot is restricted to 800 sec. to focus on the important parts; however, some outlier points are cut off. As one can see, 2BC with our cut strengthening method is still the fastest approach. However, the method by Magnanti and Wong significantly improves the running time of the approach without cut strengthening (MW:NoCS:2BC versus NoCS:2BC). Overall, NoCS:2BC is not able to solve 11 instances and MW:NoCS:2BC cannot solve 8 instances. Moreover, NoCS:2BC takes 151.83 hours for all runs and MW:NoCS:2BC requires 123.47 hours. The speedup is around 2: the median is 1.99 and on average it is 2.06 with maximum 5.32 and standard deviation 0.65.

Comparing the number of master iterations in the root node, we report that NoCS:2BC requires the largest number of iterations, and that the reduction factor obtained by our cut strengthening is 2.35 (median) and 2.4 (average) with standard deviation 0.37. The approach MW:NoCS:2BC requires even less master iterations than 2BC; here, the reduction is 1.27 (median) and 1.3 (average) with standard deviation 0.54. However, there is a computational overhead associated with MW:NoCS:2BC, induced by solving additional LPs for finding

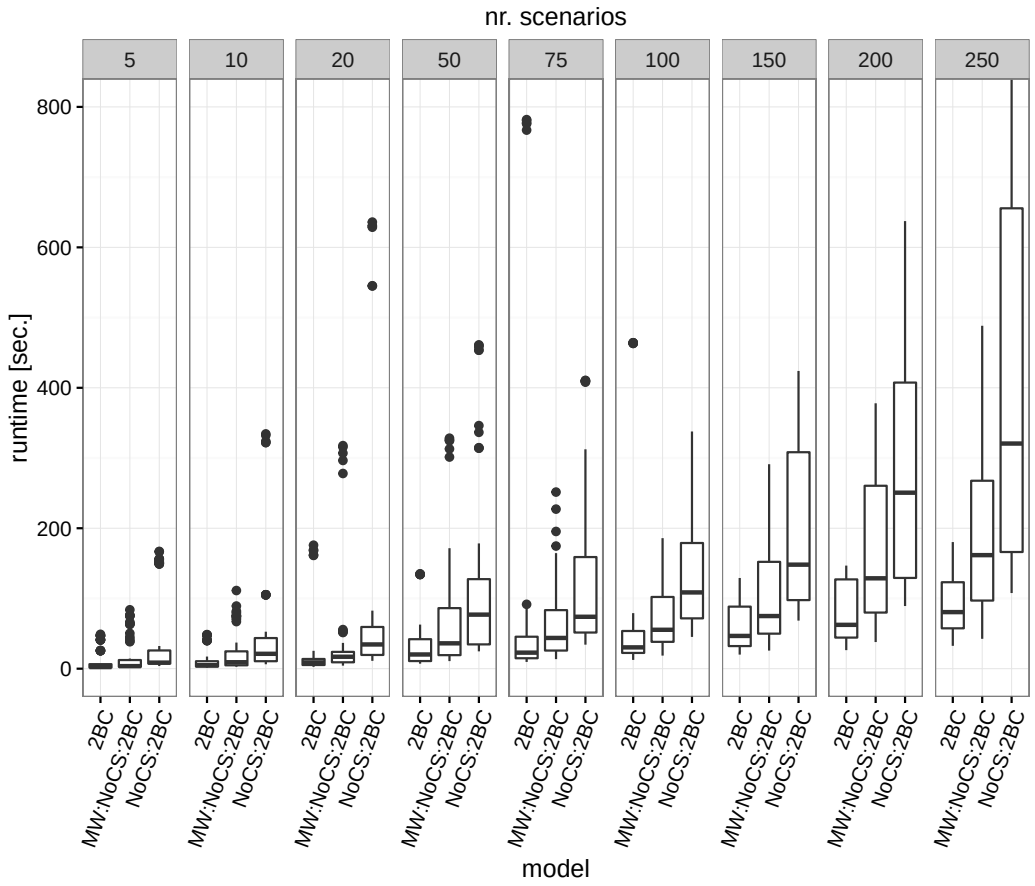


Figure 10.7: Comparison of the running times for the approach with cut strengthening (2BC), without cut strengthening (NoCS:2BC), and by additionally generating Pareto optimal cuts by the method of Magnanti and Wong (MW:NoCS:2BC) on the instances with 50 vertices and 5–250 scenarios. The plot is restricted to 800 sec. such that some outlier points are cut off.

Pareto optimal cuts. This results in the higher overall running time of MW:NoCS:2BC which is worse when compared to the running time of 2BC.

Finally, we also tried to hybridize the method by Magnanti and Wong with 2BC, but it turned out that this approach does not improve the running time. Surprisingly and interestingly, when using the method by Magnanti and Wong our cut strengthening does not have an influence on the performance such that MW:NoCS:2BC and this hybrid approach behave similarly.

Higher number of scenarios for graphs with $n = 50$. From the experiments with 30 vertices we see that our algorithm scales well with an increasing number of scenarios (average and median running time for 1 000 scenarios is less than 1 min.). Here, we consider the larger instances with 50 vertices and 300–1000 scenarios. Again, 2BC performs well: out of 100 instances the algorithm solves 90 always to optimality and is not able to solve 10 instances within the time limit of 2 hours. The overall median running time is 204.10 sec., on average 2BC takes 982.50 sec., with standard deviation 2 102.69. Detailed results about the running times, the number of b&b nodes, and the number of integer optimality

	runtime			nr. b&b nodes			nr. int. opt. cuts		
K	avg.	med.	std. dev.	avg.	med.	std. dev.	avg.	med.	std. dev.
300	822.25	108.59	2 137.89	48.04	5	145.88	26.28	3	75.33
400	860.00	136.56	2 126.05	30.52	7	79.05	17.38	4	41.40
500	957.70	175.72	2 108.05	29.04	3	75.10	16.78	4	38.16
750	1 038.70	351.58	2 078.80	20.36	6	43.32	11.38	4	20.25
1000	1 233.61	424.26	2 078.61	16.30	5	29.90	9.94	4	13.95

Table 10.6: Running time, number of b&b nodes, and number of integer optimality cuts (including no-good cuts) for 2BC on instances with 50 vertices and 300–1000 scenarios.

cuts grouped by the number of scenarios is given by Table 10.6.

Scenario generation. We tried to improve the two-stage b&c algorithm by using the idea of scenario generation (also known as scenario sampling or sample approximation), mentioned in Section 4.3, and column generation, see, e.g., Desrosiers and Lübbecke [59]. The idea is to start with a smaller subset of scenarios and solve this restricted instance to optimality. Afterwards, the remaining scenarios are added to the problem and the original instance is solved.

The resulting 2BC-SG algorithm proceeds as follows. We sort the scenarios by decreasing probability and take the first K' scenarios such that the sum of the probabilities exceeds a given threshold G, with $0 < G < 1$. Let the set of scenarios be denoted by $\mathcal{K}'$, i.e., $\sum_{k \in \mathcal{K}'} p^k \geq G$. Then, for each scenario $k \in \mathcal{K}'$ the scenario probability p^k is scaled appropriately to q^k such that $\sum_{k \in \mathcal{K}'} q^k = 1$. The resulting SSNDP instance is solved to optimality with the 2BC algorithm. Notice that the main difference is the reduced number of scenarios and the modified objective coefficient of each θ^k variable, $k \in \mathcal{K}'$, in the master problem.

Afterwards, the remaining scenarios are re-added and the probabilities are set to the original values. By using the generated set of L-shaped cuts and the generated directed cuts in the subproblems the original instance is solved as usual.

We implemented and tested two values for G: 2BC-SG-50 ($G = 50$) and 2BC-SG-80 ($G = 80$). Unfortunately, on the instances with 50 vertices and 5–250 scenarios the scenario generation method does not have a positive effect on the running time. Over all instances 2BC takes 102.86 hours, and 2BC-SG-50 and 2BC-SG-80 require 113.8 hours and 107.47 hours, respectively. Similarly, the average running time increases from 411.5 sec. to 455.2 sec. and 429.9 sec., respectively, and the same holds for the number of b&b nodes and the number of master iterations.

Graph density. To show the robustness of our decomposition method we evaluate its performance when the graph density $|E|/|V|$ is increased. For this set of experiments we consider the instances with 50 vertices and 50 scenarios and insert new edges to obtain instances with a higher density. Edge costs of the new edges are generated in the same way as before and edge connectivity requirements remain unchanged. Overall, for each of the 20 instances with $|V| = 50$ and $K = 50$ we generate graphs with density 3, 4, 5, 6, 8, 10, 12, and 14, respectively; for 50 vertices a density of 14 implies that the graph contains more than half of all possible edges.

Figure 10.8 shows the running time of 2BC and DA grouped by the graph density;

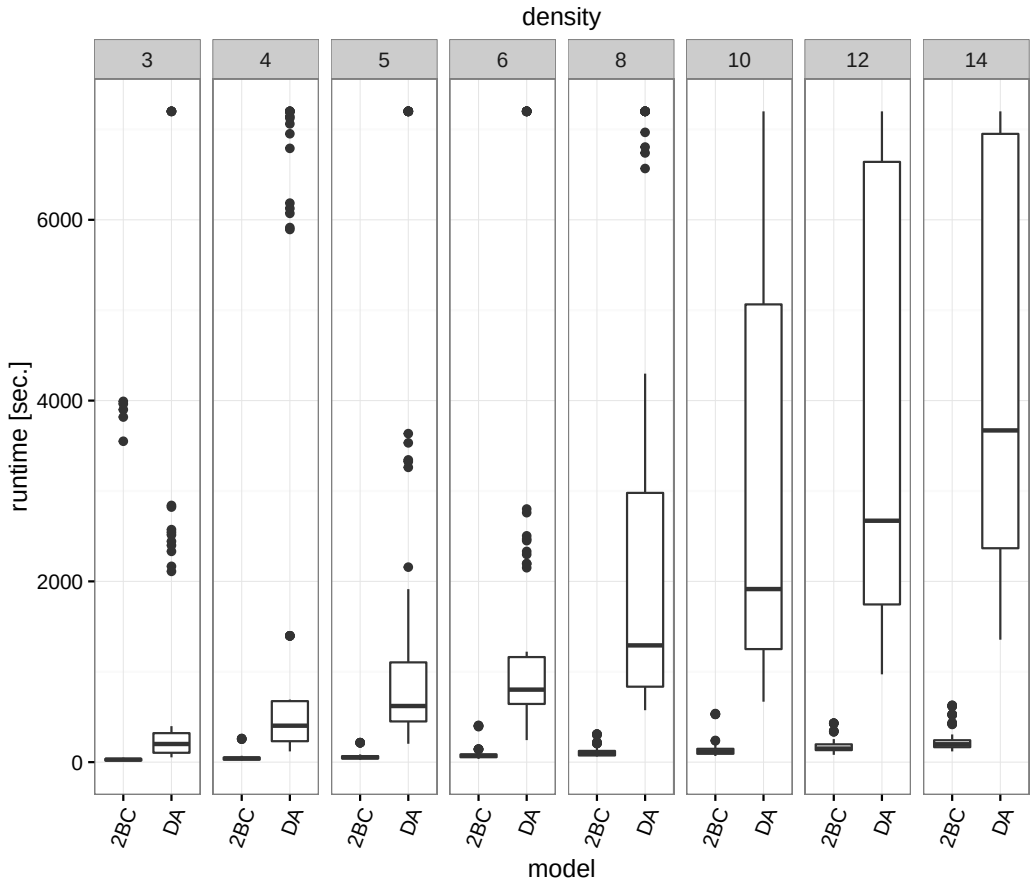


Figure 10.8: Running times of DA and 2BC when the density of the graphs increases. The used instances have 50 vertices and 50 scenarios.

Table 10.7 provides more details. As one can see the running time of 2BC increases only moderately; the median running time for density 3 is 28.09 sec. and for density 14 it is 197.86 sec. Since there are also only a few outlier points we conclude that the performance of 2BC is robust and remains stable even when the graph density increases. On the contrary, the direct approach DA performs much worse on denser graphs: the median running time increases from 200.86 sec. to 3 670.13 sec. (density 3 and 14, respectively). Moreover, DA cannot solve 25% of the densest instances within the time limit of two hours.

Second stage as penalty. We also evaluate the impact of the relative second-stage costs w.r.t. the first-stage costs. For this purpose, we take all instances with 50 vertices and simply multiply all second-stage costs with a penalty factor σ from the set $\{0.8, 2, 4, 8, 12\}$, i.e., an edge e in scenario k gets the new cost $\sigma \cdot c_e^k$. Figure 10.9 presents the results for the instances with 50 scenarios. Clearly, a more expensive second stage leads to the installation of more edges in the first stage as shown by the right plot of Figure 10.9. Moreover, our initial setting of second-stage costs seems to be reasonable as the optimum solutions consist of both first- and second-stage edges. With an increasing (or decreasing) factor this changes drastically and one stage dominates: for a factor of 0.8 or less the optimal first stage solution is always empty and for a factor of ≥ 8 this holds for the second stage. Moreover, we conclude that

$\frac{\lvert E\rvert}{\lvert V\rvert}$	avg. runtime		median runtime		std. dev. runtime		avg. nr. b&b nodes	
	DA	2BC	DA	2BC	DA	2BC	DA	2BC
3	761.88	**218.15**	200.86	**28.09**	1 640.64	**836.99**	113.24	**15.60**
4	1 352.72	**50.72**	403.27	**41.18**	2 304.47	**49.61**	107.00	**5.20**
5	1 456.56	**62.20**	621.09	**54.24**	2 052.08	**38.56**	71.60	**3.40**
6	1 543.94	**88.32**	802.38	**72.49**	1 976.20	**75.63**	50.78	**3.50**
8	2 322.52	**113.56**	1 291.70	**96.31**	2 211.72	**60.17**	34.30	**3.10**
10	3 001.70	**143.74**	1 914.96	**115.25**	2 377.57	**100.02**	22.20	**2.70**
12	3 607.12	**177.64**	2 670.86	**149.99**	2 348.28	**82.94**	17.60	**2.60**
14	4 167.50	**243.31**	3 670.12	**197.85**	2 112.16	**129.93**	12.92	**2.40**

Table 10.7: Comparison of DA and 2BC on graphs with increasing density $|E|/|V|$. The number of vertices and the number of scenarios are both fixed to 50.

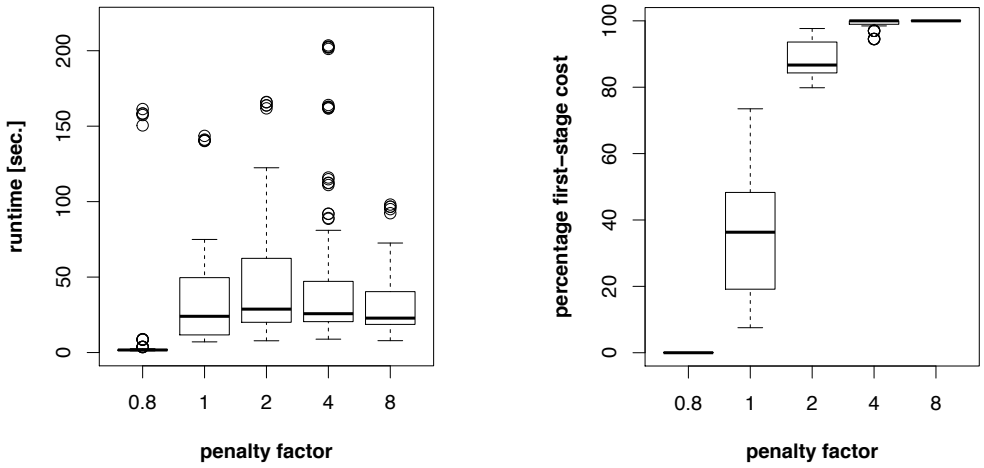


Figure 10.9: Boxplots showing the impact of a penalty factor for the second-stage cost on instances with 50 vertices and 50 scenarios: (*left*) running time of 2BC and (*right*) percentage of first-stage cost w.r.t. optimal solution value.

our algorithm performs well even when more edges are bought in the first stage, compare the left plot of Figure 10.9. This is important since the decomposed model initially contains no constraints in the master problem.

Conclusion. The computational study on the SSNDP agrees with the study on the SSTP: in general the two-stage branch&cut algorithm performs better than the direct approach. With a low number of scenarios the direct approach is faster but with an increasing scenario size the decomposition scales better and clearly outperforms the single b&c algorithm. Moreover, the experiments show that denser graphs or higher second-stage costs do not highly influence the performance.

For the decomposition the method for strengthening the L-shaped optimality cuts is very successful. On the other hand, the scenario generation approach or the new integer optimality cuts do not give the desired improvements.

Part IV

Epilogue

Chapter 11

Conclusion and outlook

Network design problems play an important role in various fields and many applications. In practice, exact knowledge of all data is often not given and uncertain data or probabilistic information prevent the application of classical deterministic algorithms. Several approaches have been developed to deal with the uncertainty; in this thesis we focus on the concept of stochastic programming.

Stochastic network design problems have been considered in the literature before, but only for developing approximation algorithms. The goals of this thesis are the development, implementation, and engineering of efficient algorithms for two-stage stochastic network design problems. For this purpose we concentrate on the two-stage stochastic Steiner tree problem—the general problem and a rooted version which requires a first-stage tree—and the two-stage stochastic survivable network design problem.

We first present linear-time algorithms based on dynamic programming for the stochastic Steiner tree problems. Afterwards, we develop the first fixed-parameter tractable algorithms. We introduce an FPT algorithm parameterized by the overall number of terminals for the rooted SSTP. Then, we consider treewidth-bounded graphs and expand an algorithm for the deterministic STP to algorithms for the stochastic Steiner tree problems. The resulting algorithms are FPT algorithms parameterized by the combination of treewidth and number of scenarios.

The main part of this thesis is dedicated to methods from mathematical programming. As a first step we develop several semi-directed cut- and flow-based models for the SSTP. These models formulate the problem by using undirected edges in the first stage and—by using orientation properties from the deterministic problem—directed arcs in the second stage. This approach leads to stronger IP models than the known undirected models from the literature. For the rooted version of the SSTP we additionally present fully directed models. The strength of the polyhedra is compared in a comprehensive study.

For solving the stochastic Steiner tree problems to optimality we apply a Benders' decomposition which results in the two-stage branch&cut algorithm. Thereby, each subproblem represents a scenario, which is a restricted Steiner tree problem. By studying the primal and dual subproblem we introduce two heuristics for improving the L-shaped optimality cuts. The first heuristic is a linear-time method which modifies the optimum dual solution by reducing the slack of the dual constraints. The second heuristic modifies the set of generated directed cuts in the primal subproblem, which results in a different set of variables in the dual problem. If applicable, both methods lead to stronger L-shaped optimality cuts.

Another important aspect of the two-stage b&c algorithm are integer optimality cuts. Due to the special structure of the stochastic problems we are able to develop new and stronger integer optimality cuts. Moreover, we additionally introduce cut-based constraints for the first-stage master problem, consider the improved integer optimality cuts, describe the disaggregation of all constraints, and mention methods for improving the lower bound for the second-stage cost.

To measure the performance of the two-stage b&c algorithm we evaluate a comprehensive computational study on expanded instances from the deterministic STP and PCSTP; to the best of our knowledge this is the first study reporting optimum solutions to the SSTP. The study shows that the decomposition is in general faster, more stable, and more robust than the direct approach (a single b&c) and that it scales better with an increasing number of scenarios. Moreover, the linear-time cut strengthening procedure is very efficient and highly reduces the running time and the number of master iterations. Since the generated instances are instances for the rooted SSTP as well, we evaluate the cost increase through an imposed first-stage tree, which is very low on our instances with at most 0.06%. On the other hand, the directed models for the rooted SSTP are much faster and solve many more instances than the semi-directed models for the SSTP. Last but not least, we show that the BuyNone heuristic (connect everything in the second stage) is a good compromise between running time and solution quality, but the BuyAll heuristic (connect everything in the first stage) mostly computes weak solutions.

In the second part of this thesis we consider the two-stage stochastic survivable network design problem. Motivated by the SSTP we develop semi-directed models which are compared with, and shown to be stronger than, the straight-forward undirected models. We decompose the strongest semi-directed model, demonstrate the applicability of the same cut strengthening method, and argue the correctness of the new integer optimality cuts. Similarly, a computational study shows the benefit of the decomposition and the cut strengthening, which significantly reduces the number of master iterations and the computational running time. The experiments further demonstrate the stable performance of the decomposition on larger instances (w.r.t. the number of scenarios) and on denser graphs. Moreover, we investigate further aspects like the effects of higher second-stage costs on the percentage of selected first-stage edges, the value of the stochastic solution, and the in-sample and out-of-sample stability.

In summary, the two-stage branch&cut algorithm is able to solve small to medium sized stochastic instances to optimality. In general, the decomposition performs better than the direct approach and in particular, it scales better with an increasing number of scenarios. The experiments clearly show that the running time of the two-stage b&c algorithm increases moderately with the number of scenarios while the direct approach is much more vulnerable to an increasing scenario set. On the other hand, the two-stage b&c algorithm performs weaker on instances where the corresponding deterministic instances are difficult; an extreme example is the artificial PUCN instance set for the (S)STP. The same behavior can be observed for the SSNDP: instances with 30 or 50 vertices can be solved for a high number of scenarios but for 75 vertices the algorithm clearly fails more often. A tremendous improvement to the two-stage b&c algorithm is the cut strengthening method which highly reduces the required running time.

The third considered problem of this thesis is the deterministic Steiner forest problem, which is related to all considered problems. On the one hand, the SFP is a generalization of the deterministic STP. On the other hand, we show that the SFP is a special case of the

stochastic Steiner tree problem: when each terminal set is interpreted as a scenario with very high edge costs, the SSTP connects all terminal sets in the first stage and directly solves the SFP. Hence, the results on the SSTP are transferable and we obtain the first FPT algorithm on treewidth-bounded graphs and new and stronger semi-directed IP models for the SFP. We additionally show how these models can be strengthened further to extended directed cut- and flow-based models and we present a polyhedral comparison of these models.

Following this conclusion, see page 205, we give a list of open problems mentioned throughout this thesis. Here, we like to discuss some further open problems and ideas for future research.

- *Applicability to further network design problems*. Many stochastic network design problems can be modeled by using capacity constraints in the second stage which enforce the selection of a second-stage edge whenever the related first-stage edge is chosen. In case the deterministic problem allows for a stronger directed model the second stage can be oriented as well which in turn leads to a stronger semi-directed model. We show the applicability for the stochastic Steiner tree problem and the stochastic survivable network design problem (we also mention the node-connectivity version), but the ideas are transferable to many two-stage stochastic network design problems.

- *Pareto optimal L-shaped cuts*. We were not able to find a (new) method (combinatorial or LP-based) for generating a Pareto optimal L-shaped cut. For the semi-directed models we know that the cut strengthening method does not give Pareto optimal cuts; for the directed models this is an open question. Since the two-stage b&c algorithm highly depends on the strength of the L-shaped optimality cuts the improvement of these cuts is an interesting research topic.

- *Preprocessing*. For the deterministic STP there exist successful preprocessing rules which reduce the instance size very effectively, see [55, 139]. For the SSTP we did not investigate preprocessing rules, but this is an interesting topic for future research.

- *Primal heuristics*. Our implementations contain only rudimentary primal heuristics which mainly consist of rounding the first stage and solving the second stage to optimality. This approach works well if the scenarios can be solved easily and if the primal heuristic is not called too often. For future research it might be interesting to develop more sophisticated primal heuristics, for example, primal heuristics that solve the second stage heuristically.

- *Numerical problems*. Solving a stochastic problem—in particular, this is known for Benders' decomposition—often causes numerical problems. Reducing the tailing off effects and numerical issues is a difficult task. In our implementation we found a setup that works well in general. But the experiments reveal some instances where numerical problems are dominating. This needs to be investigated further in the future.

- *Approximation algorithms*. The semi-directed and directed models are the strongest models known for the SSTP, rSSTP, and SFP. Perhaps it is possible to develop approximation algorithms with better approximation ratios by using these models.

- *Multistage stochastic problems, i.e., ℓ-stage stochastic network design problems*. In this thesis we consider stochastic problems with two decision stages. We are not aware of any publications considering the ℓ-stage stochastic Steiner tree or survivable network design problem, with $\ell > 2$.

- *Stronger formulation for the first stage*. The semi-directed models ($\text{SSTP}_{\text{sdc2}}$) and ($\text{SSNDP}_{\text{sdc2}}$) still use undirected edge variables in the first stage. Due to the discussion on the Steiner forest problem, where we present extended directed formulations for possibly disconnected solutions, i.e., models ($\text{SFP}_{\alpha:\text{dc}}$) and ($\text{SFP}_{\alpha:\text{df}}$), maybe it is possible to use similar ideas for strengthening the first-stage description.

List of open problems

Bibliography

[1] ABACUS – A Branch And CUt System. University of Cologne, Faculty of Mathematics and Natural Sciences, Chair of Computer Science (Prof. M. Jünger), version 3.0. URL `http://www.informatik.uni-koeln.de/abacus/`. 127

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., 1993. 36

[3] M. Albareda-Sambola, M. H. van der Vlerk, and E. Fernández. Exact solutions to a class of stochastic generalized assignment problems. *European Journal of Operational Research*, 173(2):465–487, 2006. 29

[4] E. Álvarez-Miranda, E. Fernández, and I. Ljubić. The recoverable robust facility location problem. *Transportation Research Part B: Methodological*, 79:93–120, 2015. 29, 106

[5] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980. 39

[6] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, 2007. 35

[7] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, D. G. Espinoza, M. Goycoolea, and K. Helsgaun. Certification of an optimal TSP tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15, 2009. 21

[8] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987. 18

[9] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. Connectivity-splitting models for survivable network design. *Networks*, 43(1):10–27, 2004. 170, 173

[10] E. Balas. Projection, lifting and extended formulation in integer and combinatorial optimization. *Annals of Operations Research*, 140(1):125–161, 2005. 21

[11] M. Bateni, M. Hajiaghayi, and D. Marx. Prize-collecting network design on planar graphs. *CoRR*, abs/1006.4339:1–27, 2010. 58

[12] M. Bateni, C Chekuri, A. Ene, M. Hajiaghayi, N. Korula, and D. Marx. Prize-collecting Steiner problems on planar graphs. In *Symposium on Discrete Algorithms*, pages 1028–1049. SIAM, 2011. 58

[13] M. Bateni, M. Hajiaghayi, and D. Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM*, 58(5): 21:1–21:37, 2011. 41, 57, 58, 101

[14] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009. 9

[15] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. 23

[16] D. Berend and T. Tassa. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30:185–205, 2010. 59

[17] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989. 37

[18] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. 18

[19] J. R. Birge and F. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392, 1988. 26

[20] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 2nd edition, 2011. 9, 22, 24, 25, 26, 27, 28, 29, 31, 117, 184, 185

[21] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: Fast subset convolution. In *Symposium on the Theory of Computing*, pages 67–74. ACM, 2007. 55

[22] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2): 1–22, 1993. 17

[23] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. 18

[24] H. L. Bodlaender. Treewidth: Structure and algorithms. In *International Colloquium on Structural Information and Communication Complexity*, volume 4474 of *LNCS*, pages 11–25. Springer, 2007. 17, 18

[25] F. Bökler, P. Mutzel, and B. Zey. The stochastic Steiner tree problem on partial k-trees. In *Mathematical and Engineering Methods in Computer Science*, pages 1–12. NOVPRESS Brno, 2012. 9, 10, 11, 56, 57

[26] Fritz Bökler. Algorithmen für das Stochastische Steinerbaumproblem auf Serien-Parallelen Graphen. Diploma thesis, TU Dortmund, 2012. 9, 10, 56

[27] I. M. Bomze, M. Chimani, M. Jünger, I. Ljubić, P. Mutzel, and B. Zey. Solving two-stage stochastic Steiner tree problems by two-stage branch-and-cut. In *International Symposium on Algorithms and Computation*, volume 6506 of *LNCS*, pages 427–439. Springer, 2010. 10, 11, 31, 51, 152, 164, 165

[28] G. Borradaile, C. Kenyon-Mathieu, and P. Klein. A polynomial-time approximation scheme for Steiner tree in planar graphs. In *Symposium on Discrete Algorithms*, pages 1285–1294. SIAM, 2007. 58

[29] G. Borradaile, P. Klein, and C. Mathieu. An O($n \log n$) approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5:31:1–31:31, 2009. 58

[30] Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, 25(1):13–26, 2013. 29

[31] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):1–33, 2013. 38

[32] C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999. 31

[33] C. C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(3):451–464, 1998. 31

[34] D. Catanzaro, M. Labbé, R. Pesenti, and J. J. Salazar González. Mathematical models to reconstruct phylogenetic trees under the minimum evolution criterion. *Networks*, 53(2):126–140, 2009. 8

[35] C. Chekuri, A. Ene, and N. Korula. Prize-collecting Steiner tree and forest in planar graphs. *CoRR*, abs/1006.4357:1–24, 2010. 58

[36] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997. 127

[37] M. Chimani and M. Woste. Contraction-based Steiner tree approximations in practice. In *International Symposium on Algorithms and Computation*, volume 7074 of *LNCS*, pages 40–49. Springer, 2011. 38

[38] M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Orientation-based models for {0, 1, 2}-survivable network design: theory and practice. *Mathematical Programming*, 124(1-2):413–439, 2010. 34, 177

[39] M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Obtaining optimal k-cardinality trees fast. *Journal of Experimental Algorithmics*, 14:2.5:1–2.5:23, 2010. 35

[40] M. Chimani, P. Mutzel, and B. Zey. Improved Steiner tree algorithms for bounded treewidth. In *International Workshop on Combinatorial Algorithms*, volume 7056 of *LNCS*, pages 374–386. Springer-Verlag, 2011. 10, 11

[41] M. Chimani, P. Mutzel, and B. Zey. Improved Steiner tree algorithms for bounded treewidth. *Journal of Discrete Algorithms*, 16:67–78, 2012. 10, 11, 66

[42] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 17. CRC Press, 2014. 127

[43] M. Chlebík and J. Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008. 38

[44] S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64:209–229, 1994. 38, 39, 40, 86

[45] S. Chopra and M. R. Rao. The Steiner tree problem II: Properties and classes of facets. *Mathematical Programming*, 64:231–246, 1994. 38

[46] S. Chopra and C.-Y. Tsai. Polyhedral approaches for the Steiner tree problem on graphs. In X. Z. Cheng and D.-Z. Du, editors, *Steiner Trees in Industry*, volume 11 of *Combinatorial Optimization*, pages 175–202. Kluwer Academic Publishers, 2001. 38

[47] J. C. N. Climaco and M. M. B. Pascoal. Multicriteria path and tree problems: Discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19(1-2):63–98, 2012. 35

[48] G. Codato and M. Fischetti. Combinatorial Benders' cuts. In *Integer Programming and Combinatorial Optimization*, volume 3064 of *LNCS*, pages 178–195. Springer, 2004. 114

[49] COIN-OR Foundation. COIN-OR Open Solver Interface, version 0.102. URL `https://projects.coin-or.org/Osi`. 127

[50] I. Contreras, J. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59(6):1477–1490, 2011. 29

[51] W. J. Cook. The Traveling Salesman Problem, September 2015. URL `http://www.math.uwaterloo.ca/tsp/`. 21, 35

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, third edition, 2009. 33

[53] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Foundations of Computer Science*, pages 150–159. IEEE, 2011. 58

[54] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. On interval-subgradient and no-good cuts. *Operations Research Letters*, 38(5):341–345, 2010. 114

[55] S. Daneshmand. *Algorithmic Approaches to the Steiner Problem in Networks*. PhD thesis, Universität Mannheim, 2003. 37, 39, 203

[56] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 50 (12):1764–1769, 1955. 9

[57] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954. 130

[58] C. Demetrescu, A. V. Goldberg, and D. S. Johnson. 9th DIMACS implementation challenge – shortest paths, January 2006. URL `http://www.dis.uniroma1.it/challenge9/`. 181

[59] J. Desrosiers and M. E. Lübbecke. A primer in column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 1–32. Springer, 2005. 195

[60] L. Di Puglia Pugliese and F. Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013. 35

[61] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2012. 13

[62] I. Dinur and S. Safra. On the hardness of approximating label-cover. *Information Processing Letters*, 89(5):247–254, 2004. 49

[63] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1: 195–207, 1972. 38, 53, 55, 205

[64] M. Fischetti, H. W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994. 130

[65] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders' cuts. *Mathematical Programming*, 124:175–182, 2010. 105

[66] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders decomposition for large-scale facility location. *Management Science*, 2017. 29

[67] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. 16

[68] The R Foundation. The R project for statistical computing, July 2017. URL `https://www.r-project.org`. 135

[69] B. Fuchs, W. Kern, D. Mölle, S. Richter, P. Rossmanith, and X. Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3): 493–500, 2007. 55

[70] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977. 37

[71] E. Gassner. The Steiner forest problem revisited. *Journal of Discrete Algorithms*, 8 (2):154–163, 2010. 41, 57

[72] M. X. Goemans. Arborescence polytopes for series-parallel graphs. *Discrete Applied Mathematics*, 51(3):277–289, 1994. 38, 101

[73] M. X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63(2):157–182, 1994. 38, 101, 130

[74] M. X. Goemans and Y.-S. Myung. A catalog of Steiner tree formulations. *Networks*, 23(1):19–28, 1993. 38, 40

[75] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 144–191. PWS Publishing Co., 1996. 35, 38, 41, 50

[76] M. Goerigk. *Algorithms and Concepts for Robust Optimization*. PhD thesis, Universität Göttingen, 2012. 9, 35

[77] S. Gollowitzer. *Mixed Integer Programming approaches to problems combining network design and facility location*. PhD thesis, Universität Wien, 2012. 35

[78] 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. 35

[79] L. Gouveia, M. Leitner, and I. Ljubić. The two-level diameter constrained spanning tree problem. *Mathematical Programming*, pages 1–30, 2014. 35

[80] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbook in Operations Research and Management Science*, pages 617–672. Elsevier, 1995. 36, 170

[81] M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43(6), 1995. 37

[82] J. Guo, R. Niedermeier, and O. Suchý. Parameterized complexity of arc-weighted directed Steiner problems. In *International Symposium on Algorithms and Computation*, volume 5878 of *LNCS*, pages 544–553. Springer, 2009. 41

[83] A. Gupta and A. Kumar. A constant-factor approximation for stochastic Steiner forest. In *Symposium on the Theory of Computing*, pages 659–668. ACM, 2009. 50

[84] A. Gupta and M. Pál. Stochastic Steiner trees without a root. In *Automata, Languages and Programming*, volume 3580 of *LNCS*, pages 1051–1063. Springer, 2005. 50

[85] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Symposium on the Theory of Computing*, pages 417–426. ACM, 2004. 50

[86] A. Gupta, M. Hajiaghayi, and A. Kumar. Stochastic Steiner tree with non-uniform inflation. In *APPROX/RANDOM*, volume 4627 of *LNCS*, pages 134–148. Springer, 2007. 50

[87] A. Gupta, R. Ravi, and A. Sinha. LP rounding approximation algorithms for stochastic network design. *Mathematics of Operations Research*, 32(2):345–364, 2007. 50, 75, 76

[88] A. Gupta, R. Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. In *Symposium on the Theory of Computing*, pages 685–694. ACM, 2009. 170

[89] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM Jorunal on Computing*, 40 (5):1361–1401, 2012. 50

[90] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Symposium on the Theory of Computing*, pages 585–594. ACM, 2003. 49

[91] M. Hauptmann and M. Karpinski. A compendium on Steiner tree problems, May 2016. URL `http://theory.cs.uni-bonn.de/info5/steinerkompendium/`. 35, 37

[92] G. Heilporn, J. Cordeau, and G. Laporte. An integer L-shaped algorithm for the dial-a-ride problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883–895, 2011. 29

[93] J. L. Higle. Stochastic programming: Optimization when uncertainty matters. In *Tutorials in Operations Research: Emerging Theory, Methods, and Applications*, pages 30–53. INFORMS, 2005. 31

[94] J. L. Higle and S. Sen. *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*. Kluwer Academic Publishers, 1996. 31

[95] P. Hokama, M. C. San Felice, E. C. Bracht, and F. L. Usberti. A heuristic approach for the stochastic Steiner tree problem. In *11th DIMACS Implementation Challenge*, 2014. 51, 152, 165, 166

[96] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Annals of Discrete Mathematics. Elsevier, 1992. 37, 38

[97] IBM Corporation. IBM ILOG CPLEX optimization suite, version 12.2. URL `http://www.ibm.com`. 127

[98] N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Symposium on Discrete Algorithms*, pages 691–700. SIAM, 2004. 50

[99] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. 41, 170

[100] D. S. Johnson, M. Minkoff, and S. Phillips. The prize-collecting Steiner tree problem: Theory and practice. In *Symposium on Discrete Algorithms*, pages 760–769. SIAM, 2000. 132, 181, 182

[101] D. S. Johnson, T. Koch, R. F. Werneck, and M. Zachariasen. DIMACS challenge–11th DIMACS implementation challenge in collaboration with ICERM: Steiner tree problems, July 2017. URL `http://dimacs11.zib.de`. 51, 132, 133

[102] M. Jünger and P. Mutzel. *Graph Drawing Software*. Mathematics + Visualizations. Springer, 2004. 13

[103] M. Jünger and S. Thienel. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software—Practice & Experience*, 30(11):1325–1352, 2000. 127

[104] P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley and Sons, first edition, 1994. 9

[105] M. Kandyba-Chimani. *Exact Algorithms for Network Design Problems using Graph Orientations*. PhD thesis, Technische Universität Dortmund, 2011. 21, 35, 41, 170

[106] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. 37

[107] M. Kaut and S. W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007. 185

[108] H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005. 41, 170

[109] A. J. King and S. W. Wallace. *Modeling with Stochastic Programming*. Springer Series in Operations Research and Financial Engineering, 2012. 185

[110] W. K. Klein Haneveld and M. H. van der Vlerk. Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, 85:39–57, 1999. 31

[111] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. 17

[112] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998. 39, 129, 130

[113] T. Koch, A. Martin, D. Rehfeldt, and S. Voß. SteinLib Testdata Library. Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), June 2017. URL `http://steinlib.zib.de/`. 132, 133

[114] E. Korach and N. Solel. Linear time algorithm for minimum weight Steiner tree in graphs with bounded treewidth. Technical Report 632, Israel Institute of Technology, 1990. 58

[115] D. Kurz. Parameterized algorithms for stochastic Steiner tree problems. Diploma thesis, TU Dortmund, 2012. 9, 52, 53, 55

[116] D. Kurz, P. Mutzel, and B. Zey. Parameterized algorithms for stochastic Steiner tree problems. In *Mathematical and Engineering Methods in Computer Science*, volume 7721 of *LNCS*, pages 143–154. Springer, 2013. 9, 11, 53, 55

[117] G. Laporte and F. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993. 28, 29, 115, 116

[118] G. Laporte, F. Louveaux, and L. van Hamme. Exact solution of a stochastic location problem by an integer l-shaped algorithm. *Transportation Science*, 28(2):95–103, 1994. 29

[119] G. Laporte, F. Louveaux, and L. van Hamme. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002. 29

[120] M. Leitner. *Solving Two Network Design Problems by Mixed Integer Programming and Hybrid Optimization Methods*. PhD thesis, Vienna University of Technology, 2010. 106

[121] M. Leitner, M. Ruthmair, and G. R. Raidl. Stabilizing branch-and-price for constrained tree problems. *Networks*, 61:150–170, 2013. 106

[122] M. Leitner, I. Ljubić, M. Luipersbeck, M. Prossegger, and M. Resch. New real-world instances for the Steiner tree problem in graphs. Technical report, Uni Wien, 2014. 132

[123] M. Leitner, I. Ljubić, M. Luiperbeck, M. Prosegger, and M. Resch. The Steiner tree problem in graphs. University of Vienna., September 2015. URL `http://homepage.univie.ac.at/ivana.ljubic/research/STP/`. 132

[124] I. Ljubić. Prize-collecting Steiner tree problem homepage. University of Vienna, September 2015. URL `homepage.univie.ac.at/ivana.ljubic/research/pcstp/`. 132

[125] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(2-3):427–449, 2006. 35, 92, 130, 132

[126] I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems. In *International Network Optimization Conference*, volume 41 of *Electronic Notes in Discrete Mathematics*, pages 245–252, 2013. 10, 11

[127] I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems: Theory and practice. *European Journal of Operational Research*, 256(2):333–348, 2017. 10

[128] F. Louveaux and M. H. van der Vlerk. Stochastic programming with simple integer recourse. *Mathematical Programming*, 61(1-3):301–325, 1993. 25

[129] F. Maggioni and S. W. Wallace. Analyzing the quality of the expected value solution in stochastic programming. *Annals of Operations Research*, 200:37–54, 2010. 184

[130] T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005. 34, 36, 37, 41, 42, 94, 101, 102, 170, 205

[131] T. L. Magnanti and R. T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981. 105, 109, 110, 154, 183, 193

[132] F. Margot, A. Prodon, and T. M. Liebling. Tree polytope on 2-trees. *Mathematical Programming*, 63(1-3):183–191, 1994. 38, 101

[133] F. Margot, D. R. Schmidt, and B. Zey. MIP formulations for the Steiner forest problem. Manuscript, in preparation, 2017. 10

[134] C. St. J. A. Nash-Williams. On orientations, connectivity, and odd vertex pairings in finite graphs. *Canadian Journal of Mathematics*, 12:555–567, 1960. 41

[135] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience. John Wiley & Sons, 1999. 18

[136] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 17

[137] OGDF – The Open Graph Drawing Framework. TU Dortmund, Osnabrück University, Monash University, University of Cologne, TU Ilmenau, version 2010.10. URL `http://www.ogdf.net/`. 127

[138] N. Papadakos. Practical enhancements to the Magnanti-Wong method. *Operations Research Letters*, 36(4):444–449, 2008. 105, 110

[139] T. Polzin. *Algorithms for the Steiner Problem in Network*. PhD thesis, Universität des Saarlandes, 2003. 37, 39, 203

[140] T. Polzin and S. Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1-3):241–261, 2001. 21, 39, 130

[141] T. Polzin and S. Daneshmand. Practical partitioning-based methods for the Steiner problem. In *International Workshop on Experimental and Efficient Algorithms*, volume 4007 of *LNCS*, pages 241–252. Springer, 2006. 58

[142] T. Polzin and S. Daneshmand. Approaches to the Steiner problem in networks. In *Algorithmics of Large and Complex Networks*, volume 5515 of *LNCS*, pages 81–103. Springer, 2009. 39, 84, 92

[143] András Prékopa. Probabilistic programming. In *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 267–351. Elsevier, 2003. 170

[144] Stochastic Programming Society. Stochastic programming homepage, July 2017. URL `http://stoprog.org/`. 31

[145] H. J. Prömel and A. Steger. *The Steiner Tree Problem. A Tour Through Graphs, Algorithms and Complexity*. Advanced Lectures in Mathematics. Springer Vieweg, 2002. 37, 38

[146] S. Raghavan. *Formulations and Algorithms for Network Design Problems with Connectivity Requirements*. PhD thesis, Massachusetts Institute of Technology, 1995. 101, 102, 170

[147] S. Raghavan. Low-connectivity network design on series-parallel graphs. *Networks*, 43(3):163–176, 2004. 41

[148] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3): 801–817, 2017. 29

[149] J. Rak and P. G. Sterbenz. Preface: Optimization issues in resilient network design and modeling. *Networks*, 66(4):251–252, 2015. 170

[150] R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006. 49, 170

[151] J. Riordan. *An Introduction to Combinatorial Analysis*. Wiley, 1980. 58

[152] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. 17

[153] G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *Journal on Discretete Mathematics*, 19(1):122–134, 2005. 38

[154] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115, 2005. 29

[155] A. Schrijver. *Theory of linear and integer programming*. Wiley-Interscience. John Wiley & Sons, 1998. 18

[156] R. Schultz. Stochastic programming with integer variables. *Mathematical Programming*, 97(1-2):285–309, 2003. 31

[157] R. Schultz, L. Stougie, and M. H. van der Vlerk. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50(3):404–416, 1996. 31

[158] S. Sen and J. L. Higle. The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104 (1):1–20, 2005. 31

[159] A. Shapiro. Monte carlo sampling methods. In *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 353–425. Elsevier, 2003. 31

[160] A. Shapiro and A. Philpott. A tutorial on stochastic programming, 2007. URL `http://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf`. 31

[161] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. MOS-SIAM Series on Optimization 9. SIAM, 2009. 9, 22

[162] H. D. Sherali and B. J. Lunday. On generating maximal non dominated Benders cuts. *Annals of Operations Research*, 210:57–72, 2011. 105, 110

[163] D. B. Shmoys and C. Swamy. Algorithms column: Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 37(1):33–46, 2006. 50

[164] Y. Song and J. R. Luedtke. Branch-and-cut approaches for chance-constrained formulations of reliable network design problems. *Mathematical Programming Computation*, 5(4):397–432, 2013. 170

[165] Y. Song and M. Zhang. Chance-constrained multi-terminal network design problems. *Naval Research Logistics*, 62(4):321–334, 2015. 170

[166] M. Stoer. *Design of survivable networks*, volume 1531 of *Lecture Notes in Mathematics*. Springer, 1992. 172, 173

[167] I. G. Tollis, G. Di Battista, P. Eades, and R. Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998. 13

[168] R. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17(4): 638–663, 1967. 24, 26

[169] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001. 38, 48, 49

[170] J. A. Wald and C. J. Colbourn. Steiner trees, partial 2-trees and minimum IFI networks. *Networks*, 13:159–167, 1983. 38, 56, 101

[171] P. Wentges. Accelerating Benders' decomposition for the capacitated facility location problem. *Mathematical Methods of Operations Research*, 44:267–290, 1996. 105

[172] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995. 35

[173] P. Winter. Generalized Steiner problem in series-parallel networks. *Journal of Algorithms*, 7(4):549–566, 1986. 41

[174] P. Winter. Steiner problem in networks: a survey. *Networks*, 17(2):129–167, 1987. 38

[175] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984. 39

[176] B. Zey. ILP formulations for the two-stage stochastic Steiner tree problem. *CoRR*, abs/1611.04324:1–22, 2016. 10, 11

[177] B. Zey. SSNDPLib – The stochastic survivable network design problem. TU Dortmund, July 2017. URL `http://ls11-www.cs.tu-dortmund.de/staff/zey/ssndp`. 183

[178] B. Zey. SSTPLib – The stochastic Steiner tree problem. TU Dortmund, July 2017. URL `http://ls11-www.cs.tu-dortmund.de/staff/zey/sstp`. 133, 157