University of Kaiserslautern
Department of Mathematics

Diploma Thesis

Weight-Constrained Minimum Spanning Tree Problem

by Sebastian Tobias Henn

May 2007

Supervisors:
Prof. Dr. Horst W. Hamacher
Dipl.-Math. Stefan Ruzika

Abstract

In an undirected graph G we associate costs and weights to each edge. The weight-constrained minimum spanning tree problem is to find a spanning tree of total edge weight at most a given value W and minimum total costs under this restriction. In this thesis a literature overview on this NP-hard problem, theoretical properties concerning the convex hull and the Lagrangian relaxation are given. We present also some in- and exclusion-test for this problem. We apply a ranking algorithm and the method of approximation through decomposition to our problem and design also a new branch and bound scheme. The numerical results show that this new solution approach performs better than the existing algorithms.

Acknowledgements

My special thanks go to Professor Horst W. Hamacher who made this thesis possible. I would like to express my gratitude to Stefan Ruzika, his assistance, criticism and advice was most helpful. Thanks are also due to Nico Behrent and Hans Schönemann, for their support on C++ and Singular. Thanks to my colleague Michael Busch for proofreading this thesis. Finally, I would like to thank my entire family, who supported me during my whole studies.

Contents

1 Introduction

1.1 Problem

In this diploma thesis we focus on a problem from graph theory. Therefore we consider an undirected graph $G = (V, E)$ with a set of vertices $V = \{1, 2, \ldots, n\}$ and a set of edges E with cardinality m, where we associate to each edge $e \in E$ costs c_e and a weight w_e. A very popular concept of this mathematical field are the spanning trees of a graph.

Definition 1.1
A spanning tree T is a connected subgraph of G which contains no cycle and all vertices of G.

The problem is to find a spanning tree with minimal costs under the constraint that the weight of the tree is not greater than a given constant W. We can state our problem in the following way.

Problem 1 *Weight-Constrained Minimal Spanning Tree Problem*

$$OPT := \min \sum_{e \in T} c_e \tag{1.1}$$

$$s.t. \sum_{e \in T} w_e \leq W \tag{1.2}$$

$$T \in \mathcal{T} \tag{1.3}$$

where $\mathcal{T}$ is the set of all spanning trees in G.

For simplification we denote the costs of a tree T as $c(T) = \sum_{e \in T} c_e$ and the weight of a tree T as $w(T) = \sum_{e \in T} w_e$. We call this problem *weight-constrained minimal spanning tree problem (WCMST)* and in the multidimensional case with a vector of weights on each edge *resource-constrained minimal spanning tree problem (RCMST)* where for each edge L resources are given and for each of them a constraint has to be satisfied.

Problem 2 *Resource-Constrained Minimal Spanning Tree Problem*

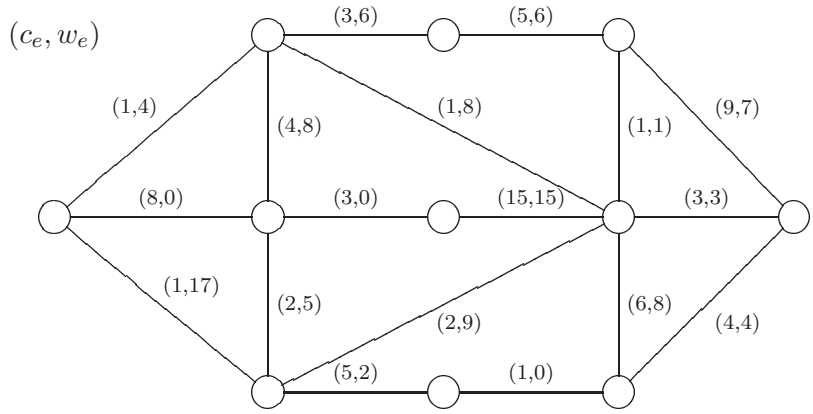
$$\min \sum_{e \in T} c_e \tag{1.4}$$

$$s.t. \sum_{e \in T} w_e^l \leq W_l \text{ for } 1 \leq l \leq L \tag{1.5}$$

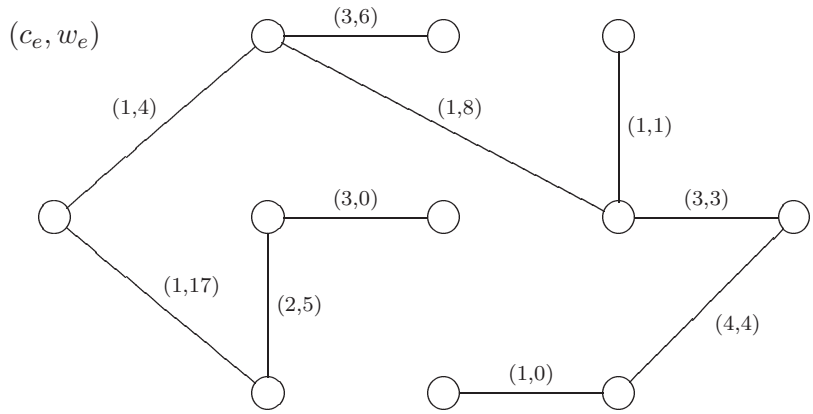
$$T \in \mathcal{T} \tag{1.6}$$

Example 1.1

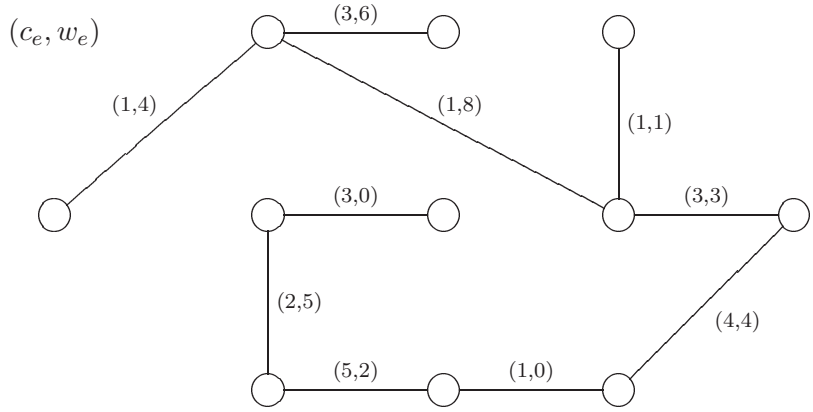
Let the following graph G with $n = 11$ and $m = 18$ be given:



A minimal spanning tree ignoring the weight constraint is:



This tree has the costs $c(T) = 20$ and the weight $w(T) = 48$. If we search for a minimal spanning tree with weight less or equal than 33, this tree is not feasible. An optimal solution with $c(T) = 24$ and $w(T) = 33$ is the following tree.



1.2 Literature Overview

In this thesis our nomenclature is related to the paper of Xue [38] and to the publication of Dumitrescu and Boland [12], which discusses the problem of finding a shortest path instead of a spanning tree. In the literature very different denotations for our problem can be found: Minimal spanning tree subject to a side constraint, minimal spanning tree subject to a budget constrained, knapsack constrained minimum spanning tree problem and constrained minimum spanning tree problem. The name 'constrained minimum spanning tree' used in two on this problem most important papers, the article of Goemans and Ravi [18] respectively the publication of Hong, Chung and Park [27], seems not to be precise enough since a large number of different problems of finding a minimal spanning tree with some constraint exists. Deo and Kumar study in [11] 29 constrained spanning tree problems. Some examples:

- *Degree-Constrained Minimal Spanning Tree Problem:*
  The goal is to find a minimal spanning tree under the condition that the degree of each vertex i (the number of edges connecting the node i) is bounded by B i.e., $\deg(i) \leq B$ for all $i \in V$.

- *Diameter-Constrained Minimal Spanning Tree Problem*
  The aim is to find a minimal spanning tree such that the largest unique path between two nodes on the tree contains at least a given number of edges. If weights for each edge are given, the problem can be extended to the problem of finding a minimal spanning tree under the condition that the weight of every path between two nodes is smaller than a given value.

- *Hop-Constrained Minimal Spanning Tree Problem*
  This is a special case of the diameter-constrained minimal spanning tree problem: We search for a minimal spanning tree such that the number of edges in the shortest path along the tree from a fixed node 0 to all other nodes is smaller than a value B.

- *Capacitated Minimal Spanning Tree Problem:*
  For every node $i \in V$ a weight or a demand d_i is given. Additionally, we have a value C. For a set $S \in V$ we define $d(S) = \sum_{i \in S} d_i$ and $\delta^-(S) = \{(i,j) \in E \; i \in V \setminus S, j \in S\}$. The problem is to find a minimal spanning tree T such that $|T \cap \delta^-(S)| \geq \lfloor \frac{d(S)}{C} \rfloor$ for all $S \subset V$.

- *Weight-Constrained Minimal Spanning Tree With Flow Requirements*:
  Additionally to Problem 1 we identify one single vertex as source and associate a demand of infinity. All other nodes $i \in V$ are interpreted as sinks and have a demand d_i. The goal is to find a minimal spanning tree which satisfies the flow requirements and the weight-constraint.

By definition of some problems the tree has to be directed. The description and some remarks to the directed version of the WCMST-problem, the so called *weight-constrained minimum arborescence problem* (WCMA), are given in Chapter 3.

Only a few considerations have been published to our problem: The WCMST was first mentioned in the paper of Aggarwal, Aneja and Nair in 1982 [1]. In the literature two main approaches are used to solve the problem exactly. On the one hand a Lagrangian relaxation to approximate a solution combined with a branch and bound strategy ([1], [37]) and on the other hand a more innovating approach by using the matrix tree theorem [27]. The most important publications to the problem are the paper of Aggarwal, Aneja and Nair and the paper of Shogan [37] which give some exact algorithms, the article of Goemans and Ravi [18] which contains an approximation scheme and the paper of Hong, Chung and Park [27] which proposed an exact algorithm by using the matrix tree theorem. The publication of Xue [38] contains only an algorithm for computing the Lagrangian relaxation, and in [23] Hassin and Levin improve the results from [18]. In [39] the authors Yamada and Wantanabe consider a maximum spanning tree problem subject to a weight constraint. We can easily apply this to the minimization case. A few sources refer to the publication of Morozov [33]. This text is written in Russian and was not found in non-Russian literature. Unfortunately, from the sources it can not be concluded what the content of this paper is like. In [4] is only mentioned that a polynomial approximation algorithm is published. A further article in Chinese published and not available in English is the paper of Li and Yao [28]. In the abstract it can be read that the NP-completeness is proven, with a dual algorithm of generalized linear programming the upper bound was estimated and the character of an optimal solution was analyzed.

1.3 Chapter Outline

In Chapter 2 we collect some basic definitions from graph theory and focus on the minimal cost spanning tree problem. These results become relevant for other parts of this thesis. Chapter 3 introduces different formulations for our problem. Chapter 4 shows the relation between spanning trees and matroids which allows us to state our problem in a more general way. In Chapter 5 we classify the complexity of our problem. The consideration of a bicriterial optimization problem allows us in Chapter 6 to state some conclusion for the convex hull of our problem which will be connect to the Lagrangian relaxation in Chapter 7. Chapter 8 reviews an alternative relaxation approach. In Chapter 9 we develop some possibilities to reduce the complexity of a problem by in- and exclusion methods. Chapter 10 focus on a relation between the costs and the weight of an edge. Chapter 11 gives an overview over all existing exact solution methods and a new branch and bound scheme. All known approximation algorithms for the WCMST are described in Chapter 12. The numerical results are presented in Chapter 13 and finally we summarize the results of this thesis and give an outlook for further research in Chapter 14.

1.4 Application

To motivate our considerations we present two applications:

Example 1.2 *Designing Physical Systems*
In Ahuja, Magnanti and Orlin [2] some possible applications for the simple minimal spanning tree problem are given. They list some problems of physical systems where a spanning tree structure is needed:

- *Pipeline Construction*
  A pipeline network connecting a number of towns should be constructed: We search for the smallest possible total length of the pipeline.
- *Linking Isolated Villages*
  There are several villages given which are connected by roads but not by telephone lines. The aim is to determine along which stretches of road the telephone lines should be placed such that the total of road length is minimized.

In both cases we have to solve a minimal spanning tree problem. For the construction costs we can remark that the pipeline / telephone line with smallest total length need not be automatically the cheapest way of connecting the towns (e.g. a mountain must be passed by a tunnel etc.). If the construction costs should not exceed a given value we have to add a budget constraint. This is a WCMST where the objective function is to minimize the total length and the constraint that the costs are smaller then the budget limit.

The second application is sketched in [38]:

Example 1.3 *Minimum Cost Reliability Constrained Spanning Tree*
An important application of our problem is the minimum cost reliability constrained spanning tree problem in communication networks: We have n stations in the plane which can communicate with each other. The goal is to find a minimum cost connection (for instance the costs are modelled by distances d_e for $e = \{i, j\}$ between the stations i and j) under the restriction that the reliability of a connection (as spanning tree) described by a probability p_e for each pair of stations i, j is greater than a limit $P \in [0, 1]$.

$$\begin{aligned} &\min \sum_{e \in T} d_e \\ s.t. &\prod_{e \in T} p_e \quad \geq \quad P \\ &T \in \mathcal{T} \end{aligned}$$

We can reformulate the constraint

$$\prod_{e \in T} p_e \geq P \Leftrightarrow \log \prod_{e \in T} p_e \geq \log P \Leftrightarrow \sum_{e \in T} \log p_e \geq \log P.$$

This is by multiplying with (-1) *equivalent to*

$$\sum_{e \in T} -\log p_e \leq -\log P.$$

So we can interpret the minimum cost reliability constrained spanning tree as weight-constrained minimal spanning tree problem.

2 Spanning Trees

The minimal spanning tree problem is one of the most popular problems in graph theory and therefore a lot of literature concerning fast algorithms can be found. In this chapter we collect some basic definitions from graph theory, properties concerning spanning trees and some facts for the minimal spanning tree problem which might become useful for our problem. Also we use the two optimality conditions for the minimal spanning tree problem to obtain optimality conditions for the weight-constrained minimal spanning tree problem.

2.1 Definitions and Properties

The denotations and results in this section are taken from Ahuja, Magnanti and Orlin [2]. In the first chapter we gave the definition of the a spanning tree. If only the denotation tree is used we talk about a connected subgraph that contains no cycle. The whole set of vertices need not necessarily be contained in the tree. For simplification we use the following notations: For an edge $e \in E$ which connects the vertices i and j we can write $e = \{i, j\}$. (In the directed case - e goes from i to j - the denotation $e = (i, j)$ is used. We call i the *source* of e and j the *target* of e.) To characterize a spanning tree we define firstly a path.

Definition 2.1 *Path*
A path P is a sequence of vertices $(i_1, \ldots, i_k, i_{k+1}, \ldots, i_K)$ with $e_{i_k i_{k+1}} = \{i_k, i_{k+1}\} \in E$ for all $k \in \{1, \ldots, k\}$ and without any repetition in the set of vertices.

For a spanning tree we can state some properties.

Property 2.1
Let T be a spanning tree of G. Then we have the following properties:

1. *T has at least two leaves, where a leaf is a vertex which is endpoint of exactly one edge.*
2. *T contains $n - 1$ edges.*
3. *Every pair of vertices in T is connected by exactly one path.*

A very important ingredient for several algorithms in the next chapters is the concept of exchanging some edges for a spanning tree.

Definition 2.2 *T-exchange*
A T-exchange is a pair of edges $[e, f]$ such that for $e \in T$ and $f \notin T$ the set $T \setminus \{e\} \cup \{f\}$ is a spanning tree. We define the costs of a $T-$exchange as $c[e, f] := c_f - c_e$.

Now we define fundamental cycles and fundamental cuts which become important in the next section.

Definition 2.3 *Cycle*
A cycle is a path $i_1, i_2, \ldots, i_k$ with the edges $\{i_1, i_2\}, \{i_2, i_3\}, \ldots, \{i_{k-1}, i_k\}$ together with the edge $\{i_k, i_1\}$.

Property 2.2 *Fundamental Cycle*
Let T be a spanning tree of G. For every T-exchange $[e, f]$ we obtain exactly one cycle. This cycle is called fundamental cycle.

For an edge $f \notin T$ let $C(T, f)$ denote this cycle in $T \cup \{f\}$. A spanning tree T has $m - n + 1$ fundamental cycles since $|E \setminus T| = m - n + 1$. If we delete in every fundamental cycle an arbitrary edge, we obtain again a spanning tree (i.e. if $f \in E \setminus T$ and if $C(T, f)$ is the uniquely defined cycle in $T \cup \{e\}$, then $T \cup \{f\} \setminus \{e\}$ for all $e \in C(T, f)$ is a spanning tree).

Definition 2.4 *Cut*
A cut is a partition of V in a set $X \subset V$ and a set $V \setminus X$. Every cut defines a set $\{X, V \setminus X\} \subset E$ with the edges in E which have one node in X and the other node in $V \setminus X$.

Property 2.3 *Fundamental Cuts*
Let T be a spanning tree of G. If we delete an arbitrary edge e of T, we get some disconnected trees T_1 and T_2. The edges which have one node in T_1 and one node in T_2 constitute a cut, called fundamental cut, of G with respect to T. Let $\{X_e, V \setminus X_e\}$ denote this set of edges.

A graph has $n - 1$ fundamental cuts with respect to any tree since a spanning tree T contains $n - 1$ edges. If we add an edge of a fundamental cut to the two subtrees T_1 and T_2, we obtain again a spanning tree (i.e. if $e \in T$ and if $Q = \{X_e, V \setminus X_e\}$ is the uniquely defined cut in G. Then $T \setminus \{e\} \cup \{f\}$ is a spanning tree for all $f \in Q$.)
For computational uses we define at last the adjacency list which allows us to store the data structure in a simple way:

Definition 2.5 *Adjacency List*
The edge adjacency list $A(i)$ of a vertex $i \in V$ is the set of edges emanating from this vertex i.e., $A(i) = \{\{i, j\} \in E | j \in V\}$. The node adjacency list $A(i)$ is the set of vertices adjacent to i i.e., $A(i) = \{j \in V | \{i, j\} \in E\}$.

Property 2.4
For an adjacency list in an undirected graph we have

$$\sum_{i \in V} |A(i)| = 2m.$$

2.2 Minimal Spanning Tree Problem

Now we give some optimality conditions for the minimal spanning tree problem which can be used to formulate two necessary optimality conditions for our WCMST. Firstly, we state the formulation of the already mentioned minimal spanning tree problem.

Problem 3 (MST) *Minimal Spanning Tree Problem*

$$\begin{aligned} &\min \sum_{e \in T} c_e \\ &s.t.\ T \in \mathcal{T} \end{aligned}$$

where $\mathcal{T}$ is the set of all spanning trees of the graph.

For the minimal spanning tree problem two optimality conditions can be formulated.

Theorem 2.5 *Cut Optimality Condition [2]*
A spanning tree $T^\star$ is a minimal spanning tree if and only if for every edge $e \in T^\star$, $c_e \leq c_f$ for every $f \in \{X_e, V \setminus X_e\}$.

Proof

- Assume there exists a minimal spanning tree T that does not satisfy this condition. Then we have an edge $e \in T$ and an edge $f \in \{X_e, V \setminus X_e\}$ with $c_f < c_e$. Now we can delete e from the tree and add f to it. So we obtain again a spanning tree with $c(T \setminus \{e\} \cup \{f\}) = c(T) - c_e + c_f < c(T)$. This is a contradiction to the optimality of T.

- For the other part of the proof we have to show that a tree $T^\star$ satisfying the cut optimality condition is optimal. Assume T' is a minimal spanning tree and $T' \neq T^\star$. At least one edge $e \in T^\star$ exists with $e \notin T'$. If we delete this edge we have a cut $X_e, V \setminus X_e$. Now we add e to the tree T'. This $T' \cup \{e\}$ must contain a cycle $C(T', e)$ with an edge $f \neq e$ where f has one node in X_e and the other in $V \setminus X_e$. Since $T^\star$ satisfies the optimality condition we have $c_e \leq c_f$. On the other hand T' is optimal and $c_f \leq c_e$ must hold, therefore $c_e = c_f$. Then
$$c(T^\star) = c(T^\star) - c_e + c_f = c(T^\star \setminus \{e\} \cup \{f\}).$$
The tree $T^\star \setminus \{e\} \cup \{f\}$ has one edge more in common with T' and satisfies also the cut optimality. So we repeat this step until we obtain T'. In total we have $c(T^\star) = c(T')$ and therefore $T^\star$ is also a minimal spanning tree.

□

The theorem leads directly to the following corollary.

Corollary 2.6 *[31]*
A spanning tree T has minimal costs if and only if no T-exchange has negative costs.

A further optimality condition follows from the property that for each pair of vertices there exists a path between both vertices in a spanning tree.

Theorem 2.7 *Path Optimality Condition [2]*
A spanning tree $T^\star$ is a minimal spanning tree if and only if for every nontree edge $f = \{k, l\}$ holds that $c_e \leq c_f$ for every edge e contained in the path in $T^\star$ connecting k and l.

Proof

- Let $T^\star$ a minimal spanning tree and the edge $e = \{i, j\}$ is contained in the path in $T^\star$ between the vertices k and l. If for the nontree edge $f = \{k, l\}$ holds that $c_e > c_f$, we could execute a $T^\star-$exchange $[e, f]$. The new spanning tree has smaller costs than $T^\star$. This contradicts the optimality of $T^\star$ and the path optimality conditions hold.
- Let $e = \{i, j\} \in T^\star$ and let X_e and $V \setminus X_e$ be the set of vertices obtained by deleting e from $T^\star$. Consider an edge $\{k, l\}$ with $k \in X_e$ and $l \in V \setminus X_e$. Since $T^\star$ contains an unique path from k to l and e is the only edge connecting the sets X_e and $V \setminus X_e$, e is element of the path connecting the vertices k and l. From the path optimality condition we know that $c_e \leq c_f$ where $f = \{k, l\}$. This condition must hold for every nontree edge $\{k, l\}$ in the cut $\{X_e, V \setminus X_e\}$ for all $e \in T^\star$.
  So T satisfies the cut optimality and from Theorem 2.5 it is a minimal spanning tree.

□

2.3 Optimality Conditions for the WCMST

The optimality conditions of the minimal spanning tree problem can be used to formulate necessary conditions for an optimal solution of the weight-constrained minimal spanning tree problem. Unfortunately, these two conditions are not very useful for practical purpose.

Theorem 2.8 *Cut Optimality Condition*
A spanning tree $T^\star$ is an optimal solution for the weight-constrained minimal spanning tree problem if for every edge $e \in T^\star$ and for every $f \in \{X_e, V \setminus X_e\}$ with $c_f < c_e$ the weight satisfies $w(T) - w_e + w_f > W$.

Proof
Suppose the theorem does not hold. Then we can make a $T^\star$-exchange $[e, f]$ and obtain a tree with

$$c(T^\star \setminus \{e\} \cup \{f\}) = c(T^\star) - c_e + c_f < c(T^\star)$$

and

$$w(T^\star \setminus \{e\} \cup \{f\}) = w(T^\star) - w_e + w_f \leq W.$$

This is a contradiction to the optimality of $T^\star$.

□

Theorem 2.9 *Path Optimality Conditions*
A spanning tree $T^\star$ is optimal for the weight-constrained minimal spanning tree problem if for every nontree edge $f = \{k, l\}$ with $c_f < c_e$ for every edge e contained in the path in $T^\star$ connecting k and l, it must hold that $w(T^\star) - w_e + w_f > W$.

Proof
Suppose the theorem does not hold. Then we can replace e by f in $T^\star$ and we get a tree with

$$c(T^\star \setminus \{e\} \cup \{f\}) = c(T^\star) - c_e + c_f < c(T^\star)$$

and

$$w(T^\star \setminus \{e\} \cup \{f\}) = w(T^\star) - w_e + w_f \leq W.$$

This is a contradiction to the optimality of $T^\star$.

□

Corollary 2.10
Let $T^\star$ be an optimal solution for the WCMST. For every $T^\star$-exchanges $[e, f]$ with $c[e, f] < 0$ it must hold that $w(T^\star) - w_e + w_f > W$.

Proof
The theorem is valid since otherwise the tree obtained by the $T^\star$-exchanges which improves the total costs is feasible. This is a contradiction to the optimality of $T^\star$.

□

2.4 Algorithms for the Minimal Spanning Tree Problem

To solve the WCMST in many approaches a 'simple' minimal spanning tree problem without constraint has to be solved. Therefore we give a short remark to MST-algorithms: In the history there are several approaches to solve the minimal spanning tree problem. The first work was published in 1926 by Boruvka [5], [19]. Chazelle [9] claimed that the minimal spanning tree problem is one of the oldest problems in computer science. The two most popular algorithms are the algorithms of Prim and Kruskal, which will be presented next. Both algorithms are 'greedy' algorithms. They add in each step an edge with minimal costs from a candidate list until a spanning tree is found.

Prims Algorithm

Algorithm 2.1 MST - Algorithm of Prim

Require: Graph $G = (V, E)$ costs c_e for all $e \in E$

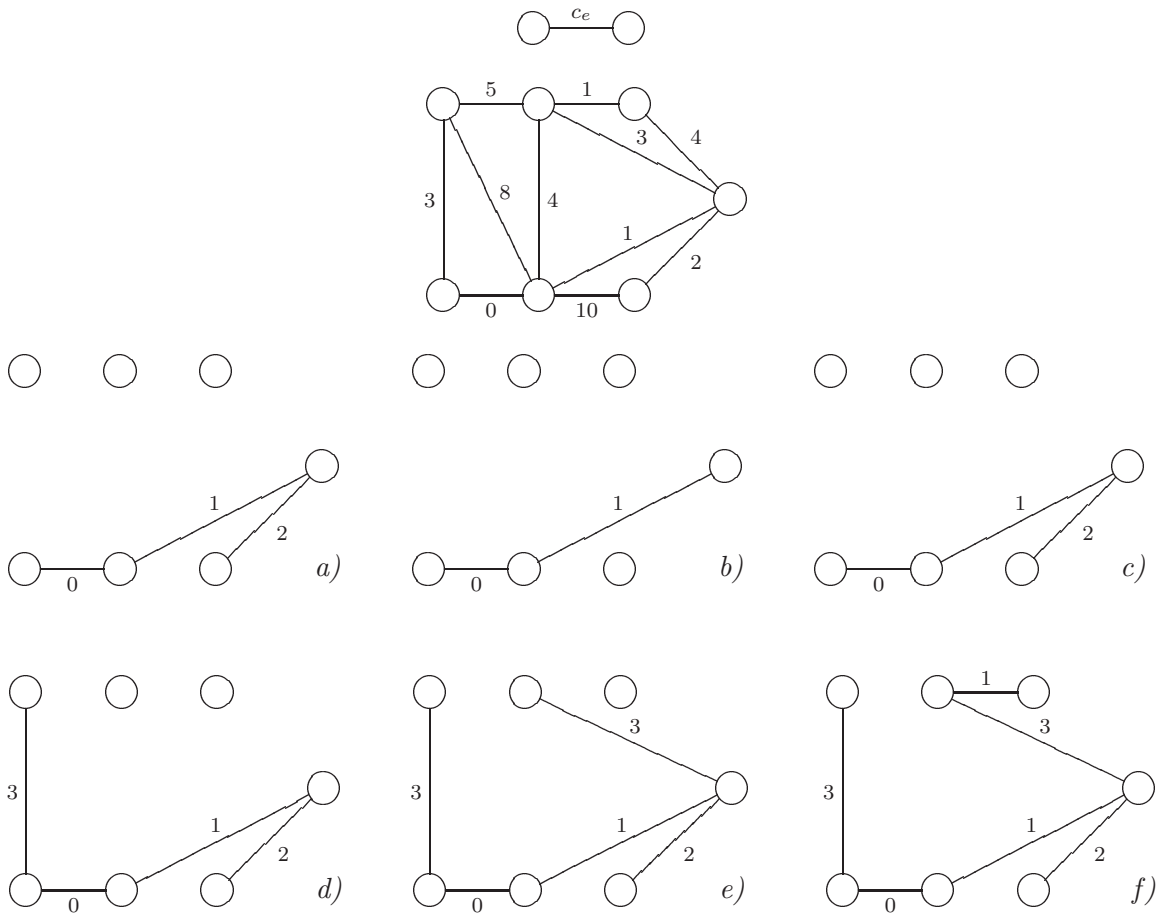
Ensure: minimal spanning tree T

choose an edge $e = \{i, j\}$ with minimal costs $X := \{i, j\}$
$T := \{e\}$
while $|T| < n - 1$ **do**
  choose $e = \{i, j\}$ with $i \in X$ and $j \in V \setminus X$ and minimal c_e
  $X := X \cup \{j\}$
  $T := T \cup \{e\}$
end while

The time complexity of this algorithm is $O(n^2)$ and can be reduced by an efficient implementation to $O(m + n \log n)$.

Example 2.1

The algorithm of Prim works in the following way:



Kruskals Algorithm

Algorithm 2.2 MST - Algorithm of Kruskal

Require: graph $G = (V, E)$ with costs c_e for all $e \in E$

Ensure: minimal spanning tree T

sort the edges of E such that $c_{e_1} \leq ... \leq c_{e_m}$ with $m = |E|$

$T := \emptyset$

while $|T| < n - 1$ **do**

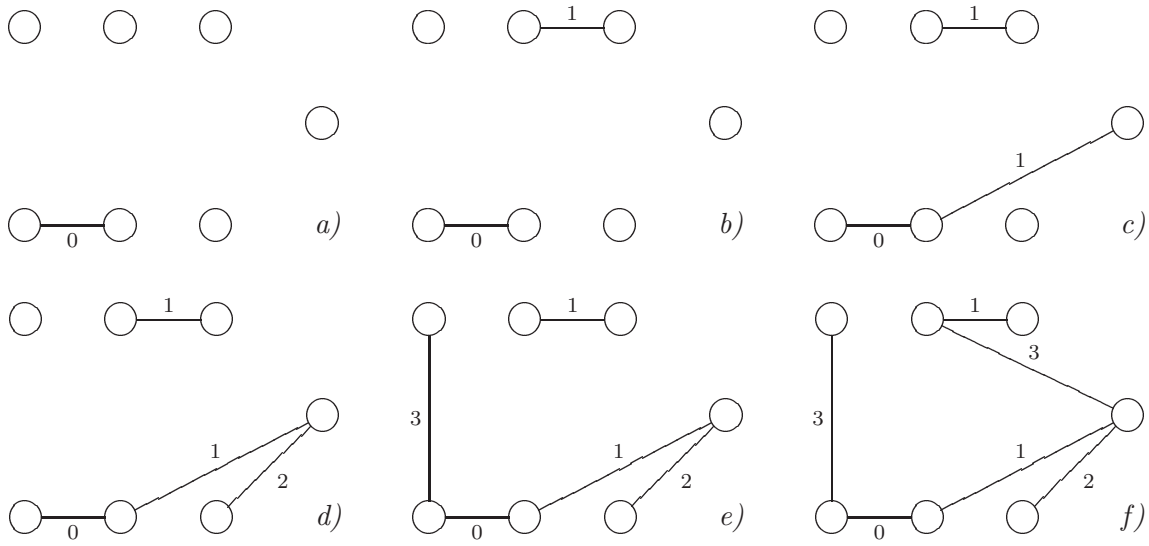
choose an edge with minimal costs such that $T \cup \{e\}$ contains no cycle.

end while

The time complexity of this second algorithm is $O(m \log m)$.

Example 2.2

To expose the difference between both algorithms we consider again the example from the previous section.



Other Algorithms

Furthermore there are some other algorithms with much lower time complexity. Yao [40] and Cheriton and Tarjan [10] independently developed algorithms with $O(m \log \log n)$. In 1986 Gabow, Galli, Spencer and Tarjan [15] published an algorithm with complexity $O(m \log \beta(m, n))$ where $\beta(m, n)$ is the number of needed log-iterations for mapping n to a number less than $\frac{m}{n}$. In [9] an algorithm for the minimal spanning tree problem is proven which runs in $O(m\alpha(m, n))$ where

$$\alpha(m, n) := \min\{i \geq 1 | A(i, 4\lceil \frac{m}{n} \rceil) > \log n\}$$

with the Ackermann's function $A : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ where

$$A(i,j) := \begin{cases} 2j & \text{if } i = 0 \\ 0 & \text{if } j = 0 \\ 2 & \text{if } i \geq 1 \text{ and } j = 1 \\ A(i-1, A(i, j-1)) & \text{if } i \geq 1, j \geq 2. \end{cases}$$

3 Different Formulations

In this chapter we are interested in modelling the problem as integer programm and present two different formulations. For the sake of simplicity we make at first a simplification for our costs and weights.

3.1 Generalization

Lemma 3.1
Without lost of generality we can assume that all weights and costs are positive.

Proof
Suppose a graph with negative costs and weights is given. Let $c_{\min} := |\min_{e\in E} c_e|$ and $w_{\min} := |\min_{e\in E} w_e|$. Then we define:

$$\bar{c}_e := c_e + c_{\min},\ \bar{w}_e := w_e + w_{\min} \text{ for all } e \in E \text{ and } \bar{W} := W + (n-1)w_{\min}$$

An optimal tree $\bar{T}^\star$ for the WCMST with costs $\bar{c}$ and weights $\bar{w}$ such that the total weight of the tree is not greater than $\bar{W}$ is also optimal for the original problem:

$$\begin{aligned}
w(\bar{T}^\star) &= \sum_{e\in\bar{T}^\star} w_e \\
&= \sum_{e\in\bar{T}^\star} (\bar{w}_e - w_{\min}) \\
&= \sum_{e\in\bar{T}^\star} \bar{w}_e - (n-1)w_{\min} \\
&= \bar{w}(\bar{T}^\star) - (n-1)w_{\min} \\
&\leq \bar{W} - (n-1)w_{\min} \\
&= W + (n-1)w_{\min} - (n-1)w_{\min} \\
&= W
\end{aligned}$$

So $\bar{T}^\star$ is feasible. Suppose it exists a tree T with $w(T) \leq W$ and $c(T) < c(\bar{T}^\star)$.

$$\begin{aligned}
\bar{c}(T) &= \sum_{e \in T} \bar{c}_e \\
&= \sum_{e \in T} (c_e + c_{\min}) \\
&= \sum_{e \in T} c_e + (n-1)c_{\min} \\
&= c(T) + (n-1)c_{\min} \\
&< c(\bar{T}^\star) + (n-1)c_{\min} \\
&= \sum_{e \in \bar{T}^\star} (c_e + c_{\min}) \\
&= \bar{c}(\bar{T}^\star)
\end{aligned}$$

Since T is also feasible for the new problem (same arguments) this is a contradiction to the optimality of $\bar{T}^\star$.

□

For the following chapters we make the assumptions that costs and weights are greater or equal than zero. Moreover we suppose that a graph does not contain multiedges and loops.

3.2 WCMST as Integer Program I

An interesting topic is to formulate our problem especially the set $\mathcal{T}$. A classical formulation is the following, which can be found in different sources [1], [2], [3].

Problem 4

$$OPT := \min \sum_{e \in E} c_e x_e \tag{3.1}$$

$$s.t. \sum_{e \in E} w_e x_e \leq W \tag{3.2}$$

$$\sum_{e \in S} x_e \leq |S| - 1, \forall S \in \mathcal{S} \text{ with } 2 \leq |S| \leq n-1 \tag{3.3}$$

$$\sum_{e \in E} x_e = n - 1 \tag{3.4}$$

$$x_e \in \{0, 1\} \tag{3.5}$$

where $\mathcal{S}$ is the set of all subgraphs of G.

The binary variable x_e is equal to 1 if the tree contains the edge e and 0 otherwise. The constraint (3.2) is the weight-restriction. The inequality (3.3) ensures that T contains no cycles and (3.4) guarantees that every vertex is connected. This formulation is not very useful for a LP-relaxation

since the cardinality of the set of all subgraphs $\mathcal{S}$ is very large. For the RCMST problem we can replace (3.2) by

$$\sum_{e \in E} w_e^l x_e \leq W_l \text{ for all } 1 \leq l \leq L \tag{3.6}$$

3.3 WCMST as Integer Program II

A second way to formulate the problem of finding a minimal spanning tree is to interpret the problem as a multicommodity network flow problem. Therefore, we need a directed edge structure: We replace each edge $e \in E$ with nodes i and j by two anti-symmetric arcs (i, j) and (j, i). We call this set $\vec{E}$. Additionally, we define $c_{ij} = c_{ji} := c_e$ and $w_{ij} = w_{ji} := w_e$. For an undirected graph $G = (V, E)$ let $\vec{G} = (V, \vec{E})$ denote the corresponding directed graph.

Definition 3.1
A spanning tree $T \in \vec{G} = (V, \vec{E})$ is a directed-in-tree rooted at node s if the unique path in the tree from any vertex to node $s \in V$ is a directed path.

Every node in the directed in-tree (except the node s) has outdegree 1.

Lemma 3.2
There is an one-to-one correspondence between the trees in G and the set of all directed-in-trees rooted at node n.

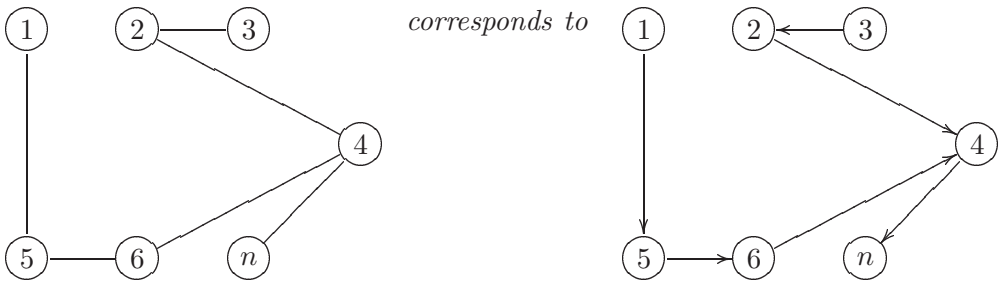
Proof
Let $\mathcal{T}_n$ denote the set of directed-in-trees rooted at node n in $\vec{G}$.

1. Let $T \in \mathcal{T}$. If we orientate all edges in T to n, we have a tree $\vec{T}$ in $\mathcal{T}_n$.
2. Let $\vec{T} \in \mathcal{T}_n$. If we ignore the directions of the tree $\vec{T}$, we get a tree T in $\mathcal{T}$.

So the one-to-one correspondence holds.

□

Example 3.1 *Correspondence*



Obviously the transformation of an undirected tree to a directed tree does not change costs and weights. If we find a directed weight-constrained minimal spanning tree in $\mathcal{T}_n$, we have an optimal solution for our problem.

For the alternative formulation, which can be found in the paper of Jörnsten and Migdalas [29], we interpret the node n as sink and all the other nodes as sources generating one unit of flow. The flow has to be sent from each source over an unique path to the sink:

Problem 5

$$\min \sum_{(i,j)\in\vec{E}} c_{ij}y_{ij} \tag{3.7}$$

$$s.t. \sum_{(i,j)\in\vec{E}} w_{ij}y_{ij} \leq W \tag{3.8}$$

$$\sum_{\substack{j\in V \\ (i,j)\in\vec{E}}} y_{ij}^k - \sum_{\substack{j\in V \\ (j,i)\in\vec{E}}} y_{ji}^k = \begin{cases} 1 & \text{if } i = k \\ -1 & \text{if } i = n \ \ \forall j \in V, \forall k \in V\backslash\{n\} \\ 0 & \text{else} \end{cases} \tag{3.9}$$

$$y_{ij}^k \leq y_{ij} \ \ \forall (i,j) \in \vec{E} \ \forall k \in V\backslash\{n\} \tag{3.10}$$

$$y_{ij}^k \geq 0 \ \ \forall (i,j) \in \vec{E} \ \forall k \in V\backslash\{n\} \tag{3.11}$$

$$y_{ij} + y_{ji} \leq 1 \ \forall (i,j) \in \vec{E} \tag{3.12}$$

$$y_{ij} \in \{0,1\} \tag{3.13}$$

$$\sum_{\substack{j\in V \\ (i,j)\in\vec{E}}} y_{ij} = 1 \ \forall i \in V\backslash\{1\} \tag{3.14}$$

The variable y_{ij} is 1 if an edge (i,j) is element of the solution and 0 otherwise. The $y_{ij}^k \in \mathbb{R}$ denote if a flow generated at node k is sent along the edge (i,j). The equalities (3.9) are the flow conserving constraints and (3.10) ensures that flow is sent along an edge only if the edge is contained in the minimum spanning tree. The inequality (3.14) ensures that only one edge goes out from a node. Combined with (3.12) this guarantees the spanning tree structure. From a resulting directed tree we go back to the undirected tree by ignoring the direction of each arc in the directed tree. To obtain a spanning tree in $G = (V,E)$ we can set $x_e = y_{ij} + y_{ji}$ if $e = \{i,j\}$.

3.4 The Weight-Constrained Minimal Arborescence Problem

In the previous formulation we have considered a directed version of our problem. More formal.

Definition 3.2 *Arborescence*
Let $G = (V,A)$ a directed graph with a set of vertices V and a set of arcs $A := \{(i,j), i,j \in V\}$. An arborescence B, rooted at s, is a connected partial graph $G' = (V,B)$ of G, $B \subset A$, such that

each vertex has out-degree 1, $|B| = n - 1$*, B contains no cycles and for each vertex j exists a unique path from s to j.*

We can extend the weight-constrained minimum spanning tree problem to a *weight-constrained minimum arborescence problem* (WCMA).

Problem 6 *Weight-Constrained Minimum Arborescence Problem*

$$\begin{aligned} \min & \sum_{(i,j)\in B} c_{ij} \\ s.t. & \sum_{(i,j)\in B} w_{ij} \leq W \\ & B \in \mathcal{B} \end{aligned}$$

where $\mathcal{B}$ *is the set of all arborescence.*

By construction we can model $B \in \mathcal{B}$ by (3.9) - (3.14). The difference between the WCMST and the WCMA is that in the WCMA c_{ij} and c_{ji} or w_{ij} and w_{ji} need not necessarily be equal.

3.5 Dual Problem

In the paper of Mehlhorn and Ziegelmann [32] a dual problem for the WCMST is stated. For each possible tree $T \in \mathcal{T}$ we introduce a variable $x_T \in \mathbb{B}$ and get:

Problem 7

$$\begin{aligned} \min & \sum_{T\in\mathcal{T}} c(T)x_T \\ s.t. & \sum_{T\in\mathcal{T}} x_T = 1 \\ & \sum_{T\in\mathcal{T}} w(T)x_T \leq W \\ & x_T \in \mathbb{B} \end{aligned}$$

This leads to the dual problem

Problem 8 *Dual Problem*

$$\begin{aligned} \max & \; u + Wv \\ s.t. & \; u + vw(T) \leq c(T) \;\; \forall T \in \mathcal{T} \\ & \; v \leq 0 \end{aligned}$$

In this formulation only two variables are given, but the number of constraints may be exponential.

4 Interpretation as Matroid Optimization Problem

In Chapter 3 we have discussed the weight-constrained minimal spanning tree problem from the Integer Programming perspective. This chapter deals with a relation to a field of combinatorial optimization, the structure of matroids. In the first section of this chapter we give some fundamental definitions of matroid theory. The second section contains the connection between matroids and the minimal spanning tree problem which is extended to the weight-constrained minimal spanning tree problem in the last part of this chapter.

4.1 Basics in Matroid Theory

Let us define a matroid:

Definition 4.1 *Matroid*
An ordered pair $M = (F, \mathcal{F})$ where $\mathcal{F}$ is a family of subsets of the set F is called matroid if and only if the following three conditions hold.

1. $\emptyset \in \mathcal{F}$
2. *If $S \in \mathcal{F}$ and $S' \in S$, then $S' \in \mathcal{F}$.*
3. *If $S_p, S_{p+1} \in \mathcal{F}$ are subsets with p respectively $p+1$ elements, then there exists a $f \in S_{p+1} \backslash S_p$ with $S_p \cup \{f\} \in \mathcal{F}$.*

We call the elements of M independent sets.

Example 4.1 *[2]*

- ***Graphic Matroid***
  We let $F = E$ where E is the set of edges in a graph G. Furthermore, we define $\mathcal{F}$ as the collection of edge sets which contains no cycle. $(E, \mathcal{F})$ satisfies the matroid properties: If S_p and S_{p+1} are some independent edge sets which contain p and $p+1$ edges, we can add some edge e from S_{p+1} to S_p and get a new set which contains no cycle.
- ***Partition Matroid***
  Let $F = F_1 \cup F_2 \cup \cdots \cup F_K$ be an union of K disjoint sets and let $u_1, \ldots, u_K \in \mathbb{N}$. Let $\mathcal{F}$ be the family of subsets $S \subset F$ such that for all $k \in \{1, \ldots, K\}$ $S \cap F_k$ contains no more than u_k.

Notice that both examples become relevant in Section 12.2. The important idea in matroid theory is the concept of finding maximal independent sets.

Definition 4.2
A set S is called maximal independent set if we cannot add an element $f \in F \setminus S$ such that $S \cup \{f\}$ is independent. A maximal independent set is also called a basis of the matroid.

In Definition 2.2 we have introduced the T-exchange. The corresponding idea in matroid theory (S is basis and $S' = S \setminus \{f\} \cup \{e\}$ is also a basis) is called elementary basis operation.

4.2 Matroid Optimization Problem

Now we associate some costs c_f to each element $f \in F$ and define the costs of a subset $S \subset F$ as

$$c(S) := \sum_{f \in S} c_f.$$

So we can formulate the following problem:

Problem 9 *Matroid Optimization Problem*

$$\min \sum_{f \in S} c_f$$
$$s.t.\ S \text{ is a basis of } M.$$

where $M = (F, \mathcal{F})$ is a matroid.

A simple approach for this problem is the use of a greedy algorithm.

Algorithm 4.1 Greedy algorithm for minimum cost basis of Ahuja, Magnanti and Orlin

order the elements of $F = \{f_1, \ldots, f_m\}$ such that $c_1 \leq c_2 \leq \cdots \leq c_m$
set LIST:= $\emptyset$
for $j = 1$ **TO** m **do**
  if LIST$\cup\{f_j\}$ is independent **then**
5: $LIST := LIST \cup \{f_j\}$
  end if
end for
LIST is a minimum cost basis;

Theorem 4.1 *[2]*
The algorithm solves the matroid optimization problem

Proof
Let $S^\star$ be an optimal solution for Problem 9 and the solution of Algorithm 4.1 is LIST:= $\{f_{j_1}, f_{j_2}, \ldots, f_{j_K}\}$. If LIST= $S^\star$ nothing is to show. Assume $S^\star \neq$ LIST. Suppose the elements of $S^\star$ are ordered in the order of increasing elements from $F = \{f_1, \ldots, f_m\}$ as $f_{j_1}, \ldots, f_{j_k}, f_q, \ldots$ with $f_q \neq f_{j_{k+1}}$ and f_q is the first element of $S^\star$ not contained in LIST. We know that the set $\{f_{j_1}, f_{j_2}, \ldots, f_{j_k}, f_q\}$ is independent. From the construction during the algorithm we can conclude from $q \geq j_{k+1}$ that $c_q \geq c_{j_{k+1}}$. The sets $S = \{f_{j_1}, \ldots, f_{j_k}, f_{j_{k+1}}\}$ and $S^\star$ are independent. So we can add some elements of $S^\star$ to S and obtain another basis S'. This basis must contain the elements $S^\star \cup \{f_{j_{k+1}}\} \setminus f_p$ for some $f_p \in S^\star$ and $p \geq j_{k+1}, c(S') \leq c(S^\star)$. So S' is also an optimal basis. This basis S' has more common elements with LIST than $S^\star$. We can iterate with the set S' until we have equality of S' and LIST.

□

We can interpret the minimal spanning tree problem as matroid optimization problem where we call like in Example 4.1 a set independent if we set E equal to F and call $\mathcal{F}$ the set of all subsets of E which contains no cycles. The Algorithm 4.1 is a generalization of Kruskal's algorithm. The if-clause testing if LIST $\cup\{e_j\}$ is independent corresponds to the test in Kruskal's algorithm if $T \cup \{e\}$ contains a cycle.

4.3 Weight-Constrained Matroid Optimization Problem

Quite obviously, we can extend our Problem 9 to a *weight-constrained matroid optimization problem* if we add a weight constraint to the formulation, if some weights are associated to each $f \in F$:

Problem 10 *Weight-Constrained Matroid Optimization Problem*

$$\begin{aligned} &\min \sum_{f \in S} c_f \\ &s.t. \sum_{f \in S} w_f \leq W \\ &\quad S \text{ is a basis of } M \end{aligned}$$

where $M = (F, \mathcal{F})$ is a matroid.

Like in the previous section this problem is a generalization of the weight-constrained minimal spanning tree problem. The idea to interpret the WCMST as matroid problem is used in Section 12.2. Also a large number of presented algorithms in this thesis can be extended to the weight-constrained minimal matroid optimization problem.

5 Complexity

The minimal spanning tree problem is polynomially solvable by the algorithms of Prim and Kruskal. For the weight-constrained minimal spanning tree problem this result is only true if $NP = P$. Aggarwal, Aneja and Nair [1] and Yamada and Wantanabe [39] prove the NP-hardness.

Theorem 5.1 *NP-Hardness*
The weight-constrained minimal spanning tree problem is NP-hard.

Proof
The idea is to reduce the *knapsack-problem* to the weight-constrained minimal spanning tree problem.
Let us consider an instance of the knapsack-problem:

$$\max \sum_{i=1}^{n} a_i x_i \tag{5.1}$$

$$s.t. \sum_{i=1}^{n} b_i x_i \leq B \tag{5.2}$$

$$x_i \in \{0,1\} \quad \text{for } 1 \leq i \leq n \tag{5.3}$$

Without loss of generality we can assume that all a_i are positive. (Otherwise we define $\tilde{a}_i := a_i + |\min_{1 \leq i \leq n} a_i|$. An optimal x for the problem with costs $\tilde{a}_i$ is also optimal for the original problem.) Using the identity $\max cx = -\min(-c)x$ we can transform the knapsack problem to:

$$-(\min \sum_{i=1}^{n} -a_i x_i) \tag{5.4}$$

$$s.t. \sum_{i=1}^{n} b_i x_i \leq B \tag{5.5}$$

$$x_i \in \{0,1\} \quad \text{for } 1 \leq i \leq n \tag{5.6}$$

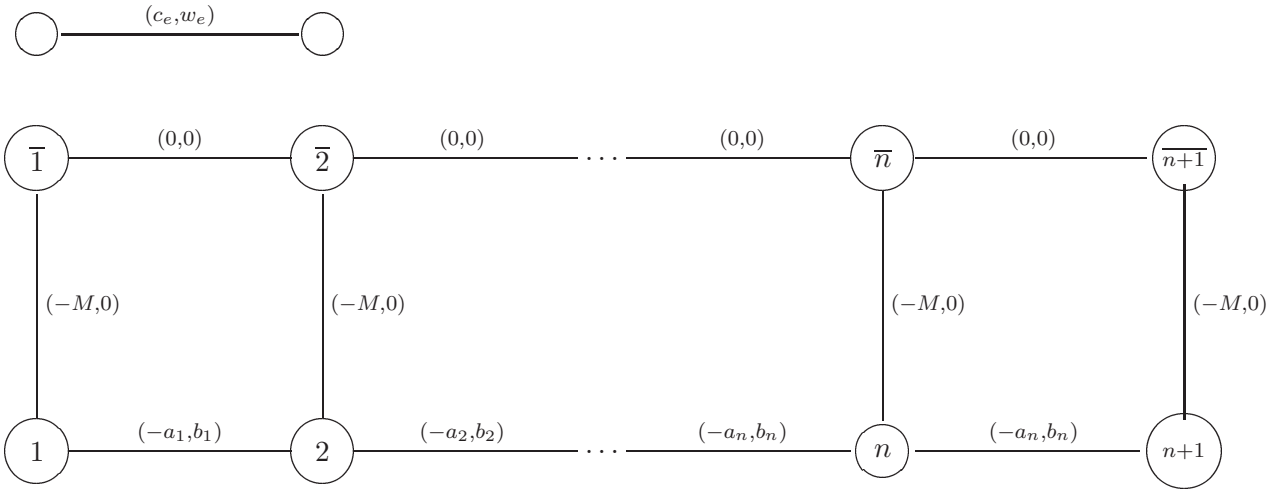
Now we construct a graph (V, E) with the set of vertices

$$V = \{1, \overline{1}, 2, \overline{2}, \ldots, n+1, \overline{n+1}\}$$

and an edge set

$$E = \{\{\overline{1}, 1\}, \{\overline{1}, \overline{2}\}, \{1, 2\}, \{\overline{2}, 2\}, \{\overline{2}, \overline{3}\}, \{2, 3\}, \ldots, \{\overline{n}, \overline{n+1}\}, \{n, n+1\}, \{\overline{n+1}, n+1\}\}.$$

Let $M := 1 + \sum_{k=1}^{n} a_k$. Now we define $c_{\bar{i}i} := -M$ for $1 \leq i \leq n+1$ and $c_{ii+1} := -a_i$ for $1 \leq i \leq n$ and $w_{\bar{i}i} := 0$ for $1 \leq i \leq n+1$ and $w_{ii+1} := b_i$ for $1 \leq i \leq n$. We set $W := B$. (Obviously this can be done in polynomial time.)
This leads to the graph:



If we solve the weight-constrained minimal spanning tree problem for this graph, all edges $\{\bar{i}, i\}$ are contained in an optimal solution. To obtain the spanning tree structure we have for each $i \in \{1, \ldots, n\}$ the opportunity either to take the edge $\{i, i+1\}$ or the edge $\{\bar{i}, \overline{i+1}\}$. If an edge $\{i, i+1\}$ is part of an optimal solution $T^\star$ of the weight-constrained minimal spanning tree problem for this graph, then $x_i = 1$. More formal

$$x_i := \begin{cases} 1 & \text{if } \{i, i+1\} \in T^\star \\ 0 & \text{if } \{\bar{i}, \overline{i+1}\} \in T^\star \end{cases}.$$

So the knapsack-problem can be reduced to the weight-constrained minimal spanning tree problem. Since the knapsack-problem is NP-hard [17] (number [MP9]) the weight-constrained minimal spanning tree problem is also NP-hard.

□

6 Convex Hull

In this chapter we make some statements for the convex hull of our problem and prove some properties of our solution. Therefore it is useful to interpret the weight-constrained minimal spanning tree problem as bicriterial optimization problem and use some already known results of a bicriterial optimization problem for our WCMST. Also we collect some properties of the relation between different trees in the convex hull which are interesting for us.

6.1 WCMST as Bicriterial Problem

The weight-constrained minimal spanning tree problem is related to a bicriterial optimization problem where we consider the constraint as second objective function:

Problem 11

$$\min \begin{pmatrix} \sum_{e \in T} c_e \\ \sum_{e \in T} w_e \end{pmatrix} \tag{6.1}$$

$$s.t.\ T \in \mathcal{T} \tag{6.2}$$

From the bicriterial problem we can classify the set of trees.

Definition 6.1

1. *A tree T is called dominated by a tree $\hat{T}$ if $c(\hat{T}) \leq c(T)$ and $w(\hat{T}) \leq w(T)$ where in at least one case '$<$' holds.*
2. *A tree $\hat{T}$ is called efficient if and only if for all trees T in $\mathcal{T}$ with $c(\hat{T}) \neq c(T)$ and $w(\hat{T}) \neq w(T)$, $c(\hat{T}) \not\geq c(T)$ and $w(\hat{T}) \not\geq w(T)$ hold.*
3. *An efficient tree $\hat{T}$ whose image $(c(T), w(T))$ lies on the border of the convex hull of $\{(c(T), w(T)) | T \in \mathcal{T}\}$ is called a supported tree.*
4. *A tree is called weakly efficient if and only if for all trees T in $\mathcal{T}$ with $c(\hat{T}) \neq c(T)$ and $w(\hat{T}) \neq w(T)$ $c(\hat{T}) \not> c(T)$ and $w(\hat{T}) \not> w(T)$.*

For the relation between the bicriterial optimization problem and the weight-constrained minimal spanning tree problem, we can state the following theorem.

Theorem 6.1

1. *An optimal solution of the weight-constrained minimal spanning tree problem is weakly efficient for the bicriterial problem.*

2. *Under all optimal trees for Problem 1 there exists at least one efficient tree for Problem 11.*

Proof

1. Assume it exists an optimal solution $T^\star$ which is not weakly efficient. Then there exists a tree T with $c(T) < c(T^\star)$ and $w(T) < w(T^\star)$. Since $w(T^\star) \leq W$ T is also feasible and has lower costs than $T^\star$ which is a contradiction to the optimality of $T^\star$.

2. Let $\mathcal{T}^\star := \{T \in \mathcal{T} | c(T) = OPT,\ w(T) \leq W\}$. Take a tree $T^\star_{\min}$ in $\mathcal{T}^\star$ with minimal weight. This tree is efficient for Problem 11 since otherwise a tree T exists with $c(T) \leq c(T^\star_{\min})$ and $w(T) \leq w(T^\star_{\min}) \leq W$ where in one case '$<$' holds. This is a contradiction either to the optimality of $T^\star_{\min}$ or to the property that $T^\star_{\min}$ has minimal weight under all optimal solutions.

□

For graphical interpretation we consider the two dimensional space with the c- and w-axes. The points in the diagram represent the cost-weight-vector of a tree. The convex hull of our Problem 1 is the boundary of the grey region. If we drop the line $W = 35$ we have the convex hull to Problem 11.

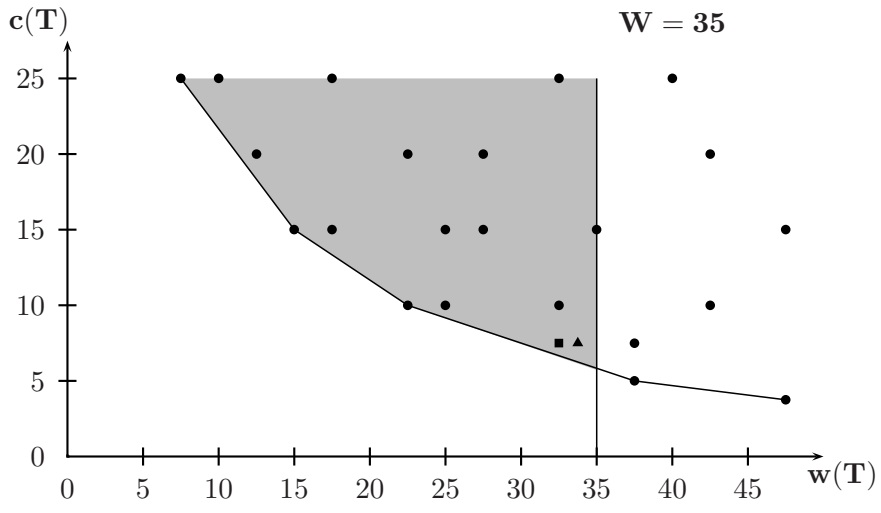


Figure 6.1: Convex hull

The optimal solution to our original problem is the tree with costs 8 and weight 32 marked with a square in the figure. This is an efficient tree. Another optimal solution is the tree with costs 8 and weight 33 marked with a triangle. This tree is weakly efficient.

In [26] Hamacher and Ruhe show that the number of efficient spanning trees is in the worst case exponential in the number of vertices n.

6.2 Adjacency and Connectedness

The next area which will be threaded, is the question whether a feasible solution can be improved by changing edges until an optimal solution is reached. Therefore we need a definition concerning the relation between two trees.

Definition 6.2

1. *Two spanning trees T_1 and T_2 are called adjacent if one T-exchange between the trees exists.*
2. *Two trees T_1 and T_K are called connected if a sequence of trees $T_1, T_2, \ldots, T_n$ exists such that for all $k = 2, \ldots, K$ T_k is adjacent to T_{k+1}.*

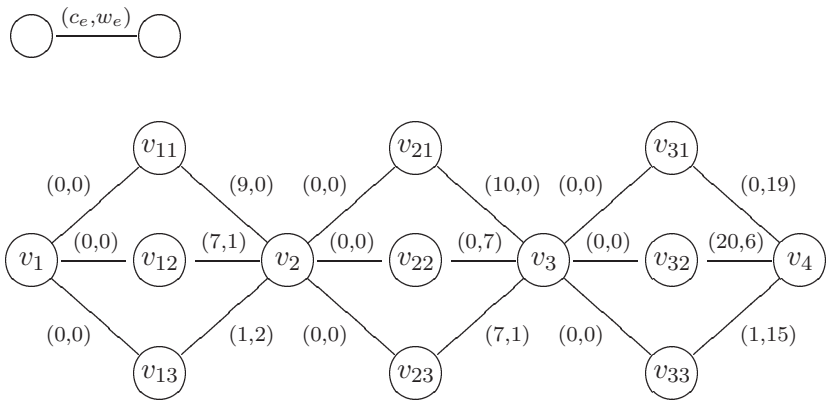
One of the most important properties is the following theorem which describes the relation between the supported trees.

Theorem 6.2 *[1]*
The set of all supported spanning trees of Problem 11 is connected.

This means that for two supported trees T_1 and T_K a sequence $T_1, T_2, \ldots, T_K$ of pairwise adjacent trees exists such that T_k is a supported tree for all $k \in \{1, \ldots, K\}$. In Section 11.1.3 we pick this property up and analyze this connectedness intensively to construct a branch and bound scheme to solve the WCMST.

The set of all efficient trees is not necessarily connected. Consider the following example from Ehrgott and Klamroth [14].

Example 6.1 *Connectedness of Efficient Solutions*



In the table only edges with positive costs and weights are listed. The edges

$$\{v_1, v_{11}\}, \{v_1, v_{12}\}, \{v_1, v_{13}\}, \{v_2, v_{21}\}, \{v_2, v_{22}\}, \{v_2, v_{23}\}, \{v_3, v_{31}\}, \{v_3, v_{32}\}, \{v_2, v_{33}\}$$

belong to every efficient tree.

Tree	*Type*	$(c(T), w(T))$	*Edges*
T_1	*Supported*	*(1, 28)*	$\{v_{13}, v_2\}, \{v_{22}, v_3\}, \{v_{31}, v_4\}$
T_2	*Supported*	*(2, 24)*	$\{v_{13}, v_2\}, \{v_{22}, v_3\}, \{v_{33}, v_4\}$
T_3	*Non-Supp.*	*(8, 22)*	$\{v_{13}, v_2\}, \{v_{23}, v_3\}, \{v_{31}, v_4\}$
T_4	*Supported*	*(9, 18)*	$\{v_{13}, v_2\}, \{v_{23}, v_3\}, \{v_{33}, v_4\}$
T_5	*Non-Supp.*	*(12, 17)*	$\{v_{13}, v_2\}, \{v_{21}, v_3\}, \{v_{33}, v_4\}$
T_6	*Non-Supp.*	*(17, 16)*	$\{v_{11}, v_2\}, \{v_{23}, v_3\}, \{v_{33}, v_4\}$
T_7	*Non-Supp.*	*(20, 15)*	$\{v_{11}, v_2\}, \{v_{21}, v_3\}, \{v_{33}, v_4\}$
T_8	*Non-Supp.*	*(27, 14)*	$\{v_{12}, v_2\}, \{v_{22}, v_3\}, \{v_{32}, v_4\}$
T_9	*Supported*	*(28, 9)*	$\{v_{13}, v_2\}, \{v_{23}, v_3\}, \{v_{32}, v_4\}$
T_{10}	*Supported*	*(31, 8)*	$\{v_{13}, v_2\}, \{v_{21}, v_3\}, \{v_{31}, v_4\}$
T_{11}	*Non-Supp.*	*(36,7)*	$\{v_{11}, v_2\}, \{v_{23}, v_3\}, \{v_{31}, v_4\}$
T_{12}	*Supported*	*(39,6)*	$\{v_{11}, v_2\}, \{v_{21}, v_3\}, \{v_{31}, v_4\}$

If we consider the not-supported tree T_8, it is clear that this tree is not adjacent to any efficient tree and the set of efficient trees is not necessarily connected.

Remark

Aggarwal, Aneja and Nair [1] claimed in their second theorem that the set of efficient solutions is connected. Example 6.1 shows that this claim is not correct. Their proof mentions only the set of supported efficient trees. In [14] it is further shown that each graph can be extended to a graph in which the set of all efficient trees is disconnected.

Conclusion

From this example we can conclude for the weight-constrained minimal spanning tree problem that an optimal solution is not necessarily adjacent to a known efficient tree. Furthermore Ruzika [35] shows that the set of all weakly efficient trees is also not connected. So an optimal solution can not be found by simple edge exchanges. If we are interested in some better upper bounds, it might be possible to improve a feasible solution by $T-$exchanges. Therefore we present in the next section the ideas of Neighborhood- and Adjacency-search.

6.3 Neighborhood- and Adjacency-Search [3]

In the publication of Andersen, Jörnsten and Lind [3] two algorithms are proposed to find efficient solutions for Problem 11. They start by the set of supported solutions. Firstly, we state the two optimality conditions which are needed for the search algorithms. These two conditions are very similar to the optimality conditions in Section 2.1 which are formulated only for costs.

Theorem 6.3 *[3]*
Assume that a tree T is not dominated by any spanning tree which is adjacent to T. Then:

1. *Let $e \in T$. Then there exists no $f \in \{X_e, V \setminus X_e\} \setminus \{e\}$ such that $c_f \leq c_e$, $w_f \leq w_e$ where in at least one case '$<$' holds.*

2. *Let $f \in E \setminus T$. Then there exists no $e \in C(T,f) \setminus \{f\}$ such that $c_f \leq c_e$, $w_f \leq w_e$ where in at least one case '$<$' holds.*

Proof

1. Assume it exists an edge in $f \in \{X_e, V \setminus X_e\} \setminus \{e\}$ with $c_f \leq c_e$, $w_f \leq w_e$ where in at least one case '$<$' holds. Then $T \setminus \{e\} \cup \{f\}$ is a spanning tree which is adjacent to T. Also $c(T \setminus \{e\} \cup \{f\}) = c(T) - c_f + c_e \leq c(T)$ and $w(T \setminus \{e\} \cup \{f\}) = w(T) - w_f + w_e \leq w(T)$ where in at least one case '$<$' holds. This is a contradiction to the assumption that T is not dominated by any adjacent spanning tree.

2. Assume it exists an edge $e \in C(T,f) \setminus \{f\}$ with $c_f \leq c_e$ and $w_f \leq w_e$ where in at least one case '$<$' holds. Then $T \setminus \{e\} \cup \{f\}$ is adjacent to T and $c(T \setminus \{e\} \cup \{f\}) = c(T) - c_e + c_f \leq c(T)$ and $w(T \setminus \{e\} \cup \{f\}) \leq w(T)$ where in at least one case '$<$' holds. This is a contradiction to the assumption that T is not dominated by any adjacent spanning tree.

□

So we have two different methods for searching supported trees.

Neighborhood-Search

Start with the set of all supported solutions. For each tree T in this set we evaluate all trees which are adjacent to this T and not dominated by T and all the trees evaluated so far. If this true, we add this tree to our set. Then we test whether one of the trees evaluated so far was dominated by this new tree. If this true, we eliminate the already known tree and go on with the next adjacent tree to T until all trees in our set are considered.

Adjacent-Search

The algorithm works in the same manner as the Neighborhood-Search but we search only for non dominated trees which are adjacent to at least two known trees.

The main difference of both approaches is that the number of found trees of the neighborhood-search is on average greater than the number of found trees by the adjacency-search. The advantage of the adjacency-search is that a smaller number of trees is examined than in the neighborhood-search.

Extension to the WCMST

Obviously, we can run both algorithms and take from the set of found trees the best efficient tree which is feasible. From Example 6.1 we know that the set of all efficient trees is in general not connected. So both search algorithms will in general not find an optimal solution. But it might be possible that better upper and lower bounds can be found. Notice that we are not interested in the whole set of all efficient trees, so we have to check whether the algorithm can be speed up. For the neighborhood-search a modified version is given in [39] which is proposed in the Section 12.3. For the adjacency-search one could think of considering not all supported solutions in the starting step. Perhaps a feasible and an infeasible supported tree. But this idea might fail: If we search in Example 6.1 for the optimal solution such that $w(T) \leq 7$ and start with T_{10} and T_{12}. We have at first to compute T_9 to find another tree which is adjacent to the optimal solution T_{11}.

7 Lagrangian Relaxation

The most used approach to handle the weight-constrained minimum spanning tree problem is the Lagrangian relaxation ([1], [18], [29], [37], [39]) which delivers a lower bound for the objective value OPT. In this chapter we introduce the Lagrangian relaxation of our problem, show a nice and very useful relation to the convex hull, present two algorithms for finding the Lagrangian dual and make a statement to the quality of the relaxation.

7.1 Definition and Properties

In the best case the value of the Lagrangian relaxation and OPT coincide but in general we have a duality gap.

Problem 12 *Lagrangian Relaxation*

$$C^\star(MST;\mu) := \min\ (c(T) + \mu w(T)) \tag{7.1}$$

$$s.t.\ T \in \mathcal{T} \tag{7.2}$$

This is a minimum spanning tree problem and thus it is easy to solve. For all $\mu \geq 0$ this relaxation is a lower bound for our problem since

$$C^\star(MST;\mu) - \mu W = \min(c(T) + \mu w(T)) - \mu W = \min(c(T) - \mu(W - w(T)) \leq OPT$$

holds for all $\mu \geq 0$. To obtain the best lower bound we have to solve the Lagrangian dual:

Problem 13 *Lagrangian Dual*

$$C^\star(D1) := \max\ (C^\star(MST;\mu) - \mu W) \tag{7.3}$$

$$s.t.\ \mu \geq 0 \tag{7.4}$$

In the following we define $\mu^\star := \arg\max(C^\star(MST;\mu) - \mu W)$. We can illustrate the function $c(T) + \mu w(T) - \mu W$ in the following figure where the dotted lines describe the function for fixed T.
For an alternative way to visualize the Lagrangian Dual we use the same figure as in Section 6.1 where $T^\star_\mu$ solves Problem 12 for $\mu^\star$.

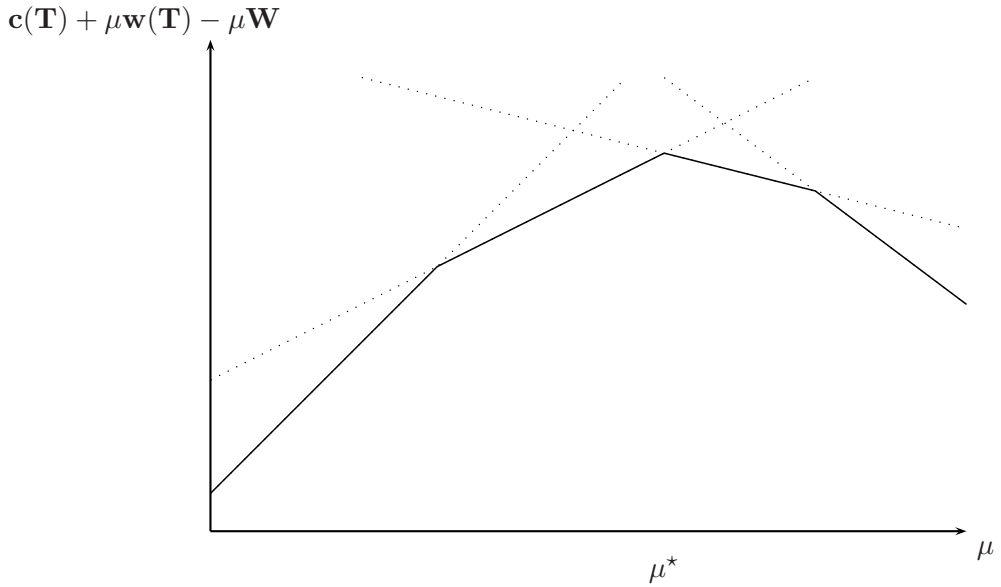


Figure 7.1: Lagrangian Relaxation I

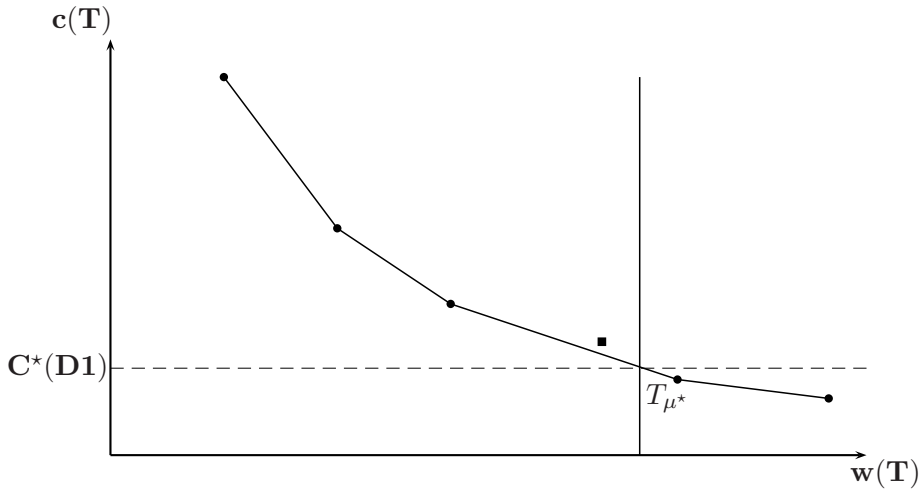


Figure 7.2: Lagrangian relaxation II

In the article of Yamada and Wantanabe [39] some properties of the Lagrangian relaxation for the maximization case are given which can easily be transformed to the minimization case.

Theorem 7.1 *[39]*

1. *$C^\star(MST;\mu) - \mu W$ is piecewise linear and concave in $[0,\infty)$.*
2. *If $C^\star(MST;\mu) - \mu W$ is differentiable at μ,*

$$\frac{dC^\star(MST;\mu) - \mu W}{d\mu} = w(T_\mu) - W$$

where T_μ is the optimal solution to the relaxed problem.

Proof

1. For a fixed tree the function $c(T)+\mu w(T)-\mu W$ is linear in μ. For the function $C^\star(MST;\mu)-\mu W$ we take the lower envelope of all the linear functions which corresponds to a tree. So we get a piecewise linear function.
2. The derivative follows directly.

□

For our $\mu^\star \geq 0$, the optimal solution for the Lagrangian dual, we can conclude from the properties that T_μ is feasible if $\mu > \mu^\star$ and infeasible if $\mu < \mu^\star$.

Theorem 7.2 *[39]*

1. *If $\mu^\star = \infty$, then the WCMST is infeasible.*
2. *Let T_0 is the optimal solution for $\min\{c(T)|T \in \mathcal{T}\}$. If T_0 is feasible, then $\mu^\star = 0$ and T_0 is the optimal solution.*
3. *If $w(T_{\mu^\star}) = W$, then $T_{\mu^\star}$ is optimal and we have no duality gap.*

Proof

1. We know that

$$OPT \geq C^\star(MST;\mu^\star) - \mu^\star W = \min(c(T) - \mu^\star(W - w(T)))$$

For a feasible solution T is $W - w(T) \geq 0$ and so

$$\min(c(T) - \mu^\star(W - w(T))) = -\infty < \min(c(T) - \mu(W - w(T)))$$

for $\mu < \infty$ which is a contradiction to the maximality of $\mu^\star$. There does not exist a feasible tree and the WCMST is infeasible.

2. If T_0, the optimal solution for $\min\{c(T)|T \in \mathcal{T}\}$, is feasible then T_0 is a optimal solution for the WCMST since no other tree has smaller costs. For the Lagrangian relaxation we know that

$$\min(c(T) - \mu^\star(W - w(T)) \leq OPT = c(T_0) = \min(c(T_0) + 0(W - w(T_0))$$

Since $\mu^\star$ maximizes the lower bound of the Lagrangian relaxation, $\mu^\star$ has to be equal to 0.

3. Since

$$OPT \geq C^\star(MST; \mu^\star) - \mu^\star W = \min(c(T_{\mu^\star}) - \mu^\star(W - w(T\mu^\star))) = \min c(T_{\mu^\star})$$

holds $c(T_{\mu^\star}) = OPT$ since $T_{\mu^\star}$ is feasible.

□

7.2 Lagrangian Relaxation and Convex Hull

Between the Lagrangian relaxation and the frontier of the convex hull (more precisely, the lower left frontier of the convex hull) there exist a very nice relation:

Theorem 7.3
Every supported tree T corresponds to a μ such that $c + \mu w$ is minimized by this tree.

Proof
Every facet of the convex hull of the bicriterial problem is a segment of a function in the $w - c$-space:

$$f(w) = -\mu w + b$$

For a supported tree T where $(w(T), c(T))$ lies on such a facet holds

$$c(T) = -\mu w(T) + b.$$

This can be reformulated to

$$b = c(T) + \mu w(T).$$

If we displace the function parallel such that an other vector $(c(T'), w(T'))$ lies on this function, we have to increase b to b'. Then we have for all T' where $(c(T'), w(T'))$ lies not on this facet

$$\sum_{e \in T'} c_e + \mu w_e = c(T') + \mu w(T') = b' > b = c(T) + \mu w(T) \sum_{e \in T} c_e + \mu w_e.$$

Therefore, T minimizes $c(T) + \mu w(T)$.

□

So every facet corresponds to a μ. For two adjacent facets with corresponding μ_1 and μ_2 all trees T where $(c(T), w(T))$ is the extreme point which has both facets in common are optimal for the costs $c + \mu w$ for all $\mu \in [\mu_1, \mu_2]$.

Theorem 7.4 *[8]*
The number of extreme points on the convex hull is polynomially bounded.

Proof
We define for every pair of edges with $w_e \neq w_f$:

$$\mu_{ef} := \frac{c_e - c_f}{w_f - w_e}.$$

We call this μ_{ef} a breakpoint and define the set of all these μ_{ef}

$$M := \{\frac{c_e - c_f}{w_f - w_e} | e, f \in E\}.$$

Let us order and number the elements of M as

$$-\infty < \mu^1 < \mu^2 < \cdots < \mu^K < \infty.$$

We know that K is bounded by $m(m-1)$. Let us now consider an extreme point and the corresponding tree T. We know that T is optimal for the cost function $c(T) + \mu w(T)$ all $\mu \in [\mu_j, \mu_{j+1}] =: \hat{I}$ where μ_j and μ_{j+1} corresponds to the two facets which have this extreme point in common. We know further from the cut optimality condition that

$$c_f + \mu w_f \geq c_e + \mu w_e \text{ for all } e \in T \text{ and for all } f \in Ł(e, T_k)$$
$$\text{where } L(e,T) := \{f \in E \setminus T | e \in C(T_k, f)\}$$

for an arbitrary $\mu \in [\mu_j, \mu_{j+1}]$. Our μ is also element of an interval $[\mu^k, \mu^{k+1}] =: I$. We show that $I \subset \hat{I}$. If $I \not\subset \hat{I}$, we have, without loss of generality, the situation that $\mu^k < \mu_j \leq \mu \leq \mu^{k+1}$. So for a $\bar{\mu} \in (\mu^k, \mu_j)$ an edge $f \in L(e,T)$ has to exists with $c_f + \bar{\mu} w_f < c_e + \bar{\mu} w_e$. Therefore, we have $\mu^k < \bar{\mu} < \mu_{fe} < \mu_j$. This is a contradiction to the fact hat M contains all breakpoints.
So $I \subset \hat{I}$ and the number extreme points is bounded since K is bounded by $m(m-1)$.

□

7.3 Algorithms for Finding the Extreme Points and the Lagrangian Dual

In this section we review two algorithms for finding the (nondominated) extreme points and the Lagrangian dual. In the article of Jüttner concerning the resource constrained spanning tree problems [30] the Handler-Zhang method is proposed to find these points. Since this method is methodically very similar to the following two algorithms it is not presented in this thesis.

Both algorithm check at the beginning whether the weight-constrained minimal spanning tree is feasible and whether a cost minimal solution is an optimal one for the WCMST. This can be done be computing the lexicographical optima:

$$\begin{aligned} &\text{lex } \min (\sum_{e \in T} c_e, \sum_{e \in T} w_e) \\ &s.t.\ T \in \mathcal{T} \end{aligned}$$

and

$$\begin{aligned} &\text{lex } \min (\sum_{e \in T} w_e, \sum_{e \in T} c_e) \\ &s.t.\ T \in \mathcal{T} \end{aligned}$$

Procedure 7.1 Feasibility

Find $T_1 = \arg \text{lex} \min\{(\sum_{e \in T} c_e, \sum_{e \in T} w_e) | T \in \mathcal{T}\}$
$C_1 := c(T_1)$
$W_1 := w(T_1)$
if $W_1 \leq W$ **then**
5: STOP (T_1 is optimal)
else
Find $T_2 = \arg \text{lex} \min\{(\sum_{e \in T} w_e, \sum_{e \in T} c_e) | T \in \mathcal{T}\}$
$C_2 := c(T_2)$
$W_2 := w(T_2)$
10: **if** $W_2 > W$ **then**
STOP (problem is infeasible)
else if $W_2 = W$ **then**
STOP (tree is optimal)
end if
15: **end if**

At the beginning the algorithm tests whether a tree which is optimal for the minimal cost spanning tree problem is also feasible for the WCMST. In the lines 7 - 11 the algorithm decides if the problem is feasible and in the case of $w(T_2) = W$ the algorithm has found an optimal solution.

Algorithm of Aggarwal, Aneja, Nair

In [1] an algorithm for computing all extreme points of the convex hull is proposed:

Algorithm 7.2 Bound algorithm of Aggarwal, Aneja, Nair

Run Procedure 7.1
$\mu := (w(T_1) - w(T_2))/(c(T_2) - c(T_1))$
compute a minimal spanning tree T_3 of G under the costs $c_e + \mu w_e$
repeat this step for (T_1, T_3) and (T_3, T_2) until no more solutions can be found

In this algorithm the quotient denotes the point where $c(T_1) + \mu w(T_1) = c(T_2) + \mu w(T_2)$. In the case that T_1 or T_2 are optimal for μ then T_1 is optimal for all $\bar{\mu} < \mu$ and T_2 is optimal for $\bar{\mu} > \mu$. If another tree T_3 is optimal, the algorithm iterates for (T_1, T_3) and (T_3, T_2).

Algorithm of Xue [38]

In the paper of Xue [38] an algorithm for identifying the closest segment of the convex hull to the optimal solution is proposed.

Definition 7.1
Let $\lambda \in [0, 1]$. We define a graph $G(\lambda) = (V, E, \lambda)$ with the same nodes and edges as G and with cost function $l_e^\lambda = \lambda c_e + (1 - \lambda) w_e \; \forall e \in E$.

We have $l^0 = w$ and $l^1 = c$.

Algorithm 7.3 The approximation scheme of Xue

Run Procedure 7.1
$\lambda := 0; \mu := 1$
while $l^\lambda(T_\mu) \neq l^\lambda(T_\lambda)$ **do**
$\gamma := \begin{cases} \frac{w(T_\mu)-w(T_\lambda)}{w(T_\mu)-w(T_\lambda)+c(T_\lambda)-c(T_\mu)} & \text{odd iteration} \\ \frac{\lambda+\mu}{2}, & \text{even iteration} \end{cases}$
Compute a minimum spanning tree T_γ with respect to l^λ
if there are more than one minimal spanning trees **then**
take one with smallest weights ($\star$)
end if
if $w(T_\gamma) \leq W$ **then**
$\lambda := \gamma$; $T_\lambda := T_\gamma$
else
$\mu := \gamma$
$T_\mu := T_\gamma$;
end if
end while
T^λ is approximation

($\star$) This line is not included in the original algorithm. The reason why we have to add it can be seen in the Example 7.1.

The algorithm finds the facet of the convex hull which is close to the optimal solution. Therefore the algorithm starts with a feasible tree for the WCMST T_λ and an infeasible tree T_μ. By updating the actual tree T_λ is always feasible and the tree T_μ is always infeasible. We try to minimize the distance between T_μ and T_λ by finding values between λ and μ until $\lambda c(T_\mu) + (1-\lambda)w(T_\mu) = \lambda c(T_\lambda) + (1-\lambda)w(T_\lambda)$ which means that T_μ is also optimal for l^λ and the facet is found. The T_λ is an upper bound and the extreme supported spanning tree of the feasible trees which has smallest costs.

Theorem 7.5 *[38]*
Let T_0 be a minimal spanning tree in $G(0)$ and T_1 a minimal spanning tree in $G(1)$.

1. *There exists an optimal tree for the WCMST if and only if $w(T_0) \leq W$.*
2. *If $w(T_1) \leq W$, then T_1 is optimal for the WCMST.*
3. *Assume that a solution for the WCMST exists but $w(T_1) > W$. The algorithm stops after a polynomial number of iterations with λ, μ, T_λ and T_μ such that $l^\lambda(T_\mu) = l^\lambda(T_\lambda), W \in [w(T_\lambda), w(T_\mu)]$, and $OPT \in [c(T_\lambda), c(T_\mu)]$.*

In [38] the theorem is stated but not proven. Only a similar theorem for the weight-constrained shortest path problem is published. For completeness we give here a modification of Xue's proof for the weight-constrained shortest path problem to our problem combined with the results established in the section above.

Proof

1. directly

2. directly

3. Let T and T' spanning trees. If $c(T) = c(T')$ and $w(T) = w(T')$ we call T and T' equivalent. (We denote this by $T \simeq T'$.) Assume that T is an optimal solution. (T' is also an optimal solution if $T' \simeq T$.) Let $\hat{\mathcal{T}}$ denote the set of all efficient spanning trees where we remove for each efficient solution all equivalent spanning trees. From Theorem 6.1 we know that $\hat{\mathcal{T}}$ contains one optimal solution.
   The number of spanning trees in a graph is bounded and therefore $K := |\hat{\mathcal{T}}| < \infty$. Let the trees in $\hat{\mathcal{T}}$ be numbered $T_{[1]}, T_{[2]}, ...T_{[K]}$, such that

   $$c(T_{[1]}) \geq c(T_{[2]}) \geq ... \geq c(T_{[K]}).$$

   Then

   $$w(T_{[1]}) \leq w(T_{[2]}) \leq ... \leq w(T_{[K]})$$

   must hold. Otherwise there are some trees in $\hat{\mathcal{T}}$ which are dominated. Let $T_{[k]}$ be an optimal solution. For $w(T_1) > W$ and $T_1 \in \hat{\mathcal{T}}$ we know that $k < K$. If $k = 1$, then T_0 is an optimal solution. Assume that $1 < k < K$. Since no tree in $\hat{\mathcal{T}}$ is dominated we have

   $$c(T_{[1]}) > c(T_{[2]}) > ... > c(T_{[K]})$$

   and

   $$w(T_{[1]}) < w(T_{[2]}) < ... < w(T_{[K]}).$$

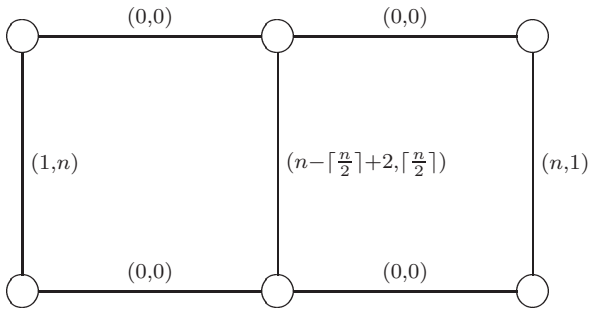
   For every $\lambda \in (0, 1)$ every minimal spanning tree in $G(\lambda)$ is equivalent to a tree in $\hat{\mathcal{T}}$. Since the algorithm only computes spanning trees which are optimal for costs of the form $(1 - \lambda)c + \lambda w$ only the trees on the facets of the convex hull are considered. The number of such trees is bounded by $m(m - 1)$.
   The algorithm stops after $O(\min\{\log K, m^2\})$ iterations (the logarithm depends on the binary search). If the algorithm stops, it is not guaranteed that the solution is optimal. We only know that segment between $(w(T_\lambda), c(T_\lambda))$ and $(c(T_\mu), w(T_\mu))$ is a facet of the convex hull. And for the optimal solution $T^\star$ it must hold that $w(T_\lambda) \leq w(T^\star) \leq w(T_\mu)$ and $c(T_\mu) \leq c(T^\star) \leq c(T_\lambda)$.

□

7.4 Quality of the Lagrangian Relaxation

The question of the goodness of the Lagrangian relaxation and the algorithm of Xue naturally occurs. Unfortunately, the gap between OPT and the result of the algorithms can be made as big as possible:

Example 7.1
Let a graph be given



Let $W = n - 1$. Obviously the optimal solution is a tree with all horizontal edges and the edge $(n - \lceil \frac{n}{2} \rceil + 2, \lceil \frac{n}{2} \rceil)$. So $OPT = n - \lceil \frac{n}{2} \rceil + 2$.

1. *Algorithm:*

Iteration	λ	μ	$(c(T_\lambda), w(T_\lambda))$	$(c(T_\mu), w(T_\mu))$	$l^\lambda(T_\mu)$	$l^\lambda(T_\lambda)$	γ	$(c(T_\gamma), w(T_\gamma))$
1	*0*	*1*	$(n, 1)$	$(1, n)$	n	*1*	$\frac{1}{2}$	$(n, 1)(\star\star)$
2	$\frac{1}{2}$	*1*	$(n, 1)$	$(1, n)$	$\frac{1}{2}n + \frac{1}{2}$	$\frac{1}{2}n + \frac{1}{2}$		*STOPP*

In $(\star\star)$ also $(1, n)$ is a solution since

$$\frac{1}{2}n + \frac{1}{2}1 = \frac{1}{2}1 + \frac{1}{2}n = \frac{n}{2} + \frac{1}{2} < \frac{n}{2} + 1 = \frac{1}{2}(n - \lceil \frac{n}{2} \rceil + 2) + \frac{1}{2}\lceil \frac{n}{2} \rceil$$

Assume we do not have line $(\star)$ in the algorithm and choose this tree:

Ite.	λ	μ	$(c(T_\lambda), w(T_\lambda))$	$(c(T_\mu), w(T_\mu))$	$l^\lambda(T_\mu)$	$l^\lambda(T_\lambda)$	γ	$(c(T_\gamma), w(T_\gamma))$
1	*0*	*1*	$(n, 1)$	$(1, n)$	n	*1*	$\frac{1}{2}$	$(1, n)$
2	*0*	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	n	*1*	$\frac{1}{4}$	$(n, 1)$
3	$\frac{1}{4}$	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	$\frac{3}{4}n + \frac{1}{4}$	$\frac{n}{4} + \frac{3}{4}$	$\frac{1}{2}$	$(1, n)$
4	$\frac{1}{4}$	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	$\frac{3}{4}n + \frac{1}{4}$	$\frac{n}{4} + \frac{3}{4}$	$\frac{3}{8}$	$(n, 1)$
5	$\frac{3}{8}$	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	$\frac{5}{8}n + \frac{3}{8}$	$\frac{3}{8}n + \frac{5}{8}$	$\frac{1}{2}$	$(1, n)$
6	$\frac{3}{8}$	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	$\frac{5}{8}n + \frac{3}{8}$	$\frac{3}{8}n + \frac{5}{8}$	$\frac{7}{16}$	$(n, 1)$
7	$\frac{7}{16}$	$\frac{1}{2}$	$(n, 1)$	$(1, n)$	$\frac{9}{16}n + \frac{7}{16}$	$\frac{7}{16}n + \frac{9}{16}$	$\frac{1}{2}$	$(1, n)$
...	...	...	...	...	...	...	...	...

It is easy to see that the algorithm does not terminate. So the modification of the algorithm is necessary.

The solution of the algorithm is a tree T with $c(T) = n$ and $w(T) = 1$. For our optimal solution we have

$$c(T) - OPT = n - (n - \lceil \frac{n}{2} \rceil + 2) = \lceil \frac{n}{2} \rceil - 2$$

and our gap is not bounded.

2. *Lagrangian relaxation*
   Additionally we will compute the Lagrangian dual by the definition above. We consider the Lagrangian relaxation

$$\begin{aligned} &\min c(T) + \mu w(T) \\ &s.t.\ T \in \mathcal{T} \end{aligned}$$

For $\mu \geq 1$ we have

$$n + \mu 1 \leq 1 + \mu n \text{ and } n + \mu 1 < n + 2 + (1 - \mu)\lceil \frac{n}{2} \rceil = (n - \lceil \frac{n}{2} \rceil + 2 + \mu \lceil \frac{n}{2} \rceil)$$

and for $\mu \leq 1$ we have

$$1 + \mu n \leq n + \mu 1$$

and

$$\begin{aligned} 1 + \mu n &< \mu n + 2 \\ &= n + 2 - n + \mu n + 3 - 3\mu \lceil \frac{n}{2} \rceil \\ &\leq n + 2 - (1 - \mu)(n - \lceil \frac{n}{2} \rceil) \\ &\leq n + 2 - (1 - \mu)\lceil \frac{n}{2} \rceil \\ &\leq n - \lceil \frac{n}{2} \rceil + 2 + \mu \lceil \frac{n}{2} \rceil \end{aligned}$$

For the Lagrangian dual we have to consider

$$\begin{aligned} &\max n + \mu 1 - \mu(n - 1) \\ &s.t.\ \mu \geq 1 \end{aligned}$$

and

$$\begin{aligned} &\max 1 + \mu n - \mu(n - 1) \\ &s.t.\ \mu \leq 1 \end{aligned}$$

In the first case we have for $\mu \geq 1$

$$n + \mu 1 - \mu(n - 1) = n(1 - \mu) + 2\mu$$

Since $\mu \geq 1$ this term is maximized for large n for $\mu = 1$.
In the second case we have for $\mu \leq 1$

$$1 + \mu n - \mu(n-1) = 1 + \mu$$

Since $\mu \leq 1$ this term is maximized for $\mu = 1$.
From both cases we obtain the value of the Lagrangian dual

$$c^*(D1) = 2$$

For the duality gap we get

$$OPT - c^\star(D1) = (n - \lceil \frac{n}{2} \rceil + 2) - 2 = (n - \lceil \frac{n}{2} \rceil)$$

For large n no approximation quality can be guaranteed.

8 Alternative Relaxation

The following alternative relaxation was published by Jörnsten and Migdalas [29]. The main idea is to combine variable splitting with the Lagrangian relaxation.
The basis for this consideration is the formulation Problem 5. We introduce some auxiliary variable z_{ij} for all $(i,j) \in \overrightarrow{E}$ and some α and β which are non negative parameters with $\alpha+\beta = 1$:

Problem 14

$$\min \alpha \sum_{(i,j)\in\overrightarrow{E}} c_{ij}y_{ij} + \beta \sum_{(i,j)\in\overrightarrow{E}} c_{ij}z_{ij} \tag{8.1}$$

$$s.t.\ (3.9) - (3.14)$$

$$\sum_{(i,j)\in\overrightarrow{E}} w_{ij}z_{ij} \leq W \tag{8.2}$$

$$z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \tag{8.3}$$

$$z_{ij} = y_{ij}\ \forall (i,j) \in \overrightarrow{E} \tag{8.4}$$

It is obvious that the objective values of Problem 5 and Problem 14 are equal.
We consider now the well-known Lagrangian relaxation and the Lagrangian dual problem

$$c^\star(D1) := \max c^\star(MST;\mu) - \mu W$$
$$s.t.\ \mu \geq 0.$$

We apply the Lagrangian relaxation to Problem 14 and split the problem into two subproblems

Problem 15

$$c^\star(MST;\lambda) = \min \sum_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \lambda_{ij})y_{ij}$$
$$s.t.\ (3.9) - (3.14)$$

and

Problem 16

$$c^\star(KNPSK;\lambda) = \min \sum_{(i,j)\in\overrightarrow{E}} (\beta c_{ij} - \lambda_{ij})z_{ij}$$
$$s.t. \sum_{(i,j)\in\overrightarrow{E}} w_{ij}z_{ij} \leq W$$
$$z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E}$$

We have the easily solvable Problem 15 of finding a minimal spanning tree and the NP-hard Problem 16 of solving a knapsack problem. We define also a dual problem to the splitten problem:

Problem 17

$$\begin{aligned} c^\star(D2) := \max\ & (c^\star(MST;\lambda) + c^\star(KNPSK;\lambda)) \\ s.t.\ & \lambda_{ij} \geq 0\ \forall (i,j) \in \overrightarrow{E} \\ & \lambda = (\lambda_{ij})_{(i,j)\in\overrightarrow{E}} \end{aligned}$$

For this problem we can conclude the following important result:

Proposition 8.1
The value $c^\star(D2)$ is a lower bound for OPT and is at least as good as the value of the Lagrangian dual, i.e.

$$c^\star(D2) \geq c^\star(D1).$$

Proof
Let $\lambda^\star$ and $\mu^\star$ be the optimal solutions to Problem 17 and Problem 13 respectively. For $\alpha+\beta=1$ we have

$$\bar{\lambda} := \mu^\star w + \beta c$$

with $w := (w_{ij})_{(i,j)\in\overrightarrow{E}}$ and $c := (c_{ij})_{(i,j)\in\overrightarrow{E}}$. Then

$$
\begin{aligned}
c^\star(D2) &= c^\star(MST;\lambda^\star) + c^\star(KNPSK;\lambda^\star) \\
&\geq c^\star(MST;\bar{\lambda}) + c^\star(KNPSK;\bar{\lambda}) \\
&= \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \bar{\lambda}_{ij}) y_{ij} \\ s.t. (3.9) - (3.14) \end{array} \right\} + \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (\beta c_{ij} - \bar{\lambda}_{ij}) z_{ij} \\ s.t. \sum\limits_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} \\
&= \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \mu^\star w_{ij} + \beta c_{ij}) y_{ij} \\ s.t. (3.9) - (3.14) \end{array} \right\} + \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (\beta c_{ij} - (\mu^\star w_{ij} + \beta c_{ij})) z_{ij} \\ s.t. \sum\limits_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} \\
&= \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (c_{ij} + \mu^\star w_{ij}) y_{ij} \\ s.t. (3.9) - (3.14) \end{array} \right\} + \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} -\mu^\star w_{ij} z_{ij} \\ s.t. \sum\limits_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} - \mu^\star W + \mu^\star W \\
&= \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} (c_{ij} + \mu^\star w_{ij}) y_{ij} - \mu^\star W \\ s.t. (3.9) - (3.14) \end{array} \right\} + \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} \mu^\star W - \mu^\star w_{ij} z_{ij} \\ s.t. \sum\limits_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} \\
&= c^\star(D1) + \left\{ \begin{array}{c} \min \sum\limits_{(i,j)\in\overrightarrow{E}} \mu^\star W - \mu^\star w_{ij} z_{ij} \\ s.t. \sum\limits_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} \\
&\geq c^\star(D1)
\end{aligned}
$$

It remains to show that Problem 17 is a relaxation i.e., $c^\star(D2) \leq OPT$. This part of the proof was not shown in [29]. We prove this in the following.

$$
\begin{aligned}
c^\star(D2) &= \max_{\lambda}(c^\star(MST;\lambda) + c^\star(KNPSK;\lambda)) \\
&= \max_{\lambda}(\left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \lambda_{ij}) y_{ij} \\ s.t. (3.9) - (3.14) \end{array} \right\} + \left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} (\beta c_{ij} - \lambda_{ij}) z_{ij} \\ s.t. \sum_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\}) \\
&= \max_{\lambda}(\left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \lambda_{ij}) y_{ij} + (\beta c_{ij} - \lambda_{ij}) z_{ij} \\ s.t. (3.9) - (3.14) \\ \sum_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\ z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\}) \\
&\leq \max_{\lambda}(\left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} (\alpha c_{ij} + \lambda_{ij}) y_{ij} + (\beta c_{ij} - \lambda_{ij}) z_{ij} \\ s.t. (3.9) - (3.14) \\ \sum_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W\ \forall (i,j) \in \overrightarrow{E} \\ z_{ij} = y_{ij} \forall (i,j) \in \overrightarrow{E} \end{array} \right\}) \\
&= \max_{\lambda}(\left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} c_{ij} y_{ij} \\ s.t. (3.9) - (3.14) \\ \sum_{(i,j)\in\overrightarrow{E}} w_{ij} y_{ij} \leq W\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\}) \\
&= \left\{ \begin{array}{c} \min \sum_{(i,j)\in\overrightarrow{E}} c_{ij} y_{ij} \\ s.t. (3.9) - (3.14) \\ \sum_{(i,j)\in\overrightarrow{E}} w_{ij} y_{ij} \leq W\ \forall (i,j) \in \overrightarrow{E} \end{array} \right\} \\
&= OPT
\end{aligned}
$$

□

Corollary 8.2

The bound of the splitting approach is strictly better if the objective value of

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in\overrightarrow{E}} \mu^\star W - \mu^\star w_{ij} z_{ij} \\
s.t. \quad & \sum_{(i,j)\in\overrightarrow{E}} w_{ij} z_{ij} \leq W \\
& z_{ij} \in \{0,1\}\ \forall (i,j) \in \overrightarrow{E}
\end{aligned}
$$

is not 0.

Proof

This follows directly from the proof of the previous proposition.

□

We can add

$$\sum_{(i,j)\in\overrightarrow{E}} z_{ij} = n - 1$$

to Problem 16 which is also a NP-hard problem. This might lead to a better bound without increasing the complexity substantially.

Conclusion

The advantage of this approach is not only of theoretical worth: Every Lagrangian multiplier corresponds to a binary variable. An infeasible solution depends on different values in a pair (y_{ij}, z_{ij}). By changing the according multipliers only this pair is affected.

Generalization

The proposed approach of variable splitting in this chapter can also be applied to several other problems of the form

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \leq 0 \\ &\quad G(x) \leq 0 \\ &\quad x \in X \end{aligned}$$

The values of the traditional Lagrangian relaxation are

$$v_1 = \min\{f(x)|G(x) \leq 0, x \in \text{Conv}[F(x) \leq 0, x \in x]\}$$

and

$$v_2 = \min\{f(x)|F(x) \leq 0, x \in \text{Conv}[G(x) \leq 0, x \in X]\}$$

Therefore the value of the reformulated problem is

$$v = \min\{f(x)|x \in \text{Conv}[F(x) \leq 0, x \in X] \cap \text{Conv}[G(x) \leq 0, x \in X]\}.$$

Then we have $v \geq \max\{v_1, v_2\}$.

9 In- and Exclusion Tests

In this chapter some possibilities are presented which allow us to decide for some edges whether the edge is in at least one optimal tree (inclusion) or whether there is no optimal tree containing this edge (exclusion). The ambition of these tests is to reduce the complexity of our problem. In contrast to the weight-constrained shortest path problem [5] in the literature only Aggarwal, Aneja and Nair [1] touch on this subject. Their algorithm will be found at the end of this chapter. For the sake of completeness also some trivial results will appear in the following. Regrettably, the most results are of more theoretical interest and not quite useful for practical applications.

9.1 Inclusion Tests

Theorem 9.1
If there is a cut X and $V \setminus X$ with $|\{X, V \setminus X\}| = 1$, then this edge is in every optimal and furthermore in every feasible solution.

Proof
Since the solution of the weight-constrained minimal spanning tree problem must be connected we have to include this edge in every tree.

□

Corollary 9.2
If our graph contains a leaf, the corresponding edge is element of every optimal solution.

In this case we can search for a minimal spanning tree in the remaining $n - 1$ vertices and with total weight less than W minus the weight of the edge of the leaf.

Theorem 9.3
If for an edge $e \in E$ there does not exist an edge $f \in E$ with $c_f < c_e$ and $w_f \leq w_e$, then there exists at least one optimal solution which contains e.

Proof
Assume the theorem does not hold. Let T be an optimal solution. We consider $T \setminus \{f\} \cup \{e\}$ where f is an arbitrary edge in the cycle $C(T, e)$. We get

$$w(T \setminus \{f\} \cup \{e\}) = w(T) - w_f + w_e \leq w(T) \leq W$$

and

$$c(T \setminus \{f\} \cup \{e\}) = c(T) - c_f + c_e < c(T)$$

which is a contradiction to the optimality of T. So the theorem holds.

□

Corollary 9.4
Let $e_1, \ldots, e_K$ be edges such that $\bigcup_{k=1}^{K} e_k$ does not contain a cycle and $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_K} < c_f$ and $w_{e_k} \leq w_f$ for all $k \in \{1, \ldots, K\}$ and for all $f \in E \setminus \bigcup_{k=1}^{K} e_k$, then there exists an optimal solution containing the edges $e_1, \ldots, e_K$.

Proof
Assume the theorem does not hold. Let T be an optimal solution such that an edge e_k exists with $e_k \notin T$. We consider $T \setminus \{f\} \cup \{e_k\}$. By adding the edge e_k the set $T \setminus \{e_k\}$ must contain a cycle. In this cycle at least one edge has to exist which is not in $\{e_1, \ldots, e_K\}$ since this set contains no cycle. We get:

$$w(T \setminus \{f\} \cup \{e_k\}) = w(T) - w_f + w_{e_k} \leq w(T) \leq W$$

and

$$c(T \setminus \{f\} \cup \{e_k\}) = c(T) - c_f + c_{e_k} < c(T)$$

which is a contradiction to the optimality of T. So the corollary holds.

□

Theorem 9.5
Let $e = \{i, j\} \in E$. If along every path d_{ij}^k from i to j in G which does not contain the edge e exists an edge e_k such that $c_e < c_{e_k}$ and $w_e \leq w_{e_k}$, then exists an optimal solution which contains e.

Proof
Assume the theorem does not hold. In the optimal solution T there exists one path d_{ij}^T from i to j. If we consider $T \cup \{e\}$, we get a cycle $d_{ij}^T \cup \{e\}$. On this cycle there exists an edge e_T such that $c_e < c_{e_T}$ and $w_e \leq w_{e_T}$. For $T \setminus \{e_T\} \cup \{e\}$ holds

$$w(T \setminus \{e_T\} \cup \{e\}) = w(T) - w_{e_T} + w_e \leq w(T) \leq W$$

and

$$c(T \setminus \{e_T\} \cup \{e\}) = c(T) - c_{e_T} + c_e < c(T)$$

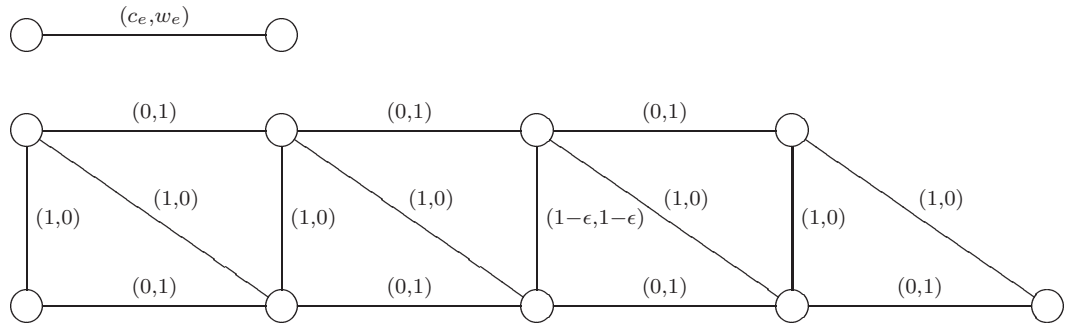
which is a contradiction to the optimality of T. So the theorem holds.

□

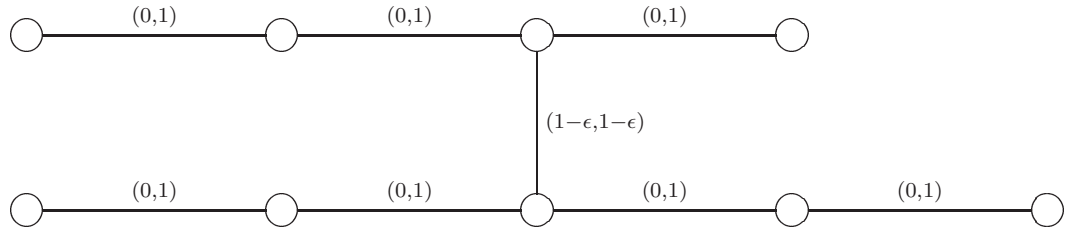
In this context one question forces on: Is an edge being contained in at least one minimal spanning tree with respect to the costs and in at least one minimal spanning tree with respect to the weights also in at least one optimal solution for the WCMST? The following example provides a negative answer.

Example 9.1

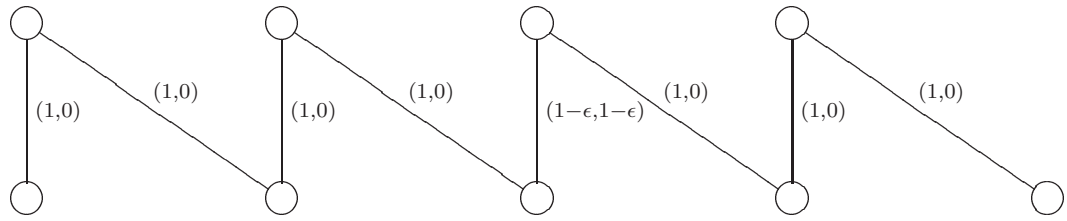
We consider the following graph with $\epsilon \in (0,1)$*:*



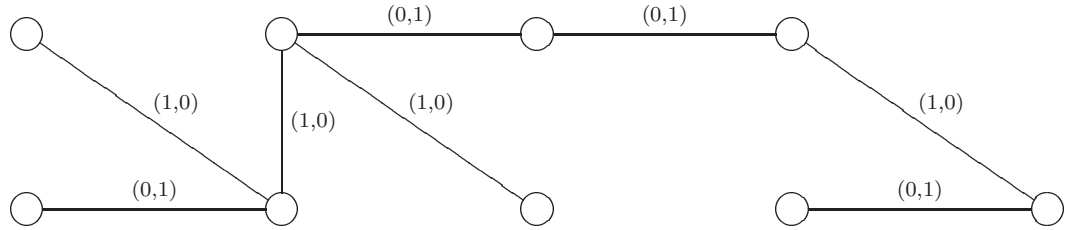
A minimal spanning tree with respect to the costs is the tree T_1 *with* $c(T_1) = 1-\epsilon$ *and* $w(T_1) = 8-\epsilon$*:*



A minimal spanning tree with respect to the weights is the tree T_0 *with* $c(T_0) = 8-\epsilon$ *and* $w(T_0) = 1-\epsilon$*:*



In both trees the edge with the cost-weight vector $(1-\epsilon, 1-\epsilon)$ *occurs. We are searching now for a WCMST with* $w(T) \leq 4$*. A feasible tree is the tree:*



This tree has costs 4 and weight 4. Assume it exists a tree T *containing the edge with cost-weight vector* $(1-\epsilon, 1-\epsilon)$ *with* $c(T) \leq 4$ *and* $w(T) \leq 4$*. The sum of the costs of the remaining seven*

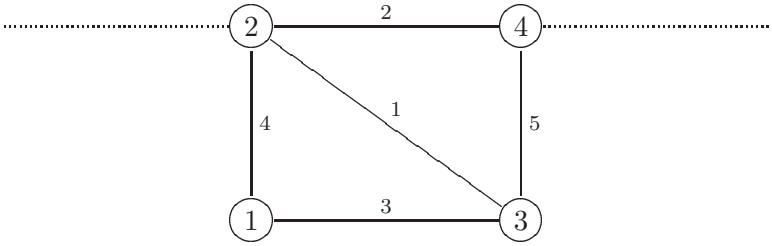
edges has to be smaller than $3+\epsilon$ and the sum of the weights of these edges has to be also smaller than $3+\epsilon$. Since all the possible edges in the graph have an integral cost-weight vector both sums have to be smaller than 3. This is not possible since seven edges have to be chosen and by this one sum is greater than 3. So no optimal solution to the WCMST-problem can contain the edge $(1-\epsilon, 1-\epsilon)$.

9.2 Exclusion Tests

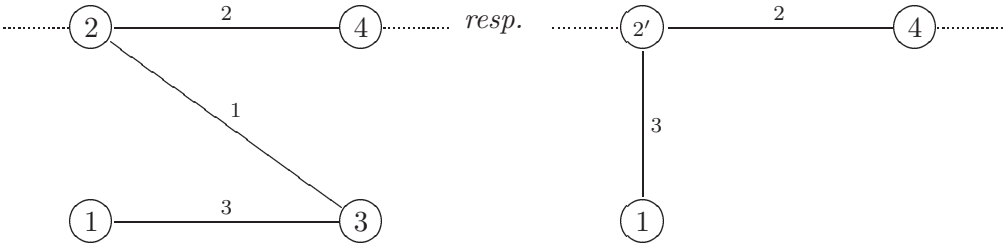
Before starting with exclusion tests we focus on the problem how to handle with edges found by an inclusion-test. Unfortunately, the way to update the graph in an ordinary minimal cost spanning tree problem after finding one edge which is element of an optimal solution can not be extended to our and other multicriterial spanning tree problems. The difficulty of updating will be described in the following example.

Example 9.2

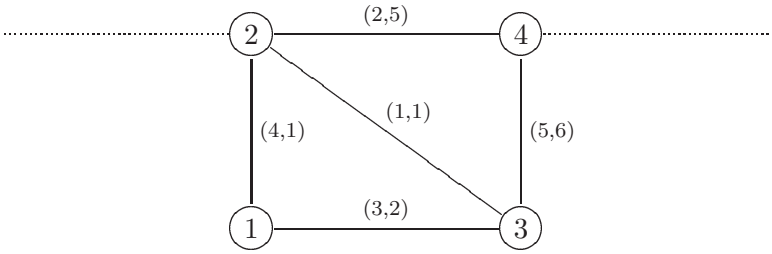
Assume that we have a graph with the following subgraph and only costs associated to each edge:



Let the edge $\{2,3\}$ be found by an inclusion test and be a member of an optimal solution for the minimal spanning tree problem. Then we can interpret the nodes 2 and 3 as one vertex and for all edges $\{2,k\},\{3,k\}$ we can exclude all edges $\{3,k\}$ with $c_{2k} \leq c_{3k}$ and all edges $\{2,k\}$ with $c_{3k} < c_{2k}$. So for every k only one edge $\{2,k\}$ or $\{3,k\}$ is in the tree and we can exclude some edges and get a graph with $n-1$ nodes. In this example the graph reduces to:



Now we consider a subgraph for our WCMST with costs and weights for each edge:



Let the edge $\{2,3\}$ be found by an inclusion test and be a member of an optimal solution for a weight-constrained minimal spanning tree. We cannot decide if the edge $\{1,2\}$ or $\{1,3\}$ can be dropped since concerning the costs $\{1,3\}$ is a better choice but $\{1,2\}$ is a better choice for the constraint. So in this case no edge can be dropped and the graph cannot be reduced to a graph with $n-1$ nodes since multiple edges are not allowed. On the other hand, the edge $\{3,4\}$ can be excluded from the graph since $c_{24} = 2 < 5 = c_{34}$ and $w_{24} = 5 < 6 = w_{34}$ and $\{3,4\}$ cannot be an edge of an optimal solution.

Now we formulate the results from this example in a more formal way.

Definition 9.1 *Dominated Edges*
Let $A, B \setminus V$ be disjoint and T_A and T_B two subtrees for the sets A and B. Let $E_{AB} \subset E$ denote the sets of edges having one node in A and one node in B. An edge $e = \{i,j\} \in E_{AB}$ is dominated by some edge $f = \{k,l\} \subset E_{AB}$ if and only if $c_f < c_e$ and $w_f \leq w_e$.

Theorem 9.6 *Exclusion of Dominated Edges*
Let we have found some disjoint subtrees T_A and T_B of an optimal tree for the WCMST. Let $A := V(T_A)$ and $B := V(T_B)$ the two connected set of nodes. All dominated edges in E_{AB} are not elements of the optimal solution which contains the subtrees T_A and T_B.

Proof
Assume that the theorem is not valid and a dominated edge e with one node in A and the other node in B is in the optimal solution T. The edge f dominating e is not in the tree since otherwise we have a cycle. Let us consider $T \setminus \{e\} \cup \{f\}$. This set is a tree since f connects the subtrees T_A and T_B. We have

$$w(T \setminus \{e\} \cup \{f\}) = w(T) - w_e + w_f \leq w(T) \leq W$$

and

$$c(T \setminus \{e\} \cup \{f\}) = c(T) - c_e + c_f < c(T)$$

which contradicts the optimality of T. So the theorem is valid.

□

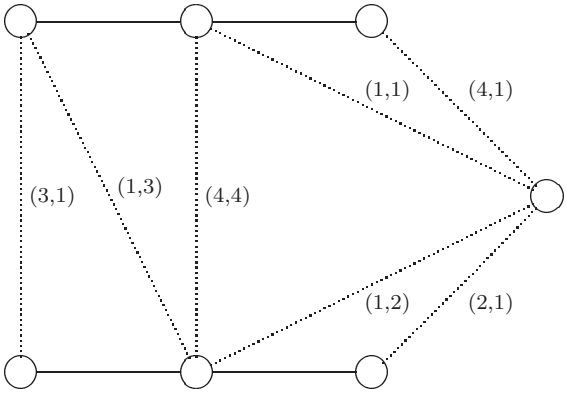
Corollary 9.7
If we have found a subtree T_A which is in an optimal solution T and which connects the node set A, then we can drop all dominated edges in $E_{A\{v\}}$ for all $v \in V \setminus A$.

Proof
Analogously to the proof of the previous theorem.

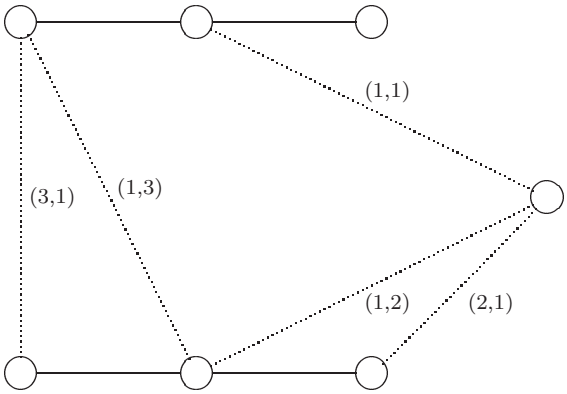
□

Example 9.3

Let in the following graph the solid edges be the edges which are in an optimal tree and the dotted edges be all possible edges that can be contained in an optimal solution.



The edges with the cost-weight vector $(4,4)$ *and* $(4,1)$ *are dominated and cannot be elements of an optimal solution. Our graph reduces to:*



Surely, it is also possible to exclude an edge e if there exists an edge f between the sets A and B with $c_f = c_e$ and $w_f < w_e$. In this case we have to give up our ambition to exclude only edges which cannot appear in every optimal solution.

Theorem 9.8

All edges e *with* $w_e > W$ *and more strictly all edges with* $w_e > W - \sum_{i=1}^{n-2} w_{e_i}$ *where* e_i, $1 \leq i \leq n-2$, *are edges with smallest weight in the graph cannot occur in a feasible solution.*

Proof

It is clear that the total weight of every tree including an edge with costs greater than W is greater than W and therefore this tree is not feasible. Further, the total weight of every tree is greater than the sum of $n-2$ times the smallest weights in the graph and therefore an edge with weights $w_e > W - \sum_{i=1}^{n-2} w_{e_i}$ is not a feasible solution.

□

If an upper bound UB for the optimal solution is found, we can reformulate the theorem by replacing W by UB and weights by costs.

In the literature only Aggarwal, Aneja and Nair [1] present an exclusion test:

Algorithm 9.1 Exclusion Test of Aggarwal, Aneja and Nair

while change = true **do**
  change: = false
  for all $e \in E$ **do**
    compute a minimal spanning tree T_e such that $w(T_e) = \min(\sum_{f \in T} w_f | T \in \mathcal{T}, e \in T)$.
    if $w(T_e) > W$ **then**
      delete e from E.
      change:=true
    end if
  end for
  for all $e \in E$ **do**
    Compute $c(T_e) = \min(\sum_{f \in T} c_f | T \in \mathcal{T}, e \in T)$
    if $c(T_{kl}) \geq UB$ **then**
      delete e from E.
      change:=true
    end if
  end for
end while

The proposed algorithm is not quite efficient with respect to the run time by the computation of the minimal spanning tree for a fixed edge: Since the minimal spanning tree with no fixed edge and the minimal spanning tree with one fixed edge differ only in one edge exchange, we show a possible way to implement the idea of Algorithm 9.1 efficiently. The approach is to take a weight minimal spanning tree T and consider for each edge $e \in E \setminus T$ the cycle $C(T, e)$. We take from this cycle without the edge e the edge f with largest weight. The tree $T \setminus \{f\} \cup \{e\}$ is a minimal tree for the fixed edge e, and if $w(T) - w_f + w_e > W$, the edge e can be excluded from the tree. Analogously, we can do this for a cost minimal spanning tree and a known upper bound UB on OPT. This improvement has the advantage that the complete spanning tree need not be recalculated in each step.

Algorithm 9.2 Refinement of Algorithm 9.1

Compute
$T_1 := \arg\min\{\sum_{e \in T} w_e | T \in \mathcal{T}\}$
$T_2 := \arg\min\{\sum_{e \in T} c_e | T \in \mathcal{T}\}$
repeat
  for all $e \in E$ **do**
    compute $C(T_1, e)$
    $f := \arg\min\{c_f | f \in C(T_1, e) \setminus \{e\}\}$
    if $w(T_1) - w_f + w_e > W$ **then**
      $E := E \setminus \{e\}$
      if $e \in T_2$ **then**
        perform a minimal cost T_2-exchange $[e, f]$ with $f \in E$
      end if
    end if
    compute $C(T_2, e)$
    $f := \arg\min\{c_f | f \in C(T_2, e) \setminus \{e\}\}$
    if $c(T_2) - c_e + c_f > UB$ **then**
      $E := E \setminus \{e\}$
      if $e \in T_1$ **then**
        perform a minimal weight T_1-exchange $[e, f]$ with $f \in E$
      end if
    end if
  end for
until E was not changed in the last iteration

10 Dependence of Costs and Weights

We focus here on the case that the weights depend on the costs: Let us consider a function $f : \mathbb{Q} \longrightarrow \mathbb{Q}$ such that $f(c_e) = w_e$ for each $e \in E$. In this chapter we deal only with simple functions. If such a dependence exists, we may be able to make some statements concerning the properties of an optimal solution.

10.1 Monotony

Theorem 10.1
Let the WCMST be feasible and a function $f : \mathbb{Q} \longrightarrow \mathbb{Q}$ with $f(c_e) = w_e$ is given. We define for a tree T $c_{max} := \max_{e \in T} c_e$ and $c_{min} := \min_{e \in T} c_e$.

1. *Let f be monotonically increasing.*
   a) *If $f(c_{max}) \leq \frac{W}{n-1}$, then T is feasible.*
   b) *If $f(c_{min}) > \frac{W}{n-1}$, then T is infeasible.*
2. *Let f be monotonically decreasing.*
   a) *If $f(c_{min}) \leq \frac{W}{n-1}$, then T is feasible.*
   b) *If $f(c_{max}) > \frac{W}{n-1}$, then T is infeasible.*

Proof

1. a) $w(T) = \sum_{e \in T} w_e = \sum_{e \in T} f(c_e) \leq (n-1) f(c_{max}) \leq (n-1)\frac{W}{n-1} = W$
   b) $w(T) = \sum_{e \in T} w_e = \sum_{e \in T} f(c_e) \geq (n-1) f(c_{min}) > (n-1)\frac{W}{n-1} = W$
2. a) $w(T) = \sum_{e \in T} w_e = \sum_{e \in T} f(c_e) \leq (n-1) f(c_{min}) \leq (n-1)\frac{W}{n-1} = W$
   b) $w(T) = \sum_{e \in T} w_e = \sum_{e \in T} f(c_e) \geq (n-1) f(c_{max}) > (n-1)\frac{W}{n-1} = W$

□

10.2 Linear Functions

Theorem 10.2
Let the function $f : \mathbb{Q} \longrightarrow \mathbb{Q}$ be linear such that $f(c_e) = ac_e + b = w_e$ for each $e \in E$. Let the WCMST be feasible.

1. *If $a \geq 0$, then an optimal tree $T^\star$ for $\min\{c(T)|T \in \mathcal{T}\}$ is optimal for the WCMST.*
2. *If $a < 0$, then an optimal tree $\hat{T}$ for $\max\{c(T)|T \in \mathcal{T}\}$ is feasible for the WCMST.*
3. *If $a < 0$, the tree $\hat{T}$ which solves $\max\{w(T)|w(T) \leq W,\ T \in \mathcal{T}\}$ is optimal for the WCMST.*

Proof

1. Let $\bar{T}$ be a feasible solution. Then

$$\begin{aligned} w(T^\star) &= \sum_{e \in T^\star} w_e = \sum_{e \in T^\star} (ac_e + b) = a \sum_{e \in T^\star} c_e + (n-1)b \\ &\leq a \sum_{e \in \bar{T}} c_e + (n-1)b = \sum_{e \in \bar{T}} (ac_e + b) = \sum_{e \in \bar{T}} w_e = w(\bar{T}) \leq W. \end{aligned}$$

   So the optimal solution for the unconstrained problem is feasible for the WCMST and therefore optimal.

2. Let $\bar{T}$ be a feasible solution. Then

$$\begin{aligned} w(\hat{T}) &= \sum_{e \in \hat{T}} w_e = \sum_{e \in \hat{T}} (ac_e + b) = a \sum_{e \in \hat{T}} c_e + (n-1)b \\ &\leq a \sum_{e \in \bar{T}} c_e + (n-1)b = \sum_{e \in \bar{T}} (ac_e + b) = \sum_{e \in \bar{T}} w_e = w(\bar{T}) \leq W. \end{aligned}$$

   So the optimal solution for the unconstrained maximization problem is feasible for the WCMST. (Notice that the optimal solution for the unconstrained maximal spanning tree problem can be found analogously to the minimal spanning tree problem by taking in the algorithm of Prim always a possible edge with greatest costs.)

3. By definition the tree $\hat{T}$ is feasible. Let us consider a tree T with $w(T) < w(\hat{T})$:

$$\begin{aligned} c(\hat{T}) &= \sum_{e \in \hat{T}} c_e = \sum_{e \in \hat{T}} \frac{1}{a} w_e - \frac{b}{a} = \frac{1}{a} w(\hat{T}) - \frac{b}{a} \\ &< \frac{1}{a} w(T) - \frac{b}{a} = \sum_{e \in T} \frac{1}{a} w_e - \frac{b}{a} = \sum_{e \in T} c_e = c(T) \end{aligned}$$

   So $\hat{T}$ is an optimal solution.

□

10.3 Proportional / Inverse Proportional Costs and Weights

Let c_e and w_e be proportional for all $e \in E$ i.e., $\frac{c_e}{w_e} = a$ for all $e \in E$.

Theorem 10.3
If $\frac{c_e}{w_e} = a$ for all $e \in E$ and the WCMST is feasible, then the weight-constraint can be ignored.

Proof
Since the WCMST is feasible a feasible tree $\bar{T}$ exists. Let $T^\star$ be the optimal solution for $\min\{c(T)|T \in \mathcal{T}\}$. Then

$$\begin{aligned} w(T^\star) &= \sum_{e \in T^\star} w_e = \sum_{e \in T^\star} \frac{1}{a} c_e = \frac{1}{a} \sum_{e \in T^\star} c_e \\ &\leq \frac{1}{a} \sum_{e \in \bar{T}} c_e = \sum_{e \in \bar{T}} \frac{1}{a} c_e = \sum_{e \in \bar{T}} w_e = w(\bar{T}) \leq W \end{aligned}$$

Since $c(T^\star) \leq c(\bar{T})$ the claim holds.

□

Let c_e and w_e be inverse proportional for all $e \in E$ i.e., $c_e w_e = a$ for all $e \in E$.

Theorem 10.4
If $c_e w_e = a$ for all $e \in E$ and the WCMST is feasible, then an optimal solution is greater than $a\frac{(n-1)^2}{W}$.

Proof
At first we show that for $a, b > 0$ the term $\frac{1}{a} + \frac{1}{b} \geq 2\frac{1}{\frac{a+b}{2}}$ holds: We now that

$$(a-b)^2 \geq 0$$

which can reformulated to

$$a^2 - 2ab + b^2 \geq 0.$$

If we add on both sides $2ab$, we get

$$a^2 + b^2 \geq 2ab$$

which can be reformulated to

$$\frac{a^2 + b^2}{ab} \geq 2$$

and further to

$$\frac{a^2 + 2ab + b^2}{ab} \geq 4.$$

By using the binomial theorem

$$\frac{(a+b)^2}{ab} \geq 4$$

holds. Since $a > 0$ and $b > 0$ we can conclude that

$$\frac{(a+b)}{ab} \geq \frac{4}{a+b}$$

which leads finally to the desired

$$\frac{1}{a} + \frac{1}{b} \geq 2\frac{1}{\frac{a+b}{2}}.$$

For $a_1, \ldots, a_K > 0$ we have

$$\sum_{k=1}^{K} \frac{1}{a_k} \geq K\frac{1}{\frac{\sum_{k=1}^{K} a_k}{K}}.$$

For a feasible solution of our problem we have:

$$\sum_{e\in T} c_e = \sum_{e\in T} \frac{a}{w_e} = a\sum_{e\in T} \frac{1}{w_e} \geq a(n-1)\frac{1}{\frac{\sum_{e\in T} w_e}{n-1}} = a\frac{(n-1)^2}{\sum\limits_{e\in T} w_e} \geq a\frac{(n-1)^2}{W}$$

□

Remark

In the previous theorem we see that the costs are minimal if each edge in the tree has costs $a\frac{n-1}{W}$. If we search for a solution with $w(T) = W$, we have:

$$\begin{aligned} \min & \sum_{e\in T} c_e \\ s.t. & \sum_{e\in T} \frac{a}{c_e} = W \\ & T \in \mathcal{T} \end{aligned}$$

So we can alternatively search for a tree with

$$\begin{aligned} \min & \sum_{e\in T} |c_e - a\frac{n-1}{W}| \\ s.t. & \sum_{e\in T} \frac{a}{c_e} = W \\ & T \in \mathcal{T} \end{aligned}$$

Since very different functions can appear the topic of a dependence between costs and weights might be interesting for further research.

11 Exact Algorithms

In the literature only a few ideas to solve the WCMST exactly can be found. We can characterize these exact algorithms in branch and bound schemes which splits the problem into several problems by fixing and excluding some edges and other exact solutions. We explain three branch and bound procedures found in the publication of Aggarwal, Aneja and Nair [1], the article of Shogan [37] and the paper of Yamada, Watanabe and Kataoka [39] and give a new own branch and bound algorithm. Additionally we state two alternative ways to solve our problem: the idea of Hong, Chung and Park [27] using the matrix tree theorem is more algebraically orientated and the idea of solving the WCMST by a ranking algorithm.

11.1 Branch and Bound Algorithms

Since branch and bound is a very popular method to solve various optimization problems, using a branch and bound method for solving the WCMST suggests itself. The four following branch and bound schemes works with the same idea: In each branching we restrict our problem by fixing and forbidding edges i.e., we introduce the disjoint sets $A, B \subset E$. A spanning tree must contain all edges of A, i.e. $x_e = 1$ for all $e \in A$. For all $e \in B$ the edge e is not in the spanning tree, i.e. $x_e = 0$.

Problem 18 (P_{AB})

$$C^\star_{AB} := \min \sum_{e \in E} c_e x_e$$
$$s.t. \sum_{e \in E} w_e x_e \leq W$$
$$x \in \mathcal{T}_{AB}$$

where $\mathcal{T}_{AB} := \mathcal{T} \cap \{x | x_e = 1,\ \forall e \in A\} \cap \{x | x_e = 0,\ \forall e \in B\} = \{T \in \mathcal{T} | A \subset T, B \cap T = \emptyset\}$.

Since $\mathcal{T}_{AB} \subset \mathcal{T}$ this (P_{AB}) is a more restrictive problem than Problem 1. And obviously

$$OPT \leq C^\star_{AB}$$

holds. The difference between the algorithms is how to choose A and B and how to handle P_{AB}. The first and second scheme use the Lagrangian relaxation and compute in each iteration a large set of spanning trees under some cost function $c_e + \mu w_e$ for all $e \in E$. The third algorithm updates only an existing tree be edge-exchanges chosen by a clever exchange rule. The last branch and bound algorithm describes a more general procedure.

11.1.1 The Algorithm of Aggarwal, Aneja and Nair [1]

The branch and bound scheme of Aggarwal, Aneja and Nair, which is the oldest idea to solve the WCMST, works in the following way: By computing the convex hull we get some best trees $T^+, T^- \in \mathcal{T}$ with $w(T^+) > W$ and $w(T^-) \leq W$ and $c(T^+) < c(T^-)$. Also we have an upper bound $UB = c(T^-)$ and a lower bound on the intersection point of the line $w(T) = W$ and the line $(c(T^+), w(T^+))((c(T^-), w(T^-))$. An efficient solution of the bicriterial problem which is optimal for the WCMST lies in the triangle uvz (see Figure 11.1) where $u = (c(T^-), w(T^-)), z = (\frac{c(T^-)-c(T^+)}{w(T^-)-w(T^+)}W - c(T^+) + \frac{c(T^-)-c(T^+)}{w(T^-)-w(T^+)}w(T^+), W)$ and $v = (c(T^-), W)$.

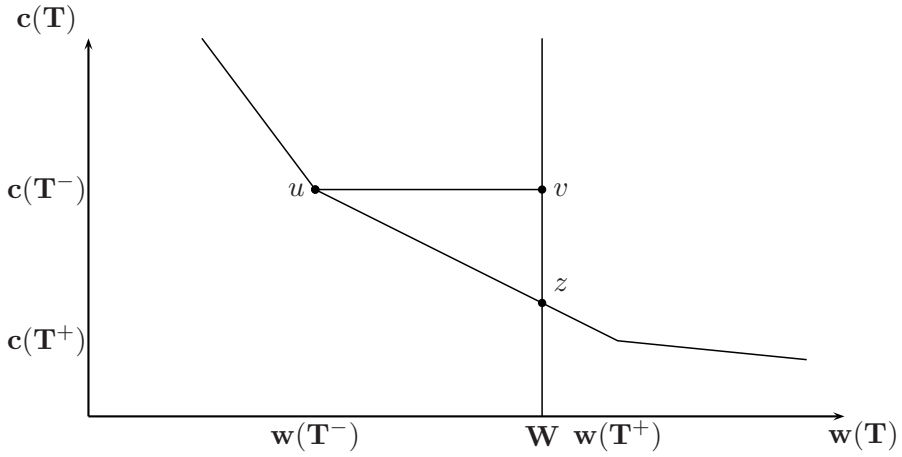


Figure 11.1: Algorithm of Aggarwal, Aneja and Nair

The idea of Aggarwal, Aneja and Nair is to start from the known upper bound on the convex hull and update by using a branch and bound scheme the triangle uvz until an optimal solution is found. Let $T^- = \{e_1, \ldots, e_{n-1}\}$. Therefore, we have $x_{e_i} = 1$ for $i < n-1$ and $x_{e_i} = 0$ for $i \geq n$.

Branching

We branch from the tree T^- by partitioning $\mathcal{T} \backslash \{T^-\}$ in $n-1$ non-empty and disjoint sets $\mathcal{T}_{A_k B_k}$ ($k \in \{1, \ldots, n-1\}$) such that

$$\mathcal{T}_{A_k B_k} = \{T \in \mathcal{T} |\ \{e_1, \ldots, e_{k-1}\} =: A_k \subset T, \{e_k\} =: B_k \cap T = \emptyset\}.$$

For $\mathcal{T}_{A_k B_k}$ we search also for trees T_k^+ and T_k^- on the convex hull of $\mathcal{T}_{A_k B_k}$ with $w(T_k^-) \leq W$, $w(T_k^+) > W$ and $c(T_k^+) < c(T_k^-)$. If the facet defined by this T^+ and T^- of this problem lies outside of uvz or does not lead to a better solution, we will not consider this set further. Otherwise we branch $\mathcal{T}_{A_k B_k}$ with T_k^-. In general, if we branch from a tree $T_s^- \in \mathcal{T}_{AB}^s$ where $T_s^- \setminus A = \{e_1, \ldots, e_K\}$, we have at level $s+1$ a partition

$$\mathcal{T}_{A_k B_k}^{s+1} := \{T \in \mathcal{T}_{AB}^s | A \cup \{e_1, \ldots, e_{k-1}\} =: A_k \subset T, B^s \cup \{e_k\} =: B_k \cap T = \emptyset\}$$

for all $k \in \{1, \ldots, K\}$. (Please notice that in [1] the used numbering of the x_i is wrong.) So we can formulate the algorithm.

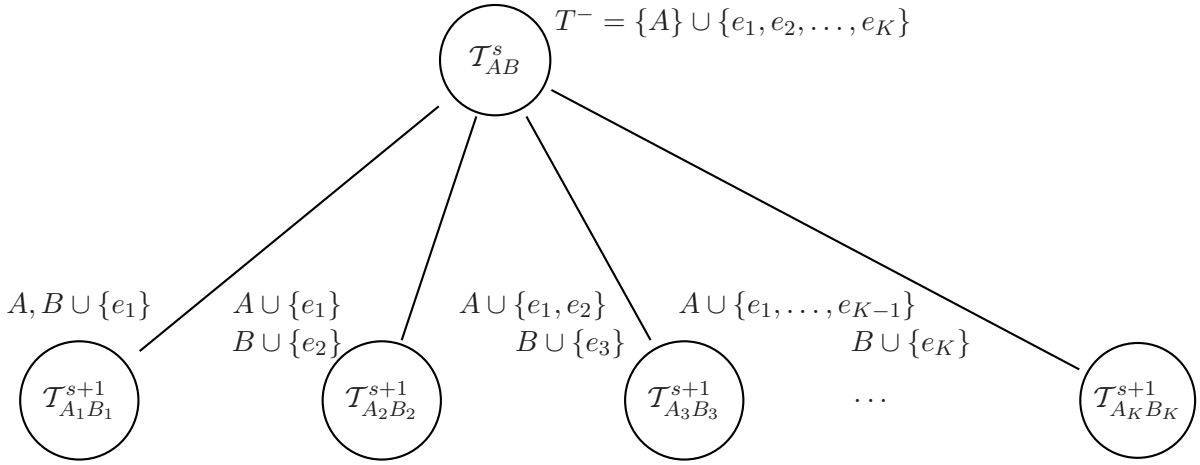


Figure 11.2: Branching in an arbitrary step

Procedure 11.1 Branch- and bound-Scheme of Aggarwal, Aneja and Nair: Efficient Frontier

run Procedure 7.1 with $\mathcal{T}_{A_k^s B_k^s}$
if $C_1 \geq UB$ or $W_2 > W$ **then**
  return to Main Case 1 {$\mathcal{T}_{A_k^s B_k^s}$ can be ignored}
end if
if $W_1 \leq W$ **then**
  define $T_k^- := T_{k1}$
  return to Main Case 2
else
  Case 3
end if
repeat
  {with (T_{k1}, T_{k3}) and (T_{k3}, T_{k2})}
  $\mu := (w(T_{k1}) - w(T_{k2}))/(c(T_{k2}) - c(T_{k1}))$
  Compute a minimal spanning tree T_{k3} under the cost $c_e + \mu w_e$ from the set $\mathcal{T}_{A_k^s B_k^s}$.
until no further efficient solution can be found
Choose two efficient trees T_k^+, T_k^- next to $w(T) = W$ such that $w(T_k^+) > W$ and $w(T_k^-) < W$
return to Main

Algorithm 11.2 Branch- and bound-Scheme of Aggarwal, Aneja and Nair: Main

while the set of eligible branches is not empty **do**
  repeat
    For the kth partition of $\mathcal{T}_{A_k^s B_k^s}$ goto Procedure 11.1
    if return under 1. case **then**
5:    the set can be ignored
    else if return under 2. case **then**
      the tree T_k^- is the best feasible tree for this branching
      $UB := c(T_k^-)$
    else {return under 3. case}
10:    **if** $\{(c,w)|(c,w) \geq \mu(c(T_k^+), w(T_k^+)) + (1-\mu)(c(T_k^-), w(T_k^-)), 0 \leq \mu \leq 1\} \cap \{(c,w)|c \leq UB, w \leq W\} = \emptyset$ **then**
        this branching need not considered
      else if $w(T_k^-) = W$ **then**
        $UB = c(T_k^-)$
      else
15:      $TLB_k := c(z_k)$ where z_k describes the intersection point of (T_k^+, T_k^-) and $w = W$.
        $T(TLB_k) := T_k^-$
      end if
      if $c(T_k^-) < UB$ **then**
        $UB := c(T_k^-)$
20:    **end if**
    end if
    $k := k + 1$
  until all partitions of the level are considered
  if $TLB_k \geq UB$ **then**
25:    prune $T(TLB_k)$ for all branchings
  end if
  $LB := \min_k TLB_k = TLB(\hat{z}_k)$
  if $LB \geq UB$ **then**
    $T^\star = T(UB)$ respectively $OPT = UB$
30:    STOPP
  end if
  $k := 1$
end while

Theorem 11.1

1. *The algorithm terminates.*
2. *The algorithm is valid.*

Proof

1. Since this is an ordinary branch and bound scheme with breadth-first search the algorithm terminates after a finite number of steps.
2. The algorithm is valid since in each step we update the triangle $(u-v-z)$. Since $(u-z)$ is a face of the convex hull of the efficient solution an optimal solution cannot lie below this face. The triangle in each branching level l can be constructed in this way:

$$\begin{aligned} z^l :=& \hat{z}_k = (TLB(\hat{z}_k), W) = (LB, W) \\ v^l :=& (UB, W) \\ u^l :=& \{(c,w) | \min_k [w | \hat{c} = UB, (c,w) = \lambda(c(T_k^+), w(T_k^+)) + (1-\lambda)(c(T_k^-), w(T_k^-)) \lambda \in [0,1]] \\ & \text{for all eligible branchings } \} \end{aligned}$$

   The facet (u^l, z^l) either coincides with (u^{l-1}, z^{l-1}) or lies above this. (v^l, z^l) lies beneath from (v^{l-1}, z^{l-1}) if UB changes. In general, we reduce in every branching by fixing and excluding edges the set of free eligible edges. Then $\Delta(u^l, v^l, z^l) \subset \Delta(u^{l-1}, v^{l-1}, z^{l-1})$. In a finite number of steps $TLB(\hat{z}_k) \geq UB$ and the triangle reduces to a line and the optimal solution was found.

□

Remark
If we stop before the algorithm is finished we get a lower and upper bound for the value of the solution.

Comment
The main disadvantage of this algorithm is that for every problem all supported trees have to be computed in the Procedure 11.1, since the quotient $\frac{w(T_1)-w(T_2)}{c(T_2)-c(T_1)}$ delivers for the best known (feasible) upper bound tree T_1 and the best known (infeasible) lower bound tree T_2 in general no μ such that a tree T which is optimal for $c_e + \mu w_e$ for all $e \in E$ satisfies $c(T_1) \leq c(T) \leq c(T_2)$ and $w(T_1) \geq w(T) \geq w(T_2)$.

11.1.2 The Algorithm of Shogan [37]

In [37] Shogan introduces an algorithm for the resource-constrained spanning tree problem (see equation (3.6)) which can be extended to a resource-constrained spanning tree problem with flow requirements. Here we show only the case $L = 1$. All other branch and bound algorithms mentioned in this thesis run only for the case $L = 1$ and cannot be extended to the resource-constrained minimal spanning tree problem.

Idea

The approach of Shogan's algorithm is a branch and bound scheme combined with the Lagrangian relaxation. The difference to the previous algorithm is a different branching rule. We use the following Notation: Let C_{cu} be the objective value of the current solution. If no solution is known let $C_{cu} = \infty$. For this algorithm we need the Lagrangian relaxation of Problem 18:

Problem 19 ($\mathbf{P}^{\mu}_{AB}$)

$$C^{\star}_{AB}(\mu) := \min \sum_{k=1}^{m} c_k x_k + \mu(\sum_{k=1}^{m} w_k x_k)$$
$$s.t.\ x \in \mathcal{T}_{AB}$$

We know

$$C^{\star}_{AB} \geq C^{\star}_{AB}(\mu) - \mu W.$$

For sake of completeness we formulate the Lagrangian dual for this problem.

Problem 20 ($\mathbf{D}_{AB}$)

$$\max C^{\star}_{AB}(\mu) - \mu W$$
$$\mu \ \geq 0$$

The solution of every P^{μ}_{AB} results in at least one of the following possibilities:

1. A better lower bound for $C^{\star}_{AB}$ can be found.
2. We can reduce the objective value of the current solution.
3. The optimal solution of P_{AB} is found.

Let $x^{\star}_{AB}(\mu)$ be the optimal solution of (P^{μ}_{AB}) and $s^{\star}_{AB}(\mu) \in \mathbb{R}$ be the slack variable defined in this manner:

$$s^{\star}_{AB}(\mu) := W - \sum_{k=1}^{n} w_k x^{\star}_{AB}(\mu)_k.$$

Algorithm

We can describe all branchings by a vector $[A, B, \mu, C^{\star}_{AB}(\mu), s^{\star}_{AB}(\mu)]$ where μ is not necessary the optimal solution of the dual problem but provides the best known lower bound for $C^{\star}_{AB}$ and store this problem on a stack.

Algorithm 11.3 Branch- and Bound-Scheme of Shogan

Require: $[\emptyset, \emptyset, 0, c^{\star}_{\emptyset\emptyset}(0), s^{\star}_{\emptyset\emptyset}(0)]$, $\epsilon > 0$

repeat
Choose a problem from the stack with largest $C^{\star}_{AB}(\mu)$
Out:=false
while $t < 4n$ or $C^{\star}_{AB}(\mu^I) - \mu W$ does increase by 0.1 percent or OUT = false **do**
$\mu^t := \max\{0, \mu^{t-1} - \theta_t s^{\star}_{AB}(\mu^{t-1})\}$
Solve $(P^{\mu^t}_{AB})$
if $(1+\epsilon)(C^{\star}_{AB}(\mu^k) - \mu W) \geq C_{cu}$ **then**
delete Problem
OUT:=true
end if
if $s^{\star}_{AB}(\mu^t) \geq 0$ **then**
if $cx^{\star}_{AB}(\mu^t) < C_{cu}$ **then**
$cu := x^{\star}_{AB}$
end if
if $\mu^t s^{\star}_{AB}(\mu^t) \leq \epsilon(C^{\star}_{AB}(\mu^t) - \mu^t W)$ **then**
delete problem
OUT:=true
end if
end if
if $C^{\star}_{AB}(\mu^t) - \mu W > C^{\star}_{AB}(\mu^I) - \mu^I W$ **then**
$I := t$
end if
t:=t+1
end while
if OUT = false **then**
Take $e^{\star} \in J_{AB}$ with minimal $c_e + \mu^I w_e$.
if $\sum_{e \in \{A \cup \{e^{\star}\}\}} w_e \leq W$ and $|A \cup \{e^{\star}\}| < n-1$ **then**
add $[A \cup \{e^{\star}\}, B, \mu^I, C^{\star}_{AB}(\mu^I), s^{\star}_{AB}(\mu^I)]$ to the stack
end if
if $(P^{\mu^I}_{A,B\cup\{j^{\star}\}})$ is feasible **then**
add $[A, B \cup \{j^{\star}\}, \mu^I, c^{\star}_{A,B\cup\{j^{\star}\}}(\mu^I), s^{\star}_{A,B\cup\{j^{\star}\}}(\mu^I)]$ to the stack
end if
end if
until the stack is empty
The current solution is optimal

Explanations and Remarks

General: The input corresponds to a minimal spanning tree problem without constraints. If $C^\star_{AB}(\mu)$ is greater than the current solution C_{cu}, this problem can be pruned. We try to solve the Lagrangian dual (D_{AB}) by a subgradient approach. The subgradient for the objective function for (D_{AB}) on μ is $-s^\star_{AB}(\mu)$. The iteration of the subgradient method is done in the while-loop:

Line 5: The difficult part of the algorithm is the choice of μ. Proposed is the following approach: For $\theta_l \in \mathbb{R}^+$ sufficiently small

$$\mu^t := \max\{0, \mu^{t-1} - \theta_t s^\star_{AB}(\mu^{t-1})\}$$

is closer to the optimal solution of (D_{AB}) than μ^{t-1} i.e.,

$$C^\star_{AB}(\mu^t) - \mu^t W \geq C^\star_{AB}(mu^{t-1}) - \mu^{t-1} W.$$

The most difficult part is the choice of θ_t. Proposed is $\theta_t = 1$ for all t.

Line 6: The problem P^μ_{AB} can be solved by any minimal spanning tree algorithm.

Line 7: If the value of the relaxation is greater than the objective value of the incumbent solution, the sets A and B cannot lead to an optimal solution since every solution of P_{AB} would be greater than $C^\star_{AB}(\mu^t) - \mu W$ and we could prune the subproblem. The ϵ denotes an approximation tolerance ($\epsilon = 0$ is possible). We can stop our subgradient method and therefore set OUT:=false.

Line 11: We check if $x^\star_{AB}(\mu)$ is feasible for (P_{AB}). This is true if and only if $s^\star_{AB}(\mu) \geq 0$. Then all conditions are fulfilled. Otherwise the problem has to be considered further.

Line 12: It holds

$$C^\star_{AB}(\mu) - \mu W = cx^\star_{AB}(\mu) - \mu s^\star_{AB}(\mu) < C_{cu}.$$

Since $x_{AB}(\mu)$ is feasible, $x_{AB}(\mu)$ becomes the current solution if $cx^\star_{AB}(\mu) < C_{cu}$.

Line 15: Since $x_{AB}(\mu)$ is feasible and $C^\star_{AB}(\mu) - \mu W$ is a lower bound for $C^\star_{AB}$:

$$C^\star_{AB}(\mu) - \mu W = cx^\star_{AB}(\mu) - \mu s^\star_{AB}(\mu) \leq C^\star_{AB} \leq cx^\star_{AB}(\mu).$$

If $\mu s^\star_{AB} = 0$, then $c^\star_{AB}(\mu) = cx^\star_{AB}(\mu) = c^\star_{AB}$. It follows that μ and $x^\star_{AB}(\mu)$ are optimal for (D_{AB}) and (P_{AB}). If $\mu s^\star_{AB}(\mu) \leq \epsilon c^\star_{AB}(\mu)$, we could also accept $x^\star_{AB}(\mu)$ as an optimal solution.

If the problem is not fathomed, we must decide whether it is meaningful to continue the subgradient method (e.g. start another turn of the loop). Notice that the sequence

$\mu^0, \mu^1, \ldots, \mu^t$ is not necessarily monotonous. So we have to store the μ^I which delivers the greatest lower bound for $c^\star_{AB}$. We start the $(t+1)$st iteration until one of the following cases occurs:

1. $t > 4n$
2. In the last 5 iterations, $c^\star_{AB}(\mu^I)$ does not increase by 0.1 percent

Line 25: If the problem was not pruned yet, we have to branch further. In contrast to the previous algorithm we partition the problem only in two branches: Let $V(A)$ denote the set of vertices which are connected by the edges of A. Let $J_{AB} \in E$ with $J_{AB} \cup B = \emptyset$ denote the set of edges with one node in $V(A)$ and one node in $V \setminus V(A)$. If $A = \emptyset$ then J_{AB} is the set of all edges which has the node 1. Let $e^\star \in J_{AB}$ be the edge that minimizes $c_e + \mu^I w_e$. We partition $\mathcal{T}_{AB}$ into $\mathcal{T}_{A\cup\{e^\star\},B}$ and $\mathcal{T}_{A,B\cup\{e^\star\}}$.
By construction $A \cup \{e^\star\}$ is connected. The choice of $e^\star$ guarantees that $e^\star$ lies in the minimal spanning tree of $\mathrm{P}^{\mu_I}_{AB}$ and of $\mathrm{P}_{A\cup\{e^\star\},B}$ since in Prim's algorithm this edge will be added to the tree. Therefore $[A \cup \{j^\star\}, B, \mu^I, c^\star_{AB}(\mu^I), s^\star_{AB}(\mu^I)]$ is added to the stack while not one of the following cases holds:

1. $\sum_{e \in A\cup\{e^\star\}} w_e > W$. The problem is not feasible.
2. $|A \cup \{e^\star\}| = n - 1$ the solution is found.

For $\mathrm{P}_{A,B\cup\{e^\star\}}$ we have to check:

1. A lower bound of $C^\star_{A,B\cup\{e^\star\}}$ can be found by solving $\mathrm{P}^{\mu^I}_{A,B\cup\{e^\star\}}$.
2. We use this $C^\star_{A,B\cup\{e^\star\}}(\mu^I), y^\star_{A,B\cup\{e^\star\}}(\mu^I), s^\star_{A,B\cup\{e^\star\}}(\mu^I)$ and check whether the problem is feasible.
3. If the problem cannot be deleted, we add $[A, B\cup\{e_j\}, \mu^I, C^\star_{A,B\cup\{e^\star\}}(\mu^I), s^\star_{A,B\cup\{e^\star\}}(\mu^I)]$ to the stack.

11.1.3 The Algorithm of Ruzika and Henn

In this section we will give a new branch and bound scheme with a more sophisticated branching rule. Therefore, we need some further knowledge about the properties of the convex hull. Let us start with a tree which has minimal weight and perform $T-$exchanges $[e, f]$ such that

$$f := \arg\min\{\frac{c_f - c_e}{w_f - w_e} | c_f < c_e, f \notin T, e \in T, T \setminus \{e\} \cup \{f\} \in \mathcal{T}\}. \tag{11.1}$$

We call these edges e and f pivot edges, and this exchange pivot operation. We can solve this minimization problem in $O(nm)$ since the tree T has $n-1$ edges and $m-(n-1)$ edges are not contained in T. Therefore, at least $(n-1)(m-(n-1))$ exchanges are possible.
In the following we show that the sequence $T_1, \ldots, T_K$ of these exchanges where T_1 is a tree with minimal weights and T_K a tree with minimal costs decreases the costs in each step strictly

(e.g. $c(T_1) > c(T_2) > \cdots > c(T_K)$), each $(c(T_k), w(T_k))$ $k \in \{1, \ldots, K\}$ lies on the frontier of the convex hull and all extreme points of the convex hull are contained in this sequence. Moreover, we see that if we perform such pivot operations starting from an arbitrary tree on the border of the convex hull, the new tree is also a tree whose cost-weight-vector lies on a facet.
In Theorem 6.2 we have already seen that the set of supported trees is connected which implies that a sequence $T_1, \ldots, T_L$ exists such that T_i and $T_{i+1},\ i \in \{1, \ldots, L-1\}$, differ by one exchange and each T_i is a supported tree. For each facet of the convex hull exist a μ such that all trees whose weight-cost vector lie on this facet are optimal for $c_e + \mu w_e$ for all $e \in E$ (see Theorem 7.3). We denote this set $\mathcal{O}_\mu$.

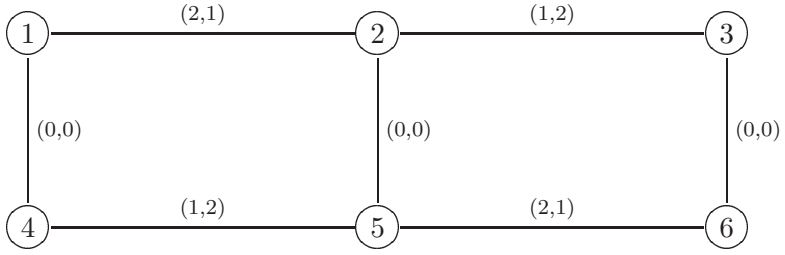
Theorem 11.2
If there exist two supported trees T_1 and T_K with $c(T_1) = c(T_K)$ and $w(T_1) = w(T_K)$ and their image is an extreme point of the convex hull then there exists a sequence $T_1, \ldots, T_K$ such that T_k and T_{k+1} are adjacent for all $k \in \{1, \ldots, K-1\}$ and $c(T_1) = c(T_2) = \cdots = c(T_K)$ and $w(T_1) = w(T_2) = \cdots = w(T_K)$.

Proof
We know for T_1 and T_2 that both trees are optimal for the costs $c_e + \mu_1 w_e$ and $c_e + \mu_2 w_e$ for all $e \in E$ where μ_1 and μ_2 corresponds to the two facets having the extreme point in common. For all $\mu \in (\mu_1, \mu_2)$ only the trees whose images correspond to the extreme point are optimal for $c_e + \mu w_e$. We know from the ranking algorithm of Katoh, Ibaraki, Mine [31] that the kth minimal cost spanning tree can be obtained by performing edge exchanges outgoing from one of the $k-1$th minimal cost spanning trees (for more information see Section 11.3). So if we search for all optimal trees for $c_e + \mu w_e$ we find by this idea all trees whose costs are $c(T_1)$ and whose weight is $w(T_1)$. Therefore, in the search also T_1 and T_2 appear. Both trees are obtained by sequences of optimal trees $T_0, \ldots, T^p = T_1$ and $T_0, \ldots, T^l = T_2$ where T_0 is the first found tree and the trees in the sequence are pairwise adjacent. If we combine both sequences, we have the desired result.

□

Unfortunately this result holds not for arbitrary trees on a facet of the convex hull. This can be seen in the next example.

Example 11.1



Obviously the trees

$$T_1 = \{\{1,4\},\{2,5\},\{3,6\},\{4,5\},\{5,6\}\}$$

and

$$T_2 = \{\{1,2\},\{1,4\},\{2,3\},\{2,5\},\{3,6\}\}$$

are optimal for $c_e + 1w_e$ and therefore supported. We have also that $c(T_1) = 3 = c(T_2)$ and $w(T_1) = 3 = w(T_2)$ and the information that T_1 and T_2 are not adjacent. As you can see there exists no tree which is adjacent to T_1 with costs 3 and weights 3 and Theorem 11.2 holds not for arbitrary trees on the convex hull.

If we perform an exchange $[e, f]$ from $T \in \mathcal{O}_\mu$ the tree $T \setminus \{e\} \cup \{f\}$ is element of $\mathcal{O}_\mu$ if $c_e + \mu w_e = c_f + \mu w_f$.
Let us consider such a set $\mathcal{O}_\mu$. We will show that if we start with an arbitrary $T_\mu^1 \in \mathcal{O}_\mu$ the edge-exchange rule (11.1) delivers a sequence $T_\mu^1, T_\mu^2, \ldots, T_\mu^K \in \mathcal{O}_\mu$ with $c(T_\mu^1) > c(T_\mu^2) > \cdots > c(T_\mu^K)$ and $c(T_\mu^K) \leq c(T)$ for all $T \in \mathcal{O}_\mu$. This T_μ^K is therefore an extreme point and optimal for other costs $c_e + \bar{\mu} w_e$ for all $e \in E$ with $c(T_\mu^K) \geq c(T)$ for all $T \in \mathcal{O}_{\bar{\mu}}$.

Lemma 11.3
Let T_1 and T_2 be in $\mathcal{O}_\mu$ with $c(T_1) = c(T_2)$. Let T_1 and T_2 differ only in one T-exchange. If a tree $T_3 \in \mathcal{O}_\mu$ exists which can be obtained by one $T-$exchange from T_2 and $c(T_3) < c(T_2)$, we can also construct a tree $T \in \mathcal{O}_\mu$ with $c(T) < c(T_1)$ by one $T-$exchange from T_1.

Proof
Since $T_1, T_2, T_3 \in \mathcal{O}_\mu$:

$$c(T_1) + \mu w(T_1) = c(T_2) + \mu w(T_2) = c(T_3) + \mu w(T_3)$$

From the definition we have for our trees T_1 and T_2

$$c(T_1) = c(T_2) > c(T_3) \text{ and } w(T_1) = w(T_2) < w(T_3).$$

We have $T_1 \setminus \{i\} \cup \{j\} = T_2$ and $T_2 \setminus \{g\} \cup \{h\} = T_3$.
Since $c(T_1) = c(T_2)$ and $T_1, T_2 \in \mathcal{O}_\mu$

$$c_i + \mu w_i = c_j + \mu w_j \tag{11.2}$$

$$\text{and } c_i = c_j,\ w_i = w_j \tag{11.3}$$

must hold. Since $T_3 \in \mathcal{O}_\mu$ and $c(T_2) > c(T_3)$ we have

$$c_g + \mu w_g = c_h + \mu w_h \tag{11.4}$$

$$\text{and } c_g > c_h,\ w_g < w_h \tag{11.5}$$

We have to make a case differentiation:

1. $T_1 \setminus \{g\} \cup \{h\}$ is a tree.
   Denote this tree with T_4. By (11.4) we see that $T_4 \in \mathcal{O}_\mu$ and by (11.5) that $c(T_4) = c(T_1) - c_g + c_h < c(T_1)$.

2. $T_1 \setminus \{g\} \cup \{h\}$ is no spanning tree
   Therefore

$$g \notin C(T_1, h) = C(T_1 \cap T_2 \cup \{i\}, h) \text{ and} \tag{11.6}$$
$$g \in C(T_2, h) = C(T_1 \cap T_2 \cup \{j\}, h) \tag{11.7}$$

   Both cycles differ only in i and j. We can conclude that

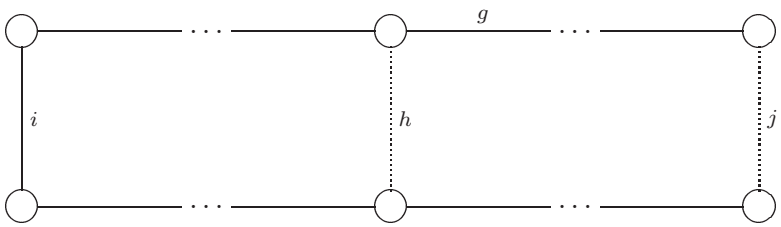
$$j, g \in C(T_2, h) = C(T_3, g). \tag{11.8}$$

   This result can be seen easily, since otherwise $g \in C(T_2 \setminus \{j\}, h) = C(T_1 \cap T_2, h)$ and $g \in C(T_1, h)$ which contradicts to the assumption of this case.

   Claim 1: $g \in C(T_1, j)$.
   A general result is that the for two cycles C_1, C_2 which have at least one edge in common ($C_1 \cap C_2 \neq \emptyset$), $C_1 \not\subseteq C_2$ and $C_2 \not\subseteq C_1$ the set $(C_1 \cup C_2) \setminus (C_1 \cap C_2)$ is also a cycle. We know $C(T_1, j) = C(T_2, i)$ and $g, j \in C(T_2, h) = C(T_1 \setminus \{i\} \cup \{j\}, h)$. Let us assume that $g \notin C(T_1, j)$. Then for $C(T_1, j)$ and $C(T_1 \setminus \{i\} \cup \{j\}, h)$ the conditions for the general result are satisfied and we can construct a cycle $D := (C(T_1 \setminus \{i\} \cup \{j\}, h) \cup C(T_1, j)) \setminus (C(T_1 \setminus \{i\} \cup \{j\}, h) \cap C(T_1, j))$ with $g, h \in D$ and $j \notin D$. So $D \subset (T_1 \cup \{h\})$ and $T_1 \setminus \{g\} \cup \{h\}$ is a tree. This is a contradiction to our assumption and thus $g \in C(T_1, j)$.
   From the claim we can conclude that $T_5 := T_1 \setminus \{g\} \cup \{j\}$ is a tree.

   Claim 2: $i \in C(T_1, h)$
   We use again the argument of the previous claim: In this case we know that $i, j \in C(T_1, j)$ and $j \in C(T_2, h) = C(T_1 \setminus \{i\} \cup \{j\}, h)$ also $g \notin C(T_1, h)$ but $g \in C(T_2, h)$ and $i \notin C(T_2, h)$. So we can get a cycle by $D := (C(T_1, j) \cup C(T_1 \setminus \{i\} \cup \{j\}, h)) \setminus (C(T_1, j) \cap C(T_1 \setminus \{i\} \cup \{j\}, h))$ with $i, h \in D$. Then $i \in C(T_1, h)$.
   So $T_6 := T_1 \setminus \{i\} \cup \{h\}$ is a tree.

   By these claims our tree T_1 has the following structure:



   Claim 3: $c_j + \mu w_j \leq c_h + \mu w_h (= c_g + \mu w_g)$
   Let us consider otherwise ($c_j + \mu w_j > c_h + \mu w_h$) the tree $T := T_2 \setminus \{j\} \cup \{h\}$. (Since j, h

are elements of the cycle $C(T_2, h)$, T is a spanning tree.) Then

$$\begin{aligned} c(T) + \mu w(T) &= c(T_2) + \mu w(T_2) - (c_j + \mu w_j) + (c_h + \mu w_h) \\ &< c(T_2) + \mu w(T_2) \end{aligned}$$

which contradicts $T_2 \in \mathcal{O}_\mu$.

Claim 4: $c_j + \mu w_j \geq c_h + \mu w_h (= c_g + \mu w_g)$
We consider otherwise ($c_j + \mu w_j < c_g + \mu w_g$) the tree T_5.

$$\begin{aligned} c(T_5) + \mu w(T_5) &= c(T_1) + \mu w(T_1) - (c_g + \mu w_g) + (c_j + \mu w_j) \\ &< c(T_1) + \mu w(T_1) \end{aligned}$$

This contradicts the fact that $T_1 \in \mathcal{O}_\mu$. So we know that

$$c_h + \mu w_h = c_g + \mu w_g = c_i + \mu w_i = c_j + \mu w_j. \tag{11.9}$$

From this equation we can conclude that the trees T_5 and T_6 are contained in $\mathcal{O}_\mu$.

Claim 5: At least $c(T_5) < c(T_1)$ or $c(T_6) < c(T_1)$ hold.
We assume otherwise that $c(T_5) \geq c(T_1)$ and $c(T_6) \geq c(T_1)$. This means that

$$c(T_5) = c(T_1) - c_g + c_j \geq c(T_1)$$

with leads to

$$c_j \geq c_g \tag{11.10}$$

and

$$c(T_6) = c(T_1) - c_i + c_h \geq c(T_1)$$

which leads to

$$c_h \geq c_i \tag{11.11}$$

We combine (11.3), (11.10) and (11.11) and get

$$c_h \geq c_i = c_j \geq c_g$$

This is a contradiction to the fact that c_g is strictly greater than c_h which we know from (11.5). So Claim 5 is valid and at least one of the two trees has lower costs than T_1.

In both cases a tree can be constructed by one edge exchange from T_1 which is optimal for $c_e + \mu w_e$ and has lower costs than T_1.

□

The question is now which relation between the obtained trees exists.

Lemma 11.4
In each case the obtained trees - proposed they exist - are adjacent.

Proof
We consider each of the three possible pairs separately.

1. T_3 and T_4
   From definition we know that $T_3 = T_2 \setminus \{g\} \cup \{h\} = T_1 \setminus \{g, i\} \cup \{h, j\}$ and $T_4 = T_1 \setminus \{g\} \cup \{h\}$.
   Obviously
   $$T_3 = T_4 \setminus \{i\} \cup \{j\}.$$

2. T_3 and T_5
   We know that $T_5 := T_1 \setminus \{g\} \cup \{j\}$ and so
   $$T_3 = T_5 \setminus \{i\} \cup \{h\}.$$

3. T_3 and T_6
   We know that $T_6 := T_1 \setminus \{i\} \cup \{h\}$ and so
   $$T_3 = T_6 \setminus \{g\} \cup \{j\}.$$

In all three cases the obtained trees - proposed they exist - are adjacent.

□

Corollary 11.5
Let T_1 and T_K in $\mathcal{O}_\mu$ with $c(T_1) = c(T_K)$ and $w(T_1) = w(T_K)$ and their image is an extreme point of the convex hull. If there exists a tree $\hat{T}_K \in \mathcal{O}_\mu$ which can be obtained by an edge-exchange from T_K and $c(\hat{T}_K) < c(T_K)$ then there exists a tree $\hat{T}_1 \in \mathcal{O}_\mu$ which can be obtained by an edge exchange starting from T_1 with $c(\hat{T}_1) < c(T_1)$.

Proof
According to Theorem 11.2 a sequence (obviously in $\mathcal{O}_\mu$) $T_1, \dots, T_K$ exists where T_k and T_{k+1} differ by one edge-exchange for $k \in \{1, \dots, K-1\}$. Since we can construct a tree $\hat{T}_K \in \mathcal{O}_\mu$ with $c(T_K) > c(\hat{T}_K)$ by an edge exchange from T_K we can also construct a tree $\hat{T}_{K-1} \in \mathcal{O}_\mu$ with $c(\hat{T}_{K-1}) < c(T_{K+1})$ by an edge exchange from T_{K+1}. We use this property iteratively and can construct a tree $\hat{T}_1 \in \mathcal{O}_\mu$ with $c(\hat{T}_1) < c(T_1)$.

□

Lemma 11.6
Let T_1 and T_2 be two adjacent trees in $\mathcal{O}_\mu$ with $c(T_2) > c(T_1)$ and we can obtain a tree $T_3 \in \mathcal{O}_\mu$ from T_2 by one edge exchange such that $c(T_2) > c(T_3)$ and $c(T_1) > c(T_3)$. Then we can construct a tree $T \in \mathcal{O}_\mu$ by an exchange outgoing from T_1 with $c(T_3) \geq c(T)$.

Proof

This proof works very close to the proof of Lemma 11.3: We use the same denotation: $T_2 := T_1 \setminus \{i\} \cup \{j\}$ and $T_3 := T_2 \setminus \{g\} \cup \{h\} = T_1 \setminus \{g, i\} \cup \{h, j\}$. In this case it must hold that

$$c(T_1) < c(T_2) = c(T_1) - c_i + c_j$$

which leads to

$$c_j > c_i \tag{11.12}$$

and

$$c(T_2) > c(T_3) = c(T_2) - c_g + c_h$$

which leads to

$$c_g > c_h. \tag{11.13}$$

Now we can proceed the same case distinction:

1. $T_1 \setminus \{g\} \cup \{h\}$ is a tree.
   Define $T_4 := T_1 \setminus \{g\} \cup \{h\}$ then we know that

   $$c(T_4) = c(T_1) - c_g - c_h < c(T_1).$$

2. $T_1 \setminus \{g\} \cup \{h\}$ is no tree.
   Analogously to Lemma 11.3 we construct trees $T_5 := T_1 \setminus \{g\} \cup \{j\}$ and $T_6 := T_1 \setminus \{i\} \cup \{h\}$ (The proof that both subgraphs are trees and elements of $\mathcal{O}_\mu$ works analogously to the proof in Lemma 11.3). Let us assume that both trees have costs greater or equal than T_1:

   $$c(T_5) = c(T_1) - c_g + c_j \geq c(T_1)$$

   which leads to

   $$c_j \geq c_g \tag{11.14}$$

   and

   $$c(T_6) = c(T_1) - c_i + c_h \geq c(T_1).$$

   This is valid if and only if

   $$c_h \geq c_i \tag{11.15}$$

   Now we sum up the inequalities (11.14), (11.15) and get

   $$c_h + c_j \geq c_g + c_i \tag{11.16}$$

   From the assumption that $c(T_3)$ is less than $c(T_1)$ we know that

   $$c(T_3) = c(T_1) - (c_g + c_i) + (c_j + c_h) < c(T_1)$$

which is equivalent to

$$c_j + c_h < c_g + c_i \tag{11.17}$$

which contradicts (11.16). At least one of the trees T_5, T_6 must have costs less than $c(T_1)$.

We can notice again that the tree T_3 is also adjacent to T_4, respectively T_5 and T_6.

□

Lemma 11.7
Let $T_1, T_K \in \mathcal{O}_\mu$ be given with $c(T_1) > c(T_K)$ and let $c(T_K) < c(T)$ for all $T \in \mathcal{O}_\mu$. Then, there exists a sequence $(T_1, \dots, T_K)$ in $\mathcal{O}_\mu$ with T_k and T_{k+1} being adjacent and satisfying $c(T_k) > c(T_{k+1})$ for all $k \in \{1, \dots, K-1\}$.

Proof
It is clear that there exists a sequence $(T_1, \dots, T_K)$ in $\mathcal{O}_\mu$ with T_k and T_{k+1} being adjacent since all trees in $\mathcal{O}_\mu$ are connected. If $c(T_k) < c(T_{k+1})$ for all $k \in \{1, \dots, K-1\}$, there remains nothing to show. Therefore, we assume that there is at least one index $k \leq K-1$ such that $c(T_k) > c(T_{k+1})$ or $c(T_k) = c(T_{k+1})$. We call these cases 'conflicts'. We denote the number of these indices by K^* and we suppose that k^* is the maximum among them. We refer to k^* as the largest index of conflict.

- Let us consider first the case $c(T_{k^*}) < c(T_{k^*+1})$. Due to Lemma 11.6 we can construct a tree $T \in \mathcal{O}_\mu$ adjacent to T_{k^*} and T_{k^*+2} with $c(T) < c(T_{k^*})$. For this tree, it holds that $c(T) < c(T_{k^*+2})$, $c(T) = c(T_{k^*+2})$, or $c(T) > c(T_{k^*+2})$.
  - Suppose $c(T) > c(T_{k^*+2})$.
    Then we substitute the subsequence $(T_k^*, {T_k}^* + 1, T_{k^*+2})$ by (T_k^*, T, T_{k^*+2}) and it holds that $c(T_k^*) < c(T) < c(T_{k^*+2})$.
  - Suppose $c(T) = c(T_{k^*+2})$ or $c(T) < c(T_{k^*+2})$.
    Then we substitute the subsequence $(T_k^*, T_{k^*+1}, T_{k^*+2})$ by (T_k^*, T, T_{k^*+2}). For the new sequence we decreased the largest index of conflict k^* by one.
- Let us now consider the case $c(T_{k^*}) = c(T_{k^*+1})$. Due to Lemma 11.3 we can construct a tree $T \in \mathcal{O}_\mu$ adjacent to T_{k^*} and T_{k^*+2} with $c(T) < c(T_{k^*})$. A similar analysis as above either resolves the 'conflict' or decreases the largest index of conflict by one.

The proof relies on the fact that a conflict is pushed towards the end of the sequence and, eventually, the application of Lemma 11.3 or Lemma 11.6 resolves this conflict and we proceed with the treatment of the next conflict.

□

Lemma 11.8
Let $(T_1^1, \ldots, T_L^1)$ be a sequence in $\mathcal{O}_\mu$ with T_l^1 and T_{l+1}^1 being adjacent and satisfying $c(T_l^1) > c(T_{l+1}^1)$ for all $l \in \{1, \ldots, L-1\}$. Suppose there does not exist a tree $T_{L+1}^1 \in \mathcal{O}_\mu$ with T_L^1 and T_{L+1}^1 being adjacent and satisfying $c(T_L^1) > c(T_{L+1}^1)$ i.e., we suppose that T_L^1 is the last element of the decreasing sequence.
Let $(T_1^1 = T_1^2, \ldots, T_K^2)$ be a sequence in $\mathcal{O}_\mu$ with T_k^2 and T_{k+1}^2 being adjacent and satisfying $c(T_k^2) > c(T_{k+1}^2)$ for all $i \in \{1, \ldots, K-1\}$. Suppose that there does not exist a tree $T_{K+1}^2 \in \mathcal{O}_\mu$ with T_K^2 and T_{K+1}^2 being adjacent and satisfying $c(T_K^2) > c(T_{K+1}^2)$, i.e., we suppose that T_K^2 is the last element of the decreasing sequence.
Then

$$c(T_K^2) = c(T_L^1).$$

Proof
Note that both sequences start with the same element since $T_1^1 = T_1^2$. It is sufficient to find a contradiction only for the case $c(T_K^2) > c(T_L^1)$. We show this by induction over K.

- *Basis:* $K = 2$
  We proof this by a second induction over L.
  - *Basis:* $L = 2$
    Assume $c(T_2^2) > c(T_2^1)$. Due to Lemma 11.6 there exists an adjacent tree T of T_2^2 with $c(T) \leq c(T_2^2)$ which is a contradiction to the assumption that the sequence stops in T_K^2. So equality must hold.
  - *Induction hypothesis:* Let the claim be valid for an arbitrary $L-1$.
  - *Induction step:* According to Lemma 11.6 we can construct a tree $T \in \mathcal{O}_\mu$ from T_2^1 which is adjacent to T_2^1 and T_2^2 with costs $c(T) \leq c(T_2^2)$. Assume that $c(T) < c(T_2^2)$. Since T and T_2^2 are adjacent, this contradicts the assumption that T_2^2 is the last tree in the sequence. Therefore, $c(T) = c(T_2^2)$. Then we consider the sequence $T_2^1, \ldots, T_L^1$ having $L-1$ elements and the sequence (T_2^2, T) consisting of two trees. We apply the induction hypothesis and conclude that $c(T_L^1) = c(T) = c(T_2^2)$.
- *Induction hypothesis:* Let the claim be valid for an arbitrary $K-1$.
- *Induction step:* To be able to apply the induction hypothesis, we start our considerations in the tree T_2^2. Note that T_2^2 and T_1^1 are adjacent. Thus, there exists a tree $T_2^3 \in \mathcal{O}_\mu$ with $c(T_2^3) \leq c(T_2^2)$ due to Lemma 11.3 and Lemma 11.6. We apply this argument iteratively and construct a sequence $(T_2^3, \ldots, T_H^3)$ with $c(T_H^3) = c(T_L^1)$. Now, we consider the sequences $(T_2^2, T_2^3, \ldots, T_H^3)$ and $(T_2^2, \ldots, T_K^2)$. Since the latter sequence has $K-1$ elements and since $c(T_H^3) = c(T_L^1)$, we apply the induction hypothesis and $c(T_K^2) = c(T_H^3) = c(T_L^1)$.

□

The lemma says that if such a sequence $(T_1^1, \ldots, T_L^1)$ exists, then a sequence of pivots according to (11.1) starting at T_1^1 yields a tree whose costs are equal to $c(T_L^1)$. It should be emphasized that this is independent of the choice of (11.1). We have shown that starting with an arbitrary tree in $\mathcal{O}_\mu$ the pivot sequence leads to a tree $T' \in \mathcal{O}_\mu$ with $c(T') \geq c(T)$ for all $T' \in \mathcal{O}_\mu$ i.e., T' is an extreme point and therefore also element of another $\mathcal{O}_{\hat{\mu}}$. We combine this fact to the next theorem.

Theorem 11.9
Starting in an arbitrary supported tree T, pivoting according to (11.1) leads to a sequence of adjacent trees (with decreasing costs) on the nondominated frontier which contains all breakpoints.

Proof
This follows from the fact that the set of supported trees is connected, the Lemma 11.7 and Lemma 11.8 and the considerations above.

□

Corollary 11.10
Let $\mathcal{O}_0$ the set of all trees which have minimal weight. If we start with an arbitrary $T_1 \in \mathcal{O}_\infty$ with $c(T_1) \leq c(T)$ for all $T \in \mathcal{O}_\infty$ and perform edge-exchanges as described in 11.1 we get a sequence such that $T_1, \ldots, T_K$ are (T_μ^K) with $c(T_\mu^K) \leq c(T)$ for all $T \in \mathcal{O}_\mu$ and $c(T_K) \leq c(T)$ for all $T \in \mathcal{T}$.

Proof
This follows directly from Theorem 11.12.

□

To apply these results to a branch and bound procedure we mention that the properties also hold for the frontier of the more restrictive problem $\{(w(T), c(T))|\mathcal{T}_{AB}\}$ where $T \in \mathcal{T}_{AB}$ if and only if $A \subset T$ and $B \cap T = \emptyset$ (as defined in the previous sections).
Since $\mathcal{T}_{AB} \subset \mathcal{T}$ we can state the following property:

Property 11.11
If T is a supported tree and $T \in \mathcal{T}_{AB}$, then T is also a supported tree for the problem

$$\min\{(c(T), w(T))|T \in \mathcal{T}_{AB}\}.$$

We also need the performing of edge exchanges by solving

$$f := \arg\min\{\frac{c_f - c_e}{w_e - w_f}|w_e > w_f \; f \notin T, e \in T, T \setminus \{e\} \cup \{f\} \in \mathcal{T}\}. \tag{11.18}$$

From the considerations for the edge-exchange rule (11.1) we can conclude the following theorem whose proof works analogously to the proof of Theorem 11.12.

Theorem 11.12
Starting in an arbitrary supported tree T, pivoting according to (11.18) leads to a sequence of adjacent trees (with increasing costs) on the nondominated frontier which contains all extreme points.

The branch and bound scheme works in the following manner: If we start with an arbitrary feasible supported tree T for a problem P_{AB}, we can pivot as mentioned in (11.1) until a weight infeasible tree is found. All intermediate trees are pairwise adjacent and located on the nondominated frontier. In each pivot the currently best upper bound for the subproblem is improved since we require $c_f - c_e < 0$. Alternatively, a weight infeasible solution is found and we have determined a feasible tree T and an infeasible tree T' where $T' = T \setminus \{e\} \cup \{f\}$. From the tree T we search for an alternative pivot under the condition that $\{f\}$ is not contained in the new tree (i.e. we search in the subproblem $P_{A,B\cup\{f\}}$) (we call this Case 1). From T', the infeasible tree, we perform pivots described as in (11.18) under the condition that $\{f\}$ is in the tree until we have found a feasible solution T'' (Case 2). This T'' is therefore a supported tree for $P_{A\cup\{f\}B}$.

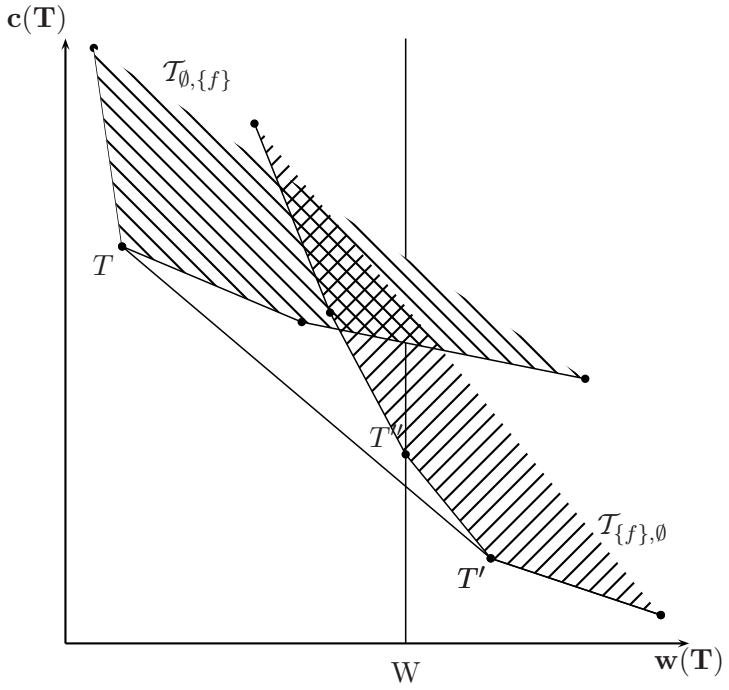


Figure 11.3: Partition into several subproblems

A subproblem P_{AB} need not be considered further if by (11.1) no pivot can be found i.e., the current tree is a minimal costs tree for P_{AB}. The weight of the current tree is equal W or a pivot leads to an infeasible tree such that the all trees on the frontier connecting these trees have costs

larger than a current upper bound, i.e

$$UB > \frac{c_f - c_e}{w_f - w_e} W - c(T) + \frac{c_f - c_e}{w_f - w_e} w(T). \tag{11.19}$$

In detail our algorithm looks in this form:

Algorithm 11.4 Branch- and Bound-Scheme of Ruzika and Henn: Main

```
Call Initialization (Procedure 11.5)
while stack is not empty do
  Take an element [A, B, T,ic,x] from the stack
  if ic < UB then
5:   if x = Case 1 then
       Call Case1 (Procedure 11.6)
     else
       Call Case 2 (Procedure 11.7)
     end if
10:  end if
end while
OPT is the optimal solution
```

Algorithm 11.5 Branch- and Bound-Scheme of Ruzika and Henn: Initialization

Solve lex $\min\{(c(T), w(T))|T \in \mathcal{T}\}$. Call this tree T
if $w(T) \leq W$ **then**
  $OPT = T$ is an optimal solution
else
  Solve lex $\min\{(w(T), c(T))|T \in \mathcal{T}\}$. Call this tree T_w
  if $w(T_w) = W$ **then**
    T_w is an optimal solution
  else if $w(T_w) > W$ **then**
    the problem is infeasible
    stopp = false
    repeat
      Find a tree T' outgoing from T by (11.1) where $T' = T \setminus \{e\} \cup \{f\}$
      if $w(T') > W$ **then**
        stopp = true
      else
        $UB := c(T')$
        $T := T'$
      end if
    until stopp = true
    if $w(T) = W$ **then**
      T is an optimal solution
    else
      $ic := \frac{c_f - c_e}{w_f - w_e} W - c(T) + \frac{c_f - c_e}{w_e - w_f} w(T)$
      add $[\emptyset, \{f\}, T$, ic, Case 1] to the stack
      add $[\{f\}, \emptyset, T'$, ic, Case 2] to the stack
    end if
  end if
end if

Procedure 11.6 Branch- and Bound-Scheme of Ruzika and Henn: Case 1

Require: Problem $[A, B, T, ic,$ Case 1$]$

stop = false
repeat
  if $c(T) < UB$ **then**
    $UB := c(T), OPT := T$
  end if
  Find a tree T' outgoing from T by 11.1 where $T' = T \setminus \{e\} \cup \{f\}$ and $T' \in \mathcal{T}_{AB}$
  if no pivot can be found **then**
    stop = true case b)
  else if $w(T') \geq W$ **then**
    stopp = true case a)
  else
    $T := T'$
  end if
until stop = true
if $w(T) = W$ **then**
  if $c(T) < UB$ **then**
    $UB = c(T)$, $OPT = T$
  end if
else if stop = true case a) and $UB > \lceil -\frac{c_f - c_e}{w_e - w_f} W - c(T) + \frac{c_f - c_e}{w_e - w_f} w(T) \rceil (=: ic)$ **then**
  push $[A, B \cup \{f\}, T,$ ic, Case 1$]$ to the stack
  push $[A \cup \{f\}, B, T',$ ic, Case 2$]$ to the stack
end if

Procedure 11.7 Branch- and Bound-Scheme of Ruzika and Henn: Case 2

Require: Problem $[A, B, T$, ic, Case 2$]$

stop = false
repeat
Find a tree T' outgoing from T by 11.18 where $T' = T \setminus \{e\} \cup \{f\}$ and $T' \in \mathcal{T}_{AB}$
if no pivot can be found **then**
stop = true case b)
else if $w(T') \leq W$ **then**
stop = true case a)
else
$T := T'$
end if
until stop = true
if $w(T) = W$ **then**
if $c(T) < UB$ **then**
$UB = c(T)$, $OPT = T$
end if
else if stop = true case a) and $UB > \lceil -\frac{c_f - c_e}{w_e - w_f} W - c(T) + \frac{c_f - c_e}{w_e - w_f} w(T) \rceil (=: ic)$ **then**
push $[A, B \cup \{f\}, T$, ic, Case 1$]$ to the stack
push $[A \cup \{f\}, B, T'$, ic, Case 2$]$ to the stack
end if

11.1.4 The Algorithm of Yamada, Watanabe and Kataoka [39]

In the publication of Yamada, Watanabe and Kataoka [39] one can find a branch and bound algorithm for the weight-constrained maximum spanning tree problem which can easily be transformed to the weight-constrained minimal spanning tree problem.

We start with an arbitrary feasible tree $T = \{e_{k_1}, \ldots, e_{k_{n-1}}\}$ and construct a branch and bound algorithm like in the algorithm of Aggarwal, Aneja and Nair: The i-th subproblem is to find an optimal solution that includes $e_1, \ldots, e_{i-1}$ and does not include e_i.

Procedure 11.8 Branch- and bound-scheme of Yamada, Watanabe and Kataoka

Require: Two sets A, B

if P_{AB} is infeasible or a lower bound on this problem is larger than the incumbent **then**
prune
else if P_{AB} is solved and the solution is smaller than the incumbent **then**
update the incumbent
5: Update the upper bound if necessary
else
Find a feasible tree T'_{AB} and partition from this tree
(Update the incumbent if necessary)
end if

We partition from a tree $T'_{AB} = A \cup \{e_{k+1}, \ldots e_{n-1}\}$ with $B \cap \{e_{k+1}, \ldots, e_{n-1}\} = \emptyset$ in P_{A_i,B_i} where $A_i := A \cup \{e_{k+1}, \ldots, e_{i-1}\}$ and $B_i := B \cup \{e_i\}$.

In order to find a feasible tree in Line 7 the local search method given in the Section 12.3 is proposed. This is the main difference to the algorithm of Aggarwal, Aneja and Nair.
Yamada, Watanabe and Kataoka mention also a shooting method to reduce the time complexity of the algorithm: In a standard branch and bound scheme we set at the beginning the objective value of the incumbent solution to infinity. In a shooting method we will take some value C. We need that $C \geq OPT$ to determine a solution. In the case that $C < OPT$ the algorithm fails.

Procedure 11.9 Branch- and bound-scheme of Yamada, Watanabe and Kataoka: Shooting Method

Start with an interval $[\underline{C}, \overline{C}]$
repeat
$C := \alpha\underline{C} + (1-\alpha)\overline{C}$
Run Algorithm 11.8 for C
5: $\underline{C} := C$
until an optimal solution is found

11.2 The Algorithm of Hong, Chung and Park [27]

Matrix tree theorem

The main ingredient for this algorithm is the Matrix Tree Theorem which was first stated in a special case by Kirchhoff in 1847. Since then many generalizations and different versions have been formulated. For this algorithm we need only a simple version. Hong, Chung and Park do not prove the theorem and refer to the paper of Chaiken and Kleitman [7]. We give a slightly modified proof in this thesis.

First, we have to define a cost matrix $c \in \mathbb{R}^n \times \mathbb{R}^n$ with $c_{ij} = c_{ji} = c_e$ if an edge in E exists with the end-nodes i and j. Also we define a cost matrix $w \in \mathbb{R}^n \times \mathbb{R}^n$ with $w_{ij} = w_{ji} = w_e$ if an edge in E exists with the end-nodes i and j. This is analogous to Problem 5.

Definition 11.1
Let $K = (k_{ij}) \in Mat(n, n, \mathbb{R})$ be a matrix defined as

$$k_{ij} = \begin{cases} \sum\limits_{1 \le l \le n,\, i \neq l} k_{il}, & \text{if } j = i \\ -c_{ij}, & \text{if } i \neq j \text{ and } e = \{i, j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

This leads (in case of a complete graph) to

$$K = \begin{pmatrix} \sum\limits_{i=1, i\neq 1}^{n} c_{1i} & -c_{12} & \dots & -c_{1l} & \dots & -c_{1n} \\ -c_{21} & \sum\limits_{i=1, i\neq 2}^{n} c_{2i} & \dots & -c_{2l} & \dots & -c_{2n} \\ \dots & \dots & & \dots & & \dots \\ -c_{l1} & -c_{2l} & \dots & \sum\limits_{i=1, i\neq l}^{n} c_{il} & \dots & -c_{ln} \\ \dots & \dots & & \dots & & \dots \\ -c_{n1} & -c_{n2} & \dots & -c_{nl} & \dots & \sum\limits_{i=1, i\neq n}^{n} c_{in} \end{pmatrix}$$

We notice that this matrix is symmetric. So we can formulate the important result for this section.

Theorem 11.13 *Matrix Tree Theorem*
Let $\check{K}[n, n]$ denote a matrix obtained from K by deleting the n-th row and column. Then

$$\det \check{K}[n, n] = \sum_{T \in \mathcal{T}} \prod_{e \in T} c_e.$$

Proof
As mentioned in Chapter 3 we can interpret each undirected spanning tree as directed spanning

tree by orienting all edges to a root n. So

$$\sum_{T\in\mathcal{T}}\prod_{e\in T} c_e = \sum_{T\in\mathcal{T}_n}\prod_{(i,j)\in T} c_{ij}$$

and it is sufficient to prove

$$\det \check{K}[n,n] = \sum_{T\in\mathcal{T}_n}\prod_{(i,j)\in T} c_{ij}.$$

We consider the matrix $K(\{z_q\})$ by

$$k\{z_q\}_{ij} = \begin{cases} \sum\limits_{(l,i)\in E} c_{li}z_i, & \text{if } j=i \\ -c_{ij}z_j, & \text{if } i\neq j \text{ and } (i,j)\in E \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, this matrix is not symmetric.

$$K(\{z_q\}) = \begin{pmatrix} \sum\limits_{(1,i)\in E} c_{1i}z_i & -c_{12}z_2 & \dots & -c_{1n}z_n \\ -c_{21}z_1 & \sum\limits_{(2,i)\in E} c_{2i}z_i & \dots & -c_{2n}z_n \\ \dots & \dots & & \dots \\ -c_{n1}z_1 & -c_{n2}z_2 & \dots & \sum\limits_{(n,i)\in E} c_{ni}z_i \end{pmatrix}$$

K is a special case of this matrix $K(\{z_q\})$ with $z_q = 1$ for all $1 \leq q \leq n$. If we prove the theorem for this matrix, then the theorem is valid. The claim is now:

$$\det \check{K}(z_q)[n,n] = \sum_{T\in\mathcal{T}_n}\prod_{(i,j)\in T} c_{ij}z_j \tag{11.20}$$

Claim 1: Both sides of the inequality (11.20) are polynomials of degree $n-1$ in the z_q.
left hand side: By definition it holds

$$\det \check{K}(z_q)[n,n] = \sum_{\sigma\in\Sigma} k\{z_q\}_{1,\sigma(1)}\dots k\{z_q\}_{n-1,\sigma(n-1)}$$

where Σ is the set of all permutations on $\{1,\dots,n-1\}$. All these products have $n-1$ factors. In each of this factors there is either a single z_q or a sum of different z_q. Consider without loss of generality a term with a sum in the first element:

$$(\sum_{(1,j)\in E} c_{1j}z_j)k\{z_q\}_{2,\sigma(2)}\dots k\{z_q\}_{n-1,\sigma(n-1)} = \sum_{(1,j)\in E}(c_{1j}z_j k\{z_q\}_{2,\sigma(2)}\dots k\{z_q\}_{n-1,\sigma(n-1)})$$

The summands now have only degree one in the first factor. Repeating this step for other factors yields to terms with only $n-1$ elements with only one z_q for each element. And each term has a degree of $n-1$.
right hand side: In a spanning tree only $n-1$ edges occur. So the degree of one spanning tree

in the z_q is $n-1$.

Claim 2: Every term of each side of (11.20) is of degree zero in some z_q for $q \neq n$.

left hand side: Since the left hand side is a polynomial of degree $n-1$ each term is of degree zero in at least one z_q. Assume that there exists a term of degree zero only in z_n. We can evaluate the sum of all such terms by setting $z_n = 0$. Then the matrix $\check{K}(z_q)$ is of the form

$$\begin{pmatrix} \sum\limits_{(1,i)\in E} c_{1i}z_i & -c_{12}z_2 & \dots & -c_{1,n-1}z_{n-1} & 0 \\ -c_{21}z_1 & \sum\limits_{(2,i)\in E} c_{2i}z_i & \dots & -c_{1,n-1}z_{n-1} & 0 \\ \dots & \dots & & & \dots \\ -c_{n1}z_1 & -c_{n2}z_2 & \dots & \sum\limits_{(n-1,i)\in E} c_{n-1,i}z_i & \sum\limits_{(n,i)\in E} c_{ni}z_i \end{pmatrix}$$

In the sums on the diagonal there is no z_n. Since we have zero row sums and the determinant of K is zero, $\det \check{K}(z_q)[n,n]$ has to be zero. This is contradiction to the assumption.

right hand side: By the orientation of the tree at least one edge (i,n) and one factor $c_{in}z_n$ have to exist. Since each term has a degree of $n-1$ the claim holds.

We prove the claim for terms independent of z_l for $l \neq n$. This follows by induction over the number of vertices in a graph.

Start Consider a graph with 2 vertices.

$$K = \begin{pmatrix} c_{12}z_2 & -c_{12}z_2 \\ -c_{21}z_1 & c_{21}z_2 \end{pmatrix}$$

Since $c_{12}z_2$ is the only possible tree

$$\det \check{K}(z_q)[2,2] = c_{12}z_2 = \sum_{T\in\mathcal{T}_2} \prod_{(i,j)\in T} c_{ij}z_j$$

holds.

Assumption: Let the claim hold for a graph with $n-1$ vertices.

Step: Consider the terms on the right hand side and left hand side with degree zero on z_l: On the right hand side there are no edges (i,l) in the tree. (Only an edge (l,i).) So l is a leaf for a tree T. The set of all trees with leaf l is the set of all trees in the $n-1$ remaining vertices (denote this set $\mathcal{T}'_n$) where each of this trees can be combined with only one edge (l,i). Therefore

$$\sum_{T\in\mathcal{T}_n, z_l=0} \prod_{(i,j)\in T} c_{ij}z_j = \sum_{(l,i)\in E} c_{li}z_i \Big(\sum_{T'\in\mathcal{T}'_n} \prod_{(i,j)\in T} c_{ij}z_j \Big).$$

In $\check{K}(z_q)[n,n]$ all the terms with degree zero in z_l can be evaluated by setting $z_l = 0$. This sets every term in the l-th row except the diagonal term equal to zero.

$$\det \check{K}(z_q)[n,n]|_{z_l=0}$$

$$= \det \begin{pmatrix} \sum\limits_{(1,i)\in E} c_{1i}z_i & \dots & -c_{1,l-1}z_{l-1} & 0 & -c_{1,l+1}z_{l+1} & \dots & -c_{1,n-1}z_{n-1} \\ -c_{21}z_1 & \dots & -c_{2,l-1}z_{l-1} & 0 & -c_{2,l+1}z_{l+1} & \dots & -c_{2,n-1}z_{n-1} \\ \dots & \dots & \dots & \dots & & \dots & \dots \\ -c_{l1}z_1 & \dots & -c_{l,l-1}z_{l-1} & \sum\limits_{(l,i)\in E} c_{li}z_i & -c_{l,l+1}z_{l+1} & \dots & -c_{l,n-1}z_{n-1} \\ \dots & \dots & \dots & \dots & & \dots & \dots \\ -c_{n1}z_1 & \dots & -c_{n-1,l-1}z_{l-1} & 0 & -c_{n-1,l+1}z_{l+1} & \dots & \sum\limits_{(n-1,i)\in E} c_{n-1,i}z_i \end{pmatrix}$$

(Developing by the l-th row)

$$= (-1)^{l+l} \sum_{(l,i)\in E} c_{li}z_i \det \begin{pmatrix} \sum\limits_{(1,i)\in E} c_{1i}z_i & \dots & -c_{1,l-1}z_{l-1} & -c_{1,l+1}z_{l+1} & \dots & -c_{1,n-1}z_{n-1} \\ -c_{21}z_1 & \dots & -c_{2,l-1}z_{l-1} & -c_{2,l+1}z_{l+1} & \dots & -c_{2,n-1}z_{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -c_{n1}z_1 & \dots & -c_{n-1,l-1}z_{l-1} & -c_{n-1,l+1}z_{l+1} & \dots & \sum\limits_{(n-1,i)\in E} c_{n-1,i}z_i \end{pmatrix}$$

$$= \sum_{(l,i)\in E} c_{li}z_i \det \begin{pmatrix} \sum\limits_{(1,i)\in E} c_{1i}z_i & \dots & -c_{1,l-1}z_{l-1} & -c_{1,l+1}z_{l+1} & \dots & -c_{1,n-1}z_{n-1} \\ -c_{21}z_1 & \dots & -c_{2,l-1}z_{l-1} & -c_{2,l+1}z_{l+1} & \dots & -c_{2,n-1}z_{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -c_{n1}z_1 & \dots & -c_{n-1,l-1}z_{l-1} & -c_{n-1,l+1}z_{l+1} & \dots & \sum\limits_{(n-1,i)\in E} c_{n-1,i}z_i \end{pmatrix}$$

(This determinant represents a graph with $n-1$ vertices and by the assumption:)

$$= \sum_{(l,i)\in E} c_{li}z_i \Big(\sum_{T'\in\mathcal{T}'_n} \prod_{(i,j)\in T} c_{ij}z_j \Big)$$

$$= \sum_{T\in\mathcal{T}_n, z_l=0} \prod_{(i,j)\in T} c_{ij}z_j$$

This computation can be done for all terms in $\check{K}(z_q)[n,n]$ with degree zero in some z_l.

$$\det \check{K}(z_q)[n,n] = \sum_{l=1}^{n-1} \det \check{K}(z_q)[n,n]|_{z_l=0} = \sum_{l=1}^{n-1} \sum_{T\in\mathcal{T}_n, z_l=0} \prod_{(i,j)\in T} c_{ij}z_j = \sum_{T\in\mathcal{T}_n} \prod_{(i,j)\in T} c_{ij}z_j.$$

So (11.20) is valid and from the first part of the proof the theorem follows.

□

The proof of the matrix tree theorem implies that it is also valid for directed trees. If we only set $c_{ij} = 1$ if the edge (i,j) is in E, the determinant will deliver the total number of possible spanning trees in the graph.

Definition 11.2
Let $K^x = (k_{ij}^x) \in Mat(n, n, \mathbb{R})$ *a matrix defined as*

$$k_{ij}^x = \begin{cases} \sum\limits_{(l,i)\in E} x^{c_{li}}, & \text{if } j = i \\ -x^{c_{ij}}, & \text{if } i \neq j \text{ and } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 11.14
Let $\check{K}^x[n, n]$ *denote a matrix obtained from* K *by deleting the* n*-th row and column. Then*

$$\det \check{K}^x[n, n] = \sum_p a_p x^p$$

where a_p *is the number of trees with costs* a*.*

Proof
If we substitute the c_e in the matrix tree theorem by x^{c_e}. We get

$$\det \check{K}^x[n, n] = \sum_{T\in\mathcal{T}} \prod_{e\in T} x^{c_e} = \sum_{T\in\mathcal{T}} x^{\sum_{e\in T} c_e} = \sum_p a_p x^p$$

□

Definition 11.3
Let $K^{xy} = (k_{ij}^{xy}) \in Mat(n, n, \mathbb{R})$ *be a matrix defined by*

$$k_{ij}^{xy} = \begin{cases} \sum\limits_{(l,i)\in E} x^{c_{li}} y^{w_{li}}, & \text{if } j = i \\ -x^{c_{ij}} y^{w_{ij}}, & \text{if } i \neq j \text{ and } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 11.15
Let $\check{K}^{xy}[n, n]$ *denote a matrix obtained from* K *by deleting the* n*-th row and column. Then*

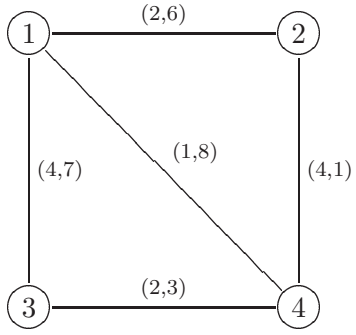
$$\det \check{K}^{xy}[n, n] = \sum_{p,q} a_{pq} x^p y^q$$

where a_{pq} *is the number of trees with costs* p *and weights* q*.*

Proof
Analogously to the proof of the previous theorem.

□

Example 11.2
Let us consider the following problem where $W = 13$ *with costs and weights:*



The corresponding matrix to this graph is:

$$\begin{pmatrix} x^2y^6+x^4y^7+x^1y^8 & -x^2y^6 & -x^4y^7 & -x^1y^8 \\ -x^2y^6 & x^2y^6+x^4y^1 & 0 & -x^4y^1 \\ -x^4y^7 & 0 & x^4y^7+x^2y^3 & -x^2y^3 \\ -x^1y^8 & -x^4y^1 & -x^2y^3 & x^1y^8+x^4y^1+x^2y^3 \end{pmatrix}$$

If we delete the fourth row and column:

$$\begin{aligned} \det \check{K}^{xy}[4,4] = &\det \begin{pmatrix} x^2y^6+x^4y^7+x^1y^8 & -x^2y^6 & -x^4y^7 \\ -x^2y^6 & x^2y^6+x^4y^1 & 0 \\ -x^4y^7 & 0 & x^4y^7+x^2y^3 \end{pmatrix} \\ =&(x^2y^6+x^4y^7+x^1y^8)(x^2y^6+x^4y^1)(x^4y^7+x^2y^3)-(-x^2y^6)(-x^2y^6)(x^4y^7+x^2y^3)- \\ &(-x^4y^7)(-x^4y^7)(x^2y^6+x^4y^1) \\ =&x^8y^{19}+x^{10}y^{14}+x^{20}y^{20}+x^{12}y^{15}+x^9y^{16}+x^6y^{15}+x^8y^{10}+x^8y^{16}+x^{10}y^{11}+x^7y^{12} \\ &-x^8y^{19}-x^6y^{15}-x^{10}y^{20}-x^{12}y^{15} \\ =&x^7y^{12}+x^8y^{10}+x^8y^{16}+x^9y^{16}+x^{10}y^{11}+x^{10}y^{14} \end{aligned}$$

Since $W = 13$ our optimal solution is a tree with costs 8 and total weight 10.

Pseudo-polynomial Algorithm

This result enables us to find the value of an optimal solution. The main question now is how to compute the determinant efficiently and how to construct the corresponding tree. There are algorithms with running time $O((n^3+p^2)p^2 \log p)$ where p is the highest degree of the polynomial. A good approach was developed by Mahajan and Vinay with time complexity $O(n^4\tau(C,D))$ where $\tau(C,D)$ is the time to multiply polynomials where C and D are upper bounds on the degrees of x and y respectively. This algorithm has the advantage that it can be modified such that degrees exceeding a limit on the degrees of x and y can be ignored. In our problem all degrees of y greater than W are not interesting. If we find an upper bound C on OPT, also larger degrees on x can be ignored. For instance we can set $C = (n-1)\max_{e\in E} c_e$. Let $\det(K^{xy}, C, W, c, w)$ denote this polynomial. If $\det(K^{xy}, C, W, c, w)$=0 there is no feasible spanning tree.

The next goal is to construct a algorithm to compute the optimal tree T. Hong, Chanc and Park give only the remark that this can be done by a recursion approach. We pick this remark up and show a possibility to construct an optimal solution: We fix an edge $e = (i, j) \in E$ and compute the determinant to see whether there is a tree with costs $OPT - c_e$ and weight at most $W - w_e$. If this is true go on with the recursion. Unfortunately, we cannot consider a reduced graph obtained from G by contracting $e \in E$ which was already mentioned in Chapter 9 concerning the exclusion tests. Therefore we need the following algorithm:

Algorithm 11.10 Constructing an optimal solution

Compute $\det(K^{xy}, (n-1)\max c_{ij}, W, c, w)$ and find OPT and the corresponding weight $\bar{W}$.
while $|T| < n - 1$ **do**
  Choose $e = (i, j) \in E$ such that T contains no cycle.
  $E := E \setminus \{e\}$
  Set $c_{ij} = c_{ji} = w_{ij} = w_{ji} = 0$ and actualize K^{xy}.
  if $\det(K^{xy}, OPT - c_e, \bar{W} - w_e, c, w) = 0$ **then**
    $k_{ij}^{xy} := 0$
    $k_{ji}^{xy} := 0$. (Update also the diagonal)
  else
    $T := T \cup \{e\}$
    $OPT := OPT - c_e$
    $\bar{W} := \bar{W} - w_e$
  end if
end while
T is optimal

Theorem 11.16
The algorithm finds an optimal solution in $O(mn^4\tau(C, D))$.

Proof
In the algorithm we set the costs and weights of an edge equal to zero such that $x^{c_{ij}}y^{w_{ij}} = 1$. So on the right hand side of the theorem all the degrees of terms corresponding to a tree containing e reducing by c_e in x and w_e in y. All the other terms remain unchanged. For the edge $e \in E$ we have to study two cases:

1. There exists an optimal solution which contains e and the edges already identified. In this case a term on the right hand side of the theorem exists which has a degree of $OPT - c_e$ and $\bar{W} - w_e$ and the determinant is not zero. So we can add e to T, update OPT and $\bar{W}$ and go to the next edge.

2. There exists no optimal solution which contains e and the edges already identified: For all these trees the costs and weights are by setting the edge costs and weights of e to 0 strictly greater than $OPT - c_e$ or $\bar{W} - w_e$. So the determinant is zero. Thus, we can exclude the

edge e and set the entry of the matrix equal to zero which might improve the computation time of the determinant in the next iterations.

The algorithm stops if a tree is found which is optimal. The time complexity follows by the fact that the computation of the determinant has a time complexity of $O(n^4\tau(C,D))$ and at least $m-1$ edges have to considered.

□

The paper claims that the number of iterations is $O(nm)$ and therefore an optimal tree can be found in $O(mn^5\tau(C,W))$.

Outlook
The algorithm can also be used for an approximation scheme which is explained in Section 12.4.

11.3 Ranking Algorithm

In literature there are several ranking algorithms to find the K best minimal spanning trees. For example Hamacher and Querente [25] give two approaches for finding the K best solutions to combinatorial optimization problems and, in detail, an algorithm to find the K best bases of a matroid. From the observations of Chapter 4 this algorithm can be used to find the best K spanning trees. This algorithm works similar to the algorithm of Katoh, Ibaraki and Mine [31] who find the K-th minimum spanning trees for a graph G. To this algorithm it is also referred in [26]. We use here this algorithm to find an optimal solution for the WCMST.

Finding K minimal spanning trees

In Chapter 2 we have seen that a spanning tree has minimal costs if and only if no T-exchange has negative costs. Recall that the costs of a T-exchange are $c[e,f] = c_f - c_e$. Katoh, Ibaraki and Mine use this property to obtain a sequence of minimal spanning trees in a graph:

Lemma 11.17 *[31]*
Let T be a spanning tree for the given sets $A \subset E$ and $B \subset E$ the condition $A \subset T$ and $B \cap T = \emptyset$ holds. A minimum spanning tree T' which is different from T and satisfy $A \subset T'$ and $B \cap T' = \emptyset$ is given by $T \setminus \{e\} \cup \{f\}$ where $[e,f]$ is a minimum T-exchange with $e \in T \setminus A$ and $f \in E \setminus (T \cup B)$.

Proof
For the required property of T', edges in A cannot be replaced by edges in B. So only the edges in $T \setminus A$ and $E \setminus (T \cup B)$ have to be considered. To obtain a minimum tree which is different from T' the minimum T-exchange with $e \in T \setminus A$ and $f \in E \setminus (T \cup B)$ has to be found.

□

The idea of the algorithm is to start with a minimal spanning tree T_0 and find a minimal T_0-exchange $[e, f]$. The obtained tree T_1 has minimal costs in $\mathcal{T} \setminus \{T_0\}$. Compute now a minimal T_1-exchange and a minimal T_0-exchange in the set of all T_0-exchanges without the exchange $[e, f]$ and take the minimum of both. The obtained tree T_2 is a minimal spanning tree in $\mathcal{T} \setminus \{T_0, T_1\}$. Now compute a minimum T_2-exchange and minimum exchanges for T_0 and T_1 without the realized exchanges. In this way we compute the K minimum spanning trees. More formal:

Definition 11.4
Assume that the first j minimum spanning trees are generated. Then a partition for the remaining trees is defined as

$$P_i^{j-1} = \{T_k | k > j-1, A_i \subset T_k, B \subset (E \setminus T_k)\} \text{ for } i = 0, 1, \ldots, j-1$$

For $i = 0, 1, 2, \ldots, j-1$ let the set of minimum exchanges denotes

$$Q_i^{j-1} = \{([e, f], r) | \text{ for each } f \in E \setminus (T_i \cup B_i), e \in T_i \setminus A_i$$

$$\text{gives the minimum } T_i\text{-exchange } [e, f] \text{ with } r = c[e, f]\}.$$

The sets A_i and B_i are initialized as follows:

$$A_0 := \emptyset \text{ and } B_0 := \emptyset$$

Let $i^\star$ be the index of the Q_i^{j-1} with the minimal exchange under all i and $[e^\star, f^\star]$ be this exchange. So $T_j := T_{i^\star} \setminus \{e^\star\} \cup \{f^\star\}$. We define

$$A_j := A_{i^\star} \cup \{f^\star\}, B_j := B_{i^\star}.$$

From $Q_{i^\star}^{j-1}$ we have to exclude the minimal exchanges with the edge $f^\star$ for the next iterations:

$$B_{i^\star} := B_{i^\star} \cup \{f^\star\}.$$

Lemma 11.18 *[31]*
Let $j \in \{1, 2, \ldots, K-1\}$.

1. *For any $i = 0, 1, \ldots, j-1$, T_i is a minimum spanning tree satisfying $A_i \subset T_i$ and $B_i \subset (E \setminus T_i)$ and no other $T_k (k = 0, 1, 2, \ldots, i-1, i+1, \ldots, j-1)$ satisfies this constraint.*
2. *Any spanning tree T satisfies $A_i \subset T$ and $B_i \subset (E \setminus T)$ for exactly one i with $0 \leq i \leq j-1$.*

Proof
This lemma follows from the definition of T_i and the construction of A_i and B_i.

□

The lemma implies that for known $T_1, T_2, \ldots, T_{j-1}$ the tree T_j is given as a minimum spanning tree in $\bigcup_{i=0}^{j-1} P_i^{j-1}$. By both lemmas a minimum spanning tree in P_i^{j-1} is given by $T_i \setminus \{e'\} \cup f'$, where $([e', f'], r')$ is a label in Q_i^{j-1} with smallest r. Therefore, $([e^\star, f^\star], r^\star)$ is a label in $\bigcup_{i=0}^{j-1} Q_i^{j-1}$ with smallest $c(T_i) + r = c(T_i) - c_e + c_f$, T_j is given by $T_j = T_{i^\star} \setminus e^\star \cup f^\star$.

In the algorithm the P_i^{j-1} are represented in a tuple

$$P_i^{j-1} = (T', [e', f'], A_i, B_i, i)$$

where $([e', f'], r')$ is the label in Q_i^{j-1} with smallest r, and $T' := c(T_i) + r'$.

Procedure 11.11 Enumeration of Katoh, Ibaraki, Mine: Procedure GENK(G=(V,E),K)

Compute Q_0^0
Find minimum weight exchange $([e', f'], r')$ in Q_0^0
$P_0^0 := (c(T_0) + r', [e', f'], \emptyset, \emptyset, 0)$
$j := 1$
while $j < K$ **do**
  $GEN(P_i^{j-1}, Q_i^{j-1} | i = 0, 1, 2, \ldots, j-1)$
  $j := j + 1$
end while

Procedure 11.12 Enumeration of Katoh, Ibaraki, Mine: (GEN($P_i^{j-1}, Q_i^{j-1} | i = 0, 1, 2, \ldots, j-1$)

Find $P_{i^\star}^{j-1} = (c^\star, [e^\star, f^\star], A_{i^\star}, B_{i^\star}, i^\star)$ with the smallest costs c' among all $P_i^{j-1} = (c(T_i) + r', [e', f'], A_i, B_i, i)$ for $i = 1, 2, \ldots, j-1$
if $c^\star = \infty$ **then**
  STOP (G has only $j-1$ trees)
end if
$Q_{i^\star}^j := Q_{i^\star}^{j-1} \setminus \{([e^\star, f^\star], c^\star - c_{i^\star})\}$
compute Q_j^j with $A_{i^\star \cup \{f^\star\}}$ and $B_{i^\star}$
$Q_i^j := Q_i^{j-1}$ for $i \neq i^\star, j$
if $Q_{i^\star}^j = \emptyset$ **then**
  $P_j^j := (\infty, \emptyset, \emptyset, \emptyset, i^\star)$
else
  $P_j^i := (c^\star + r'', [e'', f''], A_{i^\star} \cup f^\star, B_{i^\star}, j)$ here $([e'', f''], r'')$ is a label in Q_j^j with minimal r
end if
$P_i^j := P_i^{j-1}$ for $i \neq i^\star, j$

In [31] a procedure can be found which computes the set Q_j^j. In total this procedure has the complexity $O(Km + \min(n^2, m \log\log n))$.

Extension to our Problem

Can this algorithm be modified to solve our problem? Surely, a simple approach is to run the procedure GEN for the costs c until a tree is generated with $w(T) \leq W$. In particular:

Algorithm 11.13 Exact ranking

Compute Q_0^0
Find minimum cost exchange $([e', f'], r')$ in Q_0^0
$P_0^0 := (c(T_1) + r', [e', f'], \emptyset, \emptyset, 0)$
$j := 0$
repeat
  j:=j+1
  $GEN(P_i^{j-1} Q_i^{j-1} | i = 0, 1, 2, \ldots, j-1)$
until $w(T_j) \leq W$
T_j is the optimal solution.

Remark
Since this is an enumeration the validity of the algorithm is clear. In the worst case all spanning trees have to be computed. Because of the structure of the algorithm it is not possible to compute all trees starting by an arbitrary tree.

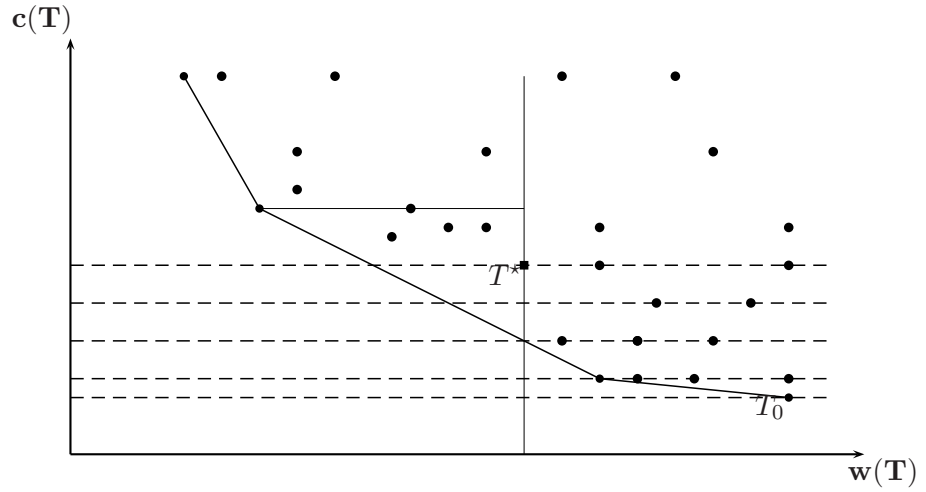
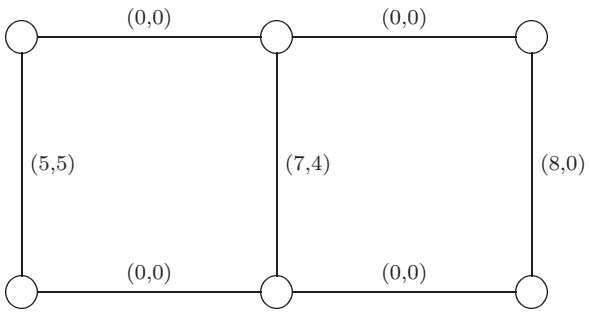


Figure 11.4: Searching for minimal cost spanning trees

Let all points describe a spanning tree. Then the algorithm starts with an optimal solution and finds all trees on the parallel line to the w-axes. If all trees are found, the algorithm moves to the next parallel line (see Figure 11.4).
The idea to compute all spanning trees under the costs $c_e + \mu^\star w_e$ for all $e \in E$ until the first feasible tree was found is not a possibility to solve the problem.

Example 11.3
Let the following graph be given:



We consider the three trees with lowest costs: T_1, T_2, T_3 *with* $c(T_1) = 5, w(T_1) = 5, c(T_2) = 7, w(T_2) = 4, c(T_3) = 8, w(T_3) = 0$ *and let* $W = 4$. *Obviously* $\mu^\star = \frac{3}{5}$ *and* T_2 *is the optimal solution for the WCMST. If we start with* T_1 *which is optimal for* $c + \frac{3}{5}w$, *the algorithm computes* T_3 *as the next tree under this cost function.* T_3 *is feasible but not the optimal solution.*

So this modification will not lead to the optimal tree if we take the first feasible spanning tree which was found by the algorithm. Therefore we have to modify the algorithm:

Algorithm 11.14 Search via Lagrangian Dual

$T^\star := T_{UB}$
Compute $\mu^\star$ and for each edge the costs $c_e + \mu^\star w_e$
Compute a minimal spanning tree $T_{\mu^\star}$ for the costs $c_e + \mu^\star w_e \; \forall e \in E$
$T_0 := T_{\mu^\star}$
Compute Q_0^0
Find minimum exchange for the costs $c_e + \mu^\star w_e$ $([e', f'], r')$ in Q_0^0
$P_0^0 := (c(T_0) + \mu^\star w(T_0) + r', [e', f'], \emptyset, \emptyset, 0)$
$j := 1$
repeat
  $GEN(P_i^{j-1}, Q_i^{j-1} | i = 0, 1, 2, \ldots, j-1)$ (for the costs $c_e + \mu^\star w_e$)
  if $w(T_j) \leq W$ and $c(T_j) \leq c(T^\star)$ **then**
    $T^\star := T_j$
  end if
  $j := j + 1$
until $-\mu^\star W + c(T_j) + \mu^\star w(T_j) > UB$
$T^\star$ is optimal

Theorem 11.19
The algorithm finds an optimal solution.

Proof
If we try to find a minimum spanning tree for the cost function $c_e+\mu^\star w_e$ for all $e \in E$ we search on parallels to the facet of the convex hull corresponding to $\mu^\star$. The extension of this segment can be described as the linear function $f : \mathbb{R} \to \mathbb{R}$ with $x \mapsto -\mu^\star x + (c(T_{\mu^\star}) + \mu^\star w(T_{\mu^\star}))$. This function passes through $(c(T_{\mu^\star}), (w(T_{\mu^\star}))$. The algorithm searches now for all trees lying on this function i.e., $c(T) = -\mu^\star w(T)+(c(T_{\mu^\star})+\mu^\star w(T_{\mu^\star}))$ (which has the same costs under $c_e+\mu^\star w_e$ for all $e \in E$ like $T_{\mu^\star}$. See Figure 11.5) If all spanning trees on this function are evaluated, the algorithm finds the next tree T with smallest $\epsilon > 0$ such that $c(T) + \mu^\star w(T) = c(T_{\mu^\star}) + \mu^\star w(T_{\mu^\star}) + \epsilon$. We search now on the function $f_\epsilon : \mathbb{R} \to \mathbb{R}$ with $x \mapsto -\mu^\star x+(c(T_{\mu^\star})+\mu^\star w(T_{\mu^\star}))+\epsilon = -\mu^\star x+c(T)+\mu^\star w(T)$. This is the function which has the smallest possible distance to f.
We know that the optimal solution has costs and weights such that $(c(T^\star), w(T^\star),)$ is contained in the triangle $(c(T_{\mu^\star}), w(T_{\mu^\star}), (UB, w(T_{UB}))$ and (UB, W). So, if the function f_ϵ lies outside the triangle we can stop. We test whether $f_\epsilon(W) \geq UB$. If this is true the function does not intersect with the triangle and we can stop. The optimal solution is the current solution $T^\star$.

□

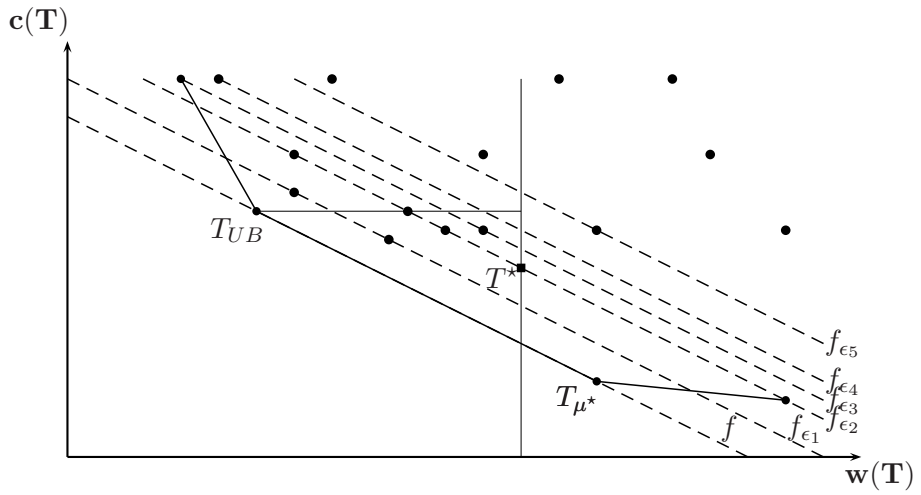


Figure 11.5: Search via Lagrangian Dual

11.4 CNOP-Software

In [32] Mehlhorn and Ziegelmann present their approach to solve constrained network optimization problems. They developed a software package which is able to solve the WCMST. They do not give an explicit insight in their algorithm. Only the main idea is published. This is

1. Compute with the Lagrangian relaxation the closest segment of the convex hull as in the algorithm of Xue.
2. Start with the lower bound solution and use a ranking algorithm to close the gap to the optimal solution.

For the ranking they refer for instance to the algorithm of Katoh, Ibariki and Mine.

12 Approximations

Approximation algorithms are an approach to handle optimization problems predominantly NP-hard problems. Since it is unlikely that there can ever be found efficient exact algorithms, one settles for non-optimal solutions, but requires them to be found in polynomial time. In this chapter we analyze the existing approximation algorithms for the weight-constrained minimal spanning tree problem and use the method of decomposition to obtain a further approximation scheme. For the sake of completeness we firstly recall the definition of an approximation.

Definition 12.1 *Approximation*

1. *A tree T is called an α-approximation if and only if T is feasible and $c(T) \leq \alpha OPT$.*
2. *A tree T is called an (α, β)-approximation if and only if $c(T) \leq \alpha OPT$ and $w(T) \leq \beta W$.*

In the literature we can find four approximation ideas to our problem. We use a similar characterization of these approximation schemes like in the previous chapter. We present first the idea of Goemans and Ravi [18] which uses the Lagrangian relaxation to construct an approximation. This idea was refined by the approximation of Hassin and Levin [23]. Then we focus on an algorithm of Yamada, Watanabe and Kataoka [39] which uses like the branch and bound scheme of Ruzika and Henn the neighborhood-structure of adjacent trees. The last algorithm from the literature is the idee of Hong, Chung and Park [27] to use their Algorithm 11.10 for an approximation. The last section is the application of the idea of approximation through decomposition to our weight-constrained minimal spanning tree problem.

12.1 Approximation of Goemans and Ravi [18]

The main ingredient for the approximation scheme of Goemans and Ravi [18] is again the Lagrangian relaxation and the Lagrangian dual:

$$\begin{aligned} C^\star(MST;\mu) := \min\ & (c(T) + \mu w(T)) \\ & s.t.\ y \in \mathcal{T} \end{aligned}$$

and

$$\begin{aligned} C^\star(D1) := \max\ & C^\star(MST;\mu) - \mu W \leq OPT \\ & s.t.\ \mu \geq 0 \end{aligned}$$

Let $\mu^\star$ be the optimal Lagrangian multiplier. Without loss of generality we consider for the following a graph where $w_e \leq W$ for all $e \in E$.

Theorem 12.1
Let $\mathcal{O}_{\mu^\star}$ be the set of all optimal trees with respect to the costs $c_e + \mu^\star w_e$ for all $e \in E$. Then there exists a $T \in \mathcal{O}_{\mu^\star}$ with $c(T) \leq C^\star(D1) \leq OPT$ and $w(T) \leq W + w_{\max}$ where $w_{\max} := \max_{e \in E} w_e$.

Proof
Let $T \in \mathcal{O}_{\mu^\star}$ then

$$c(T) = c(T) + (\mu^\star w(T) - \mu^\star W) - \mu^\star(w(T) - W) = C^\star(D1) - \mu^\star(w(T) - W)$$

The costs $c(T)$ have at least the value $C^\star(D1)$ if and only if $w(T) \geq W$.
From the proof of Theorem 7.4 it follows that for $\mu = \mu^\star + \epsilon$ or $\mu = \mu^\star - \epsilon$ for sufficiently small $\epsilon > 0$ the optimal tree with respect to $c_e + \mu w_e$ for all $e \in E$ is an element of $\mathcal{O}_{\mu^\star}$. A tree $T_\leq \in \mathcal{O}_{\mu^\star}$ has to exist with $w(T_\leq) \leq W$, because otherwise:

$$\begin{aligned} C^\star(D1) =& c(T_\leq) + \mu^\star(w(T_\leq) - W) \\ <& c(T_\leq) + (\mu^\star + \epsilon)(w(T_\leq) - W) \\ =& C^\star(MST, \mu^\star + \epsilon) - (\mu^\star + \epsilon)W \end{aligned}$$

which is a contradiction to the optimality of $C^\star(D1)$. (Remark: The tree $T_\leq$ is feasible for the weight-constrained minimal spanning tree problem, but not necessarily optimal.) On the other hand a tree $T_\geq$ with $w(T_\geq) \geq W$ in $\mathcal{O}_{\mu^\star}$ exists. It remains to show that a tree in $\mathcal{O}_{\mu^\star}$ with a weight between W and $W + w_{max}$ exists. We need again the property that $\mathcal{O}_{\mu^\star}$ is adjacent i.e., a sequence between to optimal trees T and T' with $T = T_0, T_1, T_2, \ldots, T_n = T'$ exist, such that T_i and T_{i+1} differ only in one edge-exchange. In this case we set $T = T_\leq$ and $T' = T_\geq$. Since $w(T_\geq) \geq W$ and $w(T_\leq) \leq W$ there exists a pair T_i and T_{i+1} in $\mathcal{O}_{\mu^\star}$ such that $w(T_i) \leq W$ and $w(T_{i+1}) \geq W$, and T_i and T_{i+1} differ only in one exchange. We have

$$w(T_{i+1}) = w(T_i) + w_{i+1} - w_i \leq w(T_i) + w_{max} \leq W + w_{max}$$

Since all edges with weight greater than W have been pruned

$$w(T_{i+1}) \leq W + w_{max} \leq 2W.$$

□

Algorithm 12.1 Approximation of Goemans and Ravi

Ensure: Tree T with $c(T) \leq OPT$ and $w(T) \leq 2W$
Discard all edges e with $w_e > W$
Compute $\mu^\star$ by solving the Lagrangian dual
Among all optimal trees for $c + \mu^\star w$ find a tree T satisfying $w(T) \leq W + w_{\max}$

Sketch of an implementation

In the algorithm we have to find $\mu^\star$. As mentioned in Theorem 7.4 at least m^2 possible values have to be considered. For every breakpoint we compute a minimal spanning tree with respect to $c+\mu w$. This can be found easily for every fixed μ. For all optimal trees for $c+\mu w$ we can find trees $T_{\min}$ and $T_{\max}$ with smallest and largest weights. Therefore, we use a lexicographic ordering of the edges. From the proof follows that $\mu \leq \mu^\star$ if $w(T_{\min}) > W$ and $\mu \geq \mu^\star$ if $w(T_{\max}) < W$ and μ is a possible value. So we found upper and lower bounds for $\mu^\star$ by computing this for the breakpoints.
(In an easy approach we have a time complexity $O(m^2 m \log m)$, since we have to compute the $m(m-1)$ breakpoints and for each breakpoint we have to compute a minimum spanning tree which has the time complexity $O(m \log m)$. We can improve this result by a binary search over the possible values for μ and get a time complexity of $O(m^2 \log m^2 m \log m) = O(m^3 \log m^3)$.) Goemans and Ravi claim without explanations, that a faster MST algorithm and the use of parallel sorting lead to $O(m \log^2 n + n \log^3 n)$.

Procedure 12.2 Approximation of Goemans and Ravi: Lagrangian Relaxation

```
    Order the edges in a lexicographical order {j_1, ..., j_m}.
    Start with k := 1
    μ_ := 0
    μ‾ := 1000
 5: while k < m − 1 do
      l := k + 1
      compute μ = (c_{j_k} − c_{j_l}) / (w_{j_l} − w_{j_k})
      if μ ∈ (μ_, μ‾) then
        Compute T_max and T_min under the costs c + μw
10:     if w(T_min) > W then
          μ_ := μ
        end if
        if w(T_max) < W then
          μ‾ := μ
15:     end if
      end if
      k:=k+1
    end while
    Choose μ* ∈ (μ_, μ‾)
```

The goal is to find a tree satisfying the theorem. From line 9 we know $T_{\min}$ and $T_{\max}$ which are optimal for $\mu^\star$ and $w(T_{\min}) \leq W \leq w(T_{\max})$. We start with $T_{\max}$ and swap an edge f in the tree with a minimal cost edge not in $T_{\max}$ but in a cycle closed by f. (The paper states without proof that the time complexity can be reduced to $O(n \log n)$.) In this publication Goemans and

Ravi do not mention how the trees T_i and T_{i+1} can be found. One possibility is to perform edge exchanges as described in Section 11.1.3.

Graphical interpretation

The graphical interpretation of the approach is very easy. From further considerations we know that the trees on the convex hull are adjacent. So we move along the facet of the convex hull which is next to the optimal solution until the two closest trees T_i and T_{i+1} next to the weight constraint are found.

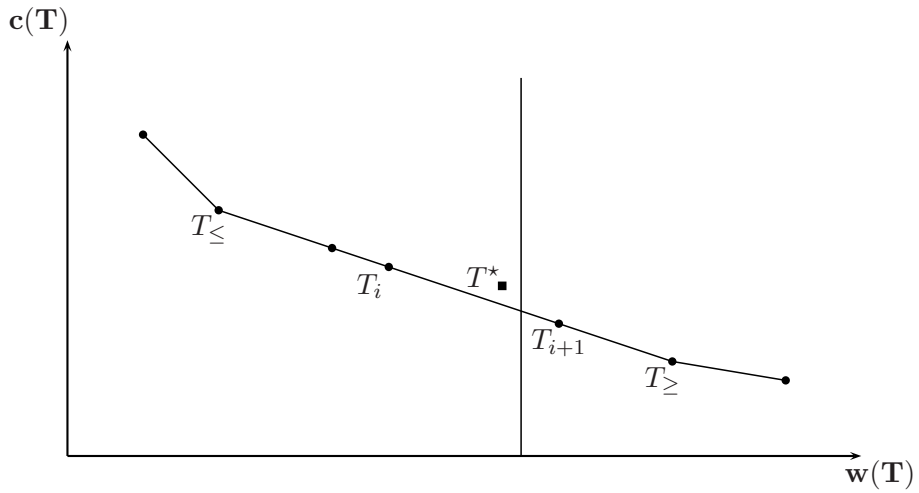


Figure 12.1: Finding adjacent trees

A polynomial-time approximation scheme

In the algorithm we discard all edges with weight greater than W. This guarantees the (1,2)-approximation. To reduce this, it is possible to prune all edges with weight greater then ϵW. This leads to a final tree with weights less than $(1+\epsilon)W$. By this procedure we might discard some edges that could be in the optimal solution. In an optimal solution at least $\frac{1}{\epsilon}$ of these edges can be contained in an optimal solution. There are only $O(n^{O(\frac{1}{\epsilon})})$ choices for those subsets. For each of these combinations we fix these edges and start our algorithm with the value W minus the length of all chosen edges and get a tree with weights at most $(1+\epsilon)W$. We take the best of all these trees.

Application to the WCMST

In general, the algorithm above does not yield a feasible solution to our problem. The idea to use this algorithm is to change the roles of costs and weights and to run the algorithm for all

possible integral costs C and find a tree with, approximately, minimum length. In detail (not mentioned in the paper):

Problem 21

$$\begin{aligned} W(C) := \min & \sum_{e \in T} w_e \\ s.t. & \sum_{e \in T} c_e \leq C \\ & T \in \mathcal{T} \end{aligned}$$

Algorithm 12.3 Approximation of Goemans and Ravi: MAIN

Solve Problem 21 with the algorithm and denote $\underline{C}$ as the costs of this solution.
$\overline{C} := (n-1) \max_{e \in E} c_e$
$C := \frac{\underline{C}+\overline{C}}{2}$
Solve Problem 21 for C with the algorithm, let $(W\tilde{(}C), \tilde{C})$ the given approximation
repeat
  if $W\tilde{(}C) \leq W$ **then**
    $\overline{C} := \tilde{C}$
  else
    $\underline{C} := C$
  end if
until $\overline{C} - \underline{C} \leq \epsilon$
$\overline{C}$ is the desired result

Theorem 12.2
The algorithm finds a $2(1+\epsilon)$*-approximation in* $O(\log(n-1) \max_{e \in E} c_e m^3 \log m)$ *time.*

Proof
The binary search has a time complexity of $O(\log(n-1) \max_{e \in E} c_e)$ and in each step the algorithm of the previous paragraph has to run. So the claimed time complexity is valid. In each iteration we get a solution which has costs at least 2 times the optimal costs corresponding to the $W(C)$. So, at the end of the algorithm we have a tree which has a weight at least $(1+\epsilon)$ times the weight of the optimal solution and has costs which is 2 times the optimal costs corresponding to this weight.

□

If we know a better upper bound than $(n-1) \max_{e \in E} c_e$, the algorithm can be started with this bound. By the suggested time complexity at the end of the paragraph about the implementation we can find the approximation in $O(\log(n-1) \max_{e \in E} c_r (m \log^2 n + n \log^3 n))$.

Graphical interpretation

In each step we find for C two trees T_i and T_{i+1} (see Figure 12.1) and know that $c(T_{i+1}) < 2C$ and check whether $w(T_{i+1}) \leq W$. If this is true, we will search between C and $\underline{C}$, otherwise in $\overline{C}$ and C. The important step in this algorithm is the discarding of edges which have costs greater than $2C$ in each run to solve Problem 21. So, we may change the frontier of the convex hull.

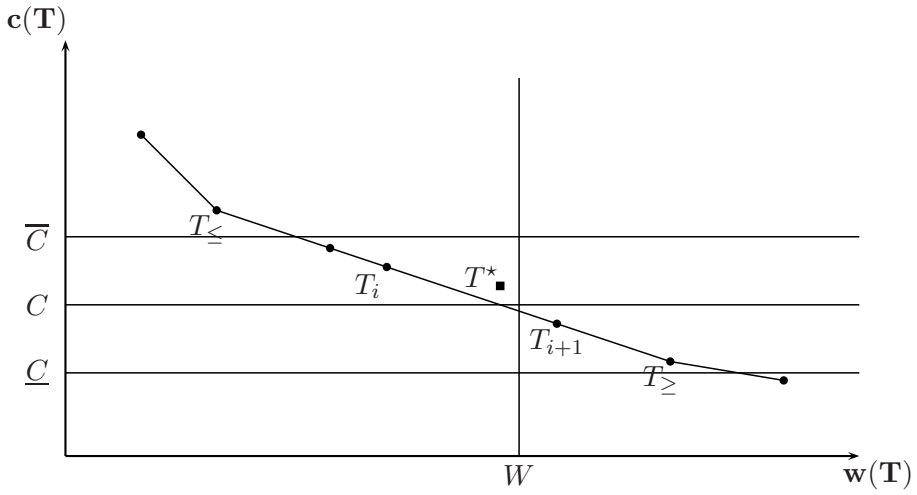


Figure 12.2: Graphical interpretation of Algorithm 12.3

The idea of Goemans and Ravi allow us now to make a new statement of the approximation quality of supported trees:

Theorem 12.3

Let us consider a graph where $c_e < (n-1)\min_{f\in E} c_f$ for all $e \in E$. Then at least for one (feasible) supported tree $T \in \mathcal{T}$ it holds that $c(T) \leq 2(1+\epsilon)OPT$.

Proof

This follows from the fact that by running the Algorithm 12.3 no edge will be discarded by calling Algorithm 12.1 and the convex hull remains unchanged. The algorithm delivers a tree T which is feasible and $c(T) \leq 2(1+\epsilon)OPT$. Also T is optimal for $c_e + \mu^\star w_e$ for all $e \in E$ and, therefore, a supported tree in the bicriterial problem.

□

12.2 Approximation of Hassin and Levin [23]

The algorithm of Hassin and Levin published in [23] is an improvement of the algorithm of Goemans and Ravi since it has a lower time complexity by using matroid intersection. The corresponding paper is the most difficult one on the weight-constrained minimal spanning tree problem. The algorithm and underlying theory have the same structure and ideas like the algorithm of Goemans and Ravi but the advantage is to consider, as announced in Chapter 4, the weight-constrained matroid optimization problem instead of the weight-constrained minimal spanning tree problem.
A first remark to improve the algorithm of Goemans and Ravi is to use a geometric-mean binary search instead of the suggested binary-mean search which reduces the number of iterations to $\log_2 \log_{1+\epsilon} \frac{UB}{LB}$ with upper and lower bounds UB and LB on OPT.

Preliminaries
For a given $\epsilon \leq \frac{1}{4}$ we compute a $2(1+\epsilon)$-approximation by using the algorithm of Goemans and Ravi. We can now remove the set $\{e \in E : c_e > C\}$ from E where C are the costs of the approximated solution.

Partition
For given $\epsilon > 0$ and C we partition E in the following way

$$E_0 := \{e \in E : c_e < \epsilon C\}$$

and for $1 \leq i \leq I := \lceil \frac{1-\epsilon}{\epsilon^2} \rceil$ we set

$$E_i := \{e \in E : (\epsilon + (i-1)\epsilon^2)C \leq c_e < (\epsilon + i\epsilon^2)C\}.$$

We have to enumerate all possible vectors with $I+1$ non-negative entries $(n_0, n_1, \ldots, n_I)$ such that

$$\sum_{i=0}^{I} n_i = n - 1 \tag{12.1}$$

$$\sum_{i=1}^{I} n_i \leq \frac{1}{\epsilon^2} \tag{12.2}$$

Interpretation: The n_i denotes the number of edges chosen from the set E_i. Note that such a vector exists, since an optimal solution may have at most $\frac{1}{\epsilon}$ edges with costs at least ϵOPT. Suppose we omit (12.2) and consider vectors with more than $\frac{1}{\epsilon}$ elements in E_i for $1 \geq i \geq I$, then we have total costs which are greater than $(\epsilon C \frac{1}{\epsilon}) = C$ which is greater than the given upper bound and useless.

Theorem 12.4
There are $O(I^{\frac{1}{\epsilon}})$ vectors that satisfy (12.1) and (12.2).

Proof
Consider a set for each E_i, $i \geq 1$ and a set for $i = 0$ that collects $\frac{1}{\epsilon} - \sum_{i=1}^{I} n_i$ items, since in $n_1, \ldots, n_I$ only $\frac{1}{\epsilon}$ items can appear. Therefore, $\frac{1}{\epsilon}$ items have to be placed in the $I + 1$ sets. The number of possibilities is at most $(I+1)^{\frac{1}{\epsilon}} = O(I^{\frac{1}{\epsilon}})$.

□

Now we define two matroids. Like in Example 4.1 a partition matroid and the already known graphic matroid from Section 4.3:
The *graphic matroid* where a set $E' \subset E$ is called independent if and only if (V, E') does not contain a cycle and the *partition matroid* over E such that a set $E' \subset E$ is independent in this matroid if and only if $|E' \cap E_i| \leq n_i$ for all $i = 0, \ldots, I$. The goal is now to search for a subset of E which is a basis of both matroids.
We have seen that for a given matroid with an element cost function, the greedy algorithm finds a minimum cost basis on M.

Definition 12.2
A polynomial time algorithm that checks for a given $S \subset F$ whether $S \in \mathcal{F}$ is called a polynomial time independence oracle.

For an algorithm with polynomial time independence oracle, the problem of finding a minimum cost base is polynomial solvable. For a pair of matroids $M = (E, \mathcal{F})$ and $M' = (E, \mathcal{F}')$ with common polynomial time independence oracle and cost function, there exists a polynomial time algorithm computing a minimal cost common base of M and M'. For our two matroids we can compute a minimal cost base for M and M'. Both of the independence tests and the rank computations in these matroids need $O(n)$. So we find a minimum cost basis in the intersection in $O(mn^2)$ time.
Let $\mathcal{S}'$ denote the set of incidence vectors of common bases of both matroids with cardinality $n - 1$. By definition of $\mathcal{S}'$ it follows that every $S \in \mathcal{S}'$ corresponds to a $T \in \mathcal{T}$. Since the optimal solution of our weight-constrained minimal spanning tree problem is contained in $\mathcal{S}$, we can formulate:

Problem 22

$$\begin{aligned} OPT = \min & \sum_{e \in S} c_e \\ s.t. & \sum_{e \in S} w_e \leq W \\ & S \in \mathcal{S}' \end{aligned}$$

where $\mathcal{S}'$ is the set of common bases of both matroids.

And the corresponding relaxation:

Problem 23

$$C^\star(\mathcal{S}';\mu) := \min \sum_{e\in S}(c_e + \mu w_e)$$
$$s.t.\ S \in \mathcal{S}'$$

As stated above this is easy to solve by concerning the cost function $c_e + \mu w_e$ for all $e \in E$ for two matroids. This can be done in $O(mn^2)$. A lower bound on OPT is given by

$$C^\star(D3) := \max C^\star(\mathcal{S};\mu) - \mu W$$
$$s.t.\ \mu \geq 0$$

Let again $\mu^\star$ denote the maximum. This can analogously be done by computing the breakpoints mentioned in the previous algorithm. This leads to $O(m^3n^2)$.

Theorem 12.5
Let $(n_0, n_1, \ldots, n_I)$ be the number of edges from $E_0, E_1, \ldots, E_I$ in an optimal solution. Let $\mathcal{O}_{\mu^\star}$ denote the set of minimum cost common bases in the matroid intersection with respect to $c_e+\mu^\star w_e$ for all $e \in E$, if the partition matroid is defined with $n_0, n_1, \ldots, n_I$. Let $T, T' \in \mathcal{O}_{\mu^\star}$. In this case there exist a series of trees $T = T_0, T_1, \ldots, T_l = T'$ with

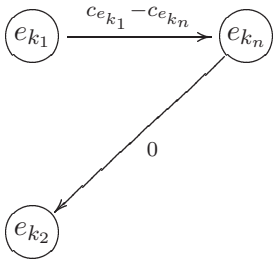
1. *$T_j \in \mathcal{O}_{\mu^\star}$ for all $j \in \{0, \ldots, l\}$*
2. *Let $E^j := T_j \setminus T_{j+1}$ and $E^{j'} = T_{j+1} \setminus T_j$. Then $|E^j \cap E_i| = |E^{j'} \cap E_i| \leq 1$ for every i and j.*

Proof by induction over $|T \setminus T'|$
Basis:
For $|T' \setminus T| = 1$ we have $|E^{0'}| = 1$ and the claim holds.
Induction hypothesis:
Assume the claim holds for $|T \setminus T'| = p - 1$.
Induction step:
Consider a cost function

$$c'_e = \begin{cases} \infty, & \text{if } e \notin T \cup T' \\ c_e + \mu^\star w_e + \epsilon', & \text{if } e \in T \setminus T' \\ c_e + \mu^\star w_e - \epsilon', & \text{if } e \in T' \setminus T \\ c_e + \mu^\star w_e, & \text{if } e \in T \cap T' \end{cases}$$

Since obviously T' is a better solution with respect to c' than T, the tree T is not the optimal solution with respect to the costs c'. A matroid intersection algorithm finds a negative cycle in an auxiliary bipartite directed graph $B(T) = (\bar{V}, \bar{V}', \bar{E})$.Where $B(T)$ is defined as: for all $e \in T$ exists a $v \in \bar{V}$ and for all $e' \in E \setminus T$ exists a $v \in \bar{V}'$. Construct for every $e_i \in T$ and $e'_j \in E \setminus T$ an arc (e_i, e'_j) with costs $c_{e_i,e'_j} = c'_{e_i} - c'_{e'_j}$ under the condition that $(T \setminus \{e_i\}) \cup \{e'_j\})$ is independent

in the graphic matroid (which guarantees the tree structure). And construct an arc (e'_j, e_i) with costs $c_{e'_j,e_i} = 0$ under the condition that $(T \setminus \{e_i\}) \cup \{e'_j\}$ is independent in the partition matroid (which guarantees that the costs are less than the upper bound.)



For $e \notin T \cup T', c'_e = \infty$, therefore a negative cycle does not contain e. From this $B(T)$ we construct a graph B by identifying the elements from $(E \setminus T) \cap E_i$ (which are nodes in $\bar{V}'$) to a single vertex v_i for all $i \in \{0, \dots, I\}$ (Parallel edges can be removed).
It can be shown that a common base T'' with smaller costs than T with respect to c' exists such that T'' is obtained from T by swapping edges along a negative cycle L in B where the number of arcs among all negative cycles is minimized by L. Let the nodes along the cycle L correspond to edges $(e_1, e'_1, e_2, e'_2, \dots e_l, e'_l, e_1)$ such that $e_k \in E_{i_k}$ for $k = 1, \dots, l$. Assume that there exists a $k \neq 1$ with $i_k = i_1$. Then there are two cycles $L' = (e_1, e'_1, e_2, e'_2, \dots, e_{k-1}, e'_{k-1}, e_1)$ and $L'' = (e_k, e'_k, e_{k+1}, \dots, e_l, e'_l, e_k)$ such that the costs of L are the sum of the costs of L' and L''. Either L' or L'' has negative costs, this contradicts the minimality of L. This leads to $T_1 := T'' \in \mathcal{O}_{\mu^\star}$. We know $T'' \subset T \cup T'$ and $T'' \neq T$. Therefore we can conclude that $|T'' \subset T'| < |T \setminus T'| = p$. By the induction assumption the claim holds.

□

Theorem 12.6
Let $n_0, \dots, n_I$ be the number of edges from $E_0, E_1, \dots, E_I$ in an optimal solution for WCMST. Let $\mathcal{O}_{\mu^\star}$ denote the set of minimum cost common bases in the intersection of the two matroids with respect to $c_e + \mu^\star w_e$ for all $e \in E$, in case that the partition matroid is defined with $n_0, n_1, \dots, n_I$. Then there exists $T \in \mathcal{O}_{\mu^\star}$ such that

$$c(T) \leq C^\star(D3) + \sum_{\{i|n_i>0\}} (\max_{e \in E_i} c_e - \min_{e \in E_i} c_e)$$

and

$$w(T) \leq W.$$

Proof
Let $\mu = \mu^\star - \epsilon'$ where $\epsilon' > 0$. There exists a $T_\geq \in \mathcal{O}_{\mu^\star}$ which is also optimal with respect to $c_e + \mu w_e$ for all $e \in E$ for sufficiently small ϵ'. Since $\mu^\star$ is the optimal value the following holds

$$c(T_\geq) + (\mu^\star - \epsilon')(w(T_\geq) - W) \leq c(T_\geq) + \mu^\star(w(T_\geq - W)) = C^\star(D3).$$

It holds that $w(T_{\geq}) \geq W$ and $c(T_{\geq}) \leq C^{\star}(D3)$. Similarly, there exists for $\mu = \mu^{\star} + \epsilon'$ a tree $T_{\leq} \in \mathcal{O}_{\mu^{\star}}$ with $w(T_{\leq}) \leq W$ and $c(T_{\leq}) \geq C$. With the theorem above there exists a series $T_{\leq} = T_0, \ldots, T_l = T_{\geq}$ in $\mathcal{O}$. Since $c(T_{\geq}) \leq C^{\star}(D3)$ and $c(T_{\leq}) \geq C^{\star}(D3)$ we can find an index j with $c(T_j) \leq C^{\star}(D3)$ and $c(T_{j+1}) \geq C^{\star}(D3)$ and $w(T_{j+1}) \leq W$. It holds

$$c(T_{j+1}) \leq c(T_j) + \sum_{\{i|n_i>0\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e).$$

□

Remark

$$\sum_{\{i|n_i>0\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e) \leq 4(1+\epsilon)\epsilon OPT \leq 5\epsilon OPT$$

Proof

$$\begin{aligned}
\sum_{\{i|n_i>0\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e) &= \max_{e\in E_0} c_e - \min_{e\in E_0} c_e + \sum_{\{i|n_i>0, i\geq 1\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e) \\
&\leq \epsilon C - \min_{e\in E_0} c_e + \sum_{\{i|n_i>0, i\geq 1\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e) \\
&\leq \epsilon C + \sum_{\{i|n_i>0, i\geq 1\}} (\max_{e\in E_i} c_e - \min_{e\in E_i} c_e) \\
&\leq 2(1+\epsilon)OPT + \sum_{\{i|n_i>0, i\geq 1\}} (\max_{e\in E_i} c_e - \min_{e\in E_i}) \\
\end{aligned}$$

In the summation there are at most $\frac{1}{\epsilon}$ elements. Each of them is not greater than $\epsilon^2 C \leq 2(1+\epsilon)\epsilon^2 OPT$. Together this gives the result. The second inequality follows from the assumption that $\epsilon \leq \frac{1}{4}$.

□

Procedure 12.4 Approximation of Hassin, Levin: Procedure

Require: $T_{\geq}$ and $T_{\leq}$

repeat

construct B by identifying the set $\{e'_j \in T' : (T \setminus \{e_i\}) \cup \{e'_j\}$ is independent in the graphic matroid$\}$

Find a negative cycle with a minimum number of edges.

Identify the next tree T

until $w(T) \leq W$

The construction of B (contains $O(n)$ vertices and $O(\frac{n}{\epsilon})$ edges) can be done in $O(\frac{n^2}{\epsilon})$ by identifying for each e_i the set $\{e'_j \in T' : (T \setminus \{e_i\}) \cup \{e'_j\}$ is independent in the graphic matroid$\}$. The second step can be done in $O(\frac{n^2}{\epsilon})$ by the Bellman-Ford algorithm. The number of trees in the series is $O(n)$. Therefore, we have $O(n)$ steps and get a complexity of $O(\frac{n^3}{\epsilon})$.

Algorithm 12.5 Approximation of Hassin, Levin: MAIN

Compute $\tilde{C}$ with $OPT \leq \hat{C} \leq 2(1+\epsilon)OPT$ by Goemans and Ravi (Algorithm 12.3)
Enumerate all possible vectors $(n_0, n_1, \ldots, n_I)$
Compute for each of them $\mu^\star$
Find for this $\mu^\star$ a tree with satisfies the condition of the theorem
5: Take the next vector $(n_0, n_1, \ldots n_I)$ and compute for them Procedure 12.4
Over all these trees pick the tree with minimal $c(T)$

For the correct $(n_0, n_1, \ldots, n_I)$ we have a minimal cost base of OPT and this leads by the algorithm to a feasible tree with costs at most $(1 + 5\epsilon)OPT$. The computing of $\tilde{C}$ takes $O((m+n\log n)\log^2 n \log\log_{1+\epsilon} n)$. The enumeration takes, by the consideration above, $O((\frac{1}{\epsilon^2})^{\frac{1}{\epsilon}})$. The solution of the Lagrangian relaxation can be done in $O(m^2n^2 + \frac{n^4}{e^2})$ and the finding of the spanning tree in $O(\frac{n^3}{\epsilon})$. The total complexity is $O(\frac{1}{\epsilon^2}^{\frac{1}{\epsilon}(m^2n^2)})$. This leads in total to the next theorem.

Theorem 12.7
A $(1+\epsilon)$-approximation can be found in $O((O\frac{1}{\epsilon^2})^{O(\frac{1}{\epsilon})}(m^2n^2))$.

Improve Complexity

For solving of the Lagrangian relaxation we can make the following improvement: For every $\mu \geq 0$ and $i \geq 0$ an edge $e \in E_i$ that participates in the minimum cost base of two matroids belongs to a minimum cost (respect to $c_e + \mu w_e$ for all $e \in E$) spanning forest $F_i(\mu)$ of (V, E_i). We define a set of potential breakpoints as $\{\mu | \exists e, e' \in E, c_e + \mu w_e = c_{e'} + \mu w_{e'}\}$. In a first step we compute all breakpoints. Then we apply a binary search over this set and find an interval (μ_1, μ_2) containing $\mu^\star$ and such that no $F_i(\mu)$ has a breakpoint in this interval besides $\mu^\star$. Then we compute $F_i(\bar{\mu})$ for every i with $n_i > 0$ for some $\bar{\mu} \in (\mu_1, \mu_2)$. Remove from the matroids all the edges not lying in $\bigcup_{i|n_i>0} F_i(\bar{\mu})$. In this set there are $O(\frac{n}{\epsilon})$ elements. The total complexity is $O(\frac{mn\log n}{\epsilon})$. In general our complexity reduces to $O((O\frac{1}{\epsilon^2})^{O(\frac{1}{\epsilon})}(n^4))$.

12.3 Neighborhood-Search of Yamada, Watanabe and Kataoka [39]

In the publication of Yamada, Watanabe and Kataoka [39] also a heuristic for the weight-constrained maximum spanning tree problem is described which can easily be transformed to the weight-constrained minimal spanning tree problem. The goal is to find a feasible solution obtained from the Lagrangian dual. Furthermore, they only claim without proof that this algorithm is a 2-approximation. Therefore we present the algorithm in this chapter and not in Section 6.3.

Definition 12.3 *Neighborhood*
A spanning tree T' is called a neighbor of T if T and T' are adjacent. We call $N(T) := \{T' | T'$ is a feasible neighbor of $T\}$ the neighborhood.

Let $T_{\tilde{\mu}}$ is represented by an extreme point on the border of the convex hull which is feasible and has smallest costs. We start from this $T_{\tilde{\mu}}$ and search in the neighborhood for a better solution.

Algorithm 12.6 Yamada, Watanabe, Kataoka: Local Search

Start with $T := T_{\tilde{\mu}}$
while There exists $T' \in N(T)$ such that $c(T') < c(T)$ **do**
  $T := T'$
end while

It is easy to see that the algorithm does in general not find the optimal solution since we have seen in Example 6.1 that the set of efficient solutions is not necessarily connected and no edge exchange can lead to the optimal solution. The remaining question is whether this fact can destroy the claimed 2-approximation.

12.4 Fully Polynomial Bicriteria Approximation [27]

In the exact algorithm using the Theorem 11.13 we have the time complexity of $O(mn^5\tau(UB, W))$ which is for large values of UB and W not very useful. Hong, Chung and Park [27] published an approximation scheme by scaling the costs and weights to improve $\tau(UB, W)$.

Scaling costs and weights

Lemma 12.8
Let $\hat{T}^\star$ be an optimal solution of the WCMST with each c_e scaled to $\hat{c}_e = \lfloor c_e(n-1)/(\epsilon C) \rfloor$ where $C \in \mathbb{N}$.

1. *$c(\hat{T}^\star) < OPT + \epsilon C$*
2. *If $\hat{c}(\hat{T}^\star) > \lfloor \frac{n-1}{\epsilon} \rfloor$, then we have $OPT > C$.*
3. *If $\hat{c}(\hat{T}^\star) \leq \lfloor \frac{n-1}{\epsilon} \rfloor$, then we have $OPT < (1+\epsilon)C$.*

Proof

1. The definition
$$\hat{c_e} = \lfloor \frac{c_e(n-1)}{\epsilon C} \rfloor$$
implies that
$$\hat{c_e} \leq \frac{c_e(n-1)}{\epsilon C}.$$

This can be formulated to

$$c_e \geq \hat{c}_e \frac{\epsilon C}{n-1}.$$

And we can conclude that

$$0 \leq c_e - \hat{c}_e \frac{\epsilon C}{n-1}$$

We know further from the definition of $\hat{c}_e$ that

$$\hat{c}_e < c_e \frac{n-1}{\epsilon C} + 1.$$

This leads to

$$c_e - \hat{c}_e \frac{\epsilon C}{n-1} < \frac{\epsilon C}{n-1}.$$

In total we have

$$0 < c_e - \hat{c}_e \frac{\epsilon C}{n-1} < \frac{\epsilon C}{n-1}.$$

Since every $T \in \mathcal{T}$ has $n-1$ edges we have

$$0 \leq c(T) - \hat{c}(T) \frac{\epsilon C}{n-1} < \epsilon C \tag{12.3}$$

and

$$c(\hat{T}^\star) < \hat{c}(\hat{T}^\star) \frac{\epsilon C}{n-1} + \epsilon C$$

By definition of the optimal solution we get

$$c(\hat{T}^\star) < \hat{c}(T^\star) \frac{\epsilon C}{n-1} + \epsilon C$$

For $T^\star$ inequality (12.3) leads to

$$\hat{c}(T^\star) \frac{\epsilon C}{n-1} \leq c(T^\star) = OPT \tag{12.4}$$

This gives

$$c(\hat{T}^\star) < OPT + \epsilon C$$

2. Let

$$\hat{c}(\hat{T}^\star) > \lfloor \frac{n-1}{\epsilon} \rfloor.$$

Since $\hat{c}(T^\star) \geq \hat{c}(\hat{T}^\star)$ and $\hat{c}(\hat{T}^\star) \in \mathbb{N}$ we get

$$\hat{c}(T^\star) \geq \hat{c}(\hat{T}^\star) > \frac{n-1}{\epsilon}$$

If we insert $T^\star$ in (12.3), we get

$$\hat{c}(T^\star) \leq c(T^\star) \frac{n-1}{\epsilon C}$$

We combine these two results to

$$\frac{n-1}{\epsilon} < c(T^\star) \frac{n-1}{\epsilon C}$$

and the claim

$$C < c(T^\star) = OPT$$

holds.

3. Let
$$\hat{c}(\hat{T}^\star) \leq \lfloor \frac{n-1}{\epsilon} \rfloor$$
If we insert $\hat{T}^\star$ in (12.3), we get
$$c(\hat{T}^\star) < \hat{c}(\hat{T}^\star)\frac{\epsilon C}{n-1} + \epsilon C$$
With the assumption
$$\begin{aligned} OPT &= c(T^\star) \\ &\leq c(\hat{T}^\star) \\ &\leq \hat{c}(\hat{T}^\star)\frac{\epsilon C}{n-1} + \epsilon C \\ &\leq \frac{\epsilon C}{n-1}\lfloor \frac{n-1}{\epsilon} \rfloor + \epsilon C \\ &\leq C(1+\epsilon) \end{aligned}$$

□

Theorem 12.9
If there exist some upper and lower bounds U and L, respectively, with $\frac{U}{L} \leq \rho$ for some constant $\rho \geq 1$, then a $(1, 1+\epsilon)$-approximation can be computed in $O(mn^5\tau(\lfloor \frac{n-1}{\epsilon} \rfloor, B))$.

Proof
We consider again a scaled problem with costs
$$\hat{c}_e = \lfloor c_e \frac{n-1}{\epsilon L} \rfloor.$$
Notice that we can apply the previous lemma. Let as above $\hat{T}^\star$ be the optimal tree to the scaled costs.
$$\hat{c}(T^\star) = \sum_{e \in T^\star} \lfloor c_e \frac{n-1}{\epsilon L} \rfloor \leq \lfloor \sum_{e \in T^\star} c_e \frac{n-1}{\epsilon L} \rfloor = \lfloor \frac{OPT(n-1)}{\epsilon L} \rfloor \leq \lfloor \frac{U(n-1)}{\epsilon L} \rfloor \leq \lfloor \frac{\rho(n-1)}{\epsilon} \rfloor.$$
This has the complexity $O(\lfloor \frac{n-1}{\epsilon} \rfloor)$! From the previous chapter we know that $\hat{c}(\hat{T})$ can be found in $O(n^4\tau(\lfloor \frac{n-1}{\epsilon} \rfloor, B))$ by computing $\det(\lfloor \frac{\rho(n-1)}{\epsilon} \rfloor, B, \hat{c}, w)$. So we can determine $\hat{T}^\star$ in $O(mn^5\tau(\lfloor \frac{n-1}{\epsilon} \rfloor, W))$.
By the lemma we know that $c(\hat{T}) < OPT + \epsilon L \leq (1+\epsilon)OPT$.

□

The advantage of considering the scaled problem is the polynomial time complexity. Remaining question is how to find appropriate bounds and reduce the time complexity further.

We can reduce the time complexity if we scale the weights to $\hat{w}_e := \lfloor w_e \frac{n-1}{\delta W} \rfloor$. In the following we compute determinants for the scaled weights only up to $\hat{W} = \lfloor \frac{n-1}{\delta} \rfloor$ in the degree of y.

Lemma 12.10

1. *Every feasible solution for the original solution is also feasible for the scaled problem (i.e.* $\{T \in \mathcal{T} | w(T) \leq W\} \subset \{T \in \mathcal{T} | \hat{w}(T) \leq \frac{n-1}{\delta}\}$*)*
2. *If* $\hat{w}(T) \leq \lfloor \frac{n-1}{\delta} \rfloor$ *then we have* $w(T) < (1+\delta)W$*.*

Proof

1. Let $T \in \mathcal{T}$ with $w(T) \leq W$.
$$\hat{w}(T) = \sum_{e \in T} \hat{w}_e = \sum_{e \in T} \lfloor w_e \frac{n-1}{\delta W} \rfloor = \lfloor \frac{n-1}{\delta W} \rfloor \sum_{e \in T} w_e \leq \lfloor \frac{n-1}{\delta W} \rfloor W = \lfloor \frac{n-1}{\delta} \rfloor$$
2. This follows from Lemma 12.8 if we replace $c, \hat{c}, C$ and ϵ by $w, \hat{w}, W$ and δ.

□

The w-scales problem has a larger feasible solution set than the original problem and a $(1, 1+\delta)$-approximation for this problem is a $(1+\epsilon, 1+\delta)$-approximation for the WCMST. So we can state the Algorithm 12.7.

Algorithm 12.7 Approximation of Hong, Chung, Park: Bicriteria FPTAS

Require: $G = (V, E), c, w, W, \epsilon, \delta$

Ensure: A spanning tree $\hat{T}$ with $c(\hat{T}) \leq (1+\epsilon)OPT$ and $w(\hat{T}) \leq (1+\delta)W$

$\hat{w}_e := \lfloor w_e \frac{n-1}{\delta W} \rfloor$
$\hat{W} := \lfloor \frac{n-1}{\delta} \rfloor$
while $\frac{U}{L} > 2$ **do**
  $C :\approx \sqrt{LU}$
5: $\hat{c}_e = \lfloor c_e \frac{n-1}{\epsilon C} \rfloor$
  if $\det(\lfloor \frac{n-1}{\epsilon} \rfloor, \hat{W}, \hat{c}, \hat{w}) = 0$ **then**
    $L := C$
  else
    $U := (1+\epsilon)C$
10: **end if**
end while
$c_e := \lfloor c_e \frac{n-1}{\epsilon L} \rfloor$
Find optimal $\hat{c}(\hat{T}^\star)$ by computing $\det(\hat{K}^{xy} \lfloor \frac{2(n-1)}{\epsilon} \rfloor, \hat{W}, \hat{c}, \hat{w})$
Construct $\hat{T}^\star$ with the Algorithm 11.10

Remark and explanations

The usage of the geometric search guarantees that $\frac{U}{L} \leq 2$ can be found in $O(\log\log(\frac{U_0}{L_0}))$ where U_0 and L_0 are initial bounds. If the determinant in line 6 is 0, there exists no tree with $\hat{c}(T) \leq \lfloor \frac{n-1}{\epsilon} \rfloor$ and from the lemma it follows that $OPT > C$. Therefore we replace the lower bound by C.

Complexity of the algorithm

The given algorithm finds a $(1+\epsilon, 1+\delta)$-approximate solution of the constrained minimum spanning tree problem in $O(\log\log(\frac{U_0}{L_0})n^4\tau(\lfloor\frac{n-1}{\epsilon}\rfloor,\lfloor\frac{n-1}{\delta}\rfloor) + mn^5\tau(\lfloor\frac{n-1}{\epsilon}\rfloor,\lfloor\frac{n-1}{\delta}\rfloor))$. This results is obtained from the fact, that the binary search has a complexity $O(\log\log(\frac{U_0}{L_0}))$ and in each iteration one determinant has to be computed which has the complexity $O(n^4\tau(\lfloor\frac{n-1}{\epsilon}\rfloor,\lfloor\frac{n-1}{\delta}\rfloor))$ and the Algorithm 11.10 needs $O(mn^5\tau(\lfloor\frac{n-1}{\epsilon}\rfloor,\lfloor\frac{n-1}{\delta}\rfloor))$. As one can see, the time complexity depends on good initial bounds $\frac{U_0}{L_0}$. Some easy upper bounds are $U_0 := (n-1)\max c_e$ and $L_0 := 1$. An other approach is the following one:

Algorithm 12.8 Approximation of Hong, Chung, Park: Starting solution

Sort the edges in nondecreasing order

Let $C_1 < C_2 < \cdots < C_l$ be the distinct edge costs. Define G_i as the subgraph with costs of at most C_i.

Find the minimum index $\hat{i}$ such that $G_{\hat{i}}$ has a spanning tree with weights in $\hat{w}$ of at most $\hat{W}$.

$L_0 := C_{\hat{i}}$

5: $U_0 := (n-1)C_{\hat{i}}$.

By using a binary search on the C_i's we can found the $\hat{i}$ in $O(n^2\log m)$ by computing in each step a spanning tree in $O(n^2)$. This procedure leads to a ratio $\frac{U_0}{L_0} = n-1$. In our algorithm the computational effort of the binary search reduces to $O(\log\log n)$.

12.5 Approximation through Decomposition

We show in this section a further approximation approach by using the packing algorithm for spanning trees of Gabow and Manu [16] to obtain a set of (directed-in-)trees and constructing from this set an approximation based on the idea of multicriteria approximation through decomposition of Burch, Krumke, Marathe, Phillips and Sundberg [6]. This approximation is more of theoretical interest.

LP - Relaxation

For this algorithm we need directed-in-trees rooted at node n which correspond according to Lemma 3.2 to undirected minimal spanning trees. Firstly, we need the LP - Relaxation for our problem outgoing from our formulation Problem 5.

Problem 24 (LP)

$$C_{LP} = \min \sum_{(i,j)\in\overrightarrow{E}} c_{ij}y_{ij} \tag{12.5}$$

$$s.t. \sum_{\substack{j\in V \\ (i,j)\in\overrightarrow{E}}} y_{ij}^k - \sum_{\substack{j\in V \\ (j,i)\in\overrightarrow{E}}} y_{ji}^k = \begin{cases} 1 & \text{if } i = k \\ -1 & \text{if } i = n \; \forall i \in V, \forall k \in V\backslash\{1\} \\ 0 & \text{else} \end{cases} \tag{12.6}$$

$$y_{ij}^k \leq y_{ij} \; \forall (i,j) \in \overrightarrow{E} \; \forall k \in V\backslash\{n\} \tag{12.7}$$

$$y_{ij}^k \geq 0 \; \forall (i,j) \in \overrightarrow{E} \; \forall k \in V\backslash\{n\} \tag{12.8}$$

$$\sum_{\substack{j\in V \\ (i,j)\in\overrightarrow{E}}} y_{ij} = 1 \; \forall i \in V\backslash\{n\} \tag{12.9}$$

$$\sum_{(i,j)\in\overrightarrow{E}} w_{ij}y_{ij} \leq W \tag{12.10}$$

$$y_{ij} \leq 1 \; \forall (i,j) \in \overrightarrow{E} \tag{12.11}$$

$$y_{ij} + y_{ji} \leq 1 \; \forall (i,j) \in E \tag{12.12}$$

$$y_{ij} \geq 0 \; \forall (i,j) \in E \tag{12.13}$$

$$y_{ij} \in [0,1] \forall (i,j) \in \overrightarrow{E} \tag{12.14}$$

We denote this solution $\tilde{T}_{LP}$ with $\tilde{y}_{ij}$.

Remark
Since for $X := \{x \in \mathbb{Z}^m | (3.2) - (3.5)\}$ it holds that $\text{conv}(X) = \{x \in \mathbb{R}^m | (3.2) - (3.5)\}$ we conclude from a fundamental result of integer programming that $C_{LP} = C^\star(D1)$ where $C^\star(D1)$ is the objective value of the Lagrangian dual.

Edmonds Decomposition

From this $\tilde{y}_{ij}$ we construct now a so called Edmonds decomposition of $\mathcal{T}_n$: A set of trees $\overrightarrow{T}_1, \ldots, \overrightarrow{T}_K \in \mathcal{T}_n$ with factors $\alpha_1, \ldots, \alpha_K \geq 0$ such that $\sum_{k=1}^K \alpha_k = 1$: First we delete all edges from the graph with $\tilde{y}_{ij} = 0$. Here we have to distinguish between $\tilde{y}_{ij}$ and $\tilde{y}_{ji}$ which might be both positive. Before starting we have to introduce some technical definitions.

Definition 12.4

1. *Two sets A and B are called intersecting if $A \cap B \neq \emptyset, A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$.*
2. *A family $\mathcal{F}$ of subsets of V is called laminar if for every pair $A, B \in \mathcal{F}$ either $A \cap B = \emptyset$, $A \subset B$ or $B \subset A$ holds.*

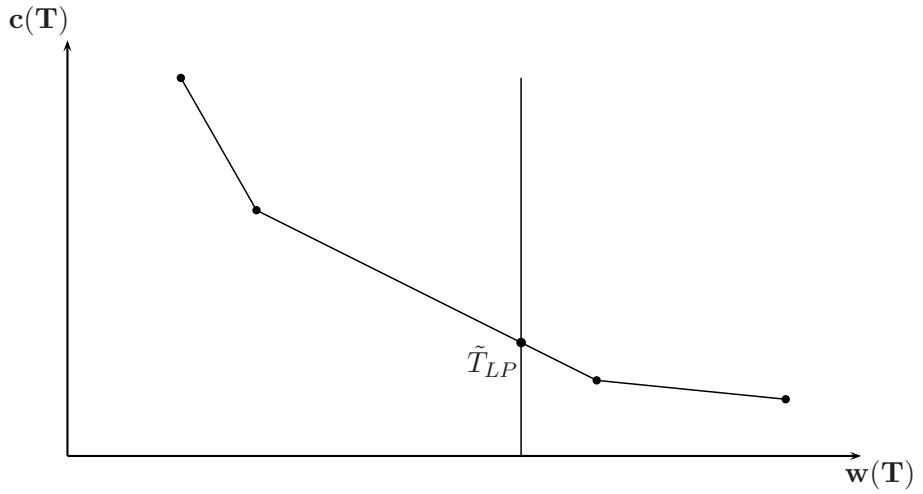


Figure 12.3: LP - relaxation

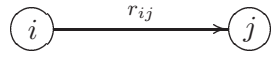
Definition 12.5
Let $\overrightarrow{G} = (V, \overrightarrow{E})$ and for each $e = (i, j)$ exist a capacity r_{ij} with $0 \leq r_{ij} \leq 1$.

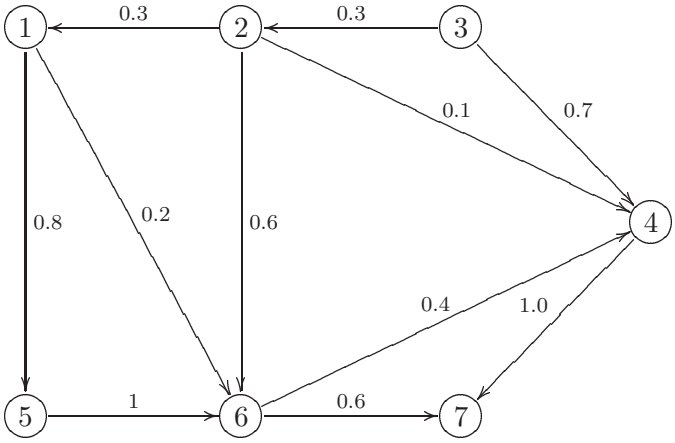
1. *For a set $D \subset V$ we denote $p_G(D) := \sum_{e=(i,j)\in E, i\in D, j\in V\setminus D} r_{ij}$.*
2. *We define for a graph $g(G) := \min\{p_G(D) | \emptyset \neq D \subset V \setminus \{1\}\}$.*
3. *For a tree $\overrightarrow{T}$ we define $r(\overrightarrow{T}) := \min\{r_{ij} |\ (i, j) \in \overrightarrow{T}\}$.*
4. *Let $\overrightarrow{G} - \beta\overrightarrow{T}$ denote the graph with capacity $r_{ij} - \beta_{ij}$ for $(i, j) \in \overrightarrow{T}$ and r_{ij} for all edges not in $\overrightarrow{T}$.*
5. *For a tree $\overrightarrow{T}$ the capacity of $\overrightarrow{T}$ is defined as*

$$\alpha(\overrightarrow{T}) := \{\max \alpha | s.t.\ \alpha \leq r(\overrightarrow{T}), g(\overrightarrow{T}) - \alpha = g(\overrightarrow{G} - \alpha\overrightarrow{T}), \alpha \geq 0\}.$$

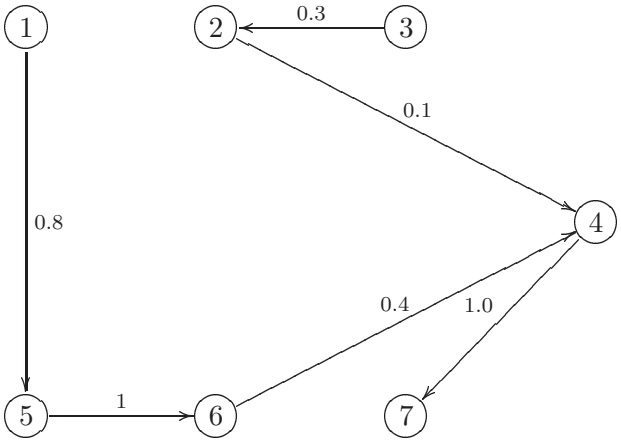
Let us start with $r_{ij} := \tilde{y}_{ij}$ for all $(i, j) \in \overrightarrow{E}$. For understandability of these definitions we give an example.

Example 12.1
Let in this example $n = 7$.

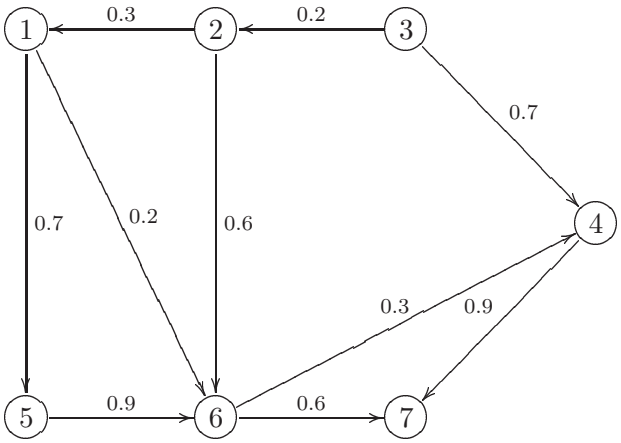




For the set $D = \{1, 2, 3\}$ we have $p_G(D) = 0.8 + 0.2 + 0.6 + 0.1 + 0.7 = 2.4$. Let us consider the following tree $\overrightarrow{T}$.



In this tree we have $r(\overrightarrow{T}) = 0.1$ and $G - 0.1\overrightarrow{T}$ leads to the following graph.



By equation (12.9) we have for every set $D \subset V \setminus \{n\}$ with $|D| = 1$ that $p_G(D) = 1$. For every set $D \subset V \setminus \{n\}$ with $|D| > 1$ that $p_G(D) \geq 1$ since a unit of flow has to be send from i to n for each $i \in D$. So $g(G) = 1$.
Next we define to construct our decomposition a so called uncrossing step. Let $\mathcal{F}$ a laminar family of sets each of them not containing the vertex n. This $\mathcal{F}$ contains at most $2n-3$ elements (the sets

$$\{1\}, \{2\}, \dots \{n-1\}, \{1,2\}, \{1,2,3\}, \dots, \{1,2,\dots,n-1\}).$$

Let $U \subset V \setminus \{n\}$ and $p_G(U) = g(G)$.

Procedure 12.9 Gabow, Manu: Uncross

Require: $U \subset V \setminus \{n\}$ with $p_G(U) = g(G)$
$X := \arg\min\{|B||B \in \mathcal{F} \cup \{V \setminus \{n\}\}, U \subset B\}$
$YI :=$ maximal subsets of X in $\mathcal{F}$ intersecting with U
$Z := U \cup \bigcup\{Y|Y \in YI\}$
$\mathcal{F} := \mathcal{F} \cup Z$

Lemma 12.11 *[16]*
The obtained $\mathcal{F} \cup Z$ is laminar.

Proof
Let $A \in \mathcal{F}$ before the uncrossing step. We have to show that either $A \cap Z = \emptyset$, $A \subset Z$ or $Z \subset A$. Therefore we consider a case distinction for the relation of A and U.

1. case: $A \cap U = \emptyset$: Then we have

   $$A \cap Z = A \cap (U \cup \bigcup\{Y|Y \in YI\}) = (A \cap U) \cup \bigcap(\{Y|Y \in YI\} \cap A) = \bigcap(\{Y|Y \in YI\} \cap A)$$

   Since all Y are elements of $\mathcal{F}$ we know that either $A \cap Y = \emptyset$ or $A \subset Y$. The case $Y \subset A$ cannot occur. Otherwise, this contradicts $A \cap U$. Now, if one Y exists with $A \subset Y$ then $A \subset Z$. Otherwise $A \cap Z = \emptyset$.

2. case: $A \cap U \neq \emptyset$ and $A \setminus U = \emptyset$. Then $A \subset U$ and $A \subset Z$.

3. case: $A \cap U \neq \emptyset$ and $U \setminus A = \emptyset$ then $U \subset A$. Then for all $Y \in YI$ it must either hold that A intersects with Y or $A \subset Y$ or $Y \subset A$. The first case cannot occur since A and Y are elements of $\mathcal{F}$ and therefore non intersecting. The last case cannot occur since Y is maximal. If the second case holds for at least one Y, we have $A \subset Z$.

4. case: Let A and U be intersecting. Then A is equal to a Y or A is a subset of a Y and so $A \subset Z$.

So $\mathcal{F} \cup Z$ is laminar.

□

If Z was already element of $\mathcal{F}$, the uncrossing procedure does not change $\mathcal{F}$. Next we have to make a statement concerning $p_G(D)$.

Lemma 12.12
For $A, B \subset V \setminus \{n\}$ we have

$$p_G(A) + p_G(B) = p_G(A \cup B) + p_G(A \cap B) + d_G(A, B) \tag{12.15}$$

where $d_G(A, B)$ is the total capacity of all edges having one end in $A \setminus B$ and the other in $B \setminus A$.

Proof

1. Let $A \cap B = \emptyset$. Then $p_G(A) + p_G(B)$ is the sum of all capacities of edges (i, j) with $i \in A$ and $j \in V \setminus A$ and the capacities of edges having one node in B and the other in $V \setminus B$. This is equal to the capacities of edges having one node in $A \cup B$ and the other in $V \setminus (A \cup B)$ and the capacities of edges between A and B. And thus the claim holds.

2. Let $A \subset B$. Then $p_G(A \cap B) = p_G(A)$ and $p_G(A \cup B) = p_G(B)$ and $d_G(A, B) = d_G(A \setminus B, B \setminus A) = d_G(\emptyset, B \setminus A) = 0$. And the claim holds.

3. Let A and B be intersecting. Then the capacity of A and B is the sum of the capacities of the union of both sets. In this union the capacities of edges between $A \setminus B$ and $B \setminus A$ are not contained, so we need $d_G(A, B)$. Additionally, in this set no capacities between A and $A \cap B$ are included such that the term $p_G(A \cap B)$ is necessary.

□

Now we need a lemma which becomes relevant in the proof of the validity of the decomposition algorithm.

Lemma 12.13 *[16]*
Let Y and Y' be two sets in YI in the uncrossing procedure for U. Then

$$p_G(Z) = p_G(Y \cap U) = g(G) \text{ and } d_G(Y, U) = d_G(Y \setminus U, Y' \setminus U) = 0.$$

Proof
Let us consider (12.15) for $Y \cup U$ and U, we get since $p_G(U) = g(G)$

$$p_G(Y \cup U) + g(G) = p_G(Y \cup U) + p_G(U) = p_G(Y \cup U \cap U) + p_G(Y \cup U \cup U) + d_G(U \cup Y, U)$$

This leads to

$$p_G(Y \cup U) + p_G(U) = p_G(Y \cap U) + p_G(Y \cup U) + d_G(Y, U)$$

Finally, we have

$$g(G) = p_G(U) = p_G(Y \cap U) + d_G(Y, U).$$

Since $g(G)$ is minimal $g(G) = p_G(U) = p_G(Y \cap U)$ and $d_G(Y, U) = 0$. Analogously, $p_G(Y \cup U) = g(G)$ and the same properties for Y' hold. We consider $Y \cup U$ and $Y' \cup U$

$$\begin{aligned} 2g(G) &= p_G(Y \cup U) + p_G(Y' \cup U) \\ &= p_G(Y \cup U \cup Y' \cup U) + p_G((Y \cup U) \cap (Y' \cup U)) + d_G(Y \cup U, Y' \cup U). \end{aligned}$$

The last term is equal to $d_G(Y \setminus U, Y' \setminus U)$ since all edges from U to U are not considered. This leads to

$$2g(G) = p_G(Y \cup Y' \cup U) + p_G(U \cup (Y \cap Y')) + d_G(Y \setminus U, Y' \setminus U).$$

By the minimality property of $g(G)$ we have $d_G(Y \setminus U, Y' \setminus U) = 0$. Since $p_G(Y \cup U) = g(G)$ we can repeat the same considerations for $Y \cup U$ and any element of YI. By repeating this procedure we get

$$p_G(Z) = p_G(U \cup \bigcup \{Y | Y \in YI\}) = g(G).$$

□

With this result we can state the decomposition algorithm.

Algorithm 12.10 Gabow, Manu: Decomposition-algorithm (Greedy - approach)

$\mathcal{F} := \{\{v\} | p_G(v) = g(G)\}$
$k := 0$
while $g(G) > 0$ **do**
  $k := k + 1$
  $\overrightarrow{T}_k :=$ spanning tree of G with $p_{\overrightarrow{T}}(X) = 1$ for every $X \in \mathcal{F}$
  $\alpha_k :=$ capacity of $\overrightarrow{T}$
  $G := G - \alpha_k \overrightarrow{T}_k$
  if $\alpha_k(\overrightarrow{T}_k) = r(\overrightarrow{T}_k)$ **then**
    $U :=$ minimal set in $U \subset V \setminus \{n\}$ with $p_G(U) = g(G)$ and $p_{\overrightarrow{T}}(U) > 1$
  end if
  uncross $\mathcal{F} \cup U$
  $g(G) := g(G) - \alpha_k$
end while

Lemma 12.14 *[16]*
It is always possible to construct a $\overrightarrow{T}_k$ *in the algorithm.*

Proof
For any $X \in \mathcal{F} \cup \{V\}$ let G_X denote the subgraph induced by the set of vertices X. We start with a spanning tree $\overrightarrow{T}$ in G_V. For every maximal set X of $\mathcal{F}$ the tree $\overrightarrow{T}$ has an unique edge directed to a node $b \in X$. Now we proceed recursively for each $X \setminus b$.
For every set S in G_X there exists with $b \notin S$ exists an edge (i, j) with $i \in X \setminus S$ and $j \in S$. This

is clear for $X = V$ since $g(G) > 0$. For $X \neq V$ we know that $p_G(S) \geq g(G) = p_G(X)$ therefore, such an (i, j) has to exist.

□

The validity of the Algorithm 12.10 which constructs the Edmonds decomposition can be seen in the next theorem.

Theorem 12.15 *[16]*
The algorithm finds a decomposition with at most m distinct trees.

Proof
If $\alpha_k = r(\overrightarrow{T}_k)$ then $G - \alpha_k \overrightarrow{T}_k$ deletes one edge from G. This occurs at most $(m - (n-1) + 1)$-times. Since after deleting $m - (n-1)$ edges only one possible tree can be chosen which vanishes in the last step. We know that $\mathcal{F}$ contains at most $2n - 3$ elements and by initialization n elements. So $\mathcal{F}$ can be enlarged only $(n-2)$-times. Therefore, the number of iterations is at most $(m - n + 2) + (n - 2) = m$. We have to show that in each run with $\alpha_k < r(\overrightarrow{T}_k)$ the uncross operation enlarges $\mathcal{F}$ i.e., $Z \notin \mathcal{F}$ where Z is found by Procedure 12.9.
Claim: $p_{\overrightarrow{T}}(Z) \geq p_{\overrightarrow{T}}(U)$ where Z is the result of the uncrossing procedure and U was found in the decomposition algorithm.
We know by definition of U that $p_{\overrightarrow{T}}(U) > 1$ and $p_{\overrightarrow{T}}(X) = 1$ for all $X \in \mathcal{F}$. So if the claim holds Z could not be in $\mathcal{F}$, since otherwise this contradicts the minimality property of U.
We consider a set $Y \in YI$. From Lemma 12.13 we know that $p_G(Y \cap U) = g(G)$. Since $p_{\overrightarrow{T}}(U) > 1$ $p_{\overrightarrow{T}}(Y \cap U) = 1$, otherwise the minimality of U would be violated.
We have also for $Y, Y' \in YI$

$$d_{\overrightarrow{T}}(Y, U) = d_{\overrightarrow{T}}(Y \setminus U, Y' \setminus U) = 0.$$

This follows by the fact that this relation holds for d_G in Lemma 12.13 and every edge of $\overrightarrow{T}$ is element of G.
To prove the claim it is sufficient to show that an edge (i, j) leaving U corresponds to an unique edge leaving Z. If (i, j) leaves both sets, the correspondence is clear. Consider an edge (i, j) with $i \in U$ and $j \in V \setminus U$ but $i \notin Z$. By construction of Z, (i, j) enters $Y \setminus U$. Since $d_{\overrightarrow{T}}(Y \setminus U, Y' \setminus U) = d_{\overrightarrow{T}}(Y, U) = 0$ this edge can only enters $Y \cap U$. Since $p_{\overrightarrow{T}}(Y \cap U) = 1$ this edge is unique. Some edge $(l, v) \in \overrightarrow{T}$ must exist with $l \in Y$ and $v \notin Y$. Since $d_T(Y, U) = 1$ the edge (l, v) has a node in $V \setminus Y$ and in $Y \setminus U$. From $d_T(Y, U) = d_T(Y - U, Y' - U) = 0$ we know that (l, v) has a node in $V \setminus Z$. Since (l, v) has a node in Z we can let (i, j) correspond to (l, v). This corresponding is unique. So the claim holds and the theorem is valid.

□

Corollary 12.16 *[16]*
The decomposition algorithm runs in $O(n^3 \log(\frac{n^2}{m}))$.

Proof
A tree $\overrightarrow{T}$ can be found by the recursion of 12.14 in $O(m)$. Suppose $\alpha_k(\overrightarrow{T}_k) = r(\overrightarrow{T}_k)$ the capacity can be found in $O(nm\log(\frac{n^2}{m}))$. Since there are at most $m \leq n^2$ iterations, the time complexity holds for these iterations. For the case $\alpha_k < r(\overrightarrow{T}_k)$ the capacity can be found in $O(n^2 m \log \frac{n^2}{m})$. Also the computing of U can be done in $O(n^2)$ by enumeration (test each $X \in \mathcal{F}$). There are at most n iterations with $\alpha_k < r(\overrightarrow{T}_k)$. So this is $O(n^3 m \log(\frac{n^2}{m}))$ and the claim holds.

□

We make now a statement for the computation of the capacity. An alternative definition of the capacity is that it is the largest value $\alpha \leq r(\overrightarrow{T})$ such that for each set $U \in V \setminus \{n\}$,

$$p_{G-\alpha(\overrightarrow{T})}(U) \geq g(G) - \alpha. \tag{12.16}$$

The left side is obviously equal to $p_G(U) - \alpha p_{\overrightarrow{T}}(U)$. If $p_{\overrightarrow{T}}(U) > 1$ then the inequality holds if and only if

$$\alpha_U := \frac{p_G(U) - g(G)}{p_{\overrightarrow{T}}(U) - 1} \geq \alpha.$$

If $p_{\overrightarrow{T}}(U) = 1$, inequality (12.16) holds for every α. So we can formulate a procedure for computing the capacity.

Procedure 12.11 Gabow, Manu: Capacity

$\alpha := r(\overrightarrow{T})$
while $g(G - \alpha\overrightarrow{T}) < g(G) - \alpha$ **do**
  $U := \arg\min_{U \subset V \setminus \{n\}} p_{G-\alpha\overrightarrow{T}}(U)$
  $\alpha := \alpha_U$
end while
α is the required capacity

Lemma 12.17 *[16]*
The procedure is valid.

Proof
We know that $p_{G-\alpha\overrightarrow{T}}(U) \geq g(G) - \alpha$ for $\alpha \leq \alpha_U$. In each iteration the value α_U changes to a smaller α_U. So the sequence of the values α is strictly decreasing during the procedure.
Claim: For all sets $X \subset V \setminus \{n\}$ with $p_{\overrightarrow{T}}(X) \geq p_{\overrightarrow{T}}(U)$ where U is found in an iteration we have $\alpha_X \geq \alpha_{\overrightarrow{T}}$.
Let α be given and U be computed. From the definition of U we have

$$p_G(X) - \alpha p_{\overrightarrow{T}}(X) \geq p_G(U) - \alpha p_{\overrightarrow{T}}(U).$$

Since $\alpha_U < \alpha$ and $p_{\overrightarrow{T}}(X) \geq p_{\overrightarrow{T}}(U)$

$$p_G(X) - \alpha_U p_T(X) \geq p_G(U) - \alpha p_T(U) = g(G) - \alpha_U.$$

We can reformulate this inequality to

$$\alpha_U \leq \frac{p_G(X) - g(X)}{p_T(X) - 1} = \alpha_X.$$

This implies that the sequence of the $p_{\overrightarrow{T}}(U)$ is strictly decreasing. Otherwise, the claim leads to a contradiction to the fact that the values for α are strictly decreasing. Since $p_{\overrightarrow{T}}(U) \leq |\overrightarrow{T}|$ at most $n-1$ iterations are needed.

□

Gabow and Manu state that the capacity α of $\overrightarrow{T}$ can be computed in $O(n^2 m \log \frac{n^2}{m})$. For the proof of this statement the authors refer to a global minimum cut algorithm of Hao and Orlin which finds U in $O(nm \log \frac{n^2}{m}))$. Since $n-1$ iterations are needed the time complexity is $O(n^2 m \log \frac{n^2}{m})$.

Approximation

The decomposition delivers a set $\{\overrightarrow{T}_1, \ldots, \overrightarrow{T}_K\} \subset \mathcal{T}_n$ with coefficients $\alpha_1, \ldots, \alpha_K$ with $\sum_{k=1}^{n} \alpha_k = 1$. We use our Lemma 3.2 and skip the orientation of our edges and get a set $\{T_1, \ldots, T_K\} \subset \mathcal{T}$. We know further that if an edge $(i,j) \in \overrightarrow{T}_k$ the edge (j,i) is not element of this tree. Otherwise $\overrightarrow{T}$ is not spanning. If we define $\tilde{x}_e := \tilde{y}_{ij} + \tilde{y}_{ji}$ for all $e \in E$ where $e = \{i,j\}$, we know from the construction of the α_k that

$$\sum_{k|e=\{i,j\} \in T_k} \alpha_k = \sum_{k|(i,j) \in \overrightarrow{T_k}} \alpha_k + \sum_{k|(i,j) \in \overrightarrow{T_k}} \alpha_k \leq \tilde{y}_{ji} + \tilde{y}_{ji} = \tilde{x}_e.$$

It holds also that $c(\tilde{T}_{LP}) = \sum_{e \in E} c_e \tilde{x}_e$ and $w(\tilde{T}_{LP}) = \sum_{e \in E} w_e \tilde{x}_e$. We apply now the more general result of Burch, Krumke, Marathe, Phillips and Sunberg [6] to our decomposition.

Theorem 12.18
For our decomposition $\{T_1, \ldots, T_K\} \subset \mathcal{T}$ with $\sum_{k|e \in T_k} \alpha_k \leq \tilde{x}_e$ and $\sum_{k=1}^{K} \alpha_k = 1$ we have

1. $$\sum_{k=1}^{K} \alpha_k c(T_k) \leq c(\tilde{T}_{LP})$$

2. $$\sum_{k=1}^{K} \alpha_k w(T_k) \leq w(\tilde{T}_{LP})$$

Proof

1. $$\sum_{k=1}^{K} \alpha_k c(T_k) = \sum_{k=1}^{K} \alpha_k \sum_{e \in T_k} c_e = \sum_{e \in E} \sum_{k|e \in T_k} \alpha_k c_e \leq \sum_{e \in E} c_e \tilde{x}_e = c(\tilde{T}_{LP})$$

2.

$$\sum_{k=1}^{K} \alpha_k w(T_k) = \sum_{k=1}^{K} \alpha_k \sum_{e \in T_k} w_e = \sum_{e \in E} \sum_{k | e \in T_k} \alpha_k w_e \leq \sum_{e \in E} w_e \tilde{x}_e = w(\tilde{T}_{LP})$$

□

Theorem 12.19 *[6]*
For the decomposition $\{T_1, \ldots, T_K\} \subset \mathcal{T}$ *and corresponding 'coefficients'* $\alpha_k \geq 0$ *and* $\sum_{k=1}^{K} \alpha_k = 1$ *with* $\sum_{k=1}^{K} \alpha_k c(T_k) \leq c(\tilde{T}_{LP})$ *and* $\sum_{k=1}^{K} \alpha_k w(T_k) \leq w(\tilde{T}_{LP})$ *where* $\tilde{T}$ *is the solution of the linear program we can find a solution* T *such that*

$$\frac{w(T)}{W} + \gamma \frac{c(T)}{OPT} \leq 1 + \gamma.$$

Proof
We can interpret the α_k as probabilities for the 'events' T_k. Consider the choice of a T from the α distribution

$$\begin{aligned} E[\frac{w(T)}{W} + \gamma \frac{c(T)}{c(\tilde{T}_{LP})}] &= \frac{E[w(T)]}{W} + \gamma \frac{E[c(T)]}{c(\tilde{T}_{LP})} \\ &\leq \frac{w(\tilde{T}_{LP})}{W} + \gamma \frac{c(\tilde{T})}{c(\tilde{T}_{LP})} \\ &= 1 + \gamma. \end{aligned}$$

By a basic principle of the probalistic method we know that there exists a realization which is less or equal than the expectation. (Provided that the expectation exists.) So there is a tree T with

$$\frac{w(T)}{W} + \gamma \frac{c(T)}{c(\tilde{T})} \leq 1 + \gamma$$

Since $OPT \geq c(\tilde{T})$ the claim holds.

□

Corollary 12.20
A solution given by the previous theorem is either a $(1 + \frac{1}{\gamma}, 1)$*-approximation or a* $(1, 1 + \gamma)$*-approximation.*

Proof

- The theorem leads to

$$\frac{c(T)}{OPT} \leq \frac{1 + \gamma - \frac{w(T)}{W}}{\gamma} = \frac{1}{\gamma} + 1 - \frac{1}{\gamma} \frac{w(T)}{W} \leq 1 + \frac{1}{\gamma}$$

  The last inequality follows from the fact that $\gamma > 0, w(T) \geq 0$ and $W \geq 0$.

- On the other hand the theorem leads to

$$\frac{w(T)}{W} \leq 1 + \gamma - \gamma \frac{C(T)}{OPT} \leq 1 + \gamma$$

  The last inequality follows from the fact that $\gamma \geq 0, c(T) \geq 0$ and $OPT \geq 0$.

- Suppose we have found a tree T that satisfies the theorem with $\frac{w(T)}{W} = 1+a$ and $\frac{c(T)}{OPT} = 1+b$ where a and b are strictly positive.

$$\frac{w(T)}{W} + \gamma \frac{c(T)}{OPT} \leq 1 + \gamma$$

  By the definition above this is

$$1 + a + \gamma(1 + b) \leq 1 + \gamma,$$

  which is equivalent to

$$a + \gamma b \leq 0.$$

  This contradicts the assumption that a, b, γ are greater than zero. Therefore, in one component we have an 1-approximation.

□

Remark

We do not know which of these two approximations our T satisfies. We only know that one tree in the decomposition satisfies the approximation. A feasible tree with lowest costs in $\{T_1, \ldots, T_K\}$ is not necessarily a $\frac{1}{\gamma}$-approximation.

13 Numerical Results

The main goal in this chapter is to compare the existing branch and bound schemes of Aggarwal, Aneja and Nair (Algorithm 11.2) and Shogan (Algorithm 11.3) with the new algorithm of Ruzika and Henn (Algorithm 11.4). Additionally, we make some statements on the exact algorithm of Hong, Chung and Park (Algorithm 11.10) and on the quality of the approximation algorithm of Goemans and Ravi (Algorithm 12.3) with $\epsilon = 1$. We combine this algorithm with the idea of finding the adjacent trees on the border of the convex hull by performing pivot operations. We forgo to implement the approximation of Hassin and Levin (Algorithm 12.5) since this was only an improvement of Algorithm 12.3, the decomposition-algorithm described in Section 12.5 since this idea is more relevant from a theoretical point of view and the exact ranking Algorithm 11.14.

13.1 Implementation

We realize our tests with C++ using the Boost Graph Library (BGL) [36]. For computing a minimal spanning tree for a parametric cost function we use the algorithm of Kruskal which was already implemented in the BGL. Since in the algorithm of Hong, Chung and Park we need to calculate a determinant of a matrix with polynomial entries (which can be interpreted as elements of the ring $\mathbb{Z}[x, y]$) the software SINGULAR, a computer algebra system for polynomial computations with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory developed at the University of Kaiserslautern Department of Mathematics and Center for Computer Algebra by Greuel, Pfister and Schönemann [20] is used. All tests are carried out on a 2x 86_64 AMD Opteron workstation.

13.2 Test Structure

Four parameters effect the result of an optimal solution: the number of nodes n in the graph, the number of edges m, the distribution of costs and weights and the choice of the constraint W. For n we choose $10, 50, 100, 150, 200, 250, 300, 350$ and 400 nodes. For the number of edges in the graph we considered a complete graph with $m = n(n-1)/2$, a graph with $m = n(n-1)/4$ and a graph with $m = n(n-1)/8$. For the costs and weights we use four different distributions:

- *uniform:* We choose uniformly distributed costs and weights in $\{1, \ldots, 100\}$.
- *outliers:* We choose costs and weights with probability 0.9 in $\{101, \ldots, 200\}$ and with probability 0.1 in $\{1, \ldots, 100\}$.

- *weak correlation:* In the third option the costs are uniformly distributed in $\{1,\dots,100\}$ and $w_e = \max\{1, X - 0.5c_e\}$ where X is uniformly distributed in $\{1,\dots,100\}$. This leads to a correlation of circa -0.4.

- *high correlation:* Our costs are again uniformly distributed in $\{1,\dots,100\}$ and $w_e = -c_e + 110 + \beta$ where β is uniformly in $\{-10,\dots,10\}$. This leads to a correlation of circa -0.9.

Also we choose the same distributions with data in $\{1,\dots,1000\}$. In detail:

- *uniform:* We choose uniformly distributed costs and weights in $\{1,\dots,1000\}$.

- *outliers:* Our costs and weights lie with probability 0.9 in $\{1001,\dots,2000\}$ and with probability 0.1 in $\{1,\dots,1000\}$.

- *weak correlation:* The costs are uniformly distributed in $\{1,\dots,1000\}$ and $w_e = \max\{1, X - 0.5c_e\}$ where X is uniformly distributed in $\{1,\dots,1000\}$.

- *high correlation:* The costs are again uniformly distributed in $\{1,\dots,1000\}$ and $w_e = -c_e + 1020 + \beta$ where β is uniformly in $\{-20,\dots,20\}$.

To obtain our constraint we compute first the weight of the lexicographical weight minimum W_2 and the weight of the lexicographical cost minimum W_1. For our constraint we choose a low limit $(W_1 + W_2)/4$, a medium limit $(W_1 + W_2)/2$ and a high limit $3(W_1 + W_2)/4$. Together this leads to $9 \times 3 \times 4 \times 2 \times 3 = 648$ different problem settings. Since the results for instances with data distributed in $\{1,\dots,100\}$ are not very meaningful we do not run our algorithms for settings with more than 150 nodes, and consider only 468 settings. For each setting we generate 40 different graphs.

13.3 Results

We have to mention that a time difference need not only be caused in the advantage of the underlying algorithmic ideas but also in a lack of implementation.
The algorithm of Shogan (Algorithm 11.3) runs for one 'easy' setting with $n = 20$ and $m = 190$ uniformly distributed costs and weights in $\{1,\dots,100\}$ more than 15 minutes and we ignore the algorithm for larger settings. The Algorithm 11.10 implemented with SINGULAR does not deliver comparable results since it runs for an instance with $n = 10$ and $m = 45$ and uniformly distributed costs and weights in $\{1,\dots,100\}$ more than 20 minutes and further computations are not useful. So we focus our considerations on the algorithm of Ruzika and Henn, the algorithm of Aggarwal, Aneja and Nair and the approximation scheme of Goemans and Ravi. For larger problems we omit running an algorithm for a problem setting if the algorithm in the next smaller setting has instances with more than 10 minutes run time. In total we have more than 523 hours of simulation.
A first result which can be seen in all settings is that our own algorithm has a significant smaller

run time than the algorithm of Aggarwal, Aneja and Nair (the time advantage is not less than one order of magnitude and for complex settings the advantage exceeds two orders of magnitude). The main reason for this advantage is the fact that in each branching of Algorithm 11.2 a large number of minimal spanning trees has to been established. More precisely, the algorithm has to spend a lot of time in sorting all edges in each computation of a minimal spanning tree. In contrast the algorithm of Ruzika and Henn updates in each iteration already existing trees. The run time of the algorithm of Shogan where a much larger number of spanning trees has to been computed and the approximation of Goemans and Ravi where we compute also a large number of spanning trees flesh out this observation. A second factor for the time advantage is the number of branchings in the Algorithm of Ruzika and Henn which is in the most cases smaller than the number of branchings of the Algorithm of Aggarwal, Aneja and Nair. Remember, Aggarwal, Aneja and Nair branch from an extreme point, Ruzika and Henn from a supported tree which is next to the weight constraint. The Algorithm of Goemans and Ravi delivers in our test an excellent approximation quality of circa 1.01. An open question is whether this quality can also be guaranteed by instances with a breather distribution of costs and weights. The large run time on this approximation is caused in the large number of running the algorithm again for different C, which does not change the approximation value in the most cases (see Theorem 12.3).
Further problems with data in $\{1,\dots,1000\}$ are more complicated than problems with data in $\{1,\dots,100\}$: In a setting with data distributed in $\{1,\dots,100\}$ the initial triangle and the whole convex hull is smaller than for a problem in the corresponding setting with data distributed in $\{1,\dots,1000\}$. The polyhedra is dispersed. Here our duality gap is larger and our algorithms has to spend more time in the computation. We investigate for graphs with more than 150 vertices only our problem settings where costs and weights are distributed in $\{1,\dots,1000\}$. Theses graphs are as announced in the previous section more significant.
By classifying the complexity of our different distributions we see that the distribution with outliers has the largest run time. The second complicated setting is the setting with highly correlated data. The problems with weakly correlated costs and weights have the lowest run time since a large set of edges with costs and respectively weights equal to 1 making the problem easier. We notice that this relation is reflected in the run time of every algorithm, we consider here. The Figure 13.1 visualizes this behavior in the settings with data in $\{1,\dots,1000\}$ for the medium weight constraint and the algorithm of Ruzika and Henn.
We number our settings for this figure in the following way:

	Vertices	Edges		Vertices	Edges		Vertices	Edges
1	50	307	7	150	2794	13	250	6219
2	50	612	8	150	5588	14	250	12438
3	50	1225	9	150	11175	15	250	24875
4	100	1238	10	200	4975	16	300	11213
5	100	2475	11	200	9950	17	300	22425
6	100	4950	12	200	19900	18	300	44850

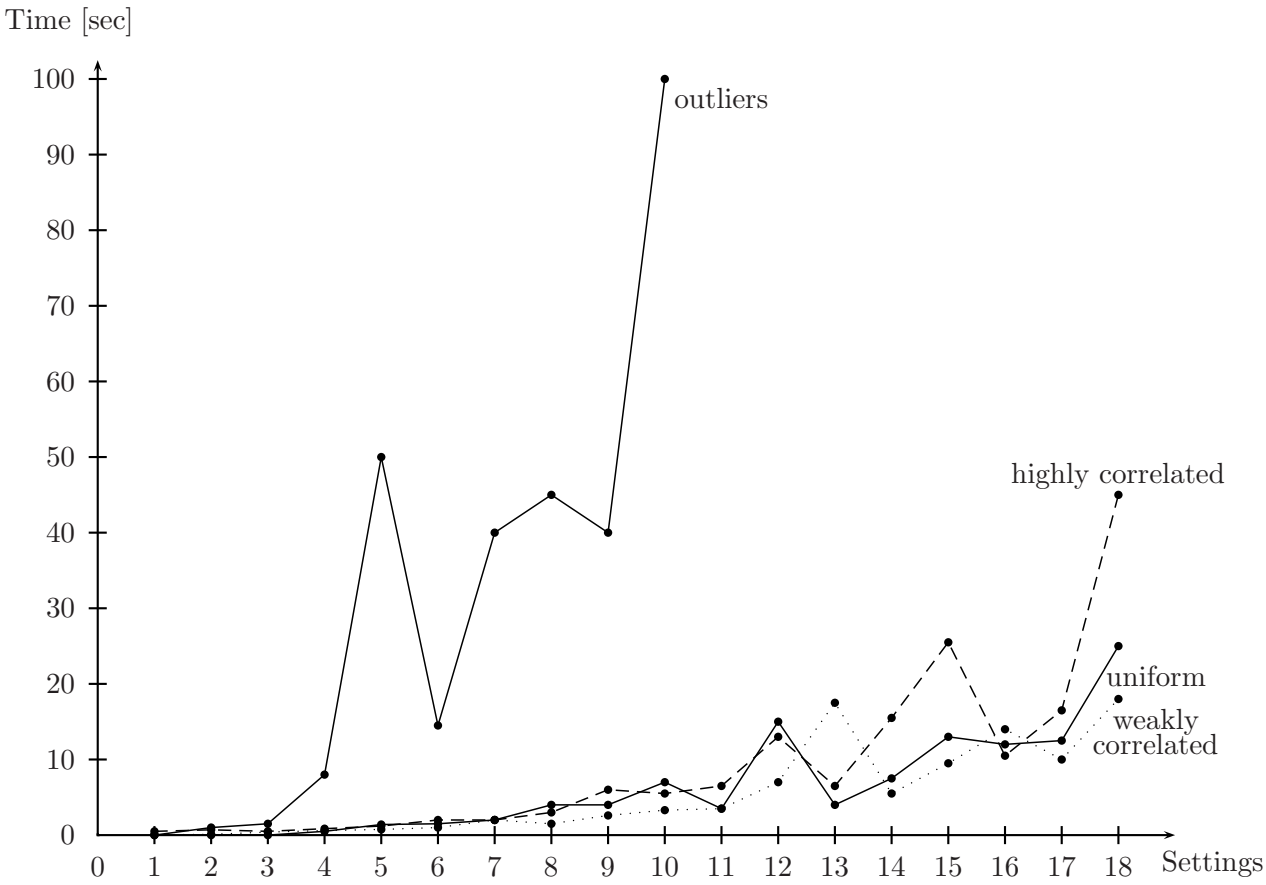


Figure 13.1: Average run time of the Algorithm of Ruzika and Henn for medium constraint in the different settings

It is not very useful to visualize our results in diagrams since we have a great dispersion between the different results for the algorithms and settings. Therefore, we present our results only in 24 tables. In the following we use these abbreviations:

$\mathbf{t_r}$: The average time of the Algorithm of Ruzika and Henn in seconds.

$\mathbf{B_r}$: The average number of branchings of the Algorithm of Ruzika and Henn.

$\mathbf{t_a}$: The average time of the Algorithm of Aggarwal, Aneja and Nair in seconds.

$\mathbf{B_a}$: The average number of branchings of the Algorithm of Aggarwal, Aneja and Nair.

$\mathbf{t_a/t_r}$: The quotient of the average time of the Algorithm of Aggarwal, Aneja and Nair and the average time of the Algorithm of Ruzika and Henn.

$\mathbf{B_a/B_r}$: The quotient of the average number of branchings of the Algorithm of Aggarwal, Aneja and Nair and the average number of branchings of the Algorithm of Ruzika and Henn.

$\mathbf{t_g}$: The average time of the Approximation of Goemans and Ravi in seconds.

α : The average approximation quality.

$\mathbf{t_g/t_r}$: The quotient of the average time of the Algorithm of Goemans and Ravi and the average time of the Algorithm of Ruzika and Henn.

Uniformly Distributed Costs and Weights

A very interesting result is that we observe in our algorithm for small n a smaller run time for a complete graph than for a graph with the same number of nodes but only $n(n-1)/4$ edges. This is clear since a complete graph has a closer distribution of the images in the $c-w$-space. The probability that the facets of our convex hull have a slope of -1 is much greater and we have more trees with weight equal W. This property of a complete graph can also be seen in the number of branchings in the algorithm of Aggarwal, Aneja and Nair. This advantage deflagrates in the run time since we have to sort more edges by the computing of minimal spanning trees. In the Algorithm of Ruzika and Henn and in the Algorithm of Goemans and Ravi we start with the weight minimal solution and pivot along the frontier of the convex hull until we reach the weight-constrained. In the low-weight limit case this occurs sooner than in the high-weight limit case and we have to spend more time for finding a solution.

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	16.3	0.00	11.8		0.72	0.00	1.0287	-
10	45	0.00	29.9	0.01	19.6	51.0	0.66	0.00	1.0414	-
50	307	0.01	148.2	1.31	293.0	95.6	1.98	0.02	1.0146	1.31
50	612	0.03	124.3	1.35	163.6	42.9	1.32	0.08	1.0153	2.50
50	1225	0.04	56.5	2.23	146.0	55.8	2.59	0.21	1.0151	5.24
100	1238	0.11	154.5	13.43	590.8	121.5	3.82	0.37	1.0087	3.33
100	2475	0.16	59.4	16.85	348.6	106.5	5.87	1.39	1.0089	8.79
100	4950	0.30	30.1	24.52	215.1	80.9	7.16	3.20	1.0075	10.54
150	2794	0.44	142.5	61.59	768.8	140.8	5.40	2.51	1.0060	5.73
150	5588	0.61	39.0	95.80	429.6	156.5	11.02	6.68	1.0048	10.90
150	11175	1.39	10.1	179.79	405.4	129.0	40.34	15.48	1.0030	11.11

Table 13.1: Low-weight limit for uniformly distributed costs and weights in $\{1,\ldots,100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	14.2	0.01	12.7	-	0.90	0.00	1.0380	-
10	45	0.00	23.3	0.01	19.5	-	0.84	0.00	1.0214	-
50	307	0.01	78.1	0.97	195.3	120.8	2.50	0.02	1.0092	2.88
50	612	0.02	52.5	1.05	107.4	63.4	2.05	0.10	1.0130	5.97
50	1225	0.03	22.0	1.87	85.0	65.0	3.87	0.25	1.0131	8.69
100	1238	0.08	72.1	13.48	399.3	177.9	5.54	0.47	1.0059	6.16
100	2475	0.16	34.7	15.82	205.1	96.4	5.92	1.67	1.0068	10.18
100	4950	0.33	6.8	30.72	139.3	93.6	20.63	3.69	1.0040	11.24
150	2794	0.30	29.1	44.49	339.0	150.6	11.67	3.10	1.0037	10.50
150	5588	0.67	9.7	121.66	378.6	182.5	39.03	7.94	1.0038	11.92
150	11175	1.67	5.3	222.59	245.6	133.5	46.05	17.92	1.0027	10.75

Table 13.2: Medium-weight limit for uniformly distributed costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	9.7	0.00	7.8	-	0.81	0.00	1.0182	-
10	45	0.00	14.5	0.01	11.5	27.0	0.79	0.00	1.0197	-
50	307	0.01	49.3	0.47	80.5	55.7	1.63	0.03	1.0080	3.15
50	612	0.02	52.5	0.79	81.1	53.6	1.54	0.11	1.0090	7.36
50	1225	0.03	13.9	1.17	49.5	38.9	3.56	0.27	1.0056	9.01
100	1238	0.05	21.2	6.67	179.4	125.3	8.48	0.53	1.0040	9.86
100	2475	0.17	10.1	13.30	116.3	80.6	11.51	1.81	1.0039	10.98
100	4950	0.37	3.6	30.09	89.5	82.5	25.22	3.97	1.0029	10.89
150	2794	0.31	11.4	49.20	267.1	157.2	23.53	3.41	1.0027	10.90
150	5588	0.74	5.9	119.00	269.3	160.5	45.64	8.55	1.0021	11.53
150	11175	1.83	2.2	252.43	248.1	137.9	112.75	19.09	1.0010	10.43

Table 13.3: High-weight limit for uniformly distributed costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	17.1	0.00	10.6	-	0.62	0.00	1.0273	-
10	45	0.00	37.4	0.01	19.8	-	0.53	0.00	1.0314	-
50	307	0.03	287.2	1.18	265.6	34.47	0.92	0.03	1.0157	0.88
50	612	0.11	337.0	2.63	343.9	23.33	1.02	0.10	1.0194	0.86
50	1225	0.23	308.0	3.34	221.3	14.81	0.72	0.30	1.0164	1.30
100	1238	0.81	865.5	17.19	891.6	21.28	1.03	0.56	1.0089	0.69
100	2475	1.55	680.4	30.54	766.2	19.74	1.13	1.78	1.0089	1.14
100	4950	1.48	320.5	56.91	634.3	38.50	1.98	4.54	1.0083	3.07
150	2794	2.92	1047.5	87.70	1562.6	30.05	1.49	3.42	1.0061	1.17
150	5588	5.43	787.7	188.23	1458.4	34.65	1.85	8.91	1.0060	1.64
150	11175	4.40	378.4	-	-	-	-	23.96	1.0052	5.44
200	4975	4.28	794.6	286.66	2202.4	67.06	2.77	10.89	1.0349	2.55
200	9950	6.95	527.3	757.75	2334.4	109.08	4.43	29.11	1.0045	4.19
200	19900	9.10	211.8	1701.22	2767.6	186.98	13.07	71.924	1.0041	7.91
250	6219	7.83	763.4	-	-	-	-	27.055	1.0023	3.46
250	12438	10.45	419.7	-	-	-	-	69.605	1.0034	6.66
250	24875	17.04	184.3	-	-	-	-	165.02	1.0038	9.69
300	11213	14.62	184.3	-	-	-	-	56.99	1.0028	3.90
300	22425	15.97	187.4	-	-	-	-	144.30	1.0034	9.04
300	44850	25.01	123.9	-	-	-	-	326.88	1.0027	13.07
350	15269	16.86	528.2	-	-	-	-	106.19	1.0023	6.30
350	30538	20.94	178.3	-	-	-	-	259.32	1.0021	12.38
350	61075	46.19	74.6	-	-	-	-	-	-	-
400	19950	29.30	649.1	-	-	-	-	-	-	-
400	39900	35.36	147.9	-	-	-	-	-	-	-
400	79800	73.07	45.7	-	-	-	-	-	-	-

Table 13.4: Low-weight limit for uniformly distributed costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	15.8	0.00	12.0	-	0.76	0.03	1.0324	-
10	45	0.00	36.1	0.02	26.7	-	0.74	0.00	1.0417	-
50	307	0.02	189.9	1.00	212.6	42.72	1.12	0.04	1.0135	1.63
50	612	0.05	175.1	1.66	186.5	32.22	1.06	0.12	1.0110	2.36
50	1225	0.20	260.8	4.01	252.6	20.07	0.97	0.35	1.0134	1.76
100	1238	0.43	463.1	15.55	692.9	36.44	1.50	0.72	1.0089	1.68
100	2475	1.46	567.5	30.99	651.0	21.17	1.15	2.17	1.0067	1.48
100	4950	1.72	278.5	94.05	961.7	54.77	3.45	5.33	1.0071	3.10
150	2794	1.92	700.6	120.73	1715.6	62.92	2.45	4.19	1.0056	2.19
150	5588	4.15	460.1	274.62	1430.0	66.20	3.11	10.66	1.0047	2.57
150	11175	3.80	200.2	-	-	-	-	27.30	1.0049	7.20
200	4975	7.38	927.1	409.65	2581.4	55.48	2.78	13.30	1.0052	1.80
200	9950	3.53	183.0	2041.31	4714.2	578.89	25.77	34.40	1.0043	9.76
200	19900	15.15	215.8	-	-	-	-	77.96	1.0039	5.14
250	6219	4.81	392.5	-	-	-	-	32.25	1.0036	6.71
250	12438	7.83	183.2	-	-	-	-	81.28	1.0033	10.39
250	24875	13.81	66.5	-	-	-	-	186.98	1.0031	13.55
300	11213	12.19	574.1	-	-	-	-	67.13	1.0024	5.51
300	22425	12.57	110.0	-	-	-	-	165.06	1.0028	13.13
300	44850	25.43	23.0	-	-	-	-	366.77	1.0020	14.42
350	15269	10.15	110.3	-	-	-	-	143.18	1.0020	12.31
350	30538	21.81	100.9	-	-	-	-	297.11	1.0022	13.62
350	61075	50.45	30.6	-	-	-	-	-	-	-
400	19950	19.90	251.2	-	-	-	-	-	-	-
400	39900	37.46	39.3	-	-	-	-	-	-	-
400	79800	83.68	31.0	-	-	-	-	-	-	-

Table 13.5: Medium-weight limit for uniformly distributed costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	11.8	0.00	7.4	-	0.62	0.00	1.0165	-
10	45	0.00	13.0	0.01	16.0	-	1.23	0.00	1.0376	-
50	307	0.02	165.8	0.78	169.2	40.47	1.02	0.05	1.0065	2.35
50	612	0.05	159.4	1.18	138.9	22.14	0.87	0.14	1.0071	2.55
50	1225	0.11	128.8	2.99	150.9	28.25	1.17	0.38	1.0089	3.60
100	1238	0.22	59.4	15.57	550.1	70.54	9.26	0.81	1.0089	3.65
100	2475	0.45	159.7	30.72	504.4	68.38	3.16	2.39	1.0039	5.31
100	4950	0.69	106.2	75.91	473.4	110.13	4.46	5.74	1.0046	8.33
150	2794	2.31	493.9	100.60	1196.9	43.54	2.42	4.65	1.0033	2.01
150	5588	5.90	328.7	211.81	899.7	35.92	2.74	11.63	1.0041	1.97
150	11175	2.10	33.0	-	-	-	-	29.26	1.0034	13.92
200	4975	3.44	401.0	607.85	2616.6	176.92	6.53	14.59	1.0024	4.25
200	9950	3.22	81.4	3566.86	5270.6	1107.81	64.79	37.26	1.0024	11.57
200	19900	6.14	21.9	-	-	-	-	80.45	1.0025	13.11
250	6219	4.91	284.7	-	-	-	-	35.17	1.0021	7.16
250	12438	6.53	71.0	-	-	-	-	87.45	1.0021	13.39
250	24875	14.02	18.1	-	-	-	-	198.48	1.0023	14.15
300	11213	5.71	101.8	-	-	-	-	73.21	1.0020	12.83
300	22425	12.70	51.6	-	-	-	-	177.38	1.0022	13.97
300	44850	27.36	9.4	-	-	-	-	388.38	1.0020	14.19
350	15269	10.15	110.3	-	-	-	-	135.22	1.0015	13.32
350	30538	22.56	41.7	-	-	-	-	318.83	1.0015	14.13
350	61075	55.83	16.7	-	-	-	-	-	-	-
400	19950	17.53	72.2	-	-	-	-	-	-	-
400	39900	41.11	34.6	-	-	-	-	-	-	-
400	79800	89.98	9.0	-	-	-	-	-	-	-

Table 13.6: High-weight limit for uniformly distributed costs and weights in $\{1, \ldots, 1000\}$

Uniformly Distributed Costs and Weights with Outliers

This distribution is the most time expensive one in relation to our other distributions. We guess that our algorithm runs slower in this setting since we have a disadvantageous adjacency-structure, where it was not possible to exchange edges with small costs and weights by edges with small costs and weights. Also the outliers may make the frontier of the convex hull more complicated which enlarges the run time. Additionally, in the cases with higher weight limit we have a lower number of branchings since we can choose more edges with cheap costs, the outliers, which does not effect the feasibility than in the low-limit case.

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	16.8	0.00	9.6	-	0.57	0.00	1.0160	-
10	45	0.00	25.6	0.00	16.6	-	0.65	0.00	1.0289	-
50	307	0.02	211.9	0.98	209.7	43.70	0.99	0.02	1.0090	0.86
50	612	0.13	476.0	2.29	277.5	17.12	0.58	0.40	1.0122	3.00
50	1225	0.37	597.3	8.31	592.6	22.34	0.99	0.15	1.0115	0.40
100	1238	0.63	901.9	24.44	1210.2	38.55	1.34	0.28	1.0063	0.44
100	2475	1.72	972.6	35.60	859.0	20.68	0.88	0.91	1.0073	0.53
100	4950	4.22	1154.1	125.26	1575.4	29.69	1.37	2.41	1.0074	0.57
150	2794	1.49	692.4	150.13	2588.1	100.88	3.74	1.70	1.0059	1.14
150	5588	3.39	790.0	419.80	3410.2	123.83	4.32	4.69	1.0053	1.38
150	11175	8.66	781.6	-	-	-	-	12.32	1.0049	1.42

Table 13.7: Low-weight limit for costs and weights with outliers in $\{1, \ldots, 200\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	12.1	0.00	8.5	-	0.70	0.00	1.0101	-
10	45	0.00	27.1	0.00	17.7	-	0.65	0.00	1.0218	-
50	307	0.02	197.3	0.86	174.5	46.61	0.88	0.02	1.0090	1.04
50	612	0.13	517.1	2.65	331.7	20.31	0.64	0.08	1.0115	0.59
50	1225	0.32	579.8	6.48	442.8	20.56	0.76	0.20	1.0144	0.64
100	1238	0.46	667.9	19.70	907.5	43.27	1.36	0.37	1.0055	0.82
100	2475	2.26	1271.7	52.04	1369.6	23.02	1.08	1.25	1.0072	0.55
100	4950	3.84	1003.3	131.34	1633.6	34.21	1.63	3.07	1.0106	0.80
150	2794	1.13	578.6	130.14	2056.0	114.99	3.55	2.36	1.0052	2.09
150	5588	3.29	756.2	439.65	3332.1	133.75	4.41	6.24	1.0063	1.89
150	11175	4.34	373.3	-	-	-	-	15.94	1.0085	3.68

Table 13.8: Medium-weight limit for costs and weights with outliers in $\{1, \ldots, 200\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	7.2	0.00	5.3	-	0.74	0.00	1.0044	-
10	45	0.00	14.3	0.01	11.4	-	0.80	0.00	1.0108	-
50	307	0.01	111.8	0.47	100.3	36.71	0.90	0.03	1.0040	2.22
50	612	0.04	163.1	1.25	151.5	28.75	0.93	0.09	1.0088	2.05
50	1225	0.24	328.7	3.97	264.4	16.33	0.80	0.24	1.0196	0.99
100	1238	0.22	227.5	8.22	331.4	37.03	1.46	0.43	1.0038	1.95
100	2475	1.55	572.1	36.30	823.3	23.42	1.44	1.52	1.0092	0.98
100	4950	4.16	341.6	43.60	454.0	10.49	1.33	3.81	1.0090	0.92
150	2794	1.21	423.8	74.85	1190.8	62.08	2.81	2.84	1.0047	2.36
150	5588	4.70	562.1	156.22	1182.1	33.24	2.10	7.60	1.0072	1.62
150	11175	2.13	110.1	-	-	-	-	19.58	1.0055	9.20

Table 13.9: High-weight limit for costs and weights with outliers in $\{1, \ldots, 200\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	16.9	0.03	7.8	-	0.46	0.00	1.0119	-
10	45	0.00	48.0	0.10	16.1	40.00	0.34	0.00	1.0199	3.00
50	307	0.07	521.9	1.12	233.1	15.58	0.45	0.03	1.0094	0.42
50	612	0.48	1514.0	8.03	955.3	16.81	0.63	0.07	1.0124	0.15
50	1225	1.75	2788.3	26.27	1790.7	15.02	0.64	0.19	1.0139	0.11
100	1238	2.93	3150.8	84.76	3530.8	28.95	1.12	0.42	1.0089	0.14
100	2475	12.01	4747.1	210.05	4682.1	17.49	0.99	1.16	1.0089	0.10
100	4950	96.11	45146.7	-	-	-	-	3.22	1.0092	0.03
150	2794	19.81	5023.5	-	-	-	-	2.16	1.0053	0.11
150	5588	164.59	36453.9	-	-	-	-	6.34	1.0054	0.04
150	11175	1060.87	179253.9	-	-	-	-	15.60	1.0098	0.01
200	4975	55.79	7565.0	-	-	-	-	6.58	1.0041	0.12
200	9950	708.38	92323.9	-	-	-	-	19.16	1.0045	0.03
200	19900	796.72	53079.8	-	-	-	-	48.91	1.0102	0.06

Table 13.10: Low-weight limit for costs and weights with outliers in $\{1, \ldots, 2000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	12.7	0.00	6.8	-	0.54	0.01	1.0038	-
10	45	0.00	30.8	0.01	14.7	18.00	0.48	0.00	1.0118	0.50
50	307	0.04	326.5	1.38	281.1	32.65	0.86	0.04	1.0092	0.91
50	612	0.29	1185.7	10.42	1288.1	35.40	1.09	0.10	1.0179	0.32
50	1225	1.82	3534.9	51.85	3353.1	28.46	0.95	0.24	1.0169	0.13
100	1238	1.74	2330.6	125.02	5058.9	71.98	2.17	0.57	1.0089	0.33
100	2475	8.09	5292.9	275.06	6085.3	33.99	1.15	1.51	1.0124	0.19
100	4950	53.91	20968.3	-	-	-	-	4.05	1.0177	0.08
150	2794	14.70	6181.9	-	-	-	-	3.01	1.0072	0.20
150	5588	39.95	7590.4	-	-	-	-	8.13	1.0089	0.20
150	11175	46.27	2602.1	-	-	-	-	19.58	1.0088	0.42
200	4975	41.94	7793.0	-	-	-	-	8.69	1.0058	0.21
200	9950	205.26	19259.8	-	-	-	-	23.68	1.0095	0.12
200	19900	114.40	3140.7	-	-	-	-	60.95	1.0068	0.53

Table 13.11: Medium-weight limit for costs and weights with outliers in $\{1, \ldots, 2000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	9.1	0.00	5.7	-	0.62	0.00	1.0043	-
10	45	0.00	22.3	0.01	11.8	-	0.53	0.00	1.0077	1.00
50	307	0.02	143.1	0.59	132.6	32.94	0.93	0.05	1.0025	2.50
50	612	0.19	659.0	7.81	952.5	40.79	1.45	0.11	1.0173	0.59
50	1225	0.71	59.4	19.90	1332.7	27.89	22.44	0.29	1.0089	0.41
100	1238	0.92	998.0	29.19	1332.6	31.66	1.34	0.68	1.0089	0.74
100	2475	4.73	2.048.8	173.91	3828.4	36.78	1.87	1.82	1.0039	0.38
100	4950	4.47	896.8	-	-	-	-	5.06	1.0148	1.13
150	2794	5.14	1837.2	-	-	-	-	3.70	1.0094	0.72
150	5588	8.41	1095.3	-	-	-	-	10.17	1.0053	1.21
150	11175	23.07	1377.3	-	-	-	-	23.31	1.0069	1.01
200	4975	35.42	3891.0	-	-	-	-	10.54	1.0106	0.30
200	9950	46.95	2045.9	-	-	-	-	29.96	1.0066	0.64
200	19900	86.99	1739.0	-	-	-	-	71.10	1.0069	0.82

Table 13.12: High-weight limit for costs and weights with outliers in $\{1, \ldots, 2000\}$

Weakly Negatively Correlated Costs and Weights

As announced in the beginning this problems are relatively easily to solve since we have a large set of edges with a weight equal to 1. This effects also the sequence of decreasing number of branchings by increasing the number of edges for a fixed number of vertices. Furthermore this instances underlines that our own algorithm has a much smaller run time than the algorithm of Aggarwal, Aneja and Nair.

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	21.0	0.01	19.0	-	0.91	0.00	1.0230	-
10	45	0.00	43.7	0.02	31.1	-	0.71	0.00	1.0406	-
50	307	0.02	225.0	1.85	433.0	86.80	1.92	0.02	1.0130	1.04
50	612	0.04	143.1	1.67	206.0	41.11	1.44	0.07	1.0144	1.80
50	1225	0.06	106.6	2.48	157.9	40.71	1.48	0.15	1.0128	2.44
100	1238	0.10	143.0	14.81	691.9	156.35	4.84	0.34	1.0066	3.58
100	2475	0.17	65.5	14.24	254.3	84.37	3.89	0.99	1.0069	5.89
100	4950	0.23	29.5	32.62	253.3	142.61	8.60	2.08	1.0083	9.10
150	2794	0.22	51.2	-	-	-	-	1.81	1.0255	8.23
150	5588	0.41	16.5	-	-	-	-	4.02	1.0062	9.89
150	11175	0.88	13.1	-	-	-	-	8.54	1.0042	9.73

Table 13.13: Low-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	16.5	0.01	15.6	-	0.94	0.00	1.0333	-
10	45	0.00	29.5	0.02	25.1	-	0.85	0.00	1.0261	-
50	307	0.02	225.0	1.85	433.0	86.80	1.92	0.02	1.0130	1.04
50	612	0.02	77.4	1.21	122.9	52.45	1.59	0.09	1.0131	3.89
50	1225	0.02	20.5	1.64	68.8	72.74	3.36	0.18	1.0126	8.16
100	1238	0.13	127.8	11.63	371.2	86.98	2.91	0.43	1.0073	3.24
100	2475	0.13	23.6	19.46	241.9	153.19	10.25	1.25	1.0065	9.87
100	4950	0.25	8.2	37.09	209.5	150.32	25.71	2.53	1.0043	10.25
150	2794	0.24	34.0	-	-	-	-	2.34	1.0046	9.69
150	5588	0.51	9.4	-	-	-	-	5.20	1.0033	10.28
150	11175	1.10	5.3	-	-	-	-	10.65	1.0037	9.68

Table 13.14: Medium-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	12.2	0.01	10.2	-	0.83	0.00	1.0230	-
10	45	0.00	17.1	0.01	14.6	-	0.85	0.00	1.0273	-
50	307	0.42	70.5	0.01	34.7	0.01	0.49	0.03	1.0064	0.07
50	612	0.01	23.8	0.78	72.9	58.55	3.06	0.10	1.0079	7.64
50	1225	0.02	9.5	1.29	51.4	57.11	5.41	0.20	1.0044	8.99
100	1238	0.05	24.9	7.01	171.5	132.83	6.90	0.49	1.0046	9.33
100	2475	0.14	9.6	13.80	117.9	102.59	12.34	1.39	1.0027	10.30
100	4950	0.28	3.5	30.10	92.6	108.18	26.46	2.78	1.0024	10.00
150	2794	0.25	12.5	-	-		-	2.61	1.0030	10.29
150	5588	0.58	4.8	-	-		-	5.72	1.0021	9.93
150	11175	1.24	2.7	-	-	-	-	11.63	1.0013	9.35

Table 13.15: High-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	22.2	0.01	16.8	-	0.76	0.00	1.0236	-
10	45	0.00	52.0	0.00	52.0	1.00	1.00	0.00	1.0455	3.00
50	307	0.05	458.9	2.34	442.4	43.61	0.96	0.03	1.0154	0.60
50	612	0.13	423.6	3.19	345.5	24.68	0.82	0.09	1.0172	0.73
50	1225	0.29	394.7	4.37	258.1	14.86	0.65	0.21	1.0464	0.72
100	1238	1.04	1099.1	24.04	1007.6	23.21	0.92	0.54	1.0190	0.52
100	2475	1.61	706.8	31.33	737.8	19.50	1.04	1.27	1.0372	0.79
100	4950	4.80	679.2	60.03	688.3	12.50	1.01	2.92	1.0087	0.61
150	2794	3.23	1122.5	145.28	2471.0	45.03	2.20	2.79	1.0207	0.86
150	5588	4.83	779.1	-	-	-	-	6.03	1.0062	1.25
150	11175	4.38	370.5	-	-	-	-	12.91	1.0063	2.95
200	4975	7.09	1117.7	-	-	-	-	7.56	1.0054	1.07
200	9950	5.09	467.7	-	-	-	-	17.27	1.0043	3.39
200	19900	8.33	248.3	-	-	-	-	38.31	1.0037	4.60
250	6219	10.52	889.3	-	-	-	-	19.64	1.0033	1.87
250	12438	11.48	502.4	-	-	-	-	44.71	1.0032	3.89
250	24875	9.93	140.0	-	-	-	-	97.62	1.0035	9.83
300	11213	12.23	783.2	-	-	-	-	39.23	1.0025	3.21
300	22425	17.85	450.0	-	-	-	-	93.80	1.0024	5.26
300	44850	15.93	78.3	-	-	-	-	194.73	1.0022	12.22
350	15269	16.94	683.0	-	-	-	-	70.95	1.0023	4.19
350	30538	18.65	245.4	-	-	-	-	162.84	1.0023	8.73
350	61075	28.27	110.0	-	-	-	-	335.85	1.0026	11.88
400	19950	15.48	350.0	-	-	-	-	123.87	1.0017	8.00
400	39900	23.16	148.1	-	-	-	-	266.28	1.0020	11.50
400	79800	44.26	89.3	-	-	-	-	554.45	1.0024	12.53

Table 13.16: Low-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	19.2	0.01	18.5	-	0.97	0.00	1.0367	-
10	45	0.00	27.2	0.01	21.7	50.00	0.80	0.00	1.0310	1.00
50	307	0.03	269.6	1.29	233.5	39.34	0.87	0.04	1.0124	1.33
50	612	0.12	371.1	2.82	293.4	23.78	0.79	0.12	1.0137	1.01
50	1225	0.42	459.1	4.54	262.2	10.87	0.57	0.27	1.0149	0.64
100	1238	0.61	642.3	17.57	706.5	28.83	1.10	0.69	1.0087	1.13
100	2475	0.74	374.6	22.89	520.2	31.11	1.39	1.63	1.0076	2.21
100	4950	1.09	260.2	39.04	424.3	35.89	1.63	3.63	1.0061	3.33
150	2794	1.76	719.5	156.99	2168.1	89.22	3.01	3.51	1.0052	2.00
150	5588	1.48	253.4	-	-	-	-	7.45	1.0058	5.02
150	11175	2.60	181.9	-	-	-	-	15.77	1.0058	6.07
200	4975	3.29	582.6	-	-	-	-	9.56	1.0043	2.90
200	9950	3.62	254.3	-	-	-	-	21.63	1.0047	5.97
200	19900	5.24	111.5	-	-	-	-	47.06	1.0037	8.99
250	6219	10.78	713.3	-	-	-	-	24.50	1.0035	2.27
250	12438	5.55	153.8	-	-	-	-	54.81	1.0043	9.88
250	24875	9.78	59.0	-	-	-	-	120.12	1.0031	12.28
300	11213	14.22	598.8	-	-	-	-	48.48	1.0028	3.41
300	22425	9.95	125.8	-	-	-	-	114.62	1.0020	11.52
300	44850	18.25	42.4	-	-	-	-	232.02	1.0023	12.71
350	15269	9.48	229.9	-	-	-	-	88.93	1.0025	9.38
350	30538	17.91	139.7	-	-	-	-	199.29	1.0026	11.12
350	61075	31.13	34.4	-	-	-	-	401.93	1.0021	12.91
400	19950	15.01	222.4	-	-	-	-	153.34	1.0025	10.22
400	39900	24.65	53.4	-	-	-	-	326.68	1.0020	13.25
400	79800	50.68	26.7	-	-	-	-	662.17	1.0016	13.07

Table 13.17: Medium-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	11.8	0.00	9.9	-	0.84	0.00	1.0204	2.00
10	45	0.00	23.5	0.01	18.0	-	0.76	0.00	1.0323	
50	307	0.03	185.0	0.85	153.5	33.00	0.83	0.05	1.0093	1.87
50	612	0.05	125.4	1.25	131.4	25.99	1.05	0.13	1.0065	2.74
50	1225	0.09	108.5	2.16	115.5	23.78	1.06	0.29	1.0117	3.21
100	1238	0.40	401.6	11.50	466.9	28.95	1.16	0.77	1.0050	1.94
100	2475	0.74	264.1	17.27	381.7	23.44	1.45	1.82	1.0048	2.47
100	4950	0.65	114.5	27.87	277.3	42.60	2.42	4.03	1.0056	6.16
150	2794	1.83	384.1	121.77	1461.4	66.42	3.80	3.93	1.0031	2.14
150	5588	1.24	182.1	-	-	-	-	8.29	1.0034	6.67
150	11175	1.58	46.6	-	-	-	-	17.24	1.0038	10.92
200	4975	2.45	318.8	-	-	-	-	10.70	1.0029	4.38
200	9950	2.85	156.1	-	-	-	-	23.96	1.0033	8.41
200	19900	4.52	37.7	-	-	-	-	51.72	1.0031	11.43
250	6219	3.50	233.0	-	-	-	-	27.48	1.0024	7.86
250	12438	4.95	66.1	-	-	-	-	60.53	1.0020	12.23
250	24875	10.22	22.5	-	-	-	-	131.38	1.0025	12.28
300	11213	5.59	183.5	-	-	-	-	54.13	1.0014	9.69
300	22425	12.19	574.1	-	-	-	-	67.13	1.0024	11.52
300	44850	19.72	13.6	-	-	-	-	253.77	1.0010	12.87
350	15269	8.38	91.2	-	-	-	-	99.34	1.0018	11.85
350	30538	17.27	45.2	-	-	-	-	220.25	1.0016	12.76
350	61075	34.66	14.3	-	-	-	-	441.11	1.0014	12.73
400	19950	12.74	60.5	-	-	-	-	169.97	1.0013	13.34
400	39900	27.22	27.3	-	-	-	-	358.87	1.0012	13.18
400	79800	56.20	10.7	-	-	-	-	717.87	1.0012	12.77

Table 13.18: High-weight limit for weakly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

Highly Negatively Correlated Costs and Weights

In contrast to uniformly distributed data our lexicographical cost minimum has a much larger weight under highly negatively correlated costs and weights which causes the larger run time for these problems.

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	65.6	0.02	37.0	-	0.56	0.00	1.0536	-
10	45	0.00	220.0	0.07	93.8	35.5	0.43	0.00	1.0398	0.25
50	307	0.02	253.4	4.65	634.4	204.5	2.50	0.04	1.0082	1.74
50	612	0.03	161.0	5.75	352.1	185.6	2.19	0.13	1.0090	4.22
50	1225	1.76	3517.8	146.78	7455.3	83.5	2.12	0.34	1.0085	0.19
100	1238	0.16	327.3	90.15	1205.3	565.2	3.68	0.77	1.0038	4.84
100	2475	1.25	1956.2	197.44	958.5	157.7	0.48	2.10	1.0042	1.68
100	4950	0.43	85.5	-	-	-	-	4.43	1.0033	10.36
150	2794	8.47	10238.4	-	-	-	-	4.10	1.0025	0.48
150	5588	0.85	92.2	-	-	-	-	9.85	1.0028	11.62
150	11175	1.67	108.8	-	-	-	-	20.23	1.0029	12.15

Table 13.19: Low-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	88.8	0.03	48.3	-	0.54	0.00	1.0561	-
10	45	0.00	180.3	0.05	70.1	30.4	0.39	0.00	1.0540	0.71
50	307	0.02	188.9	4.23	628.3	228.5	3.33	0.06	1.0113	3.16
50	612	0.04	177.8	7.62	573.9	181.5	3.23	0.18	1.0134	4.39
50	1225	0.15	219.7	39.38	1362.4	271.0	6.20	0.47	1.0221	3.25
100	1238	0.16	247.5	183.82	2434.0	1136.5	9.84	1.08	1.0194	6.68
100	2475	0.30	80.6	248.99	1517.5	834.83	18.84	2.91	1.0042	9.76
100	4950	0.56	65.7	-	-	-	-	5.98	1.0051	10.72
150	2794	0.82	488.3	-	-	-	-	5.84	1.0049	7.13
150	5588	1.10	58.6	-	-	-	-	13.64	1.0026	12.40
150	11175	2.10	63.0	-	-	-	-	26.26	1.0037	12.50

Table 13.20: Medium-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.01	68.8	0.02	38.2	-	0.55	0.00	1.0667	0.50
10	45	0.02	133.5	0.05	65.6	33.7	0.49	0.01	1.0858	0.67
50	307	0.02	181.4	2.69	417.6	115.5	2.30	0.07	1.0151	3.06
50	612	0.04	102.1	2.06	160.2	52.0	1.57	0.22	1.0145	5.65
50	1225	0.13	176.8	8.86	369.7	66.1	2.09	0.57	1.1074	4.28
100	1238	0.14	115.5	31.91	818.6	227.1	7.09	1.32	1.0096	9.36
100	2475	0.41	188.4	191.49	4041.9	171.36	2.13	3.74	1.0105	9.12
100	4950	1.39	653.8	-	-	-	-	8.44	1.0105	6.06
150	2794	0.64	169.1	-	-	-	-	6.91	1.0064	10.78
150	5588	5.40	2.393.2	-	-	-	-	18.32	1.0053	3.39
150	11175	3.09	73.8	-	-	-	-	38.64	1.0062	12.49

Table 13.21: High-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 100\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	184.3	0.04	76.6	-	0.42	0.00	1.0538	-
10	45	0.01	809.4	0.21	280.6	20.48	0.35	0.00	1.0452	0.13
50	307	0.38	2981.4	29.71	4733.3	78.55	1.59	0.06	1.0074	0.17
50	612	1.52	4772.0	90.64	6771.9	59.52	1.42	0.19	1.0063	0.12
50	1225	0.69	1568.3	97.01	3656.4	140.80	2.33	0.51	1.0077	0.73
100	1238	1.12	1897.7	191.50	4041.9	171.36	2.13	1.077	1.0041	0.96
100	2475	3.44	2385.4	-	-	-	-	3.08	1.0037	0.90
100	4950	3.00	1757.0	-	-	-	-	7.02	1.0032	2.34
150	2794	4.32	2245.1	-	-	-	-	5.86	1.0024	1.36
150	5588	7.19	3021.0	-	-	-	-	15.36	1.0027	2.14
150	11175	54.40	16514.5	-	-	-	-	34.26	1.0019	0.63
200	4975	3.67	1273.9	-	-	-	-	18.47	1.0017	5.03
200	9950	39.32	10067.6	-	-	-	-	43.98	1.0017	1.12
200	19900	41.62	5256.7	-	-	-	-	100.80	1.0020	2.42
250	6219	14.86	3035.7	-	-	-	-	46.73	1.0013	3.14
250	12438	665.82	103813.4	-	-	-	-	114.27	1.0012	0.17
250	24875	23.95	604.8	-	-	-	-	240.13	1.0015	10.03
300	11213	67.81	10322.3	-	-	-	-	91.12	1.0011	1.34
300	22425	1622.20	159124.3	-	-	-	-	224.71	1.0013	0.14
300	44850	45.02	827.0	-	-	-	-	460.06	1.0012	10.22
350	15269	549.94	64875.0	-	-	-	-	168.90	1.0007	0.31

Table 13.22: Low-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	306.1	0.07	146.1	-	0.48	0.000	1.0619	0.33
10	45	0.01	1014.1	0.25	340.8	32.65	0.34	0.002	1.0558	0.19
50	307	0.42	3535.7	32.85	5405.3	77.57	1.53	0.084	1.0111	0.20
50	612	0.69	2515.7	64.50	5631.2	93.45	2.24	0.269	1.0147	0.39
50	1225	0.42	1148.2	201.52	6965.5	475.00	6.07	0.697	1.0136	1.64
100	1238	0.87	1425.5	434.21	9403.9	501.25	6.60	1.538	1.0173	1.78
100	2475	1.26	1114.1	-	-	-	-	4.18	1.0055	3.31
100	4950	2.21	919.3	-	-	-	-	9.44	1.0048	4.27
150	2794	2.09	1341.8	-	-	-	-	8.19	1.0041	3.92
150	5588	3.42	879.6	-	-	-	-	20.48	1.0043	5.99
150	11175	6.27	773.2	-	-	-	-	46.17	1.0049	7.36
200	4975	5.53	1646.6	-	-	-	-	25.32	1.0020	4.58
200	9950	6.88	668.3	-	-	-	-	59.35	1.0026	8.62
200	19900	13.73	598.6	-	-	-	-	136.68	1.0022	9.96
250	6219	6.89	797.6	-	-	-	-	62.99	1.0021	9.15
250	12438	15.59	854.6	-	-	-	-	154.22	1.0020	9.89
250	24875	26.62	588.2	-	-	-	-	318.55	1.0022	11.97
300	11213	10.82	635.6	-	-	-	-	122.17	1.0020	11.29
300	22425	26.39	654.7	-	-	-	-	302.52	1.0013	11.46
300	44850	46.46	548.6	-	-	-	-	596.44	1.0015	12.84
350	15269	21.09	861.1	-	-	-	-	227.71	1.0014	10.80

Table 13.23: Medium-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

		Ruzika, Henn		**Aggarwal, Aneja, Nair**				**Goemans, Ravi**		
n	m	t_r	B_r	t_a	B_a	t_a/t_r	B_a/B_r	t_g	α	t_g/t_r
10	22	0.00	246.4	0.06	122.4	-	0.50	0.00	1.0857	0.20
10	45	0.01	771.7	0.25	329.1	39.52	0.43	0.00	1.1203	0.28
50	307	0.63	4444.7	29.55	4474.5	46.66	1.01	0.10	1.0201	0.16
50	612	1.30	3407.5	120.54	10224.9	92.60	3.00	0.33	1.0160	0.25
50	1225	1.18	1822.0	40.56	1584.8	34.38	0.87	0.83	1.0189	0.70
100	1238	1.64	2251.7	152.24	3670.7	93.07	1.63	1.88	1.0101	1.15
100	2475	2.79	1702.2	-	-	-	-	4.94	1.0064	1.77
100	4950	2.15	854.5	-	-	-	-	11.13	1.0072	5.18
150	2794	2.93	1450.9	-	-	-	-	9.73	1.0064	3.32
150	5588	6.76	2478.0	-	-	-	-	23.93	1.0057	3.54
150	11175	56.94	14497.9	-	-	-	-	57.32	1.0066	1.01
200	4975	4.27	925.1	-	-	-	-	29.95	1.0043	7.01
200	9950	35.99	5941.8	-	-	-	-	70.79	1.0053	1.97
200	19900	142.32	21245.2	-	-	-	-	178.58	1.0050	1.25
250	6219	18.73	3258.5	-	-	-	-	73.71	1.0043	3.94
250	12438	205.21	26416.4	-	-	-	-	185.43	1.0033	0.90
250	24875	64.14	2549.8	-	-	-	-	422.55	1.0043	6.59
300	11213	54.86	7182.4	-	-	-	-	143.82	1.0027	2.62
300	22425	386.37	36494.7	-	-	-	-	397.18	1.0035	1.03
300	44850	57.17	392.3	-	-	-	-	788.43	1.0024	13.79
350	15269	198.71	20474.2	-	-	-	-	266.93	1.0029	1.34

Table 13.24: High-weight limit for highly negatively correlated costs and weights in $\{1, \ldots, 1000\}$

14 Conclusion

This chapter summarizes the results of this diploma thesis and gives an outlook of possible further research topics.

14.1 Summary of the Main Results

In this diploma thesis we have discussed the NP-hard problem of finding a minimal cost spanning tree under the restriction that the weight of this tree does not exceed a given value. We have approached this problem from a spanning tree problem without restrictions (Chapter 2) and from a multicriterial optimization problem (Chapter 6). Also we have noticed that the Lagrangian relaxation is a very usual method to handle this problem (Chapter 7) and stated some in- and exclusion tests (Chapter 9). In the Chapters 11 and 12 we gave an overview over all solution and approximation approaches to our problem which can be found in literature and carried out a few refinements. Also we applied two more general methods (ranking and approximation through decomposition) to our problem. Main part of this thesis was the designing of an alternative solving algorithm using the tree structure of supported trees. Our detailed numerical results has shown that this algorithm was the best way to solve a weight-constrained minimal spanning tree problem. The Algorithm of Aggarwal, Aneja and Nair needed much more time to find the exact tree and the algorithms of Shogan and Hong, Chung and Park were not applicable ways to solve the weight-constrained minimal spanning tree problem.

14.2 Further Research

The weight-constrained minimal spanning tree problem and the work of this thesis offer a lot of possible topics.

In- and Exclusion for Supported Trees

Based on the chapter concerning in- and exclusion tests and the section 11.1.3 where we made some statements for the adjacency structure of the supported trees the question occurs whether for two supported trees T_1, T_K with an edge $e \in T_1 \cap T_K$ a sequence of adjacent trees $T_1, T_2, \ldots, T_K$ like in Theorem 11.12 exists such that e is contained in each tree of this sequence.

Alternative Problem Settings

It is possible to evaluate the behavior of the algorithm tested in Chapter 13 with other problem settings, like uniform distributed data in $\{1, \ldots, 10000\}$ or other distributions. Another field of interest is which effect a faster minimal spanning tree algorithm (see Section 2.4) implemented in the algorithm of Aggarwal, Aneja and Nair will have.

Directed Case

In Chapter 3.5 we have introduced the weight-constrained minimal arborescence problem

$$\begin{aligned} \min & \sum_{(i,j)\in B} c_{ij} \\ s.t. & \sum_{(i,j)\in B} w_{ij} \leq W \\ & B \in \mathcal{B} \end{aligned}$$

where $\mathcal{B}$ is the set of all arborescence. For this problem can be investigated which results from our undirected graph problem can be transformed to this directed case. In principle our Algorithm 11.4 can easily be modified to solve this kind of problem.

Special Cases

A further idea is to go one with the treatment of special relations between costs and weights (already started in Chapter 10) and the considering of a special graph structure (for instance grid graphs or bipartite graphs).

Multidimensional

A next field of work is the dealing with two- or more dimensional constraints the resource-constrained minimal spanning tree problem (Problem 2).

$$\begin{aligned} \min & \sum_{e\in T} c_e \\ s.t. & \sum_{e\in T} w_e^l \leq W_l \text{ for } 1 \leq l \leq L \\ & T \in \mathcal{T} \end{aligned}$$

This problem was stated in the article of Shogan [37] and Algorithm 11.3 can be used to solve this problem. Can other ideas of our WCMST be applied to this problem? Unfortunately, we may lose some nice properties we have in the one-dimensional case like the adjacency property of the convex hull.

Combination with other Restrictions

It is also possible to combine a weight constraint with several other constraints for instance flow requirements, degree constraints or hop constraints (see for more possibilities [11]).

Matroid

In Chapter 4 we have seen the connection between a spanning tree in a graph and basis of a matroid. Since the set of extreme points in a bicriterial matroid optimization problem can be obtained by a sequence of elementary basis operations we complete our outlook with the idea of applying our results from Section 11.1.3 to the theory of matroids.

List of Algorithms

Bibliography

[1] V. Aggarwal, Y. P. Aneja, K. P. K. Nair, Minimal spanning tree subject to a side constraint, Comput. Oper. Res. 9 (4) (1982), 287-296

[2] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, Englewood Cliffs, NJ, 1993

[3] K. A. Andersen, K. Jörnsten, M. Lind, On bicriterion minimal spanning trees: an approximation, Comput. Oper. Res. 23 (12) (1996), 1171-1182

[4] D. Blokh, G. Gutin, An approximation algorithm for combinatorial optimization problems with two parameters, Australasian Journal of Combinatorics 14, 1996

[5] O. Boruvka, O jistém problému minimálním, Práce Moravské Prirodovedecké Spolecnosti, 3 1926, 37-58 (In Czech.)

[6] C. Burch, S. O. Krumke, M. Marathe, C. Phillips, E. Sundberg, Multicriteria approximation through decomposition, 1998

[7] S. Chaiken, D. J. Kleitman, Matrix trees theorems, Journal of combinatorial theory, Series A 24 1978, 377-481

[8] R. Chandrasekaran, Minimal ratio spanning trees, Networks 7 1977, 335-342

[9] B. Chazelle, A minimum spanning tree algorithm with Inverse-Ackermann Type Complexity, Journal of the ACM, 47(6), 2000, 1028-1047

[10] D. Cheriton, R. E. Tarjan, Finding minimum spanning trees, SIAM J. Comput. 5, 1976, 724-742

[11] N. Deo, N. Kumar, Computation of constrained spanning trees; A unified approach, Network Optimization: Lecture Notes in Economics and Mathematical Systems 450, 1997, 194-220

[12] I. Dumitrescu, N. Boland, Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortes path problem, Networks 42(3) 2003, 135-153

[13] M. Ehrgott, Multicriteria Optimization, Berlin, Springer, 2000

[14] M. Ehrgott, K. Klamroth, Connectedness of efficient solutions in multiple criteria combinatorial optimizations, European Journal of Operational Research 97 1997, 155-166

[15] H. N. Gabow, Z. Galil, T. Spencer, R. E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica, 6 1986, 109-122

[16] H. N. Gabow, K. S. Manu, Packing algorithms for arborescoences (and spanning trees) in capacicated graphs, Proceedings of 4th MPS Conference on Integer Programming and Combinatorial Optimization (IPCO), 1995, 388-402

[17] M. Garey, David S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness

[18] M. X. Goemans, R. Ravi, A constrained minimum spanning tree problem, In: Proccedings of the Scandinavian Workshop on Algorithmic Theory (SWAT), Springer, 1996, 66-75

[19] R. L. Graham, P. Hell, On the history of the minium spanning tree problem, Ann. Hist. Comput., 7 1985, 43-57

[20] G.-M. Greuel, G. Pfister, and H. Schönemann, Singular 3.0, A computer algebra system for polynomial computations Centre for Computer Algebra, University of Kaiserslautern 2005, `http://www.singular.uni-kl.de`

[21] J. Gorski, K. Klamroth, S. Ruzika, Connectedness of efficient solutions in multiple objective combinatorial optimization / Technical Report, University of Kaiserslautern, Department of Mathematics, 2006. Report in Wirtschaftsmathematik Nr. 102 2006.

[22] M. Guignard, M. Rosenwein, An application of Lagrangean decomposition to the resource-constrained minimum weighted arborescence problem, Networks 20 1990, 345-359

[23] R. Hassin, A. Levin, An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection, SIAM Journal on Computing. 33 2004 2, 261-268

[24] H. W. Hamacher, K. Klamroth, Lineare und Netzwerk-Optimierung, Vieweg, 2000

[25] H. W. Hamacher, M. Queyranne, K best solutions to combinatorial optimization problems, Annals of Operations Research 4 (1985), 123 - 143

[26] H. W. Hamacher, G. Ruhe, On spanning tree problems with multiple objectives, Annals of Operations Research 52, 1994, 209-230

[27] S-P. Hong, S-J. Chung, B. H. Park, A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem, Operations Research Letters 32 (2004) 233-239

[28] B Li, E Yao, The constrained minimum spanning tree problem: complexity and estimation of the bound, Journal-Zhejiang University-science edition, 2000 (in Chinese)

[29] K. Jörnsten, S. Migdalas, Designing a minimal spanning tree network subject to a budget constraint, Optimization 19 (1988) 4 475-484

[30] A. Jüttner, On resource constraint optimization problems, 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, 2005

[31] N. Katoh, T. Ibaraki, H. Mine, An algorithm for finding k minimum spanning trees, Siam Journal on Computing 10(2):246-255 1981

[32] K. Mehlhorn, M. Ziegelmann, CNOP - A package for constrained network optimization, In: Revised Papers From the Third international Workshop on Algorithm Engineering and Experimentation (January 05 - 06, 2001), A. L. Buchsbaum and J. Snoeyink, Eds. Lecture Notes In Computer Science Vol. 2153. Springer-Verlag, London, 17-31

[33] S.A. Morozov, On the minimal spanning tree problem with restrictions, In Uprvlianye systemy, 14 1976, 40-47 (in Russian)

[34] J. Naor, B. Schieber, Improved approximation for shallow-light spanning trees, In FOCS, 1997, 536-541

[35] S. Ruzika, On multiple objective combinatorial optimization, Dissertation, Department of Mathematics, University of Kaiserslautern, 2007

[36] J. Siek, L-Q. Lee, A. Lumsdaine, The Boost Graph Library (BGL), 2000-2001, `http://www.boost.org`

[37] A. Shogan, Constructing a minimal-cost spanning tree subject to resource constraints and flow requirements, Networks 13. 1983 169-190

[38] G. Xue, Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees, In 19th IEEE International Performance, Computing, and Communications Conference (IPCCC 2000) p. 271-277

[39] T. Yamada, K. Wantanabe, Algorithms to solve the knapsack constrained maximum spanning tree problem, International Journal of Computer Mathematics Vol. 82 No. 1 2005 23-34

[40] A. Yao, An $O(|E| \log \log |V|)$ algorithm for finding minimum spanning trees, Inf. Process. Lett. 4, 1975, 21-23

Appendix

Part of this Diploma Thesis is a compact disk which contains the C++ - Files of the implementation, the SINGULAR source code for Algorithm 11.10, a random generator, all testfiles and MicrosoftExcel sheets with the test results.

Statement of authorship

I hereby certify that this diploma thesis has been composed by myself, and describes my own work, unless otherwise acknowledged in the text. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged. It has not been accepted in any previous application for a degree.

Kaiserlautern, 14 May 2007 Sebastian T. Henn