INAUGURAL-DISSERTATION

zur Erlangung der Doktorwürde der
Naturwissenschaftlich-Mathematischen
Gesamtfakultät der
Ruprechts-Karls-Universität Heidelberg

vorgelegt von
Dipl.-Math. Achim Hildenbrandt
aus Ilmenau

Tag der mündlichen Prüfung: ....

The Target Visitation Problem

Betreuer:
Prof. Dr. Gerhard Reinelt

Abstract

The thesis considers the target visitation problem, a combinatorial optimization problem, which merges the classical traveling salesman problem with the linear ordering problem. In more detail, we are looking for a tour which visits a set of targets and which is optimal with respect to two different aspects: On the one hand, we have given a travel cost from each target to every other. On the other hand, we have preference values which tell us how much we would like to visit one target before another one. The objective is now to maximize the difference of the sum of the met preferences and the total travel costs.

We test several different integer programming formulations and examine the associated polytopes concerning their facets and combinatorial structure. We come to the result that a model based on the combination of integer programming formulations for the traveling salesman problem and the linear ordering problem is most suitable for being used in practical computations. For this model we then develop an extended formulation.

Besides the theoretical studies, this thesis also contains a practical part, where we apply the various methods of combinatorial optimization to the target visitation problem. We examine their performance and the amount of memory they need on a set of self-defined benchmark instances. We also realize that the target visitation problem is, from a practical point of view, a really tough problem. Therefore, we cannot only implement the basic methods, but we have to apply special techniques to obtain exact solutions for instances with thirty or more targets. The best results are achieved by a branch-and-cut approach which uses the facet classes we discovered in the theoretical part of the thesis. Besides the exact approaches, we also examine different heuristics which are inspired by approximation approaches for the traveling salesman problem.

Zusammenfassung

Das Thema dieser Arbeit ist das Target Visitation Problem. Dabei handelt es sich um ein kombinatorisches Optimierungsproblem, dass eine Verbindung des Problems des Handlungsreisenden und des linearen Ordnungsproblems darstellt. Genauer gesagt gilt es bei diesem Problem eine Tour zu finden, die eine vorgegebene Menge von Zielen besucht und welche hinsichtlich zweier Gesichtspunkte optimal sein soll. Einerseits haben wir Reisekosten, anderseits gibt es für je zwei Ziele einen Prioritätswert der besagt, wie sehr wir das eine vor dem anderen Ziel besuchen möchten. Die Aufgabe ist es dann eine Tour zu finden, bei der die Differenz aus der Summe der erfüllten Präferenzen und den gesamten Reisekosten maximal ist.

Wir betrachten mehrere Formulierungen ganzzahliger Programme und untersuchen die zugehörigen Polyeder ausführlich hinsichtlich ihrer Facetten und ihres kombinatorischen Aufbaus. Wir kommen zu dem Ergebnis das ein Modell basierend auf der Kombination von ganzzahligen Programmen für das Problem des Handlungsreisenden und des linearen Ordnungsproblems am geeignetsten erscheint, vor allem auch im Hinblick auf den Einsatz für praktische Berechnungen. Für dieses Modell geben wird dann zusätzlich eine erweiterte Formulierung an.

Die Arbeit enthält auch einen großen praktischen Teil bei dem wir verschiedene Lösungsverfahren der kombinatorischen Optimierung auf das Target Visitation Problem anwenden. Wir vergleichen die unterschiedlichen Ansätze hinsichtlich ihrer Laufzeit und ihres Speicherbedarfs, die sie zur Lösung eines von uns selbst definierten Menge von Benchmark-Instanzen benötigen. Wir werden dabei feststellen, dass das Target Visitation Problem auch in der Praxis ein wirklich schweres Optimierungsproblem darstellt, bei dem die bloße Implementierung von Standardverfahren ohne spezielle Techniken es nicht ermöglicht Instanzen mit dreißig oder mehr Zielen exakt zu lösen. Die besten Ergebnisse zeigt ein „branch-and-cut"-Ansatz der Facettenklassen benutzt, welche wir bei unseren theoretischen Untersuchungen gefunden haben. Neben den exakten Lösungsansätzen haben wir auch verschiedene Heuristiken untersucht, welche auf approximativen Ansätzen für das Problem des Handlungsreisenden basieren.

Acknowledgment

A project like writing a Ph. D. thesis can hardly be accomplished without help and support. Also in my case there were a lot of people involved who gave me assistance and encouraged me. I would like to dedicate the following lines to these people.

There are three people without whom this thesis would probably never have been finished and whom I therefore owe the most thanks: Firstly I am grateful to my supervisor Prof. Reinelt for introducing me to a wonderful part of combinatorial optimization and for giving me the opportunity to do research very independently. Secondly I have to thank my mother Regina Hildenbrandt who has supported me my whole live and especially during my whole scientific education. And then thirdly I send my deepest gratitude to my colleague, best friend and my bastion of calm, Olga Heismann. It is nearly impossible to describe what she has done for me in the last five years. So I would just like to say thank you for the time we have spent together all over the world and which was filled with lots of math, fun and so many productive and interesting discussions.

Then I would also like to say thank you to a lot of other people: Our secretary Catherine Proux-Wieland for supporting the group by keeping the administrative tasks going in an excellent way. Stefan Lörwald for helping me out a countless number of times with my computer problems and also for very interesting mathematical discussions. I am grateful to Francesco Silvestri for proof-reading this thesis, and his very valuable help in the HUHFA project. Further, I would like to say thank you to Stefan Lörwald for the very helpful last-minute proof-reading.

Jens Fielenbach, Kathrin Ronellenfitsch, Francesco Silvestri and Stefan Lörwald for many hours of productive joint work. Markus Speth for sharing the office with me for more than four years in a very peaceful and productive atmosphere and for calming me down every time when my computer was doing strange, stupid things. Finally to my former colleagues Marcus Oswald and Thorsten Bonato who helped me much during my first year and always had an open ear for a lot of silly Ph. D. beginners questions.

Contents

1 Preliminaries **1**
- 1.1 Basic Definitions . . . 1
  - 1.1.1 Linear Algebra . . . 1
  - 1.1.2 Graphs . . . 2
  - 1.1.3 Complexity Theory . . . 3
- 1.2 Combinatorial Optimization . . . 4
  - 1.2.1 Linear and Integer Programming . . . 4
  - 1.2.2 Polyhedral Combinatorics . . . 6
  - 1.2.3 Traveling Salesman Problem . . . 6
  - 1.2.4 Linear Ordering Problem . . . 9
- 1.3 Solving Methods for Integer Programs . . . 10
  - 1.3.1 Branch-and-Bound . . . 10
  - 1.3.2 The Cutting Plane Approach . . . 11
  - 1.3.3 Branch-and-Cut . . . 12

2 The Target Visitation Problem **15**

3 Integer Programming Models for the TVP **19**
- 3.1 An IP Model Based on the Combination of LOP and HP Models . . 19
  - 3.1.1 Extended Formulation of the TVP_{HP} Model . . . 24
- 3.2 The Edge-Node Formulation . . . 26
- 3.3 A Model Based on Distance Variables . . . 29

4 Polyhedral results **31**
- 4.1 Polyhedral Results of the TVP_{HP} Model . . . 34
  - 4.1.1 Dimension of the P^n_{TV} Polytope . . . 34
  - 4.1.2 Facets of the P^n_{TV} Polytope . . . 36
  - 4.1.3 Zero-Lifting for the P^n_{TV} Polytope . . . 38
  - 4.1.4 General Lifting procedures . . . 44
- 4.2 Polyhedral Results for the TVP_E Model . . . 49
  - 4.2.1 Dimension . . . 49
  - 4.2.2 Projection of the P^n_{ETV} Polytope to the P^n_{TV} Polytope . . . 52

5 Approaches for Solving the TVP 55
5.1 Lagrangean Decomposition 55
5.2 Dynamic Programming 58
5.3 Branch-and-Bound 59

6 Heuristic Approaches 61
6.1 The Nearest-Neighbor Method 62
6.2 Best-Insertion Method 62
6.3 Becker Heuristic 63
6.4 The k-Opt Method 64
6.5 Kernighan-Lin Heuristic 65

7 Computational Experiments 67
7.1 Test-Instances 67
7.2 Heuristics 69
7.3 Branch-and-Cut for the TVP$_{HP}$ 71
7.4 Branch-and-Bound with active/inactive variables for the TVP_E . . 73
7.5 Dynamic Programming 74
7.6 Branch-and-Bound with Dynamic Programming 75
7.7 Lagrange Decomposition 76
7.8 Comparison 76

8 Variants of the TVP 79
8.1 The original TVP formulation 79
8.2 The Symmetric Target Visitation Problem 82

9 Conclusion and Further Research 85

A Description of Small TVP-Polytopes 91
A.1 Complete Description of P^4_{TV} 91
A.2 Complete Description of P^4_{ETV} 93
A.3 Complete Description of P^4_{ENTV} 93
A.4 Complete Description of P^4_{STV} 94

B Overview of the TVP Instances 97

C Bounds of the LP Relaxation of the Different TVP Models 101

D Results of the Heuristic Approaches 103
D.1 Results of the Constructive Heuristics 103
D.2 Results of the Improvement Heuristics 108

E Results of the Branch-and-Bound Approach 117

F Results of the Branch-and-Cut Approach 121
F.1 Test of different cutting planes 121
F.2 Test of combinations of different cutting planes 124
F.3 Detailed Information for the Runs with different facet combinations 128

G Results of the Branch-and-Bound with Active/Nonactive variables 135

H CPLEX Computation Times 137

I Results of the Lagrangean Decomposition 141

Preface

The traveling salesman problem is one of the oldest and best-known combinatorial optimization problems. It has been studied for decades and the results achieved in all these years are quite satisfying. It also has a wide field of applications like route planning, chip design and genome sequencing. Of course not all practical routing problems can be modeled by this problem. Imagine for example the following situation: After some huge natural catastrophe one wants to distribute food and medicine in the disaster area. For this task it is necessary to plan a route through the wasted area, which on the one hand should be as short as possible in order to quickly provide assistance in the entire area. But on the other hand, it should be taken into account that in some spots the situation is much worse than in other ones, which means that these places should be visited as fast as possible.

These kinds of problems can be modeled as a combination of the traveling salesman problem and the linear ordering problem, another well-known combinatorial optimization problem. In this thesis, we want to study this combination of both problems which is called target visitation problem. It is a relatively new problem which occurred during the planing of missions of unmanned aerial vehicles.

More formally, we face the objective of finding a tour which visits a set of spots and takes two different criteria into account: The first one are the travel costs which should be as small as possible. The second one are the preference values which tell us for each two spots how much we would like to visit one before the other. Then the sum of all met preferences should be maximal. In combination this means that the difference of the met preferences and the distance costs should be as big as possible. So this problem is an example of a problem with a multicriterial objective function.

This thesis is divided into two general parts. In the first part we study the theoretical background of the problem. There we focus mainly on finding a good integer programming formulation including an analysis of the polyhedral structure of their associated polytopes. In the second part of the thesis, we implement practical approaches which solve the target visitation problem to optimality. For this purpose we use the theoretical results obtained before. In more detail, the thesis is structured as follows.

We begin in Chapter 1 with a brief introduction to the necessary definitions and notations. In Chapter 2 we introduce the target visitation problem and present a reformulation as a path problem.

Then in Chapter 3 we try to find a suitable integer programming formulation. We examine several approaches like combining the integer programming formulation of the traveling salesman and linear ordering problem, a model with variables which state the relation between an edge and a node, as well as a model based on distance variables. We also introduce an extended formulation.

In Chapter 4 we examine the polytopes associated with these models. We take special regards to their usability in a practical branch-and-cut approach. The two most promising polytopes are examined further. We prove their dimension as well as a general zero-lifting result. Some general facet classes are also presented. Additionally we study the connection between the polytope of the extended formulation and its basic formulation without the additional variables.

In the next chapter, we present approaches which solve the target visitation problem to optimality. Namely we present a branch-and-bound and a dynamic programming approach as well as a combination of both. Lagrange decomposition is also considered.

Chapter 6 is then dedicated to approximation approaches. We present several heuristics which are adopted from the research on the traveling salesman problem and the linear ordering problem. We examine constructive heuristics which create a feasible tour from scratch as well as methods which improve a given solution. Both types will be combined to obtain satisfying results. So even for many instances with forty or more nodes, some combination of heuristics find the optimal value in nearly every run.

Chapter 7 finally compares these different approaches by doing practical computations. The instances we solved with the various algorithms are hereby defined by ourselves since there are no benchmark problems available so far. It shows that none of the standard approaches are able to solve target visitation problem instances with thirty or more nodes. Only the branch-and-cut approach, which uses facets we have described in the sections before, solves instances up to forty-five nodes exactly, which is way more than the commercial integer programming solver CPLEX is able to do.

In Chapter 8 we make some remarks on how we can transfer results of Chapter 4 to the original tour formulation as well as giving a brief examination of the symmetric version of the target visitation problem.

The final Chapter 9 sums up the results and gives some ideas for further research.

Chapter 1

Preliminaries

1.1 Basic Definitions

1.1.1 Linear Algebra

In this subsection we want to state a few definitions from linear algebra which are used in this thesis. For more details consult one of the standard textbooks like [FQ12]. A **matrix** M with m rows and n columns is called an (m, n)-matrix. We call a matrix a **row vector** if $m = 1$ and a **column vector** if $n = 1$. A matrix is said to be a **binary** matrix if all entries are either 0 or 1. By a_{ij} we will denote the entry in the j-th column and the i-th row. In this thesis we will use the term **vector** when we mean a column vector if not stated otherwise. By $\mathbf{0}$ ($\mathbf{1}$) we denote the column vector with all entries equal to zero (one).

A^T denotes the **transposed** matrix of A, and A^{-1} denotes the **inverse** matrix. Let $I \subset \{1, \ldots, m\}$ and $J \subset \{1, \ldots, n\}$ be ordered sets. Then $A_{I,J}$ which contains all entries a_{ij} of A where $i \in I$ and $j \in J$ is called a **submatrix** of A. The sum $\sum_{i=2}^{n} \sum_{j=1}^{i-1} a_{ji}$ will be called the **upper diagonal** sum. The **lower diagonal** sum is defined analogously.

A **linear combination** of a set of vectors $\{x_1, \ldots, x_k\} \in \mathbb{R}$ is defined by $\sum_{i=1}^{k} a_i x_i$ where $a_1, a_2, \ldots, a_k \in \mathbb{R}$, are called coefficients. It is called an **affine combination** if $\sum_{i=1}^{k} a_i = 1$, a **conic combination** if $a_i \geq 0$ for $i = 1, 2, \ldots, k$ and a **convex combination** if $\sum_{i=1}^{k} a_i = 1$ and $a_i \geq 0$ for $i = 1, 2, \ldots, k$.

The **convex hull** of a vector set S is defined as the set of all convex combinations of finitely many vectors from S. The **conic, linear and affine hull** are then defined analogously. In the following we denote the convex hull by Conv and the conic hull by Cone. A vector x is called **linearly (affinely) independent** from a set of vectors S if x is not contained in the linear (affine) hull of S.

The cardinality of the smallest set $X \subseteq S$ so that S is a subset of the affine hull of X is called the **affine rank** of S, denoted by $\operatorname{arank}(S)$. The **dimension** of

S is then defined by arank$(S) - 1$.

1.1.2 Graphs

We define an **(undirected) graph** $G = (V, E)$ as a pair of two finite sets where every element of E is a 2-subset of V. The elements of V are called the **vertices** or **nodes** and the elements of E are called the **edges**. If all 2-subsets of V are contained in E then the graph is said to be **complete**. An edge e is defined as a **weighted edge** if some **weight** w_e is assigned to e. If all edges are weighted edges, then the graph is called a weighted graph $G = (V, E, w)$ where w is the **weight function** $w : E \to \mathbb{R}$. In a graph more than one weight could be assigned to one edge. In this case we have $w : E \to \mathbb{R}^k$ where k denotes the number of weights which are assigned to each edge.

A graph $D = (V, A)$ is called **directed** if all edges $a = (i, j)$ are ordered pairs, where i is called the **start node** and j is called the **end node** of a. In the directed case we call the edges **arcs**.

In an undirected graph a node v is **incident** to an edge e if v is contained in e. A node which is incident to an edge e is also called an **end** of e. Two nodes are called **adjacent** to each other or **neighbors** if they are both ends of the same edge e. Two edges are called **adjacent** if they have an end node in common. In a directed graph we denote by $\delta^-(v)$ the set of **outgoing edges** (edges which have v as start node) and analogously by $\delta^+(v)$ the set of **incoming edges** (edges which have v as end node) of a node $v \in V$. When a graph is undirected, we denote by $\delta(v)$ the set of edges which have $v \in V$ as an end. Additionally, $|\delta(v)|$ is referred to as the **degree** of v.

A graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$ is called a **subgraph** of G. A **path** in a directed or undirected graph $G = (V, E)$ is a set of edges $P = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$, where k is called the **length** of the path. If all the nodes contained in the path P are pairwise different and P has length $|V| - 1$, the path is called a **Hamiltonian path**. If $v_0 = v_k$ then a path is called a **cycle**. If the length of a cycle is $|V|$ and all nodes (except start and end node) are pairwise different, then the cycle is called a **Hamiltonian cycle**. An undirected graph $G = (V, E)$ is called a tree if it does not contain any cycles and for each two nodes $i, j \in V$ there exist exactly one path between i and j.

The **adjacency matrix** of a graph $G = (V, E)$ is a $(|V|, |V|)$-matrix so that the entry (i, j) is equal to one if $(i, j) \in E$ and equal to zero otherwise. In this thesis we often need to represent the adjacency matrix as a vector, where the rows of the matrix are concatenated. We call this vector the **characteristic vector** of the graph.

An undirected graph is called **finite** if and only if V and E are both finite. In this thesis we are only dealing with finite graphs.

1.1.3 Complexity Theory

In combinatorial optimization it is important to know how complex a problem will be. Again this is just a short introduction to this topic. For further information [GJ79] gives a good survey.

Firstly we have to distinguish between optimization problems and decision problems. A **decision problem** can be answered with "yes" or "no" while an **optimization problem** is defined as:

$$\max_{x \in X} f(x)$$

where X is the set of feasible solutions of the problem, and f is a function which assigns a real number to each such solution.

The complexity of an algorithm is a measure of the number of elementary operations it takes to execute it. In order to describe the running time of a given algorithm, we introduce a **time function** $t_a : \mathbb{N} \to \mathbb{N}$, which provides for each instance of size n the maximal running time.

We say that an algorithm has a **computation time** of $\mathcal{O}(f(n))$ if there exist constants c and n_0 so that $t_a(n) \leq c \cdot f(n)$, $\forall n > n_0$. One could also state that a computation time of $\mathcal{O}(f(n))$ means that the execution of the algorithm needs a magnitude of $c \cdot f(n)$ elementary operations. The $\mathcal{O}$ is called a Landau symbol.

In complexity theory, optimization problems are mainly classified in two classes (which are not disjoint). If $f(n)$ is a polynomial function, then we say that an algorithm which has a computation time of $\mathcal{O}(f(n))$ is a **polynomial time** algorithm. All problems for which a polynomial time algorithm is known belong to the **class P**. The **class NP** on the other hand is defined as the set of problems for which a "yes" instance of the assigned decision problem can be verified in polynomial time. Therefore it is clear that $P \subseteq NP$. Still it is one of the most important questions in complexity theory whether NP is also a subset of P and therefore:

$$NP = P.$$

We call a decision problem D **polynomially reducible** to a decision problem D' if there exists a polynomial time algorithm which transforms each instance d of D to an instance d' of D' in a way that d has a positive answer if and only if d' has a positive answer. Informally, we could say that this definition characterizes a problem D' at least as hard as problem D.

We call a decision problem D **NP-complete** if each problem in NP is polynomially reducible to D. Lastly we are defining an optimization problem *Opt* to be **NP-hard** if each problem in NP is polynomially reducible to *Opt*. This means that the complexity class NP-hard is the set of all problems that are at least as hard as the problems in the complexity class NP-complete. In particular, it contains the optimization versions of the NP-complete decision problems.

1.2 Combinatorial Optimization

Contrary to other parts of mathematical optimization, a **combinatorial optimization problem** (COP) consists of finding an optimal solution in a finite set of feasible solutions. More formally we can define:

Definition 1.1. *Let E be a finite set and $\mathcal{F} \subseteq 2^E$, where 2^E denotes the power set of E. Further a function $c : \mathcal{F} \to \mathbb{R}$ is given. A combinatorial optimization problem $(E, \mathcal{F}, c)$ is then defined as*

$$\max_{F \in \mathcal{F}} c(F).$$

The function c is called the objective function and the elements $F \in \mathcal{F}$ are called the feasible solutions. If the objective function can be written as $c(F) = \sum_{e \in F} c'(e)$, where $c' : E \to \mathbb{R}$, we call such a problem a linear optimization problem. Of course this definition could also be used for minimization problems by multiplying the objective function with minus one.

To obtain the optimal solution of a linear combinatorial optimization problem, there exist several different concepts. The most obvious method is called **complete enumeration**, which consists of simply calculating the objective value for every element of $\mathcal{F}$. Unfortunately for most COPs the set $\mathcal{F}$ has an exponential number of elements in E so this approach is infeasible.

A more sophisticated idea is the so-called **branch-and-bound** method (see Section 1.3.1), which systematically examines the elements of $\mathcal{F}$ and excludes subsets from investigation when they are proven not to contain an optimal solution.

Another approach are **heuristics** which are algorithms that examine just a subset of the elements of $\mathcal{F}$, which most promisingly contains the optimal solution. That means that the result of a heuristic is often not an optimal solution and even if it is, there is no proof for that.

But in most cases COPs are solved with more complicated methods which are based on the concept of integer programming. The basic theory of this idea will be introduced in the next section.

1.2.1 Linear and Integer Programming

Linear programming is one of the most important methods in combinatorial optimization. Its main concept are **linear programs** (LP), which describe linear optimization problems by putting them in some normal form, stated as follows:

$$\begin{aligned} &\max cx && (1.1)\\ &Ax \leq b, \\ &x \in \mathbb{R}^n_+ \end{aligned}$$

where A is an $(m, n)-$matrix with entries in $\mathbb{R}$, b is a vector in $\mathbb{R}^m$ and c is a row vector in $\mathbb{R}^n$. The entries of the vector $x = (x_1, \dots, x_n)$ are called the **variables**, and the m conditions $Ax \leq b$ are referred to as **constraints**.

The set of all feasible solutions $S := \{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$ is called the **feasible set** of the LP. If $S \neq \emptyset$, the LP is called **feasible**. The linear function $cx : \mathbb{R}^n_+ \to \mathbb{R}$ is called the **objective function**. A feasible solution x^* is called the **optimal solution** if and only if $cx^* \geq cx \; \forall x \in S$. Then cx^* is called the **optimal value** of the LP. There can exist more than one $x \in X$ with $cx = cx^*$.

We would like to point out that (1.1) is a standard form of an LP to which all linear programs can be transformed by the following modifications. For minimization problems the objective function could be multiplied by minus one. Equality constraints like $ex = e_0$ can be split up in two inequalities $ex \leq e_0$ and $ex \geq e_0$. And of course $\geq$-inequalities can be transformed to $\leq$-inequalities by multiplying them with minus one.

One big advantage of linear programs is that there exist fast solving methods. The most common ones are interior point methods and the **simplex method**. Additionally, there exists the ellipsoid method, but this approach is in general not used for practical computations. For details on this methods please see [KV12]. In this thesis we use only the simplex algorithm for the computation of linear programs.

Unfortunately not all combinatorial optimization problems can be formulated as an LP in a practical way, that means with a suitable number of constraints. Often it is more sensible to formulate them as integer programs (IP). A (linear) **integer program** is the same as a linear program except that its solution must be integral:

$$\begin{aligned} &\max cx \\ &Ax \leq b, \\ &x \in \mathbb{Z}^n. \end{aligned} \tag{1.2}$$

Unfortunately for solving integer programs in general there exists so far no algorithm with a polynomial computation time. Nevertheless, there exist different techniques for dealing with integer programs like **branch-and-bound** (see Section 1.3.1), the **cutting plane** approach (see Section 1.3.2) and **branch-and-cut** (see Section 1.3.3).

A special case of an integer program is the case when $x \in \{0, 1\}^n$. The program is then called a **0/1 linear program**. For practical purposes 0/1 linear programs are in most cases easier to handle with the common techniques than integer programs. In some solution methods it is necessary to use the **LP relaxation** of an IP, which is defined as the IP where the integrality constraint has been left out.

1.2.2 Polyhedral Combinatorics

The set $\{x \in \mathbb{R}^n \mid ax = a_0\}$ is called a **hyperplane** and the set $\{x \in \mathbb{R}^n \mid ax \leq a_0\}$ a **halfspace**. A **polyhedron** P is then defined as an intersection of finitely many halfspaces. That means we can describe each polyhedron as:

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\},$$

which is also called the **outer description** or **H-representation**. It is an important result of combinatorial optimization (see for example [Zie95]) that every polyhedron has also an **inner description** defined by $P = \text{Conv}(X) + \text{Cone}(Y)$ for some finite sets X and Y. This is also called the **V-representation**. We call the polyhedron a **polytope** if it can be described by a convex hull only, i. e. $P = \text{Conv}(X)$. The **dimension** (denoted by dim) of a polytope or polyhedron is the maximum number of affinely independent points minus one. A polytope P is **full-dimensional** if $P \subseteq \mathbb{R}^n$ and $\dim P = n$.

An inequality $ax \leq a_0$ is called **valid** for a polytope P if the inequality is satisfied by all points of P. For a valid inequality $ax \leq a_0$, the set $f = P \cap \{ax = a_0\}$ is called **face** of the polytope. When $\dim f = \dim P - 1$ then f is called a **facet**. If $\dim f = 0$ then f is called a **vertex**. The set of all vertices is denoted by **vert**. The associated inequality with f is called **face-defining** or **facet-defining**, respectively. Two inequalities are said to be **equivalent** if they define the same face. An outer description of a polytope is minimal when every facet is defined through exactly one inequality, and there exists no valid linearly dependent equation. An inner description of a polytope is minimal when no $x \in X$ is contained in $\text{Conv}(X \setminus \{x\})$.

When we have an integer program like (1.2), we denote the set $\{x \in \mathbb{Z} \mid Ax \leq b\}$ as **associated polytope**.

We say that two faces f_1 and f_2 of a polytope $P \subseteq \mathbb{R}^n$ are equivalent with respect to some bijection $s : x \to Mx + r$ where $M \in \mathbb{R}^n \times \mathbb{R}^n$ and a vector $r \in \mathbb{R}^n$ if $s(\text{vert}(f_1)) = \text{vert}(f_2)$. If S is a group of bijections, then f_1 and f_2 are equivalent with respect to S if they are equivalent with respect to some $s \in S$. A bijection s is said to be a symmetry for P if it maps every vertex of P to another vertex of P. If two facets are equivalent with respect to a set of symmetries, we say that these facets belong to the same class. A facet class regarding to a given group of symmetries contains all facets which are equivalent with respect to this group. A class of inequalities is defined analogously.

1.2.3 Traveling Salesman Problem

In the following we want to introduce two well-known combinatorial optimization problems which are very important for this thesis since they are both contained as subproblems in the target visitation problem.

The first one is the **traveling salesman problem (TSP)**, which is probably the most studied problem in combinatorial optimization.

Definition 1.2. *Let $D = (V, A)$ be a complete directed graph with n nodes. Furthermore a weight function $d : A \mapsto \mathbb{R}$, which assigns a distance (cost) to every arc (i, j), $1 \leq i, j \leq n$, $i \neq j$ is given. The traveling salesman problem then consists of finding a tour with minimal weight which visits all nodes exactly once.*

The TSP can also be defined on an undirected graph. It is then called symmetric traveling salesman problem (STSP). Defined on a directed graph it is also often called asymmetric traveling salesman problem (ATSP).

Next we want to present one standard IP formulation which models the ATSP. In the model we use the variables x_{ij}, $1 \leq i, j \leq n$, $i \neq j$ with the interpretation ($\pi(i) = j$ when j is visited as the i-th node in the tour):

$$x_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(k+1) \text{ for some } k \in \{1, \ldots, n-1\}, \\ 1 & \text{if } i = \pi(n) \text{ and } j = \pi(1), \\ 0 & \text{otherwise.} \end{cases}$$

An IP model for the TSP can be then formulated as follows.

$$\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij} \tag{1.3}$$

$$\text{s.}t.$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \ j \in V, \tag{1.4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1, \ i \in V, \tag{1.5}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ i \neq j}} x_{ij} \leq |S| - 1, \ S \subset V, 2 \leq |S| < n, \tag{1.6}$$

$$x_{ij} \in \{0, 1\}, \ i, j \in V, \ i \neq j. \tag{1.7}$$

The constraints (1.4) and (1.5) describe the fact that each node is visited exactly once, while (1.6) assures that there exist no sub-cycles in the tour.

This model is just one possible formulation for the TSP. There are other models like for example the Millin-Tucker-Zemplin formulation[MTZ60] which is based on position variables.

There exists a tremendous amount of research on the TSP (see for example [LLKS85] or [ABCC06]). With the help of this research, it is now possible to solve instances with metric distances up to 85900 nodes. This is remarkable since the TSP is an NP-hard problem.

A problem related to the TSP is the so-called **Hamiltonian path problem (HP)**. The HP consist of finding a Hamiltonian path (instead of a tour) in a weighted, complete graph with minimum costs. The HP can be formulated as an IP which is very similar to (1.3)–(1.7):

$$\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij} \tag{1.8}$$

$$\text{s.}t.$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = n-1, \tag{1.9}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} \leq 1, \quad j \in V, \tag{1.10}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \leq 1, \quad i \in V, \tag{1.11}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ i \neq j}} x_{ij} \leq |S| - 1, \;\; S \subset V, \; 2 \leq |S| < n, \tag{1.12}$$

$$x_{ij} \in \{0,1\}, \; i,j \in V, \; i \neq j. \tag{1.13}$$

In this formulation the number of equations has been reduced to one. This single equation describes the fact that a Hamilitonian path on n nodes must contain exactly $n-1$ edges. So the equation families (1.4) and (1.5) have now turned into inequalities since there no longer necessarily exists an ingoing/outgoing edge for every node.

1.2.4 Linear Ordering Problem

The **linear ordering problem (LOP)** is another well-known combinatorial optimization problem.

Definition 1.3. *Let $D = (V, A)$ be a complete directed graph with n nodes. Furthermore a weight function $p : A \to \mathbb{R}$ is given. The objective is to find a cycle-free subgraph where the sum of weights is maximal and which contains exactly one of the edges (i, j) or (j, i) for each two distinct nodes i and j.*

Besides this graph definition, the LOP can also be seen as an ordering problem on a matrix. The objective is in this case to permute rows and columns simultaneously so that the upper diagonal sum becomes maximal.

For more details on this problem, see for example [RR11] or [Rei85].

Definition 1.4. *If there exist a solution for a given linear ordering instance where all preferences > 0 have been met, we say that the instance contains a complete ordering.*

The LOP can be expressed by a very simple IP model for which we define the binary variables w_{ij}, $1 \leq i, j \leq n$, $i \neq j$ with the definition ($\pi(i) = j$ when j is the i-th node in the ordering):

$$w_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(l) \text{ for some } 1 \leq k < l \leq n, \\ 0 & \text{otherwise.} \end{cases}$$

We can then formulate the following IP model:

$$\max \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} w_{ij} \tag{1.14}$$

$$\text{s.}t.$$

$$w_{ij} + w_{jk} + w_{ki} \leq 2, \quad i, j, k \in V, \; i < j, \; i < k, \; j \neq k, \tag{1.15}$$

$$w_{ij} + w_{ji} = 1, \quad i, j \in V, \quad i \neq j \tag{1.16}$$

$$w_{ij} \in \{0, 1\}, \quad i, j \in V \tag{1.17}$$

As one can see the model only contains one type of constraint. This constraint models the fact that it is not possible to order a before b before c before a. The LOP is way harder to solve than the TSP: While it is possible to solve metric instances with thousands of nodes with modern TSP codes, it is still a challenge to solve LOP instances with 200 nodes, even if they contain a nearly complete ordering.

1.3 Solving Methods for Integer Programs

Next we give a short introduction to the most common approaches for solving integer programs to optimality. We will present all these algorithmic approaches for the case of maximization problems, but of course it is possible to use them for minimization problems as well. It should be clear that these approaches cannot have a polynomial computation time in general. But still they work much more efficient on NP-hard problems than complete enumeration.

1.3.1 Branch-and-Bound

The key idea of branch-and-bound is to split up a problem into smaller subproblems, which can then either be solved, fathomed or split up again. This leads to the construction of a search tree with the original problem as root node. In addition we keep a global lower bound, which allows us to fathom branches of the search tree if we can prove that the value of the best optimal solution in this branch is less than the global lower bound. So if we are able to obtain strong global bounds quickly, then the search tree becomes in most cases much smaller than the one which occurs when we are doing complete enumeration.

This method has the advantage that the basic principle is simple, and it can be adopted, without much effort, to every combinatorial optimization problem.

We can also use branch-and-bound for solving IPs in the following way. First we solve the LP relaxation of an IP problem. If the LP solution is not fully integral, we select some variable x which has some non-integer value z in the LP solution. We then create two new problems: In the first problem we add the constraint $x \leq \lfloor z \rfloor$. In the second problem we add the constraint $x \geq \lceil z \rceil$. Then we execute the procedure on the new subproblems. The process stops when there exists no open subproblem. If a subproblem has an integer solution we keep this solution, if it is better than the currently best solution, otherwise we can fathom the subproblem. Also if a subproblem has a non-integral solution, which is less than the currently best integral solution, the subproblem can be fathomed and does not need to be examined any further. The details of this method can be seen in Algorithm 1.

Algorithm 1: Branch-and-Bound

```
1  Initialize Active_Subproblem_List with the global problem;
2  Initialize Best_Solution with $-\infty$;
3  if Active_Subproblem_List $=\emptyset$ then
4      Return: Best_Solution
5  else
6      Choose a problem $P$ of Active_Subproblem_List and solve the associated
       LP relaxation, denote solution by $x^*$;
7      if $P$ is infeasible or $x^* \leq$ Best_Solution then
8          Fathom P;
9          Goto 3;
10     if $x^* \in \mathbb{Z}$ and $x^* >$ Best_Solution then
11         Best_Solution= $x^*$;
12         Fathom P;
13         Goto 3;
14     else
15         Split the Subproblem into two new subproblems and add them to
           Active_Subproblem_List;
16         Fathom P;
17         Goto 3;
```

The critical issue of branch-and-bound is to obtain good global lower bounds in short time. We could do this in two ways: The first is running a heuristic before we execute the algorithm. The other one is to select at the beginning such subproblems that lead to a solution which is close to the optimal solution. If we choose the second option, then thinking about a good selection criteria in Step 6 of the algorithm is absolutely necessary.

1.3.2 The Cutting Plane Approach

Another technique for solving integer programs is the so-called cutting plane method. Consider an integer program and the associated set of feasible points

$$Q = \{x \in \mathbb{Z}^n \mid Ax \leq b\}. \tag{1.18}$$

We then solve the LP relaxation of the integer program and obtain a solution x^*. As long as $x^* \notin Q$ we add an inequality $ax \leq a_0$ to the integer program which is valid for all points $x \in Q$ but is not satisfied by x^*. We continue this process until the LP solution is fully integral. More formally the process is shown in Algorithm 2.

Algorithm 2: Cutting Plane

```
1 Initialize $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$;
2 Solve LP associated with P, obtain $x^*$;
3 if $x^* \in \mathbb{Z}^n$ then
4     Return $x^*$;
5 else
6     Obtain an inequality $ax \leq a_0$ with $ax^* > a_0$ but $ax \leq a_0 \ \forall x \in \mathbb{Z}^n \cap P$;
7     Update $P = P \cap \{ax \leq a_0\}$;
8     Goto 2;
```

The difficulty of this method is to find inequalities which cut off a big area of the actual polytope (described by the current set of inequalities). Good classes of inequalities for such a purpose are those which define facets of the polytope associated with the IP. Please note that it is not useful to add all inequalities of an inequality class (even if they are facets) at once to the LP because depending on the class there can exist a great number of inequalities of this type. So adding them all at once will slow down the process of solving the LP. Thus it is much better just to add inequalities which violate the current solution of the LP. The process of finding such a violated inequality is called **separation**.

Separation is the crucial part of the algorithm and must be done in a sensible way to obtain good results. In general, we have to pay attention to the following two aspects. On the one hand, it is important that we obtain violated inequalities quickly, on the other hand, the inequality should cut off a big area of the polytope.

So the best inequality classes are facet-defining inequalities of the associated polytope of the IP for which fast separation algorithms exist. In most cases we need knowledge about the optimization problem and its associated polytope to obtain such classes of inequalities.

1.3.3 Branch-and-Cut

Branch-and-cut is a technique which combines the branch-and-bound strategy with the cutting plane method. It is one of the frequently used methods for computing the exact solution of NP-hard combinatorial optimization problems.

The basic idea of this technique can be stated therefore as follows: First we compute the solution of the LP relaxation of the integer program. Then we successively add violated inequalities to strengthen the LP relaxation (cutting phase). The phase ends when no violated inequalities can be found anymore. Then we begin with the branching phase that works like we describe in Section 1.3.1 and obtain two new subproblems. Now we go on and apply the algorithm to both subproblems. The complete method is shown in Algorithm 3.

Algorithm 3: Branch-and-Cut

```
1  Initialize $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$;
2  Initialize Active_Subproblem_List with P;
3  Initialize Global_Lower_Bound with $-\infty$;
4  if $Active\_Subproblem\_List = \emptyset$ then Return: Global_Lower_Bound;
5  Take a subproblem $S$ from Active_Subproblem_List and solve the associated
   LP, obtain $x^*$;
6  if $x^* \in \mathbb{Z}^n$ and $x^* \geq Global\_Lower\_Bound$ then
7      Global_Lower_Bound = $x^*$;
8      Fathom $S$;
9      Goto 4;
10 if $x^* \leq Global\_Lower\_Bound$ then
11     Fathom $S$;
12     Goto 4;
13 else
14     while an inequality $ax \leq a_0$ with $ax^* > a_0$ but $ax \leq a_0 \; \forall x \in \mathbb{Z}^n \cap S$ can
       be found do
15         Update $S = S \cap ax \leq a_0$;
16         Update LP and obtain new $x^*$;
17         if $x^* \leq Global\_Lower\_Bound$ then
18             Fathom subproblem;
19             Goto 4;
20     Choose a variable $x_i$ and its actual LP solution $x_i^*$. Then split the
       problem so that $S_1 = S \cap \{x_i \geq \lceil x_i^* \rceil\}$ and $S_2 = S \cap \{x_i \leq \lfloor x_i^* \rfloor\}$;
21     Add these two problems to Active_Subproblem_List;
22     Goto 4;
```

Chapter 2

The Target Visitation Problem

Given a base position and a set of targets, the objective of the target visitation problem (TVP) consists of finding a tour which starts at the base, visits all targets exactly once in some order and returns to the starting point. So we are essentially looking for a Hamiltonian tour, but for the TVP the profit of a tour is depending on two weights, i. e., the value of a tour is the sum of pairwise preferences between the targets corresponding to their visiting sequence minus the sum of distances traveled. So the TVP combines the two combinatorial optimization problems, TSP and LOP. An illustration of the objective can be found in Figure 2.1.

Example: Given a base node, four targets and distances plus preference values, which are stated in the following two matrices (D gives the distances and P the preferences, B denotes the base node).

D	B	1	2	3	4
B		2	5	1	2
1	1		5	2	4
2	1	2		4	2
3	2	2	4		2
4	1	2	5	3	

P	1	2	3	4
1		3	2	1
2	2		1	2
3	5	4		2
4	3	5	1	

We now want to calculate the profit of the tour $B \to 3 \to 4 \to 2 \to 1 \to B$. Firstly we calculate the distance cost:

$$d_{B3} + d_{34} + d_{42} + d_{21} + d_{1B} = 1 + 2 + 5 + 2 + 1 = 11.$$

Then we calculate the sum of the met preferences:

$$p_{31} + p_{32} + p_{34} + p_{41} + p_{42} + p_{21} = 5 + 4 + 2 + 3 + 5 + 2 = 21.$$

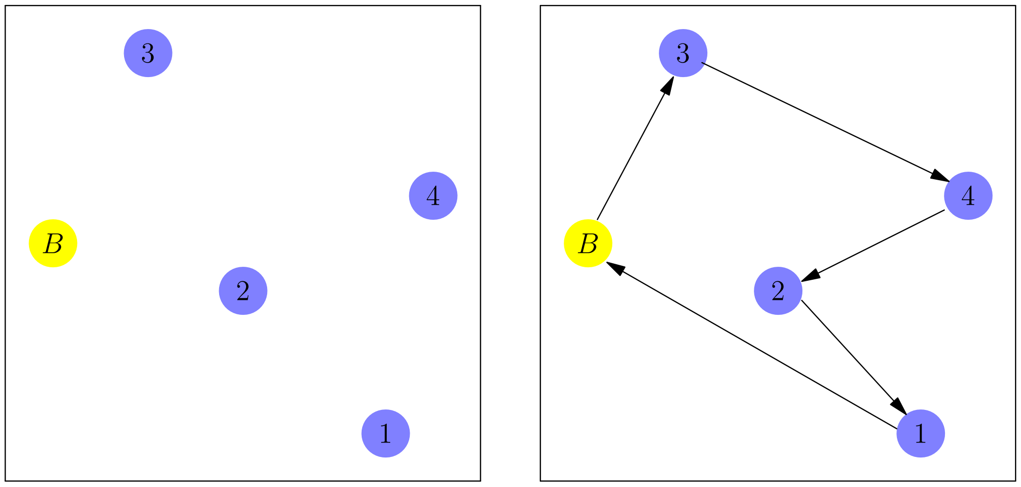


Figure 2.1: Objective of the TVP

In the end we subtract the distance cost from the sum of the met preferences and obtain the objective value of the tour:

$$21 - 11 = \mathbf{10}.$$

Next we state a formal definition of the target visitation problem.

Definition 2.1. *Let $D = (V, A)$ be the complete directed graph with n nodes where we set $V = \{0, 1, \ldots, n-1\}$. Furthermore there are two types of arc weights: weights d_{ij} (**distances**) for every arc (i, j), $0 \leq i, j \leq n-1$, $i \neq j$, and weights p_{ij} (**preferences**) associated with every arc (i, j), $1 \leq i, j \leq n-1$, $i \neq j$. The profit of a tour which starts at node 0, then visits all remaining nodes (called* targets*) exactly once in some order and returns to node 0 is defined as the difference of the sum of all met preferences and the sum of all distances traveled. The **target visitation problem** then consists of finding a most profitable tour.*

It is clear that every feasible TVP tour can be represented by a permutation π of $\{1, 2, \ldots, n-1\}$ where $\pi(i) = j$ if target j is visited as the i-th target. For convenience we also define $\pi(0) = 0$ and $\pi(n) = 0$. With this information we can formally express the cost of a TVP tour by:

$$\sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} p_{\pi(i)\pi(j)} - \sum_{i=0}^{n-1} d_{\pi(i)\pi(i+1)}.$$

Additionally to Definition 2.1, the TVP can be also defined on an undirected graph, which means that distances satisfy $d_{ij} = d_{ji}$ for all pairs of nodes. In this case the underlying TSP could then be treated in its symmetric version. To distinguish between these two variants we will call the symmetric case STVP and the asymmetric one just TVP since we deal with this case in most parts of the thesis.

Obviously, the TVP is NP-hard because it contains the traveling salesman problem ($p_{ij} = 0 \ \forall i, j \in V$) and the linear ordering problem ($d_{ij} = 0 \ \forall i, j \in V$) as special cases.

The TVP is a relatively new problem. Therefore the literature on this topic is very limited and consists only of a few papers. The problem was introduced in 2004 by Jeffcoat and Grundel in [GJ04]. Besides this there exists some work by Pardalos et al., which consist of two papers. Both papers examine a heuristic approach using genetic algorithms. The first one focuses on a hybrid genetic algorithm [ACP08] while the second one deals with a random key genetic algorithm. Also Blázsik et al. tried some other heuristics, which can be found in [BBI$^+$06]. Lastly [Hun14] considers a semidefinite programming approach on the target visitation problem, but its success is very limited. To the best of our knowledge there is no research besides these mentioned papers, especially so far there exists no approach for solving the problem to optimality.

The original application, which was the motivation resulting in the formulation of the TVP is the planning of optimal routes for unmanned aerial vehicles in military missions (see [GJ04] for a detailed description).

Besides this, there is a wide range of applications. The most important one is probably the scheduling of rescue missions in disaster areas. The problem in this case is to distribute food, medicine and other sorts of relief supplies. Hereby one has to take into account that on the one hand the distribution should be fast while on the other hand the goods are needed more eagerly in some places than in others.

Also any other routing problem where additional preferences (e. g., town cleaning, snow-plowing service, etc.) have to be taken into account can be a possible application of the TVP.

The base node plays a special role in the formulation. To make it easier for computations we want to get rid of this node. For this purpose we adapt an idea which has been used in TSP studies before (e. g., see in [QW93]). The key idea is to exploit the fact that each tour has to start at the base and return to it and that no preferences have to be taken into account for the base. So we can reduce the target visitation problem to just finding a path which visits all targets exactly once and forget the base node completely. This leads to the final definition of the TVP, which we want to use from now on in this thesis:

Definition 2.2. *Let $D = (V, A)$ be a complete directed graph with n nodes, where we set $V = \{1, \ldots, n\}$. Furthermore there are two types of arc weights: weights d_{ij} (**distances**) and weights p_{ij} (**preferences**) associated with every arc (i, j), $1 \leq i, j \leq n$, $i \neq j$. The profit of a path which visits all targets exactly once is defined as the difference of the sum of all met preferences and the sum of all distances traveled. The **reformulated target visitation problem** then consists of finding a most profitable path.*

An instance of the original TVP (on n nodes) can be transformed to an instance of the reformulated problem (on $n-1$ nodes) as follows. First we have to modify the distance matrix by setting: $d'_{ij} = d_{ij} - d_{i0} - d_{0j}$, for $1 \leq i, j \leq n-1$. Secondly we subtract the two constants $\sum_{i=1}^{n-1} d_{i0}$ and $\sum_{i=1}^{n-1} d_{0i}$ from the profit of a TVP path.

We will work with the path formulation in the rest of this thesis. An exception is Section 8.1, where we will give a short examination of the original formulation.

Chapter 3

Integer Programming Models for the TVP

As mentioned in the preliminaries, it is in most cases necessary to find a good IP formulation to be able to solve an NP-hard combinatorial optimization problem. In this chapter we therefore want to present different approaches for modeling the target visitation problem as an IP.

There exist different possibilities and we will start with modeling the TVP by combining the IP models of the two subproblems. Based on this first approach, we develop an extended formulation by adding a new type of variables. In the second part of this chapter we present two other models, one based on distance variables while the other uses the idea of variables which state the relation between an edge and a node.

3.1 An IP Model Based on the Combination of LOP and HP Models

First we combine the IP models of the two subproblems LOP and HP. For this we introduce two types of binary variables.

We define the variables x_{ij}, $1 \leq i, j \leq n$, $i \neq j$ with the interpretation:

$$x_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(k+1) \text{ for some } k \in \{1, \dots, n-1\} \\ 0 & \text{otherwise.} \end{cases}$$

The fact that some target i is visited before some target j is modeled with binary variables w_{ij}, $1 \leq i, j \leq n$, $i \neq j$ with the definition:

$$w_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(l) \text{ for some } 1 \leq k < l \leq n, \\ 0 & \text{otherwise.} \end{cases}$$

We denote the x-variables as *HP-variables* and the w-variables as *LOP-variables.*

By combining the IP formulations of the two subproblems (as stated in Sections 1.2.3 and 1.2.4.) we obtain a first IP model for the TVP:

$$\max \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} w_{ij} - \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij} \right) \tag{3.1}$$

$$s.t.$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = n - 1, \tag{3.2}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1, \; S \subseteq \{1, \ldots, n\}, \; 2 \leq |S| \leq n - 1, \tag{3.3}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} \leq 1, \qquad j \in V, \tag{3.4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \leq 1, \qquad i \in V, \tag{3.5}$$

$$w_{ij} + w_{jk} + w_{ki} \leq 2, \qquad i, j, k \in V, \; i < j, \; i < k, \; j \neq k, \tag{3.6}$$

$$w_{ji} + w_{ij} = 1, \qquad i, j \in V, \; i \neq j, \tag{3.7}$$

$$x_{ij} - w_{ij} \leq 0, \qquad i, j \in V, \; i \neq j, \tag{3.8}$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in V, \; i \neq j, \tag{3.9}$$

$$w_{ij} \in \{0, 1\}, \quad i, j \in V, \; i \neq j. \tag{3.10}$$

The correctness of the model is shown in the following lemma.

Lemma 3.1. *The model presented in (3.1)–(3.10) is a correct IP model for the TVP.*

Proof. Firstly, we have to prove that every (w, x) vector which describes a feasible TVP path fulfills the model. Since (3.2)–(3.5), (3.9) is an IP model for the HP and

(3.6)–(3.7), (3.10) is a model for the LOP it is clear that if the x- and w-variables describe the same TVP path (3.8) must also be fulfilled.

Secondly we have to show that every feasible solution of (3.1)–(3.10) is a correct TVP path. It is clear that every feasible integer solution must induce a feasible linear ordering and a feasible Hamiltonian path. So it is sufficient to show that the values of x_{ij} and the values of w_{ij} match or, in other words, that both types of variables describe the same Hamiltonian path. To assure this, it is sufficient to prove the following two facts:

a) $x_{ij} = 1 \Rightarrow w_{ij} = 1$,

b) $w_{ij} = 1 \Rightarrow i$ is visited before j in the path described by the x-variables.

Because (3.8) must be fulfilled, a) is obvious. To prove b), we assume that $w_{ij} = 1$ and j is visited before i in the path described by the x-variables. This means that there exist indices $k_1, \dots, k_l$ so that $(j, k_1, \dots, k_l, i)$ is part of the path. So it follows that $x_{jk_1} = x_{k_1k_2} = \cdots = x_{k_li} = 1$. With a) we get that $w_{jk_1} = w_{k_1k_2} = \cdots = w_{k_li} = 1$. Because of (3.6) and (3.7) we can then iteratively conclude that $w_{jk_2} = 1,\ w_{jk_3} = 1, \dots, w_{ji} = 1$. But with (3.7) this is a contradiction to our assumption. □

As an interesting fact we note that the subtour elimination constraints (3.3) are actually not needed.

Lemma 3.2. *If a vector (w, x) satisfies (3.2) and (3.4)–(3.10), then it satisfies also the subtour elimination constraints (3.3).*

Proof. Let $S \subseteq V$ and $|S| = 2$. If $x_{ij} + x_{ji} > 1$, then $x_{ij} = x_{ji} = 1$ and therefore $w_{ij} + w_{ji} = 2$ which is a contradiction to (3.7). Now let $|S| \geq 3$. If (w, x) satisfies (3.2), (3.4)–(3.10), then the only way inequality (3.3) can be violated is the existence of a subtour, i. e., there exists a subset of k nodes $k < |S|$ (where w.l.o.g. we can assume that the set is $\{1, 2, \dots, k\}$) such that

$$x_{12} = x_{23} = \cdots = x_{k-1k} = x_{k1} = 1.$$

Because of that it follows that

$$w_{12} = w_{23} = \cdots = w_{k-1k} = w_{k1} = 1.$$

For some nodes $i, j, k \in V$ it must be true that if we have $w_{ij} = w_{jk} = 1$ then because of (3.6) it follows $w_{ki} = 0$. Together with (3.7) it follows that $w_{ik} = 1$. With this argument we can iteratively conclude that because of

$$w_{34} = ... = w_{k-1k} = w_{k1} = 1$$

it must be true that $w_{31} = 1$. But together with $w_{12} = w_{23} = 1$ this would violate $w_{13} + w_{23} + w_{31} \leq 2$.

□

x	1	2	3	4	5
1	$\times$	$\frac{1}{2}$	$\frac{1}{2}$	0	0
2	$\frac{1}{2}$	$\times$	0	$\frac{1}{2}$	0
3	$\frac{1}{2}$	0	$\times$	$\frac{1}{2}$	0
4	0	$\frac{1}{2}$	$\frac{1}{2}$	$\times$	0
5	0	0	0	0	$\times$

w	1	2	3	4	5
1	$\times$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
2	$\frac{1}{2}$	$\times$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
3	$\frac{1}{2}$	$\frac{1}{2}$	$\times$	$\frac{1}{2}$	$\frac{1}{2}$
4	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\times$	$\frac{1}{2}$
5	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\times$

Table 3.1: Example of an LP solution which is only violated by the subtour elimination constraint

As a consequence we can eliminate Constraint (3.3) from the model. An interesting fact is that with this we also have an IP formulation for the TSP with a polynomial number of constraints. We would like to point out that there exist non-integral solutions which satisfy all constraints but (3.3). An example illustrating this fact is shown next:

Example: Let $V = \{1, \dots, 5\}$ and consider the LP solution in Table 3.1.

Then (3.2), (3.4)–(3.10) are satisfied. But for the subtour elimination constraint on $\{1, 2, 3, 4\}$ we obtain

$$x_{12} + x_{13} + x_{14} + x_{21} + x_{23} + x_{24} + x_{31} + x_{32} + x_{34} + x_{41} + x_{42} + x_{43} = 4$$

i. e., it is violated.

The binary Constraints (3.9) are also not needed because if all w_{ij} are integral, they force the x_{ij} to be integral as well. This fact is shown in the next lemma.

Lemma 3.3. *If a vector (w, x) satisfies (3.2), (3.4)–(3.8), (3.10) and $x_{ij} > 0$, $\forall i, j$ it also satisfies (3.9).*

Proof. Let the w_{ij}-variables be integral and describe correctly w.l.o.g. the path $1 \to 2 \to \cdots \to n$.

Because of (3.8), all x_{ij} with $i > j$ must be zero. This means that the n inequalities

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} \leq 1, \ j \in V$$

turn to the $n - 1$ inequalities

$$\sum_{i=1}^{j-1} x_{ij} \leq 1, \ j \in V.$$

x_{12}	$+$	x_{13}	$+$	x_{14}	$+$	$\dots$	$+$	x_{1n-1}	$+$	x_{1n}	$=1$
		$+$		$+$				$+$		$+$	
		x_{23}	$+$	x_{24}	$+$	$\dots$	$+$	x_{2n-1}	$+$	x_{2n}	$=1$
				$+$				$+$		$+$	
				x_{34}	$+$	$\dots$	$+$	x_{3n-1}	$+$	x_{3n}	$=1$
								$+$		$+$	
								$\vdots$		$\vdots$	
								$+$		$+$	
								x_{n-2n-1}	$+$	x_{n-2n}	$=1$
										$+$	
										x_{n-1n}	$=1$
$=$		$=$		$=$				$=$		$=$	
1		1		1				1		1	

Table 3.2: Sketch for the proof of Lemma 3.3

This applies also for the n inequalities $\sum\limits_{\substack{j=1\\j\neq i}}^{n} x_{ij} \leq 1,\ i \in V$.

It must also be true that all these inequalities must be satisfied with equality or (3.2) would be violated. But this means $x_{12} = 1$. Because of $\sum_{j=2}^{n} x_{1j} = 1$, we can conclude that $x_{1j} = 0$ for $3 \leq j \leq n$. Together with $x_{13} + x_{23} = 1$ this means $x_{23} = 1$.

Then because of $\sum_{j=3}^{n} x_{2j} = 1$, it is clear that $x_{2j} = 0$ for $4 \leq j \leq n$, which together with $x_{14} + x_{24} + x_{34} = 1$ leads to $x_{34} = 1$. We can continue this process until the integrality of all x_{ij} is proven. The main idea of the proof is sketched in Table 3.2.

□

Remark 3.4. *Note that obviously the equations $w_{ij} + w_{ji} = 1$ (called tournament equations) have to hold for all $1 \leq i < j \leq n$ because either i is visited before j or j is visited before i. By replacing all $\{w_{ij} \mid j < i\}$ by $1 - w_{ji}$ we can reduce the LOP-variables to the set $\{w_{ij} \mid i < j\}$. Then the 3-cycle inequalities turn into $0 \leq w_{ij} + w_{jk} - w_{ik} \leq 1$ for all $1 \leq i < j < k \leq n$ and the part of the objective function for the LOP-variables reads:*

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [(p_{ij} - p_{ji})w_{ij} + p_{ji}].$$

As a result of Lemma 3.2 and Remark 3.4 we get an improved version of the model stated in (3.1)–(3.10):

$$\max\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}[(p_{ij}-p_{ji})w_{ij}+p_{ji}]-\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}d_{ij}x_{ij}\right) \tag{3.11}$$

$$s.t.$$

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}x_{ij}=n-1, \tag{3.12}$$

$$\sum_{\substack{i=1\\i\neq j}}^{n}x_{ij}\leq 1, \qquad j\in V, \tag{3.13}$$

$$\sum_{\substack{j=1\\j\neq i}}^{n}x_{ij}\leq 1, \qquad i\in V, \tag{3.14}$$

$$0\leq w_{ij}+w_{jk}-w_{ik}\leq 1, \qquad 1\leq i<j<k\leq n, \tag{3.15}$$

$$x_{ij}-w_{ij}\leq 0, \qquad i,j\in V,\ i<j, \tag{3.16}$$

$$x_{ij}\in\{0,1\}, \quad i,j\in V,\ i\neq j, \tag{3.17}$$

$$w_{ij}\in\{0,1\}, \quad i,j\in V,\ i<j. \tag{3.18}$$

We denote this model by TVP_{HP} and the convex hull of the feasible 0/1 solutions of this model by P^n_{TV}.

Finally we like to point out that it is possible to calculate the position of node i in the path with the help of the LOP-variables as:

$$n-\sum_{\substack{j=1\\i\neq j}}^{n}w_{ij}.$$

3.1.1 Extended Formulation of the TVP_{HP} Model

The use of extended formulations is a common technique, which is used to strengthen the LP relaxation of a combinatorial optimization problem. The key idea of this approach is to add new variables and constraints to a given IP formulation so that the gap between the solution of the LP relaxation and the optimal integral solution becomes smaller. We will see in the following that it can be useful to formulate such an extended formulation for the TVP_{HP} model.

We construct an extended formulation for the TVP by adding three-indexed variables, which are a generalization of the linear ordering variables from the TVP_{HP} model. These new variables w_{ijk} are defined as follows:

$$w_{ijk} := \begin{cases} 1 & \text{if } i = \pi(a),\ j = \pi(b) \text{ and } k = \pi(c) \text{ for} \\ & \text{some } 1 \leq a < b < c \leq n, \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

So this new type of variables is an extension of the w_{ij}-variables. In the objective function we assign zero coefficients to the new variables. In order to extend our standard model, we also need to introduce two new classes of constraints to make sure that the solution of the new variables and the old x_{ij}- and w_{ij}-variables are consistent. The extended formulation looks as follows:

$$\max \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [(p_{ij} - p_{ji}) w_{ij} + p_{ji}] - \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij} \right) \tag{3.20}$$

$$s.t.$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = n - 1, \tag{3.21}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} \leq 1,\ j \in V, \tag{3.22}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \leq 1,\ i \in V, \tag{3.23}$$

$$0 \leq w_{ij} + w_{jk} - w_{ik} \leq 1,\ 1 \leq i < j < k \leq n, \tag{3.24}$$

$$w_{ij} + w_{jik} + w_{jki} + w_{kji} = 1,\ i, j, k \in V,\ i < j,\ i \neq k,\ j \neq k, \tag{3.25}$$

$$x_{ij} - w_{ijk} - w_{kij} \leq 0\ i, j, k \in V,\ i \neq j,\ j \neq k,\ i \neq k, \tag{3.26}$$

$$x_{ij} - w_{ij} \leq 0,\ i, j \in V,\ i < j, \tag{3.27}$$

$$x_{ij} \in \{0, 1\},\ i, j \in V,\ i \neq j, \tag{3.28}$$

$$w_{ij} \in \{0, 1\},\ i, j \in V,\ i < j, \tag{3.29}$$

$$w_{ijk} \in \{0, 1\},\ i, j, k \in V,\ i \neq j,\ j \neq k,\ i \neq k. \tag{3.30}$$

In the following we will denote this model by TVP_E and the associated polytope by P^n_{ETV}. As one can see (3.25) matches the new variables and the w_{ij}-variables and (3.26) matches the new variables and the x_{ij}-variables. We would like to point out that the model would still be correct without (3.25). But experiments have shown that the gap between the LP relaxation and the optimal integer solution would be far worse without it.

Lemma 3.5. *The formulation (3.20)–(3.30) is a correct model for the TVP.*

Proof. Without (3.25), (3.26) and (3.30) this model is correct since it is equivalent to the TVP_{HP} model. Thus, it remains to show that (3.25) and (3.26) are correct and sufficient to describe the w_{ijk}-variables as defined in (3.19).

The correctness of (3.25) is quite obvious: If i is not visited before j, then j must be visited before i and a third node k must be visited before, in between or after these two nodes. Constraint (3.26) just encodes the fact that when i is visited directly after j, a third node k must be visited either before or after these two nodes.

It remains to show that (3.25) is sufficient to describe the w_{ijk}-variables as defined in (3.19). If w_{ij}, w_{jk} and w_{ik} are equal to one, then $w_{ji} = 0$ and because of (3.25) the variables w_{ikj} and w_{kij} must be equal to zero. Which means that we get from $w_{ji} + w_{ijk} + w_{ikj} + w_{kij} = 1$ that $w_{ijk} = 1$. On the other hand: if $w_{ijk} = 1$ it follows from (3.25) that w_{ji}, w_{kj} and w_{ki} are zero and therefore w_{ij}, w_{jk} and w_{ik} are equal to 1.

□

3.2 The Edge-Node Formulation

The key idea of the next model is to combine the w and x-variables of the TVP_{HP} model to new three-indexed variables which state the relation between a node k and a fixed edge (i, j). More precisely we define for $i, j, k \in V$ $i \neq j,\ k \neq i$ the variables

$$w_k^{ij} := \begin{cases} 1 & \text{if } i = \pi(a),\ j = \pi(a+1) \text{ and } k = \pi(b) \text{ for} \\ & \text{some } 1 \leq a < b \leq n, \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

A valid IP model based one these variables can then be formulated as follows.

$$\max \left(\sum_{i=1}^{n} \sum_{\substack{j=1\\ i\neq j}}^{n} p_{ij} (\sum_{\substack{m=1\\ m\neq i}}^{n} w_j^{im}) + \sum_{i=1}^{n} \sum_{\substack{j=1\\ i\neq j}}^{n} d_{ij} w_j^{ij} \right) \tag{3.32}$$

$$s.t.$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1\\ j\neq i}}^{n} (w_j^{ij}) = n-1, \tag{3.33}$$

$$\sum_{\substack{i=1\\ i\neq j}}^{n} w_j^{ij} \leq 1, \quad j \in V, \tag{3.34}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} w_j^{ij} \leq 1, \quad i \in V, \tag{3.35}$$

$$\sum_{\substack{l=1\\ l\neq i}}^{n} w_j^{il} + \sum_{\substack{l=1\\ l\neq j}}^{n} w_k^{jl} + \sum_{\substack{l=1\\ l\neq k}}^{n} w_i^{kl} + w_k^{ik} \leq 2 \ \ i,j,k \in V, \ i \neq j, \ j \neq k, \ i \neq k, \tag{3.36}$$

$$w_k^{ij} - w_j^{ij} \leq 0, \ i,j,k \in V, \ i \neq j, \ j \neq k, \ i \neq k, \tag{3.37}$$

$$w_j^{ij} + w_l^{jk} - w_l^{ij} \leq 1, \ i,j,k,l \in V, \ i \neq j \neq k, \ i \neq l \neq j, \ i \neq k, \tag{3.38}$$

$$w_k^{ij} \in \{0,1\}, \ i,j,k \in V, \ i \neq j, \ i \neq k. \tag{3.39}$$

In the following we will denote this model by TVP_{EN} and its associated polytope with P^n_{ENTV}.

Lemma 3.6. *The model presented in (3.32)–(3.39) is a correct IP model for the TVP.*

Proof. The variables of type w_j^{ij} together with the Constraints (3.33)–(3.35) are equal to the model of the Hamiltonian path problem presented in Section 1.2.3. We can therefore assume that the variables w_j^{ij} must be correct for all $i, j \in V$. Therefore it remains to show that:

a) The Constraints (3.36)–(3.38) are correct.

b) If and only if $w_k^{ij} = 1$, then k must be visited after i and j and edge (i, j) must be contained in the path (i. e., the Constraints (3.33)–(3.39) describe the w_k^{ij}-variables correctly according to (3.31).

c) The objective function (3.32) is correct.

Proof of a):

- Constraint (3.37): It is obvious that if k is visited after edge (i, j), edge (i, j) must be contained in the path.
- Constraint (3.38): It is also obvious that if l is visited after edge (j, k) and edge (i, j) is contained in the path, then l must be visited after edge (i, j).
- Constraint (3.36): This type of inequality is similar to the three-cycle constraint (3.6) in the TVP_{HP} model. It states that of the four constraints j is visited after i, k is visited after j, i is visited after k and k is visited directly after i only two could be fulfilled at the same time.

Proof of b): We can conclude from (3.37) that all w_k^{ij} must be zero if edge (i, j) is not contained in the path implied by the w_j^{ij}-variables. So it remains to show that $w_k^{ij} = 1$ if k is visited after (i, j) and $w_k^{ij} = 0$ if k is not visited after (i, j) in the path.

We start by showing the first statement. Assume that k is visited after (i, j) and consider the part of the path $(\dots, i, j, l, m, \dots)$. With $w_j^{ij} = 1$ and w_l^{jl} it follows because of (3.38) that $w_l^{ij} = 1$. With the same argument we can show that $w_m^{jl} = 1$. Again, with (3.38), it follows then that $w_m^{ij} = 1$. We can continue this process until we obtain the result that $w_k^{ij} = 1$.

Now assume that $w_k^{ij} = 1$ and k is not visited after (i, j).
Case 1: k is not the direct predecessor of i: Then denote the successor of k by l. It is obvious that there must exist an m so that $w_i^{lm} = 1$. But together with $w_l^{kl} = 1$ we get a contradiction because of (3.36).
Case 2: k is the direct predecessor of i: Because $w_i^{ki} = 1$ and $w_j^{ij} = 1$ a contradiction also follows with (3.36).

Proof of c): Since the w_j^{ij}-variables are equal to the x_{ij}-variables of the Hamiltonian path problem it is obvious the the distance cost are computed correctly. Because exactly one of $n - 1$ the variables w_j^{im}, $m = 1, \dots, i-1, i+1, \dots, n$ is equal to 1 if and only if j is visited after i, the met preferences are also calculated correctly.

$\square$

The model would be also correct without (3.37). But experiments have shown that the gap between the LP relaxation and the optimal integer solution would be far worse without this constraint.

An advantage of this model is that we only have one type of variables. Unfortunately the price we have to pay for this is a cubic number of variables.

It is also possible to formulate an analogous IP model with variables (defined for $i, j, k \in V$ and $1 \leq i, j, k \leq n,\ i \neq j, k \neq j$) of the following type:

$$w_{ij}^k := \begin{cases} 1 & \text{if } k = \pi(a),\ i = \pi(b) \text{ and } j = \pi(b+1) \text{ for some } 1 \leq a \leq b \leq n-1\ , \\ 0 & \text{otherwise.} \end{cases}$$

3.3 A Model Based on Distance Variables

Another idea for constructing an IP model for the TVP is due to E. Fernandez from the UPC Barcelona. The key idea of this approach is the use of distance variables. In detail we define variables z_{ij}^t which describe whether there are exactly t arcs between i and j in the path or not. More formally we define for $i, j \in V$ and $1 \leq i, j, \leq n,\ i \neq j, t \in \{1, 2, \ldots, n-1\}$ the variables:

$$z_{ij}^t = \begin{cases} 1 & \text{if the solution contains a path with } t \text{ arcs from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$$

Since we do not longer distinguish between distance and ordering variables, it is necessary to adjust the coefficients in the following way:

$$h_{ij}^t = \begin{cases} p_{ij} - d_{ij} & \text{if } t = 1, \\ p_{ij} & \text{otherwise.} \end{cases}$$

With this modification we are now able to formulate a TVP model with distance variables:

$$\max \sum_{i=1}^{n} \sum_{\substack{j=1\\j\neq i}}^{n} \sum_{t=1}^{n-1} h_{ij}^t z_{ij}^t \tag{3.40}$$

$$s.t.$$

$$\sum_{\substack{i=1\\i\neq j}}^{n} z_{ij}^1 \leq 1, \quad j \in V, \tag{3.41}$$

$$\sum_{\substack{j=1\\j\neq i}}^{n} z_{ij}^1 \leq 1, \quad i \in V, \tag{3.42}$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1\\j\neq i}}^{n} z_{ij}^k = n-k, \quad k \in \mathbb{N}, k \leq n-1, \tag{3.43}$$

$$z_{ij}^{t_1} + z_{jk}^{t_2} \leq z_{ik}^{t_1+t_2} + 1, \; i,j,k \in V, t_1, t_2 \in \mathbb{N}, i \neq j, j \neq k, \; i \neq k, t_1+t_2 \leq n-1, \tag{3.44}$$

$$\sum_{t=1}^{n-1} z_{ij}^t + z_{ji}^t = 1, \quad i,j \in V, i \neq j, \tag{3.45}$$

$$z_{ij}^t \in \{0,1\} \;\; i,j \in V, i \neq j, t \in \mathbb{N}, t \leq n-1. \tag{3.46}$$

We will denote this model by TVP_D and the associated polytope by P_{DTV}^n. Again we have the advantage that this model contains just one type of variables. Nevertheless the z_{ij}^1-variables still play a special role, for example in the objective function.

Lemma 3.7. *The model presented in (3.40)–(3.46) is a correct IP model for the TVP.*

Proof. The Constraints (3.41), (3.42) together with (3.43) for $k = 1$ describe an IP model for the Hamiltonian path problem shown in Section (1.2.3). That means that the z_{ij}^1-variables have to describe a Hamiltonian path. Because of (3.44) it is clear that if z_{ij}^1 and z_{jk}^1 are equal to one, then z_{ik}^2 must be also equal to one. So we get $n-2$ variables of the type z_{ij}^2 which must be equal to one. Because of (3.43) it is clear that all the other variables of this type must then be zero. Iteratively we can conclude that the z_{ij}^k-variables with $k > 2$ are also correct.

It remains to show that (3.45) is correct. But this is obvious, since there is exactly one path from i to j or from j to i. □

Chapter 4

Polyhedral results

In this chapter we want to examine the associated polytopes of the IP models which have been presented in the last section. One of the main aims of this examination is to obtain results which can be used for successfully implementing a branch-and-cut approach for solving the TVP to optimality.

So first we evaluate which of the proposed models may be useful for such an approach. An interesting fact in this context is the strength of the LP relaxation (e. g., the average size of the gap between the solution of the LP relaxation and the optimal integer solution). We make several computations with CPLEX 12.1 to investigate this gap for the different TVP models. An excerpt of these test runs is shown in Table 4.1. The results of all tests can be found in Appendix C. For a description of the test instances listed in Table 4.1 see Section 7.1.

When considering Table 4.1, one can observe the following facts: The smallest gap between the optimal integral solution and the LP solution can be found in case of the TVP_E model. This is not so surprising since this model was constructed with the aim to have tight bounds. Another observation is that in case of the TVP_D model the gap is very large. Also because of the big number of constraints, it was not even possible to solve the LP relaxation of the instances with 26 nodes. So we have to state that for practical purposes this model seems not to fit as long as we are not able to strengthen it with some additional constraint classes and get rid of the five-indexed Constraint (3.44).

The remaining two models have similar gaps which are far worse than the ones of the extended formulation. But the computation of the LP relaxation for the TVP_{EN} model takes much more time than for the TVP_{HP} formulation.

Because of these facts we will focus in the polyhedral examination mainly on the polytopes P^n_{TV} and P^n_{ETV}.

Another concern for a successful implementation of a branch-and-cut approach is knowledge of the facial structure of the associated polytope. A first measure for

Name	Opt.	TVP_{HP}	TVP_E	TVP_{EN}	TVP_D
ER_CFO_15_1	−6435	94.0	−6406.1	2370.0	7920.5
LD_CFO_15_1	107620	115361.5	111246.2	112091.8	124233.8
ER_MCO_15_2	−5738	−3099.8	−4668.6	1632.5	3256.3
LB_MCO_15_1	3798	10007.6	3861.1	7314.1	14929.6
LB_CFO_20_2	9494	13743.5	9496.8	11848.8	21376.0
ER_CFO_20_1	−7417	2699.5	103.0	2985.8	11652.4
LB_MCO_20_1	14272	22189.7	17660.7	20036.2	34352.1
ER_MCO_20_1	905	8018.3	1208.5	7226.6	13713.7
ER_BCO_20_2	−9682	−3751.5	−6640.3	−4844.8	8190.0
ER_CFO_23_1	−18453	−4124.8	−15426.9	−3465.5	3798.3
ER_CFO_23_3	−8966	1560.1	−6751.9	1998.7	15666.5
LD_BCO_23_1	131109	154173.0	138900.8	144670.8	180248.8

Table 4.1: Solution of the LP relaxation of the different TVP models for instances with 15–23 nodes

the complexity of the polyhedral structure is the complexity of the description of the polytopes for small n. To examine small TVP polytopes we use two algorithms. Namely this is PORTA [CL09] for calculating all facets and HUHFA for classifying these facets due to symmetries. Details on the HUHFA algorithm are presented in the following.

We have developed an algorithm which is able to classify facets of a given polytope due to symmetries. Since we used this algorithm to classify small TVP polytopes we will in following give a short overview of this topic. For further information on this subject consult [HHS$^+$13].

As we have seen in Section 1.3.2, it is necessary to obtain good cutting planes which can be used in the algorithm. This is equivalent to trying to obtain facet classes of a given polytope. One way facet classes can be obtained, is to study the complete linear description of small polytopes in order to generalize the equations and inequalities. But “small polytopes” might actually not look so small at first sight: There is often a huge number of facet-defining inequalities already for very small problem sizes.

However, there are also often many symmetries implied by the combinatorial structure of the problem which can be used to classify the facets. These symmetries act on the feasible solutions and naturally form a group. In their representation as maps on the variable values they can be extended to symmetries acting on the polytope, and one can easily prove that they map vertices of the polytope to vertices of the polytope. We say that those facet-defining inequalities which are similar in the sense that they can be transformed onto each other by some symmetry belong to one class.

Understanding all the facet-defining inequalities of a combinatorial optimization problem polytope then reduces to understanding one facet from each class.

To do this classification, one applies the symmetries to the facet-defining inequalities and then checks whether any two facets can be transformed into each other and hence belong to the same class. Often, this check is not so easy as two linear expressions describing the same facet might differ by the sum of multiples of several equalities from the problem description.

The check can be accomplished by defining a so-called normal form for the representation of inequalities which have the same normal form describe the same facet. To this end, problem-specific normal forms were developed for some extensively studied combinatorial optimization problems. For the TSP, every facet-defining inequality can be efficiently transformed to the so-called tight triangular normal form [NR93]. For an example of a normal form for the LOP, see [Rei85]. In general, the representation of facet-defining inequalities in the orthogonal complement of the linear subspace spanned by the equations can be of course used as a normal form for the facets of a polytope. However, this needs techniques from linear algebra and can therefore raise numerical issues. Unfortunately, normal forms that can be described combinatorially are often not known. Hence, having a method that can be applied to every combinatorial optimization problem and relies solely on the combinatorial structure of the polytope is desirable.

For this problem, we have developed a novel technique for classifying facets without using normal forms. The main idea there is to identify every facet defining inequality with the vertices of the polytope which satisfy it with equality. With this method, complete descriptions of polytopes computed by a software like PORTA [CL09] (or a similar package) can be analyzed to divide the facets into equivalence classes according to groups generated by given symmetry mappings.

We also developed an algorithm for the classification called HUHFA which was implemented in C++ and which has been used in this thesis.

In case of the TVP polytopes we are able to calculate P^4_{TV}, P^4_{ENTV} and P^4_{ETV} with the help of PORTA [CL09]. We then sort the inequalities which describe these polytopes with the HUHFA algorithm by symmetry classes. We take into account permutation symmetries (i. e., interchanges of the nodes) and switching symmetries (i. e., change of the orientation of all edges). A summary of the results of these investigations is shown in Table 4.2. The full description of all polytopes can be found in Appendix A. In case of TVP_{HP} and TVP_E we will for the sake of a clearer representation present all facet classes in a form which includes also w_{ij}-variables with $i > j$. We will also do this from now on every time we talk about some fixed facet class. However, general results for facets will be proved for the reduced model which contains the w_{ij}-variables only for $i < j$.

For TVP polytopes with $n > 4$ the computation becomes more difficult because the number of inequalities is tremendous. We therefore cannot use PORTA anymore but we have to think about different approaches. Details on this topic can be found in Section 4.1.2.

Polytope	# Facets	# Facet-Classes
P^4_{TV}	1280	48
P^4_{ENTV}	512	24
P^4_{ETV}	144	7

Table 4.2: Key facts of different TVP polytopes

4.1 Polyhedral Results of the TVP_{HP} Model

In this section we first discuss the dimension of the P^n_{TV} polytope. Then we present some facet classes and prove a zero-lifting result. Also some general facet classes will be introduced in the end of this section.

4.1.1 Dimension of the P^n_{TV} Polytope

Let $(w^S, x^S,)$ be the characteristic vector of a TVP path S on the node set $\{1, \ldots, n\}$. Then x^S is the characteristic vector of a directed Hamiltonian path and w^S is the characteristic vector of a linear ordering.

The dimension of the corresponding Hamiltonian path polytope P^n_{HP} is n^2-n-1 as shown in [QW93], and the dimension of the linear ordering polytope P^{n-1}_{LOP} is $\frac{(n-1)n}{2}$ as shown in [GJR85]. (The valid equations for both polytopes are given in the TVP_{HP} model) Therefore

$$\dim P^n_{TV} \leq \dim P^n_{HP} + \dim P^n_{LOP} = \frac{3n^2-3n-2}{2}. \tag{4.1}$$

In the following we will show that in fact equality holds in this statement for $n \geq 4$. For $n = 3$ explicit calculations yield $\dim P^3_{TV} = 1$.

Let MTV_n denote the matrix of all characteristic tour vectors for the TVP_{HP} model with n nodes (written row-wise). Of course, this matrix is not unique as the order of the rows can be changed. The column associated with variable x_{ij} is denoted by T_{ij} and the column associated with variable w_{ij} is denoted by L_{ij}. Together there are exactly $n(n-1)+\frac{n(n-1)}{2}=\frac{3n^2-3n}{2}$ columns.

Theorem 4.1. *Every column L_{ij} of MTV_n is linearly independent of all other columns for $n \geq 4$.*

Proof. By using a computer algebra system one gets that all solutions of

$$\sum_{i=1}^{3}\sum_{j=i+1}^{4} L_{ij}\alpha_{ij} + \sum_{i=1}^{4}\sum_{\substack{j=1\\ j\neq i}}^{4} T_{ij}\beta_{ij} = \mathbf{0}$$

satisfy $\alpha_{ij}=0$ for all $1 \leq i < j \leq 4$. Therefore the statement holds for $n=4$.

We proceed by induction and assume that the statement is true for some $n \geq 4$, i. e., for all solutions of

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} L_{ij}\alpha_{ij} + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} T_{ij}\beta_{ij} = \mathbf{0} \tag{4.2}$$

we have $\alpha_{ij}=0\ \forall i,j \in V$.

Consider the equation

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} L_{ij}\alpha_{ij} + \sum_{i=1}^{n+1}\sum_{\substack{j=1\\ j\neq i}}^{n+1} T_{ij}\beta_{ij} = \mathbf{0} \tag{4.3}$$

and assume that it is solved by a vector (α,β) of variables such that $\alpha_{pq} \neq 0$ for some p and q. We will now show that in this case also System (4.2) has a solution with a nonzero value in the linear ordering columns.

For any n and $i \leq n$ let MTV_n^i be the block of rows of MTV_n corresponding to the TVP paths where target i is visited first. Choose i so that the block MTV_{n+1}^i where the coefficients column corresponding to α_{pq} is neither equal to $\mathbf{1}$ nor to $\mathbf{0}$. It is easy to see that such a block must exist. The subsystem of (4.3) defined by the rows of MTV_{n+1}^i is also satisfied by (α,β). We can make the following observations:

– The n columns $T_{ji},\ j \in \{1,\ldots,i-1,i+1,\ldots,n+1\}$ are zero vectors in the subsystem since i has no predecessor.

– The $i-1$ columns $L_{ji},\ j \in \{1,\ldots,i-1\}$ are zero vectors in the subsystem since i is the first node which will be visited in the tour.

– The $n-i$ columns $L_{ij},\ j \in \{i+1,\dots,n\}$ are equal to 1. Since i is the start node of the path, we know that $\sum_{k=1,k\neq j}^{n+1} T_{kj} = \mathbf{1}$, for all $j \neq i$. With this we can modify (α,β) so that it is still a valid solution of the subsytem: $\alpha_{ij}L_{ij} = \alpha_{ij}\mathbf{1} = \alpha_{ij}\sum\limits_{\substack{k=1\\k\neq j}}^{n} T_{kj}$. So we can define the vector (α',β') by setting

$$\beta'_{kj} = \begin{cases} \beta_{kj} + \alpha_{ij} & k \in \{1,\dots,n\}, j \in \{i+1,\dots,n\}, \\ \beta_{kj} & \text{otherwise} \end{cases}$$

and

$$\alpha'_{kj} = \begin{cases} 0 & k = i, j \in \{i+1,\dots,n\}, \\ \alpha_{kj} & \text{otherwise.} \end{cases}$$

– With $\sum\limits_{\substack{j=1,\\j\neq k}}^{n+1} T_{jk} = \mathbf{1}$ for all $k \neq i$ we can also substitute all T_{ik} by $\mathbf{1} - \sum\limits_{\substack{j=1,\\j\neq i,\, j\neq k}}^{n+1} T_{jk}$ and adjust the coefficients as above.

Now we relabel all nodes $j > i$ with $j = j-1$. We also relabel all variables and coefficient columns in the same way and denote them with L''_{kj}, T''_{kj}, w''_{kj}, x''_{kj}. Now the system will be equal to (4.2). By construction the vector (α'',β'') satisfies this system and we still have $\alpha''_{pq} \neq 0$. This is a contradiction and therefore the theorem has been proved. □

Corollary 4.2. *For $n \geq 4$ we have* $\dim P^n_{TV} = \frac{3n^2-3n-2}{2}$.

Proof. Because of Theorem 4.1 and the fact that the dimension of the Hamiltonian path polytope is equal to n^2-n-1 the statement is true. □

4.1.2 Facets of the P^n_{TV} Polytope

In this section we want to state some results about the facial structure of the P^n_{TV} polytope. Firstly we will present some facet classes and in the second part of this section we will present some lifting results with which we are able to prove that some facet classes of P^4_{TV} are facets in general.

Firstly we want to compute the complete description of P^n_{TV} for small n. We used the software package PORTA [CL09] which uses the Fourier-Motzkin elimination method for this purpose. Unfortunately we were only able to compute P^3_{TV} (which is trival and could not be used for general results) and P^4_{TV}.

The P^4_{TV} polytope can be described by 1280 inequalities. If we sort them by symmetries with the help of the HUHFA algorithm we get 48 facet classes. We used permutation symmetries which are permuting the nodes and the switching symmetries which change the orientation of all edges for the classification. The 48 classes can be found in Appendix A.1. If we only sort the facets using permutation symmetries we get 72 classes of inequalities.

For P^5_{TV} the computation time of PORTA is too long to obtain results. Therefore we have implemented a heuristic approach for computing facets of P^5_{TV} which uses the following technique. We only want to give a short overview of this topic, since a more detailed description can been found in [Lör14].

1. Compute a start facet f (or take a trivial facet $-x_{ij} \leq 0$).
2. Compute the set S of all integral points in P^5_{TV} which satisfy f with equality.
3. Calculate subset $T \subseteq S$ with $\dim(S) = \dim(T)$ and T is minimal.
4. Select an integral point $p \in T$ and an integral point $q \in P^5_{TV}$, $q \notin S$
5. Let $T' = \{T \setminus p\} \cup \{q\}$. Check for the set $\mathrm{Conv}(T')$ whether this is a facet or not with the help of Gaussian elimination.
6. In case it is go to Step 2 and apply procedure to new facet. Otherwise go to Step 4.

With the help of this heuristic it is possible to find 35 829 461 097 facets which can be sorted to 298 603 288 classes.

Of course all facet classes found in the case of $n = 4$ or $n = 5$ are not automatically facets for all n. So it is very important to find ways to generalize these inequality classes. This generalization process is called lifting and is defined as follows:

Definition 4.1. *An algorithm which takes a facet f of a polytope P and converts f to a facet of a polytope Q is called a lifting.*

By trivial lifting or zero-lifting we denote a special lifting of a facet f where the coefficients of the old variables are kept and the coefficients of all new variables are set to zero. In other words if a facet of P^k_{TV} is zero-liftable it is valid for all $n \geq k$. Since such facets are very preferable for the use in branch-and-cut algorithms, we prove a general zero-lifting result in the next section.

Additionally we also present some other general facet classes, which can be found in Section 4.1.4.

4.1.3 Zero-Lifting for the P^n_{TV} Polytope

Before we present a zero-lifting theorem we need to introduce some additional notation:

Definition 4.2. *Let f be a facet of P^n_{TV} defined by $a_w w + a_x x \leq a_0$. We will denote the zero-lifting of f for P^k_{TV}, $k > n$ by $f^{(k)}$ or in expanded form with $a^{(k)}_w w^{(k)} + a^{(k)}_x x^{(k)} \leq a^{(k)}_0$.*

To prove that a zero-lifting $f^{(k)}$ is a facet for all $k \geq n$ we have to assure two conditions: First $f^{(k)}$ must be feasible for all $k \geq n$ and second that it is a facet, which means that the dimension of $f^{(k)}$ is equal to $\dim P^k_{TV} - 1$.

Lemma 4.3. *Let f be a facet of P^n_{TV} defined by $a_w w + a_x x \leq a_0$ where all coefficients are non-negative. Then the zero-lifting of f is feasible for all P^k_{TV} with $k \geq n$.*

Proof. We will prove the statement by induction. For $k = n$ the statement is true by assumption. Now assume the statement is true for $k \geq n$ and it is not true for $k + 1$. Then consider a feasible integral point $p \in P^{k+1}_{TV}$ which violates $f^{(k+1)}$. It is clear that p can be constructed from a feasible point p' of P^k_{TV} by inserting node $k + 1$ somewhere in the path. Obviously this operation does not change the values of $w_{ij},\ \ i, j \leq k$. It can change the value of one $x_{ij},\ \ i, j \leq k$ from 1 to 0 if node $k + 1$ is inserted between i and j. But since all coefficients are non-negative, it follows then that $a^{(k)}_w w^{(k)} + a^{(k)}_x x^{(k)} \geq a^{(k+1)}_w w^{(k+1)} + a^{(k+1)}_x x^{(k+1)}$. That means p' has to violate $f^{(k)}$, which is a contradiction. □

We now want to develop a general lifting theorem. With this result we will be able to show for many facet classes that zero-lifting is possible.

Definition 4.3. *Let f be a valid inequality for P^n_{TV}. We call a node v a free node of f if for all $j \neq v$ the coefficients of the variables w_{vj}, w_{jv}, x_{jv}, x_{vj} in f are zero.*

Theorem 4.4. *Let f be a facet of P^n_{TV} defined by $a_w w + a_x x \leq a_0$ and let i be a free node of $f^{(n+1)}$ in P^{n+1}_{TV}. Then there exists a bijection I between the rows of MTV_n and MTV^i_{n+1} (i.e. between S_n and S_{n+1}) such that if and only if a characteristic vector (w_r, x_r) of P^n_{TV} satisfies f with equality, then $I(w_r, x_r)$ satisfies the zero-lifting $f^{(n+1)}$ with equality.*

Proof. We will prove this theorem in two steps. First let $i = n + 1$. Then we can define the following bijection between MTV_n and MTV^{n+1}_{n+1} via (S_n denotes the symmetric group on n nodes):

$$I:\ S_n \to S_{n+1},\ \ \pi'(k) := \begin{cases} n+1 & \text{if } k=1, \\ \pi(k-1) & \text{otherwise.} \end{cases}$$

We want to point out that $n+1$ is a free node of the zero-lifting of f and that this bijection does not affect the values of $w_{ij},\ x_{ij}$ with $i, j \neq n+1$. So it is clear that if a row (w_r, x_r) of MTV_n satisfies f with equality then the row $I(w_r, x_r)$ of MTV_{n+1}^{n+1} must satisfy $f^{(n+1)}$ with equality.

Now let $i < n+1$ and consider an arbitrary row (w_r, x_r) of MTV_{n+1}^{i}. We relabel node i with $n+1$ and vice versa. Since node i and node $n+1$ are free nodes of f it is still true that $a_w w_r + a_x x_r \leq a_0$. It is easy to see that after we have done this modification for all rows MTV_{n+1}^{i} is equal to MTV_{n+1}^{n+1} (maybe some rows have to be interchanged). So we can define the following bijection between MTV_n and MTV_{n+1}^{i} (p_i denotes the old position of i in the path):

$$I:\ S_n \to S_{n+1},\ \ \pi'(k) := \begin{cases} i & \text{if } k=1, \\ n+1 & \text{if } k=p_i+1, \\ \pi(k-1) & \text{otherwise.} \end{cases}$$

With the facts we mentioned in the first step, it should be clear that this bijection fulfills the desired criteria. □

Definition 4.4. *Let f be a valid inequality for P_{TV}^{n}. Then $MTVEq_n(f)$ denotes the matrix of all rows of MTV_n which fulfill f with equality. Analogously we define $MTVEq_n^i(f)$.*

For the following theorem we want to recall a definition from above: The column of MTV_n which is associated with variable w_{ij} is denoted by L_{ij} and analogously the column associated with x_{ij} is denoted by T_{ij}. We extend this definition also to $MTVEq_n(f)$ and $MTVEq_n^i(f)$.

Definition 4.5. *Let f be a facet defined by $a_w w + a_x x \leq a_0$. With $f^{=}$ we denote the equation $a_w w + a_x x = a_0$.*

We will in the following also use the term $f^{=}$ for the column equality implied by f in $MTVEq_n(f)$.

We known that

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} T_{ij} = (n-1) \cdot \mathbf{1} \tag{4.4}$$

is a valid column equation for $MTVEq_n(f)$. If f is a facet only one other linear dependence of columns in $MTVEq_n(f)$ exists. This is of course $f^=$. So showing that f is a facet is equal to showing that no valid column equation besides $f^=$, (4.4) and linear combinations of both exist. For a better understanding we will denote the set of $f^=$, (4.4) and all linear combinations of these two equations (for a given n) with $G(n)$.

Theorem 4.5. *Let f be a facet of P_{TV}^n with free node i. Let the zero-lifting of f also be feasible for P_{TV}^{n+1}. Then $f^{(n+1)}$ is also a facet of P_{TV}^{n+1}.*

Proof. Assume the statement is not true. This is equivalent with the fact that there exists a column equation C which is valid in $MTVEq_{n+1}(f^{(n+1)})$ but $C \notin G(n+1)$. Let C have w.l.o.g. the following form:

$$\begin{aligned} \alpha_{12}L_{12} + \cdots + \alpha_{1n+1}L_{1n+1} + \\ \vdots \\ +\alpha_{nn+1}L_{nn+1} + \\ \beta_{12}T_{12} + \cdots + \beta_{1n+1}T_{1n+1} + \\ \vdots \\ \beta_{n+11}T_{n+11} + \cdots + \beta_{n+1n}T_{n+1n} = c_0. \end{aligned}$$

We will prove the statement in 9 steps.

Step 1: For a column equation valid in $MTVEq_{n+1}(f^{(n+1)})$ we define the following minimal form: Subtract or add a multiple of $f^=$ so that $\sum_{k=1}^{n+1} \sum_{j=k+1}^{n+1} |\alpha_{kj}|$ is minimal. Of course this minimal form does not have to be unique. From now on we let w.l.o.g. C be in such a minimal form.

Step 2: We will introduce a transformation converting our column equation C of $MTVEq_{n+1}(f^{(n+1)})$ to a column equation of $MTVEq_n(f^{(n)})$.

Consider $MTVEq_{n+1}^{n+1}(f^{(n+1)})$. Since $n+1$ is the start node of the path it is clear that in this block all L_{jn+1} are equal to zero and therefore are not important for the column equation C. The same applies to all T_{jn+1} which are also equal to zero. Furthermore it follows that in $MTVEq_{n+1}^{n+1}(f^{(n+1)})$ the following degree column equations are valid:

$$\sum_{\substack{j=1\\ j\neq k}}^{n+1} T_{jk} = \mathbf{1} \ \ \forall k \in \{1,\dots,n\}.$$

So we can replace T_{n+1k} in C by $\mathbf{1} - \sum_{j=1,j\neq k}^{n} T_{jk}$. So the new β-coefficients look as follows:

$$\beta'_{jk} = \beta_{jk} - \beta_{n+1k},\ k,j \in \{1,\dots,n\},\ j \neq k.$$

Now we have an equation (denoted by C') which does not contain T_{jn+1}, T_{n+1j}, L_{jn+1}. With Theorem 4.4 (applied to C) we know that C' must be valid in $MTVEq_n(f)$.

Step 3 By assumption f is a facet of P_{TV}^{n}, which means that C' has to be in $G(n)$. If we consider the transformation in Step 2, we observe that all $\alpha_{kj}, 1 \leq k < j \leq n$ remain the same in C'. But since C is per definition in a minimal form defined in Step 1, we can conclude that in C all $\alpha_{kj} = 0$ for all $k, j \leq n$ or C' is not in $G(n)$.

Step 4 We will introduce a second transformation converting the column equation C of $MTVEq_{n+1}(f^{(n+1)})$ to a column equation of $MTVEq_n(f^{(n)})$. It is similar to the first one but this time we consider $MTVEq^{i}_{n+1}(f^{(n+1)})$. From Step 3 we know that all α_{kj} with $1 \leq k < j \leq n$ are zero. It is clear that L_{in+1} is equal to 1. Therefore this column is not important for the equation. The same applies to all T_{ji} for $j \in \{1,\dots,n+1\},\ j \neq i$ which are also equal to zero. Since i is the start node of the path it is clear that in $MTVEq^{i}_{n+1}(f^{(n+1)})$ the following degree column equations are valid:

$$\sum_{\substack{j=1\\ j\neq k}}^{n+1} T_{jk} = \mathbf{1} \ \ \forall k \in \{1,\dots i-1, i+1,\dots,n+1\}.$$

So we can replace T_{ik} in C by $\mathbf{1} - \sum_{j=1,j\neq k,i\neq j}^{n+1} T_{jk}$. So the new β-coefficients look as follows:

$$\beta''_{jk} = \beta_{jk} - \beta_{ik},\ j,k \in \{1,\dots,n+1\},\ j \neq i,\ j \neq k,\ i \neq k.$$

Now we have an equation (denoted by C'') which does not contain the variables T_{ji}, T_{ij}, L_{ji} and L_{ij}. If we relabel $n+1$ with i, we know with Theorem 4.4 (applied to C) that C'' must be valid in $MTVEq_n(f)$.

Step 5 With a similar argument as in Step 3 we can conclude that also all α_{jn+1} with $1 \leq j \leq n$ must be equal to zero.

Step 6 From Steps 3 and 5 we can conclude that all α_{jk} must be zero. Therefore C' and C'' must be each a multiple of the degree equation (4.4). But that means (because of the transformation in Step 2) that for $1 \leq j \neq k \leq n$ the differences $\beta_{jk} - \beta_{n+1k}$ must be all equal to a fixed value B'. But then it follows that $\beta_{jk} = \beta_{\circ k}$ for all $1 \leq j, k \leq n$ and $j \neq k$. Analogously we can conclude from the transformation presented in Step 4 that the differences $\beta_{jk} - \beta_{ik}$ for $k, j \in \{1, \ldots, i-1, i+1, \ldots, n+1\}$ must be all equal to a fixed value B''.

Step 7 Let $NTVEq^{k}_{n+1}(f^{(n+1)})$ denote that block of rows of $MTVEq_{n+1}(f^{(n+1)})$ which correspond to paths which end with node k. We will introduce a third transformation. This time we consider $NTVEq^{n+1}_{n+1}(f^{(n+1)})$. From Steps 3 and 5 we know that all α_{kj} are zero. The same applies to all T_{n+1k} for $k \in \{1, \ldots, n\}$ which are also equal to zero. Since $n+1$ is the end node of the path, it is clear that in $NTVEq^{n+1}_{n+1}(f^{(n+1)})$ the following degree column equations are valid:

$$\sum_{\substack{j=1 \\ j \neq k}}^{n+1} T_{kj} = \mathbf{1} \;\; \forall k \in \{1, \ldots, n\}.$$

So we can replace T_{kn+1} in the equation through $\mathbf{1} - \sum_{j=1, j\neq k}^{n} T_{kj}$. So the new β-coefficients look as follows:

$$\beta'''_{kj} = \beta_{kj} - \beta_{kn+1}, \; k, j \in \{1, \ldots, n\}, \; j \neq k.$$

Now we have an equation (denoted by C''') which does not contain T_{jn+1}, T_{n+1j} and L_{jn+1}. With an argument analogous to Theorem 4.4 we can conclude that C''' must also be valid in $MTVEq_n(f)$.

Step 8 With a similar argumentation as in Step 6 we can conclude together with Step 7 that $\beta_{kj} = \beta_{k\circ}$ for all $1 \leq j, k \leq n$. Together with Step 6 we can conclude that $\beta_{kj} = \beta, \; \forall j, k \in \{1, \ldots, n\}$.

Step 9 We know from Step 6 that $B'' = \beta_{n+1k} - \beta_{ik} = \beta_{jk} - \beta_{ik}$ for some $j \in \{1, \ldots, n\}$ But since $\beta_{jk} = \beta$ for $j, k \in \{1, \ldots, n\}$ this means $\beta_{n+1k} = \beta$ for $k \in \{1, \ldots, n\}$. We can define a fourth transformation analogous to the one in Step 7 but between $NTVEq^{i}_{n+1}(f^{(n+1)})$ and $MTVEq_n(f^{(n)})$. With the help of this transformation we can conclude with the same argumentation as above that $\beta_{kn+1} = \beta$ for $k \in \{1, \ldots, n\}$. But that means that all β_{jk} are equal to β and C is equal to the degree equation. This is a contradiction to the assumption. So we can conclude that no such column equation can exist.

□

Theorem 4.6. *Let f be a facet of the P^n_{TV} with at least one free node i and the facet defining inequality is valid for P^k_{TV} with $k \geq n$. Then $f^{(k)}$ is a facet of P^k_{TV} with $k \geq n$.*

Name	Facet
C2	$w_{il} + w_{kl} + w_{lj} + x_{ji} + x_{jk} + x_{jl} + x_{li} + x_{lk} \leq 3$
	$w_{ji} + w_{jk} + w_{lj} + x_{ij} + x_{il} + x_{jl} + x_{kj} + x_{kl} \leq 3$
C7	$w_{ij} + 2w_{jk} + 2w_{kl} + w_{li} + w_{lj} + x_{il} + x_{ji} + x_{jl} + x_{kj} + x_{lk} \leq 5$
C11	$-x_{ki} \leq 0$
C13	$w_{il} + w_{ji} + w_{jk} + w_{kl} + w_{lj} + x_{ij} + x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 4$
C14	$w_{ij} + 2w_{jk} + w_{ki} + w_{kl} + w_{lj} + x_{ji} + x_{jl} + x_{kj} \leq 4$
	$w_{il} + w_{ji} + w_{jk} + w_{kl} + 2w_{lj} + x_{ij} + x_{jl} + x_{kj} \leq 4$
C20	$w_{ij} + w_{il} + 2w_{jk} + w_{ki} + w_{lj} + x_{ji} + x_{jl} + x_{ki} + 2x_{kj} + x_{kl} + x_{li} \leq 5$
	$w_{il} + 2w_{jk} + w_{ki} + w_{kl} + w_{lj} + x_{ij} + x_{ik} + 2x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 5$
C25	$w_{il} + w_{jk} + w_{ki} + w_{lj} + x_{jl} + x_{kj} + x_{kl} \leq 3$
C29	$w_{jk} + w_{kl} + w_{lj} + x_{kj} \leq 2$
C30	$2w_{jk} + 2w_{kl} + 2w_{lj} + x_{jl} + x_{kj} + x_{lk} \leq 4$
C39	$w_{ij} + 2w_{jk} + w_{kl} + x_{ji} + x_{ki} + 2x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 4$
C41	$w_{jk} + w_{kl} + x_{kj} + x_{lj} + x_{lk} \leq 2$
C46	$w_{jk} + x_{kj} \leq 1$
C47	$w_{il} + w_{kj} + w_{lk} + x_{ji} + x_{jk} + x_{jl} + x_{kl} + x_{li} \leq 3$
	$w_{ij} + w_{jk} + w_{li} + x_{il} + x_{ji} + x_{jl} + x_{kj} + x_{kl} \leq 3$

Table 4.3: Facet-defining inequality classes of P_{TV}^4 for which zero-lifting is possible ($i, j, k, l \in \{1, 2, 3, 4\}$ and pairwise different)

Proof. We will show this theorem by induction. For $k = n$ the statement is true by assumption. Now let the statement be true for $k \geq n$. Since node i is still a free node of $f^{(k)}$ we can apply Theorem 4.5 and therefore the statement must be also true for $f^{(k+1)}$. □

Corollary 4.7. *The inequality classes listed in Table 4.3 are facet-defining for P_{TV}^n for all n greater than 3.*

Proof. It is obvious that the inequalities of class 11 are feasible for all n in P_{TV}^n. All other inequalities are feasible for all n due to Lemma 4.3 since all coefficients are non-negative. For $n \in \{4, 5\}$ the statement can be checked by hand or with a computer algebra system. Since every inequality has at least one free node (5 for example) in P_{TV}^5 we can then apply Theorem 4.6 to prove the statement for $n \geq 6$. □

4.1.4 General Lifting procedures

Some classes of P^4_{TV} can be generalized to facet classes valid for P^n_{TV}, $n > 4$. In more detail, we are able to present lifting procedures for classes 37 and 45 (from the table in Appendix A.1). We first present the proof for class 45. The concept for the proof of the following theorem is to refine the main idea of the Zero-Lifting Theorem 4.5.

Theorem 4.8. *The inequalities* $\sum_{\substack{i=1,\\ i\neq k}}^{n} x_{ik} \leq 1$ *and* $\sum_{\substack{i=1,\\ i\neq k}}^{n} x_{ki} \leq 1$ *are facet-defining for* P^n_{TV} *for all* $n \geq 4$ *and for all* $k \in V$*.*

Proof. Consider the inequality

$$\sum_{\substack{i=1,\\ i\neq k}}^{n} x_{ik} \leq 1$$

for some $k \in V$. In the following we will denote that inequality in dimension n with $f^{<n>}$. This inequality is fulfilled with equality by all TVP paths where k is not at the first position.

Now let the statement be true for n. We will show that it is then also true for $n+1$.

First consider the case $k \in \{1, \ldots, n\}$. Let p be a path through the targets $\{1, \ldots, n\}$ which fulfills $f^{<n>}$ with equality. It is clear that a path p' which visits $n+1$ and then follows p fulfills $f^{<n+1>}$ with equality. This remains also true if we interchange some node $i \neq k$ and $n+1$. Also when we move $n+1$ at the end of p' this remains true. With this it is for $i \neq k$ possible to extend every vector of $MTVEq_n(f^{<n>})$ to $MTVEq^i_{n+1}(f^{<n+1>})$ (or $NTVEq^i_{n+1}(f^{<n+1>})$ respectively). So we can define the same bijection between $MTVEq_n(f^{<n>})$ and $MTVEq^i_{n+1}(f^{<n+1>})$ (and $NTVEq^i_{n+1}(f^{<n+1>})$) as in Theorem 4.4. But this time we do not need a free node for this purpose, but we can rely on the properties of the vectors which fulfill $f^{<n>}$ with equality.

With the help of these bijections it is then possible to prove that $f^{<n+1>}$ is a facet in the same way as we proved Theorem 4.5. The requirement of a free node in Theorem 4.5 is only needed for the definition of the bijections.

Since the statement is true for P^4_{TV} the theorem can be proven by induction. That the statement is also true for $k = n+1$ follows because of the symmetry of the polytope. For the inequality $\sum_{\substack{i=1\\ i\neq k}}^{n} x_{ki} \leq 1$ the proof is analogous. □

Remark 4.9. *In general a lifting $f^{<n+1>}$ of a facet f of P^n_{TV} is a facet when every vector of $MTVEq_n(f^{<n>})$ can be extended to a vector of $MTVEq^i_{n+1}(f^{<n+1>})$ and to a vector of $NTVEq^i_{n+1}(f^{<n+1>})$ for at least two different nodes i. Then we can define the same bijection between $MTVEq_n(f^{<n>})$ and $MTVEq^i_{n+1}(f^{<n+1>})$ as in Theorem 4.4 and the idea of the proof of Theorem 4.5 can be applied to prove that $f^{<n+1>}$ is a facet as well.*

Next we present a lifting for class 37 where we need to extend the idea of Corollary 4.9.

Theorem 4.10. *The inequality classes $w_{kl} - \sum\limits_{\substack{j=1,\\ j\neq k}}^{n} x_{kj} \leq 0$ and $w_{lk} - \sum\limits_{\substack{j=1,\\ j\neq k}}^{n} x_{jk} \leq 0$ are facet-defining for P^n_{TV} for all $n \geq 4$ and for all $k, l \in V$ with $k \neq l$.*

Proof. Consider w. l. o. g. the following inequality of this class:

$$w_{n-1n} - \sum_{\substack{j=1,\\ j\neq n-1}}^{n} x_{n-1j} \leq 0. \tag{4.5}$$

We again denote this inequality in dimension n by $f^{<n>}$. It is easy to see that $f^{<k>}$ is feasible for P^k_{TV} for all $k > n$.

Let the statement true for n: Assume there exists some column equation C in $MTVEq_{n+1}(f^{<n+1>})$ which is not linearly dependent on (4.4) and $f^{<n+1>=}$. Let C have w.l.o.g. the following form:

$$\begin{aligned}
\alpha_{12}L_{12} + \cdots + \alpha_{1n+1}L_{1n+1}+& \\
\vdots& \\
+\alpha_{nn+1}L_{nn+1}+& \\
\beta_{12}T_{12} + \cdots + \beta_{1n+1}T_{1n+1}+& \\
\vdots& \\
\beta_{n+11}T_{n+11} + \cdots + \beta_{n+1n}T_{n+1n} =\ & c_0.
\end{aligned}$$

Consider a TVP path p that fulfills (4.5) with equality. Then n is visited after $n-1$ (and node $n-1$ is not the last node in the path) or $n-1$ is the last node in the path. It is clear that the path p' which starts in $n+1$ and then follows p fulfill $f^{<n+1>}$ with equality. Again this remains true if we exchange $n+1$ with any other node besides n and $n-1$ (Unfortunately it is not possible to move $n+1$ at the end of the path. So we could not just apply Remark 4.9). So every vector of $MTVEq_n(f^{<n>})$ can be extended to a vector of $MTVEq^i_{n+1}(f^{<n+1>})$ for all $i \neq n-1, n$. That means we can apply the same argumentation as in the proof of Theorem 4.5 until Step 6. From this we can conclude that all $\alpha_{i,j}$ are equal to zero and that:

$$\beta_{jq} = \beta_{\circ q}$$

for all $j \in 1, \ldots, n+1$ and $q \in 1, \ldots, n$. To show that $\beta_{jn+1} = \beta_{\circ n+1}$ we need to define a third transformation from $MTVEq^{k}_{n+1}(f^{<n+1>})$ to $MTVEq_{n}(f^{<n>})$ for some node $k \in V \setminus \{i, n-1, n, n+1\}$. After this C simplifies to

$$\beta_{\circ 1} \sum_{j=2}^{n+1} T_{j1} + \beta_{\circ 2} \sum_{\substack{j=1\\ j\neq 2}}^{n+1} T_{j2} + \cdots + \beta_{\circ n+1} \sum_{j=1}^{n} T_{jn+1} = c_0.$$

It is obvious that $\sum_{j=1,\, j\neq q}^{n+1} T_{jq} = 0$ if q is at first position and equal to 1 if not. For every $q \in \{1, \ldots, n+1\}$ we find some row r of MTV^{q}_{n+1} that fulfills $f^{<n+1>}$ with equality. Since each such r must fulfill C with equality the following equation system must be valid:

$$\begin{aligned}
\beta_{\circ 2} + \beta_{\circ 3} + \cdots + \beta_{\circ n+1} &= c_0,\\
\beta_{\circ 1} + \beta_{\circ 3} + \cdots + \beta_{\circ n+1} &= c_0,\\
&\vdots\\
\beta_{\circ 1} + \beta_{\circ 2} + \cdots + \beta_{\circ n} &= c_0.
\end{aligned}$$

This equation system has an unique solution. And that is $\beta_{\circ q} = \frac{c_0}{n}$ for all $q \in \{1, \ldots, n+1\}$. So C is equal to the degree equation (4.4), which is a contradiction.

We can show in the same way that:

$$w_{lk} - \sum_{\substack{j=1,\\ j\neq k}}^{n} x_{jk} \leq 0 \tag{4.6}$$

is facet-defining for P^{n}_{TV} for all $n \geq 4$ and for all $k, l \in V$ with $k \neq l$. □

The lifting of class 37 is shown in Figure 4.1. Hereby (and also in Figure 4.2 and 4.3) a blue arrow from i to j denotes $+w_{ij}$ and a red arrow denotes $+x_{ij}$. A dash arrow denotes a coefficient of -1 while the thick dashed arrows in Figure 4.2 denote the coefficient $-(n-2)$.

We also strongly suspect that the inequalities of class 38 (shown in Figure 4.2) can be lifted to general facet classes of the form:

- $\sum_{\substack{i=1\\i\neq k}}^{n} w_{ki} - (n-2)\sum_{\substack{i=1\\i\neq k}}^{n} x_{ki} + \sum_{\substack{i=1\\i\neq k}}^{n} x_{ik} \leq 1$

- $\sum_{\substack{i=1\\i\neq k}}^{n} w_{ik} - (n-2)\sum_{\substack{i=1\\i\neq k}}^{n} x_{ik} + \sum_{\substack{i=1\\i\neq k}}^{n} x_{ki} \leq 1.$

For every fixed order of all nodes also class 47 can be generalized (shown in Figure 4.3). Here we used the order $1, 2 \ldots, n$:

- $\sum_{i=1}^{n-1} w_{i+1i} + \sum_{i=1}^{n-1} x_{in} + \sum_{i=1}^{n-2} x_{ii+1} \leq n-1$

- $\sum_{i=1}^{n-1} w_{ii+1} + \sum_{i=1}^{n-1} x_{ni} + \sum_{i=1}^{n-2} x_{i+1i} \leq n-1.$

Unfortunately in case of these two inequality classes we are not able to find two different nodes so that we can apply Remark 4.9. We have checked the statement by hand for P_{TV}^n with $n \leq 8$. But if these inequalities are facet-defining in general remains an open question.

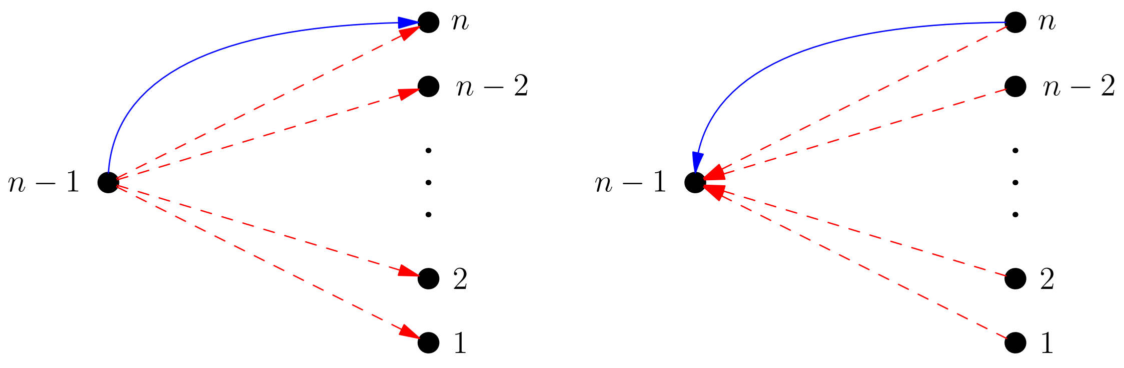
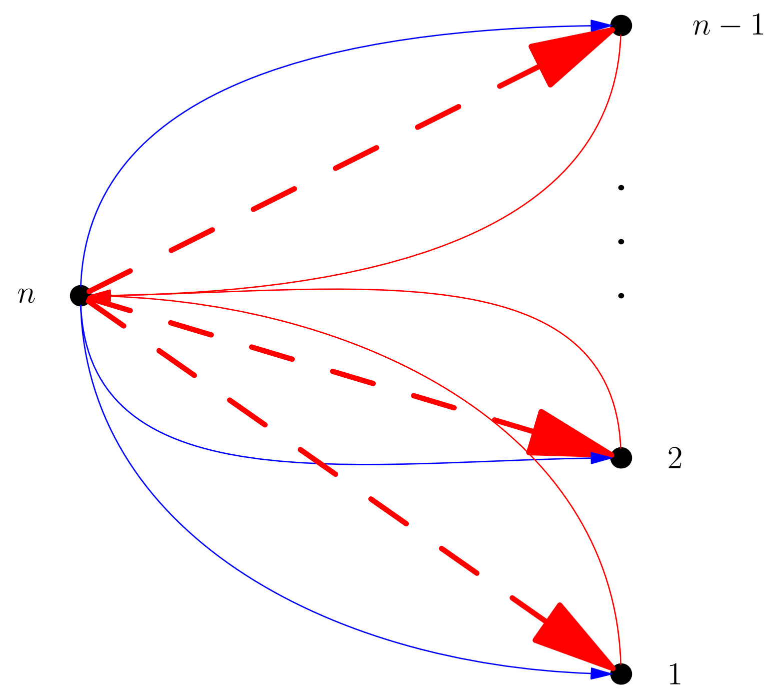


Figure 4.1: Generalized lifting of facet class 37 of P_{TV}^4



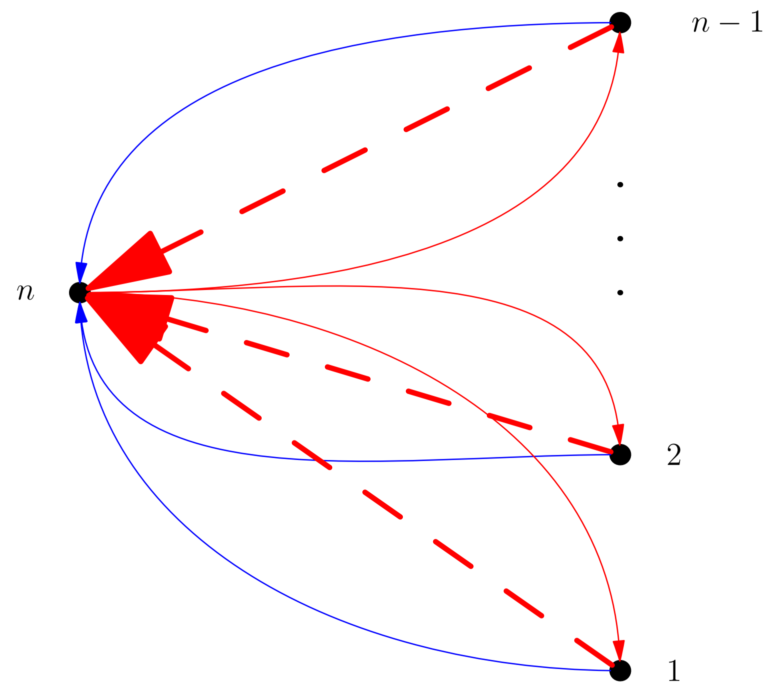


Figure 4.2: Generalized lifting of facet class 38 of P^4_{TV}

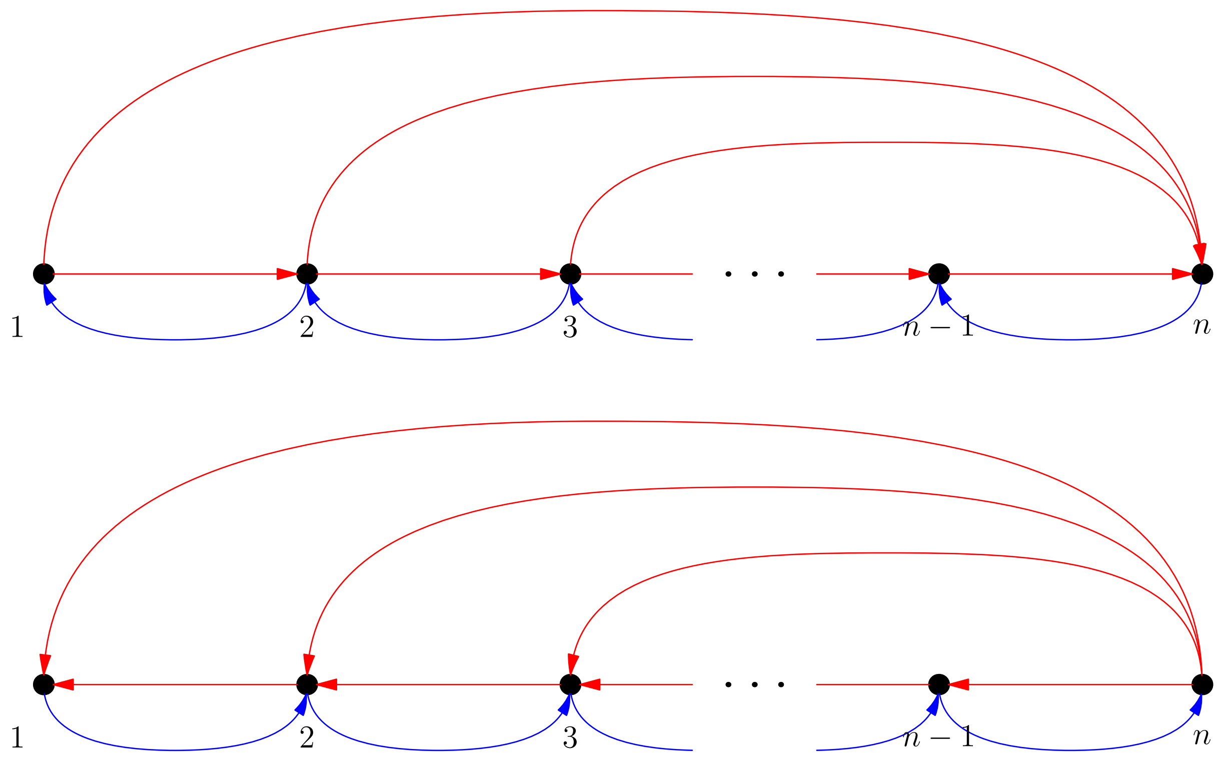


Figure 4.3: Generalized lifting of facet class 47 of P^4_{TV}

4.2 Polyhedral Results for the TVP_E Model

As we have seen at the beginning of this chapter in case of the TVP_E model the gap between the solution of the LP relaxation and the optimal solution is in the average very small. Because of that we want to present in the following a closer examination of the structure of P^n_{ETV} and the connection between P^n_{TV} and P^n_{ETV}.

4.2.1 Dimension

We start with an examination of the dimension of P^n_{ETV}. Because of the additional variables there exist way more equations than in the TVP_{HP} model. To determine the dimension of the polytope we have to find a minimal equation system.

We can describe the set of valid equations by the following three basic types. The first class of equations is already contained in the IP model:

$$w_{ij} + w_{jik} + w_{jki} + w_{kji} = 1,\ 1 \le i,j,k \le n,\ i < j,\ i \ne k,\ j \ne k. \tag{4.7}$$

Also the equation which sums up all x-variables is already contained in the model:

$$\sum_{i=1}^{n} \sum_{\substack{j=1\\ j\neq i}}^{n} x_{ij} = n-1. \tag{4.8}$$

Additionally to these two classes there exists a family of equations which is the generalization of the tournament equations. In other words we have equations which describe the fact that three nodes i, j and k must have some fixed order:

$$w_{ijk} + w_{ikj} + w_{jki} + w_{jik} + w_{kij} + w_{kji} = 1,\ 1 \leq i < j < k \leq n. \tag{4.9}$$

With this result we are able to prove the dimension of the polytope. We will do this in two steps: First we will show that the set of equations (4.7)–(4.9) is minimal. Second we show that apart from this set no other linearly independent equation is valid for the polytope.

Lemma 4.11. *The equations (4.7)–(4.9) form a system of linearly independent equations.*

Proof. It is clear that the equations of family (4.9) are linearly independent of each other since each w_{ijk} is only part of one equation. Since all of these equations do contain a w_{ijk} with $k > j > i$ they also cannot be linearly dependent on a subset of equations of (4.7). Since (4.8) is the only equation which contains x-variables it is clear that it is not linearly dependent on a set of other equations. It remains to show that all inequalities of (4.7) are linearly independent of each other. For $i < j < k$ there exist exactly three equations of (4.7) which contain three indexed w-variables with the indices i, j, k:

$$w_{ij} + w_{jik} + w_{jki} + w_{kji} = 1, \tag{4.10}$$

$$w_{jk} + w_{kji} + w_{kij} + w_{ikj} = 1, \tag{4.11}$$

$$w_{ik} + w_{kij} + w_{kji} + w_{jki} = 1. \tag{4.12}$$

The variable w_{jik} (w_{ikj}) is only contained in the first (second) equation. So these two equations cannot be linearly depend on a subset of equations of (4.7). For (4.12) we can state that this equation contains w_{jki} which is contained only in one other equation of (4.7). But this is (4.10) which contains w_{jik}. So with the same argument as in the first two cases there cannot exist a linear dependency. □

Theorem 4.12. *The dimension of P^n_{ETV} is equal to $\frac{n(n-1)(2n+5)}{6} - 1$ for $n \geq 4$.*

Proof. The IP model contains $\frac{n(n-1)}{2}$ LOP-variables, $n(n-1)$ HP-variables and $n(n-1)(n-2)$ w_{ijk}-variables.

Due to Lemma 4.11 we are able to compute a bound for the number of valid linearly independent equations. In more detail we have $\frac{n(n-1)(n-2)}{2}$ equations of type (4.7), one equation of type (4.8) and $\frac{n(n-1)(n-2)}{6}$ equations of type (4.9).

Therefore it follows for $\dim P^n_{ETV}$:

$$\begin{aligned}\dim P^n_{ETV} &\leq \frac{n(n-1)}{2} + n(n-1) + n(n-1)(n-2)\\ &\quad - \frac{n(n-1)(n-2)}{2} - 1 - \frac{n(n-1)(n-2)}{6}\\ &= \frac{2n^3 - 3n^2 + n}{2} - \frac{4n^3 - 12n^2 + 8n}{6} - 1\\ &= \frac{2n^3 + 3n^2 - 5n}{6} - 1\\ &= \frac{n(n-1)(2n+5)}{6} - 1.\end{aligned}$$

The next step is to prove that no equation exists which is valid for the polytope and which is linearly independent of the equations listed in Lemma 4.11. We will prove this statement with similar methods as we have proved Theorem 4.1.

Given $i < j < k \in V$, three-indexed w-variables which have all three of these nodes as index are only contained in (4.10)–(4.12) and in

$$w_{ijk} + w_{ikj} + w_{jki} + w_{jik} + w_{kij} + w_{kji} = 1, \; 1 \leq i < j < k \leq n.$$

It is obvious that we can use these four equations to eliminate the variables w_{ikj}, w_{jki}, w_{jik}, w_{kij} from the IP model. The conjecture is now that in this reduced IP model no other equation besides (4.8) is valid.

With the help of a computer algebra system we can check that the statement is true for the case of $n = 4$. Now let the statement be true for P^n_{ETV} with $n \geq 4$. We apply the idea of the proof of Theorem 4.1. Define L_{ij}, T_{ij}, MTV_n and MTV^i_n similar as in Section 4.1.1. Additionally we denote by E_{ijk} the column in MTV_n associated to w_{ijk}.

Now consider MTV_{n+1}. It follows from Theorem 4.1 that there cannot exist an additional column equation in MTV_{n+1} which only contains L_{ij} and T_{ij}. Next assume there exists a column equation which contains at least one column $E_{k_1k_2k_3}$. Denote this equation by C.

Now we consider a block MTV^i_{n+1} where $E_{k_1k_2k_3} \neq \mathbf{0}$ or $E_{k_1k_2k_3} \neq \mathbf{1}$. We can observe that:

- The column vectors E_{jli} and E_{jil}, $j, l \in \{1, \ldots, i-1, i+1, \ldots, n+1\}$, $j \neq l$ are zero vectors since i is the first node of the tour.

- For the column vectors E_{ijl}, $j, l \in \{1, \ldots, i-1, i+1, \ldots, n+1\}$, $j \neq l$ it is true that $E_{ijl} = L_{jl}$. So we can unite these columns.

We can then apply the arguments of the proof of Theorem 4.1. Also the relabeling can be done in the same way. In consequence we get a contradiction since the new system is equal to MTV_n and does contain an equation which is not linearly dependent on the equations listed in Lemma 4.11. □

Next we want to have a closer look on the facial-structure of the TVP_E. Calculations with PORTA yield that the description of P^4_{ETV} contains less inequalities than the description of P^4_{TV}. In detail the polytope can be described by only 144 facets and 20 equations. Using HUHFA we can sort the inequalities in 9 classes which are listed in Appendix A.2.

4.2.2 Projection of the P^n_{ETV} Polytope to the P^n_{TV} Polytope

Since the LP bounds of the TVP_E model are so tight we want to have a closer look at the facets of the P^n_{TV} polytope which are implied by the extended formulation. For this purpose we try to project P^n_{ETV} onto P^n_{TV}. This can be done by eliminating all w_{ijk}-variables from the extended model. We use the so-called Fourier-Motzkin elimination method (see [Zie95] for details) for this purpose.

Fortunately, in this case it is possible to eliminate the w_{ijk}, w_{ikj}, w_{jik}, w_{jki}, w_{kij}, w_{kji} separately for each $i, j, k \in V$. So we have to eliminate a system of three equations (3.25), six inequalities (3.26) and six inequalities (3.30) from the model. This leads to the result that the following four classes of equations are implied by the extended formulation:

$$2w_{jk} + 2w_{kl} + 2w_{lj} + x_{jl} + x_{kj} + x_{lk} \leq 4, \tag{4.13}$$

$$w_{jk} + w_{kl} + x_{kj} + x_{lj} + x_{lk} \leq 2, \tag{4.14}$$

$$w_{jk} + w_{kl} + w_{lj} + x_{kj} \leq 2, \tag{4.15}$$

$$x_{ij} - w_{ij} \leq 0. \tag{4.16}$$

We observe that these classes are all facet-defining for P^k_{TV} with $k \geq 4$. Moreover they are the only inequalities on three or less nodes describing P^4_{TV} and P^5_{TV}. It remains as an open question whether this is also true for $n > 5$. Another related open question which arises in this context is: Does a generalized model which contains the variables $w_{i_1 i_2 \ldots i_k}$ imply all facets on k or less nodes?

At last we would like to make a comment on the practical aspect of this result. Since (4.15) and (4.16) are already contained in the model we can only use (4.13) and (4.14) as cutting planes in a branch-and-cut approach. We will see in Chapter 7 that both classes do work quite good for this purpose.

Chapter 5

Approaches for Solving the TVP

In this chapter we want to briefly introduce three approaches for solving the target visitation problem. We will give a short overview of the different topics and encourage the reader to consult the given references for more detailed information.

This chapter will also contain only the theoretical background of the methods. Results of practical computations can be found in Chapter 7.

5.1 Lagrangean Decomposition

In order to explain the method of Lagrangean decomposition we have to introduce the Lagrangean relaxation of an integer program. Hereby we are follow mainly [KV12].

This is a relaxation tailored for cases where we have an integer program which becomes substantially easier to solve when some constraints are removed from the program. So the Lagrangean relaxation is a method to get rid of troublesome constraints. In more detail instead of explicitly enforcing some constraints we modify the objective function in the way that we penalize infeasible solutions. Formally, instead of optimizing the program:

$$\begin{aligned}
&\max cx \qquad (5.1)\\
&\text{s.t.}\\
&Ax \leq b,\\
&A'x \leq b',\\
&x \in \mathbb{Z}^n
\end{aligned}$$

we are considering the following program for some vector $\lambda \geq 0$:

$$\begin{aligned} &\max cx + \lambda(b' - A'x) \\ &\text{s.t.} \\ &Ax \leq b, \\ &x \in \mathbb{Z}^n. \end{aligned} \tag{5.2}$$

It is clear that for each $\lambda \geq 0$ the solution of (5.2) is an upper bound for the solution of (5.1). So in order, to make sense the system (5.2) should be easier to solve than (5.1). We denote (5.2) as a Lagrangean relaxation of (5.1) and the components of λ are called the Lagrange multipliers.

Since we are interested in a good upper bound we have to determine the best λ. We define $LR(\lambda)$ as the optimal solution of (5.2) which is a convex function. The problem is now to minimize $LR(\lambda)$ which is also called the Lagrangian dual of (5.1). One method for minimizing LR is the so-called sub-gradient method which is described by Algorithm 4.

Algorithm 4: Sub-gradient method

```
1 Start with random $\lambda^{(0)} \geq 0$;
2 $i = 0$;
3 while $i < Max$ do
4     $x^{(i)} = \max_{x \in \mathbb{Z}^n} \{cx + \lambda^{(i)}(b' - A'x) \mid Ax \leq b\}$;
5     $\lambda^{(i+1)} = \max\{0, \lambda^{(i)} - t_i(b' - A'x^{(i)})\}$ for some $t_i > 0$;
```

We would like to make a few comments on this approach. First it has been shown that if

$$\lim_{i \to \infty} t_i = 0 \text{ and } \sum_{i=0}^{\infty} t_i = \infty,$$

then

$$\lim_{i \to \infty} LR(\lambda^{(i)}) = \min\{LR(\lambda) \mid \lambda \geq 0\}.$$

Secondly it can be shown that the minimum is always attained unless

$$\{x \in \mathbb{Z}^n \mid Ax \leq b, A'x \leq b'\} = \emptyset.$$

Concerning the question of the quality of the solution we would like to state the following theorem:

Theorem 5.1. *Let $c \in \mathbb{R}^n, A' \in \mathbb{R}^{m\times n}$ and $b' \in \mathbb{R}^m$. Let $Q \subseteq \mathbb{R}^n$ such that $Conv(Q)$ is a polyhedron. Suppose that $max\{cx \mid A'x \leq b', x \in conv(Q)\}$ has an optimal solution. Let $LR(\lambda) := max\{cx + \lambda(b' - A'x) \mid x \in Q\}$. Then $inf\{LR(\lambda) \mid \lambda \geq 0\}$ (the optimal value of the Lagrangean dual) is attained by some λ and the minimum is equal to* $\max\{cx \mid A'x \leq b', x \in conv(Q)\}$.

The proof of this theorem can be found for example in [KV12]. Also the following remark can be found there:

If we have an integer program $\max\{cx \mid A'x \leq b', Ax \leq b, x \text{ is integral}\}$ where $\{x \mid Ax \leq b\}$ is integral, then the Lagrangean dual (when relaxing $A'x \leq b'$ as above) yields the same upper bound as the standard LP relaxation

$$\max\{cx \mid A'x \leq b',\ Ax \leq b,\ x \in \mathbb{R}^n\}. \tag{5.3}$$

If $\{x \mid Ax \leq b\}$ is not integral, the upper bound can be stronger (but can be difficult to compute).

Lagrangean decomposition is a technique based on Lagrangean relaxation. In detail, we relax a set of constraints so that the integer program is decomposed in two or more independent subprograms, which are then solved separately from each other.

As one can see, the TVP_{HP} model is very suitable to be used with this method since it is naturally constructed by merging two independent subproblems with only one connecting constraint. So the idea is to relax Constraint (3.8) and decompose the problem. Afterwards we can use the existing solvers for the LOP and TSP part. In more detail, the decomposition works as follows: (X denotes the set of characteristic vectors of feasible TSP tours and W the set of characteristic vectors of feasible orderings on a graph $G = (V, E)$, $\lambda \geq 0$)

$$\begin{aligned}
\mathrm{LR}_{\mathrm{TVP}}(\lambda) &= \max_{\substack{w\in W\\ x\in X}}(Pw - \lambda(x - w) - Dx)\\
&= \max_{\substack{w\in W\\ x\in X}}(Pw + \lambda w - \lambda x - Dx)\\
&= \max_{\substack{w\in W\\ x\in X}}((P+\lambda)w - (D+\lambda)x)\\
&= \max_{w\in W}(Pw + \lambda w) - \min_{x\in X}(Dx + \lambda x)\\
&= LOP(P+\lambda) - HP(D+\lambda).
\end{aligned}$$

We use this decomposition for several practical computations. For solving the Lagrangean relaxation we implement the sub-gradient method as well as a proximal bundle method and a conic bundle method. More details on the practical implementation can be found in Chapter 7.

For more detailed information on this topic consult [Fie12].

5.2 Dynamic Programming

Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems. The key idea is to use already computed partial solutions for the computation of unexplored solutions. So if we deal with NP-hard problems, this means in most cases that the method needs an exponentially growing amount of space since all partial solution must be stored. On the other hand the method does not need additional branching-methods or other special knowledge of the problem structure.

In case of the TVP the method works recursively in the following way. For each node in the search tree we store the set of already visited nodes, the node which has been visited last and the value of the best path which visits the remaining nodes if already computed. We explicitly do not store the order in which the targets have been visited so far. We can do this because we just need to know the end node of a partial TVP path to compute an optimal path through the remaining targets. So in consequence we are able to melt nodes of the search tree if the list of visited targets as well as the last node visited are the same.

The practical implementation basically consists of the recursive procedure which is shown in Algorithm 5. To start the computation we use Dynamic-Programming($\emptyset, 0$) where 0 is a dummy node which has distance zero to all other nodes.

Algorithm 5: Dynamic-Programming

```
Data:
Visited_Nodes: All targets which have been visited so far in the TVP path
(denoted by $V_n$)
Last_Node: The node which has been visited last in the partial TVP path
1  if optimal solution for Dynamic-Programing(Visited_Nodes, Last_Node) has
   not been computed yet then
2      Optimum = $-\infty$;
3      for all $s \in \{V \setminus V_n\}$ do
4          sum = $\sum_{i \in V \setminus V_n,\, i \neq s} p_{s,i} - d_{\text{Last\_Node},s}$;
5          sum = sum + Dynamic-Programming($V_n \cup s, s$);
6          if sum>Optimum then Optimum=sum;
7      Store Optimum;
8      Return: Optimum;
9  else
10     Return: Stored Optimum;
```

5.3 Branch-and-Bound

In this subsection we present a branch-and-bound approach for the TVP which does not use linear programming. The basic idea is to successively construct a feasible TVP path from the first target visited till the last one. This procedure uses the advantage that we can compute a lot of met preferences for a given partial solution (even for a short one). For example, in a partial solution where half of the targets have been visited, more than 75 percent of the preferences are already known.

Since we do not want to use linear programming, we have to find other methods to compute upper bounds of a given partial solution. For this purpose we use ideas which have been used to estimate partial LOP or TVP solutions. So we will compute a separate bound for each subproblem. In detail, we use the following four ways of estimation (N_V denotes the set of targets which have not been visited so far, l the node which has been visited as last node in the partial solution):

1.) Since every partial solution contains the beginning of the TVP path, every target which has not been visited so far must have an incoming edge. So we can sum up the minimum distance costs for entering each node and get a lower bound for the necessary cost of connecting the remaining targets. Formally this idea can be expressed by:

$$\text{Bound}_{\text{TSP}} = \sum_{i \in N_V} \min_{\substack{j \in N_V \cup\, l \\ j \neq i}} d_{ji}.$$

2.) It is clear that for all targets $i, j \in N_V$ it must be true that i is visited before j or j before i. So we can sum up:

$$\text{Bound}_{\text{LOP}} = \sum_{\substack{i,j \in N_V \\ i \neq j}} \max\{w_{ij}, w_{ji}\}.$$

3.) We use the so-called Helmstädter condition [Hel64]. Let $W \subset V$ be a subset with k elements and $P = (v_1, v_2, \ldots, v_k)$ be a path which visits all targets of W. Then the Helmstädter condition is fulfilled if for every path P' on W which begins in v_1 and ends in v_k it is true that the objective value of P' is less or equal than the value objective of P. In other words if we move a node in the path, then its objective value will not increase. It is trivial that a partial solution which does not fufill the Helmstädter condition cannot lead to an optimal solution and should therefore be discarded.

4.) Minimal aborescence: We can sharpen the bound for the TSP found in 1.) by summing up not only the cheapest incoming edge but by constructing a minimal aborescence (i. e. a set of directed trees in which, for some vertex u called the root and any other vertex v, there is exactly one directed path from u to v) on the remaining nodes. Since a path is also an aborescence it is clear that this is a lower bound for the TSP cost. There exist a lot of algorithms (e. g., Edmonds algorithm) which can be used for computing a minimal aborescence. But even despite the fact that these algorithms are polynomial, test runs have shown that this method does not work so well since the computation of the minimal aborescence takes way longer than the time saved by cutting off some branches of the search tree.

When we try to implement these conditions in practice, it shows that in the upper levels (e. g., where the partial solutions are short) it is nearly impossible to discard solutions and the search tree is mostly complete.

In consequence we come to the idea to combine the branch-and-bound method with dynamic programming. In more detail, we perform in the upper levels dynamic programming and from a defined depth we shift to branch-and-bound. The determination of the optimal depth where to switch between the two methods is not trivial, so computational tests are needed.

We will present some computational results in Chapter 7. For more detailed information on this topic consult [Sil12].

Chapter 6

Heuristic Approaches

In this chapter we present heuristics for the TVP. As we have described in Section 1.2, heuristics are approximation algorithms which examine just a subset of all feasible solutions of a combinatorial optimization problem and therefore do not find the optimal solution in every case. Nevertheless heuristics can be very useful in some cases. An example are applications where it is obligatory to find a good (but not necessary an optimal) solution fast. Heuristics are also used in support of other solution methods like branch-and-cut algorithms in order to obtain good bounds.

The quality of the results of a heuristic depends of course on the strategy which decides which feasible solutions are examined. To obtain good selection strategies for the TVP we examine already existing heuristics for the TSP and the LOP (see for example [RR11]) and extend their principles to the TVP. In this chapter we will present six such heuristics and explain the ideas behind them. Computational results can then be found in Chapter 7.

The set of implemented heuristics can be divided in two basic types. The first type constructs a feasible TVP path from scratch. The second type tries to improve an already existing feasible solution. From now on we will call the approaches of type one constructive heuristics and the ones of type two improvement heuristics. In detail, the constructive heuristics are a nearest-neighbor approach, the best-insertion method and a modified Becker heuristic. As improvement heuristics we implemented two different versions of the Kernighan-Lin heuristic as well as a simple 2-opt and 3-opt exchange.

In the following subsections we will describe each of the heuristics in detail. We denote by $[k_1, \ldots, k_n]$ a partial TVP path, by $[\,]$ an empty path, by $\mathrm{obj}(O)$ the objective value of a (partial) TVP path and by $O[k]$ the node which is on k-th position in O. By definition we set $\mathrm{obj}([\,])=-\infty$.

6.1 The Nearest-Neighbor Method

The nearest-neighbor method is a classical TSP heuristic. It is basically a greedy algorithm which starts at some node and then travels in each step to the unvisited node where the travel cost is minimal. The extension of this method to the TVP works similarly: We start with an empty path and construct the TVP path from the first target visited to the last one. In each step we check all remaining targets for which the following condition holds: If visited next, the target will lead to the largest increase (the increase can also be negative in some cases) of the objective function. In other words we compute for each remaining target the sum of the met preferences if it is visited next and subtract the distance cost of getting there. As one can see, this method is completely deterministic and produces the same result in each run. The formal description of this method can be found in Algorithm 6.

Algorithm 6: Nearest-Neighbor Method

```
1  $S := \{1 \dots n\},\ O := [\,],\ m := 1$ ;
2  while $S \neq \emptyset$ do
3      for each $s \in S$ do
4          $\text{pref}_s = \sum_{k \in S, k \neq s} p_{sk}$;
5          if $m \neq 1$ then $\text{dist}_s = d_{O[m-1],s}$;
6          else $\text{dist}_s = 0$;
7          $t_s = \text{pref}_s - \text{dist}_s$;
8      Compute $k$ so that $t_k = \max_{i \in S} t_i$;
9      Set $O[m] = k$;
10     $m = m + 1,\ S = \{S \setminus k\}$ ;
11 Return: $O$;
```

6.2 Best-Insertion Method

The best-insertion method is another classical heuristic which can been used on both of the subproblems. The extension to the TVP works as follows: We start with an empty path. Then, in each step we pick a random target which is not already contained in the path and insert it in the position where the resulting partial path has the maximal objective value. We repeat this insertion procedure until all targets are contained in the path. From that point we can either stop or repeat for a defined iteration number the following procedure: remove a random target from the path and reinsert it at the best position.

It is obvious that this heuristic is not deterministic. But nevertheless if the number of insertion operations is big enough, the calculated objective values of different runs are very similar. So a problem when using this method is to find a suitable number of inserting operations which is neither too small so that the result is too far away from the optimum nor too great so that the algorithms makes unnecessarily time consuming computations. A detailed description of this method is given in Algorithm 7.

Algorithm 7: Best-Insertion Heuristic

```
Data: NUM: Number of times the heuristic is executed, must be equal to or
      greater than $n$
1  Best_Path=[ ];
2  $S = \{1 \dots n\}$; $O = [\,]$;
3  for $a = 1$ to $NUM$ do
4      if $a \leq n$ then Choose per random $s \in S$;
5      else Remove a random target $s$ from the path $O$;
6      for $l = 1$ to $\min(a, n)$ do
7          $t_l = \sum_{k=1}^{l-1} p_{O[k],s} + \sum_{k=l}^{\min(a,n-1)} p_{s,O[k]}$;
8          if $l \neq 1$ and $l \neq \min(a, n-1)$ then
9              $t_l = t_l - d_{O[l-1],s} - d_{s,O[l]} + d_{O[l-1],O[l]} + \text{obj}(O)$
10         if $l = 1$ then $t_l = t_l - d_{s,O[1]} + \text{obj}(O)$;
11         if $l = \min(a, n)$ then $t_l = t_l - d_{O[l],s} + \text{obj}(O)$;
12     Compute $k$ so that $t_k = \max_{1 \leq i \leq \min(a,n)} t_i$;
13     Insert $s$ at position $k$;
14     if $a \leq n$ then $S = S \backslash s$;
15 Return: $O$;
```

6.3 Becker Heuristic

The Becker heuristic is another possible way to construct a TVP path without a start solution. It consists of two different phases which are repeated until the maximum iteration number is reached.

The first phase consists of generating a random TVP tour. Then the tour is transformed into a path by removing one edge. This is done for each of the n edges and for each of the resulting paths the objective value is computed. The best among these paths is then compared to the best know path so far and kept if better and fathomed if not. Then we start again with a new random path. In detail the procedure is described in Algorithm 8.

Algorithm 8: Becker Heuristic

```
Data: NUM: Number of times the heuristic is executed
1 Best_Path=[ ];
2 for $a = 1$ to $NUM$ do
3     Choose random TVP path $[O[1], \dots, O[n]]$;
4     for $m = 1$ to $n$ do
5         Set $O_m = [O[m], \dots, O[n], O[1], \dots, O[m-1]]$;
6         Compute $v_m$= obj($O_m$);
7         if $v_m$ >obj(Best_Path) then Best_Path=$O_m$;
8 Return: Best_Path;
```

6.4 The k-Opt Method

The k-opt method is a very simple improvement heuristic, which can be applied to many combinatorial optimization problems. The basic principle is to select k objects and locally re-optimize them. For the TVP this approach consists of selecting k targets from an existing TVP path and interchange their positions so that the objective value of the path is the greatest of all such possible interchanges. Because the number of possible interchanges grows exponentially if k is increasing, for practical purposes only $k = 2$ or $k = 3$ should be considered. A detailed description of k-opt is presented in Algorithm 9 (2-opt) and Algorithm 10 (3-opt).

Algorithm 9: 2-opt Heuristic

```
Data: NUM: Number of times the heuristic should be executed
O: A feasible TVP path
1 for $l = 1$ to $NUM$ do
2     Select $1 \leq i, j \leq n$ at random;
3     Interchange $O[i]$ and $O[j]$ if objective value of the path is increasing;
4 Return: $O$;
```

Algorithm 10: 3-opt Heuristic

```
Data: NUM: Number of times the heuristic should be executed
O: A feasible TVP path
1 for $l = 1$ to $NUM$ do
2     Select $1 \leq i, j, k \leq n$ at random;
3     Re-optimize the positions of $O[i]$, and $O[j]$ and $O[k]$ so that the
      objective value of the resulting TVP path is maximal;
4 Return: $O$;
```

6.5 Kernighan-Lin Heuristic

The Kernighan-Lin heuristic is one of the best-known methods for the computation of strong approximate solutions for the TSP. Its basic idea is to efficiently execute a series of simple operations to improve a given solution. In the case of the TVP we have implemented two variants of this approach. The first one is based on 2-opt exchanges, while the second approach uses insertion operations instead. The algorithm stops in both cases when every element has been moved at least once in the path. Then the exchange/insert operations are executed to the point where the overall objective is maximal. In more detail, one can see the two variants of Kernighan-Lin heuristics in Algorithm 11 and Algorithm 12.

Algorithm 11: Kernighan-Lin-1 Heuristic

```
   Data: O: A feasible TVP path
 1 $m = 1$;
 2 $S_m = \{1, 2, \dots, n\}$;
 3 Find $s, t \in S_m,\ s \neq t$ so that the interchange of these two nodes in the path
   leads to the largest increase $g_m$ among all such interchanges;
 4 Interchange $s$ and $t$;
 5 $s_m = s$ and $t_m = t$;
 6 if $m < n/2$ then
 7     $S_{m+1} = S_m \backslash \{s, t\}$;
 8     $m = m + 1$;
 9     Goto 3;
10 Find $k < m$ that maximizes $G = \sum_{l=1}^{k} g_l$;
11 if $G > 0$ then
12     Starting from the original TVP path interchange successively $s_i$ and $t_i$
       for $i = 1, \dots, k$;
13     Denote this new path by $O'$;
14 Return: $O'$;
```

Algorithm 12: Kernighan-Lin-2 Heuristic

```
   Data: O: A feasible TVP path
1  $m = 1$;
2  $S_m = \{1, 2, \ldots, n\}$;
3  Find the object $s \in S_m$ and the position $p$ so that $s$ moved to $p$ results in
   the largest increase $g_m$ of the objective value of all such moves;
4  Insert $s$ at position $p$;
5  $s_m = s$; $\text{pos}_m = p$;
6  if $m < n$ then
7      $S_{m+1} = S_m \backslash s$;
8      $m = m + 1$;
9      Goto 3;
10 Find $k \leq n$ that maximizes $G = \sum_{l=1}^{k} g_l$;
11 if $G > 0$ then
12     Starting from the original path insert $s_i$ at position $\text{pos}_i$ for
       $i = \{1, \ldots, k\}$;
13     Denote this new path by $O'$;
14 Return: $O'$;
```

Chapter 7

Computational Experiments

In this chapter we compare the different approaches for solving the TVP. The bases of these methods have been presented in Chapters 4, 5 and 6. To our knowledge there does not exist a set of benchmark instances for the TVP. Also no instance describing practical problems is available at the moment. So in order to compare the different solving methods we first have to define a set of test instances. Details on this topic will be presented in Section 7.1.

To compare the different methods directly against each other all computations were done on the same machine, namely an Intel® Pentium® Dual CPU E2200 with 2.20GHz and 8 GB RAM, running Open Suse 10.4.

7.1 Test-Instances

To give a detailed analysis of the implemented algorithms we need a sensible set of test instances. Since such a set does not exist, we have to define it by ourselves.

There are different possibilities how to create such test data sets. We decide to generate data which is closely related to real world applications. As a basis for this task we use a list which contains the coordinates and the numbers of inhabitants of all 15112 villages and cities of Germany.

The instances are then created in the following way. Firstly a certain number of cities is randomly selected from the list. Then we compute the distances of two cities a and b by using the Euclidean distance, namely (x_a denotes the x-coordinate and y_a the y-coordinate of a city a):

$$d_{ab} := \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}.$$

To reduce unnecessary high cost values in the problem we do the following for each city: First, we subtract the smallest cost for entering a city from all other costs for entering this city. Afterwards we do the same for the costs of leaving the city. We then delete randomly one city and proceed to calculate the preference values. This is done by using the number of inhabitants of two cities in the following way (i_a denotes the number of inhabitants of a city a):

$$p_{ab} := \left\lfloor \frac{i_a}{i_b} \right\rfloor \cdot f$$

The factor f denotes a weight factor which is used to control the ratio of the optimal LOP solution and the optimal TSP solution. This computation strategy has the advantage that for the resulting preference matrix it must be true that in nearly every case $\min\{p_{ab}, p_{ba}\} = 0$. In the unlikely event that both cities have the same number of inhabitants we set $p_{ab} = f$ and $p_{ba} = 0$.

Based on these ideas we construct the following different types of instances. Firstly we compute instances using just the methods described above. That means that the resulting instances have the properties that all distances fulfill the triangular equation and that they contain a complete ordering.

Next we create instances with the methods described above but afterwards we randomly interchange some of the preferences values. We do this exchange with two different stages of intensity: $\frac{1}{2}N^2$ and $\frac{1}{4}N^2$ (N denotes from now on the number of targets of a certain instance).

To distinguish between the different types of instances we introduce a name scheme where each instance is labeled as follows:

XX_OOO_N_ID

Hereby the letters have the following meaning:

XX: The ratio of the optimal LOP solution and the optimal TSP solution. We divide in three different classes:

ER: if $0.5 \leq \frac{\text{Sol. LOP}}{\text{Sol. TSP}} \leq 1.5$

LB: if $1.5 < \frac{\text{Sol. LOP}}{\text{Sol. TSP}} \leq 3$

LD: if $3 < \frac{\text{Sol. LOP}}{\text{Sol. TSP}}$

OOO: The term encodes the situation of the preference matrix:

CFO: No changes have been made

MCO: $\frac{1}{4}N^2$ interchanges have been made

BCO: $\frac{1}{2}N^2$ interchanges have been made

N: The number of nodes

ID: Number of the instance of that specific type (starting by 1).

An overview over all instances can be found in Appendix B.

7.2 Heuristics

In this section we examine the performance of the heuristics presented in Chapter 6. In order to obtain satisfying results we want to use an improvement heuristic together with a constructive heuristic.

But first we have to determine a suitable number of random tour generating (insertion) operations for the Becker (best-insertion) heuristic. For this purpose we make some experiments on the instances with 15–26 nodes with four different magnitudes (each depending on the number of nodes) for the number of such operations. In detail, we use the values: $5 \cdot N$, $10 \cdot N$, $50 \cdot N$ and $100 \cdot N$. Since both heuristics are not deterministic, we make for each instance and magnitude 500 different computations and calculate the mean value of all these experiments. The results can be observed in Appendix D in Tables D.3 and D.4 for the Becker heuristic and in Tables D.5 and D.6 for the best-insertion method.

We can make the following observation: In general a higher number of executed operations brings a result closer to the optimum. However, the improvement per additional operation decreases with the number of executions. In the case of the best-insertion heuristic it is often not necessary to execute $100 \cdot N$ insertion operations since the optimum is in most cases already achieved with a fewer number of such operations.

We also make some experiments on the performance of the nearest-neighbor heuristic. These results are presented in Tables D.1 and D.2.

If we compare the constructive heuristics we can state a clear order of the quality of their results. The far best results can be achieved with the best-insertion heuristic. The results of the other two methods are in most cases far away from the optimal solution. Nevertheless the nearest-neighbor heuristic performs still way better than the Becker heuristic (even if the number of randomly generated tours is very high). The computation times are in general short (at most 2 seconds for best-insertion with $100 \cdot N$ operations on 26 nodes), but the best-insertion method needs more time than the other two heuristics (computation time less than a second in all cases).

In the second part of this section we want to examine the performance of combinations of a constructive heuristic and an improvement heuristic: So we compute a solution with the Becker heuristic, the best-insertion method (both with $100 \cdot N$ operations) and the nearest-neighbor heuristic and try to improve that solution with one of the different improvement heuristics. For the k-opt methods we use the following values for the maximal number of changes/local re-optimizations: $100 \cdot N$ and $1000 \cdot N$ for the 2-opt method and $100 \cdot N$, $500 \cdot N$ and $1000 \cdot N$ for the 3-opt method. Again 500 different computations per instance are made. The results of this experiments can be observed in Appendix D in Tables D.7, D.8 and D.9 (Becker with $100 \cdot N$ operations as start heuristic), in Table D.10 (best-insertion with $100 \cdot N$ operations as start heuristic) and in Tables D.11, D.12 and D.13 (nearest-neighbor as start heuristic).

In general we can observe that the Kernighan-Lin-2 heuristic achieves the best results and finds in almost every case the optimal solution if we use Becker with $100 \cdot N$ operations or best-insertion with $100 \cdot N$ operations as start heuristic. Only with the nearest-neighbor heuristic the results are not so close to the optimal solution. The Kernighan-Lin-1 heuristic does not perform so well. Only with Becker as start heuristic the results are close to the optimum. In the other cases the method achieves in the average worse results than the k-opt methods.

For the k-opt methods we can observe that the number of executions does not have that much effect on the results. It seems that in both cases $100 \cdot N$ is already a suitable number of interchanges to find the local maximum. Of course the results of the 3-opt method are slightly better in the average than of the 2-opt approach, but the difference is small. We also observe that the standard derivation is very high, so the k-opt method is highly unpredictable.

In summary of these facts we can conclude that the Kernighan-Lin-2 heuristic in connection with Becker with $100 \cdot N$ operations or best-insertion with $100 \cdot N$ operations produces the best results. So finally we computed the mean values for these two combinations on instances with 30–50 nodes. The results can be observed in Tables D.14 and D.15.

It can be concluded that Kernighan-Lin-2 with best-insertion as start heuristic performs better. The running times for an instance with 50 nodes are 10 seconds (Becker) and 12 seconds (best-insertion).

So as a conclusion we can say that the combination of best-insertion and Kernighan-Lin-2 heuristic is a very useful tool for the calculation of solutions which are very close to the optimum.

7.3 Branch-and-Cut for the TVP$_{HP}$

In this section we present a branch-and-cut algorithm which solves the TVP. For this purpose we will use the TVP_{HP} model as a basis as well as the facets we obtained in Chapter 4 which will be used as cutting planes. We have implemented the algorithm in C++ and use ABACUS [JT00] as the branch-and-cut framework. The LP computations where done by CPLEX 12.1.

Since the TVP_{HP} model is a 0/1 linear program the branching step simplifies to adding the constraints x_{ij}=0 (w_{ij}=0) or x_{ij}=1 (w_{ij}=1) to the new subproblems. This means we only have to make one branching step per variable. So the depth of the search tree is bounded by the number of variables. We also use logical implications which arise by fixing an x_{ij} or w_{ij} to 0 or 1. So we are able to shrink the search tree even more. Nevertheless these implications do not have that much effect on the running time of the algorithm.

Additionally we employ a heuristic to quickly find a lower bound for the best integral solution in order to discard subproblems with bad LP solutions. Because of the results shown in the last section we decide to use the Kernighan-Lin-2 heuristic. To obtain a feasible start solution we take the non integral solution of the LP relaxation and discretize it in the following way. We compute "positions" by $W_i = n - \sum_{j=1}^{n} w_{ij}$ and sort them in nondecreasing order. Then we construct a Hamiltonian path by setting the node i on the first position which has the smallest W_i, the node with the second smallest W_i on the second position and so on.

As parameter settings for CPLEX and ABACUS we mainly use the default settings with a few exceptions: We enable the ABACUS function "tailing-of-control", which allows us to quit the cutting phase when the LP solution has not achieved a certain improvement in the last X iterations. The constraint elimination strategy is set to non-binding. Last the maximal number of violated inequalities which are added in each cutting step is fixed to 50.

In the cutting phase we try to use the facet classes we have listed in Table 4.3 (with the exception of classes 11 and 46, which are already contained in the IP model). So in our first experiment we want to check for each of these facet classes whether it makes sense to use them as cutting planes or not. For this task we combine the TVP_{HP} model with one additional facet class and test the performance on the instances with 15–23 nodes. In case of class 29 (extended 3-cycle) we leave out the normal 3-cycles since they are redundant then.

The results of these tests are shown in Appendix F in Tables F.1. F.2 and F.3.

As one can see, the results for the various facet classes are of course very different. But in nearly all cases the use of additional facets makes the algorithm faster. Nevertheless the magnitude of speedup is very different per facet class and also per instance. Overall we can observe that by far the best results are achieved by classes 25, 29 and 41. Also class 30 and class 39 show good results for most instances.

In the next series of experiments we check whether the combined use of two or more facet classes will speed up the algorithm even more. In detail we first try every possible combination of the classes 25, 29 and 41 and then add classes 30 and 39 to those two combinations which show the best results. As instances for these tests we use the ones with 20 and 23 nodes. The results of these experiments are shown in Table F.4 and F.5. As one can observe, the best results are achieved by the following combinations of facet classes:

- class 29 and class 41,
- class 29, class 39 and class 41,
- class 29, class 30, class 39 and class 41,
- class 25, class 29 and class 41,
- class 25, class 29, class 30 and class 41,
- class 25, class 29, class 39 and class 41,
- class 25, class 29, class 30, class 39 and class 41.

We can observe that the results of combination 29/39/41 and combination 29/30/39/41 are very similar. In other words, adding class 30 does in that case speed up the computation. So we do not consider the combination 29/30/39/41 anymore. The same applies for the combinations 25/29/39/41 and 25/29/30/39/41 where we leave out 25/29/39/41. We apply the remaining five combinations to the instances with 26 and more nodes. The results of the runs can be seen in Table F.6 and Table F.7.

We can make the observation that there is no general rule which combination works best. On most instances the combination 25/29/41 produces better results than the combination 29/41. The addition of classes 30 and 39 then improves these results even more in a lot of cases. But in some cases it also slows down the solving time. The instances with 45 nodes and also some instances with 40 nodes can only be solved with the combination of all five facet classes. So overall this seems to be the most promising approach.

But with any of these combinations the algorithm produces good results and is way more efficient than if we just work with the basic model where we are in most cases already unable to solve instances with 23 nodes.

Last we compare the root bounds of the test runs with the different facet combinations. Tables of these bounds and some other miscellaneous statistics on the test runs can be found in Appendix F.3.

Not surprisingly the combination 29/41 does produce the worst root bounds. Interestingly the combination 29/39/41 produces root bounds which lie much closer to the optimum than the bounds of 25/29/41 and 25/29/30/41. It seems that in most cases class 39 cuts off a big part of the polytope (at least considering our type of instances). If we compare 25/29/41 and 25/29/30/41 it seems that class 30 only reduces the depth of the search tree but does not cut off a big area of the polytope.

7.4 Branch-and-Bound with active/inactive variables for the TVP_E

We have seen in Chapter 4 and in the tables in Appendix C that the gap between the optimal integer solution and the LP relaxation of the TVP_E model is on average about 50 % better than in case of the TVP_{HP} model. On the other hand we face the disadvantage that the extended formulation contains a cubic number of variables. This means that it is not so suitable to use the common branch-and-bound or branch-and-cut technique since it takes a long time to solve the LP relaxation.

An approach for coping with this problem is to use the concept of active and inactive variables which we introduce in the following.

The idea of active and inactive variables is a possible extension of branch-and-bound or branch-and-cut algorithms. This technique has been developed for polytopes which contain a lot of variables. Such polytopes, for example, occur when we extend IP models by adding variables and constraints (so-called extended formulations).

The key idea is then to reduce the amount of time which is needed for solving the linear program in each branching step. This is done by solving the LP with the simplex method which contains just a subset of the variables, while all other variables which are currently not used are fixed to zero. After each solution of the linear program, the reduced costs of all variables which are currently not part of the basis are calculated. Then we decide which of these variables should become a base variable in the next iteration of the simplex algorithm.

This process is done until no variable with negative reduced cost exists. Then we continue with the branching step like in the branch-and-bound method. Formally the additional step is stated in Algorithm 13.

Algorithm 13: Reduced costs

```
1 Solve LP, obtain $x^*$;
2 if variables with negative reduced cost exist then
3     Add variable with biggest negative reduced cost to the basis;
4     Goto 1;
5 else
6     Return $x^*$
```

In case of the TVP_E model it seems natural to select the w_{ijk}-variables as active/inactive variables.

The algorithm was written in C++ and uses the framework SCIP [Ach09]. The computation times (with the default settings of SCIP) of this method can be observed in Appendix G. When we compare the results of this method to the performance of the branch-and-cut approach we have to admit that in all cases the computation time is much worse. Also a modification of the parameters of SCIP does not change this fact in general.

Because of these results we decide not to pursue this approach any further. Maybe an extension of this method to a branch-and-cut algorithm with active/inactive variables would speed up the computation. But for this idea more facts about P^n_{ETV} must be obtained.

7.5 Dynamic Programming

In this section we present the results of the implementation (written in C++) of the dynamic programming approach which we described in Section 5.2.

The performance of a dynamic programming algorithm for an NP-hard problem is normally bounded by the available disk space. In the case of the TVP, the amount of needed disk space (roughly $\mathcal{O}(n \cdot 2^n)$) doubles with each additional node but it is independent from the given instance. The same applies for the computation time. So the only remaining parameter is the number of nodes. Table 7.1 shows the computation times of an instance with N nodes.

N	15	20	23	26	30
Time	< 1	00:00:10	00:02:15	00:47:40	Mem.

Table 7.1: Computation times of the dynamic programming approach

As one can see, the largest instances we are able to solve have 26 nodes. But the computation times for 23 and 26 nodes are much worse than the ones of the branch-and-cut algorithm or even the CPLEX computation times. So the dynamic programming approach is, as expected, not a good alternative for big instances. Only for instances up to 20 nodes it may be in some cases a fast and suitable alternative method.

7.6 Branch-and-Bound with Dynamic Programming

As stated in Section 5.3 the branch-and-bound approach, which does not use IP modeling, does not perform good enough to be used for the TVP. So we decide to combine the branch-and-bound approach with dynamic programming. More precisely we use dynamic programming in the upper levels where there is not much chance to branch and shift at one point to branch-and-bound which is then used to computed an optimal partial path through the remaining nodes. We also examined the idea of using dynamic programming in the last levels.

For practical computations we use the algorithm presented in [Sil12] which was written in C++ and contains the following three parameters:

- Depth where to switch from dynamic programming to branch-and-bound.
- The frequency how often the Helmstädter condition should be checked.
- The information whether dynamic programming should be used at the last level. And if so at which level it should start.

The outcomes of [Sil12] suggest that the Helmstädter condition should be checked at every level and that it is not useful to use dynamic programming at the end. Test runs on our instances support this result. So we focus on the determination on the right depth to switch between dynamic programming and branch-and-bound.

For this purpose we make experiments on the instances with 15–26 nodes. We test different switch depths between 1 (i. e. no dynamic programming) and 12. We do not test greater switching depths because we wanted to focus on branch-and-bound in this context. The results are shown in Appendix E in Table E.1 (instances with 15 nodes), Table E.2 (instances with 20 nodes), Table E.3 (instances with 23 nodes) and Table E.4 (instances with 26 nodes).

We come to the conclusion that for a given number of nodes the optimal switch-depth cannot be determined in general since it strongly depends on the specific instance. We also observe that a wrong switching depth produces in some cases a much longer running time compared to pure dynamic programming. So for small instances it seems more sensible to use dynamic programming alone since the additional effort of determining the right switch depth is too big compared to the time which can be saved.

For the big instances, on the other hand the computation time even in best case is far worse compared to the result of the branch-and-cut approach. Together this makes this method not capable of being used in practice.

7.7 Lagrange Decomposition

In this section we present the results of an algorithm which uses Lagrangrean relaxation. For this purpose we want to use the Lagrangean decomposition approach we described in Section 5.1. We implemented three different methods for solving the Lagrangean function. In detail we tried the subgradient method, a conic bundle and a proximal bundle approach. The subgradient method has been described in Section 5.1. For a description of the two bundle methods consult [Fie12].

The algorithms were written in C++. We used the LOPSUM library [Rei08] for solving the partial LOP problem and the Concord [ABCC03] code for solving the TSP subproblem. For solving the linear programs we again used CPLEX 12.1.

We test the three algorithms on the instances with 15 and 20 nodes. The results of these experiments are shown in Appendix 7. As one can observe the results are very poor. Also the computation times are quite long. So for practical purposes this methods seems really not suitable.

7.8 Comparison

If we compare the different solving methods we come to the conclusion that in nearly every case the branch-and-cut approach is the best (and for instances with more than 30 nodes often the only way) to obtain an optimal solution.

The use of dynamic programming does, as expected, only make sense for small instances where it can sometimes be the fastest of all examined methods. The hybrid branch-and-bound/dynamic programming algorithm is also only suitable for small instances, but for these cases the use of pure dynamic programming is preferable since a wrong switching depth produces longer running times.

The branch-and-bound approach with active variables based on the extended formulation is able to solve instances up to thirty nodes. But the computation times are much worse compared to the branch-and-cut approach. In most cases even CPLEX is faster.

Perhaps it is possible to make the algorithm faster by adding an additional cutting phase. But at the moment it seems that this approach can even then not compete with the branch-and-cut approach.

The results of the branch-and-cut approach depend very heavily on the used cutting planes. So with the wrong choice of facet classes the algorithm already fails to compute instances with 23 nodes. With the right choice instead it is possible to achieve results that are magnitudes better than the results of CPLEX or any other approach. The border of calculability so far lies at fifty nodes in cases of our types of instances.

For practical purposes the combination of best-insertion and Kerninghan-Lin-2 heuristics may be a good approximate alternative since it obtains very good solutions in fast time. So in the case that we do not need a guaranteed optimal solution but only a solution which is near the optimum this combination of heuristics is probably the best approach. The other combinations of heuristics work not that much successful and are therefore not useful for practical combinations.

The Lagrangean approach is also not useful for practical computations because of its poor results and long computation times.

So in summa this shows that the target visitation problem is way harder to solve than the linear ordering problem (here instances which contain a nearly complete ordering can be solved up to 200 nodes) or the TSP (metric instances up to 85900 nodes can be solved).

It also shows that the TVP is a really hard problem in practice and that it is not possible to solve it with standard methods. Instead we have to invest a lot of time in specialized algorithms to obtain satisfying results.

Chapter 8

Variants of the TVP

In this chapter we will provide some information of two variants of the TVP. First this is the original tour formulation for which we will make a few comments how results from Chapters 3 and 4 can be transferred. Secondly we like to give a short overview of the properties of the symmetric variant of the TVP.

8.1 The original TVP formulation

It is possible to model the tour variant of the TVP with an IP formulation which is (analogous to the TVP_{HP} model) constructed out of the IP models of the two subproblems, the LOP and the TSP. The variables we need to introduce are very similar to the ones used in the TVP_{HP} model but differ slightly in some points. In the following the base node is denoted by node 0 while the targets are denoted by $1, \ldots, n-1$.

So the TSP-variables x_{ij}, $0 \leq i, j \leq n-1, i \neq j$ are this time defined as follows:

$$x_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(k+1) \text{ for some } k \in \{0, \ldots, n-2\}, \\ 1 & \text{if } i = \pi(n-1) \text{ and } j = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The LOP-variables w_{ij}, $1 \leq i, j \leq n-1$, $i \neq j$ are defined exactly in the same way as in the path model. However the preference are only defined between two targets and not between the base and a target. With this we are able to set up the following IP model:

$$\max \left(\sum_{i=1}^{n-1} \sum_{\substack{j=1 \\ j\neq i}}^{n-1} p_{ij} w_{ij} - \sum_{i=0}^{n-1} \sum_{\substack{j=0 \\ j\neq i}}^{n-1} d_{ij} x_{ij} \right) \tag{8.1}$$

$$s.t.$$

$$\sum_{\substack{i=0 \\ i\neq j}}^{n-1} x_{ij} = 1,\ 0 \leq j \leq n-1, \tag{8.2}$$

$$\sum_{\substack{j=0 \\ j\neq i}}^{n-1} x_{ij} = 1,\ 0 \leq i \leq n-1, \tag{8.3}$$

$$\sum_{i\in S} \sum_{\substack{j\in S \\ j\neq i}} x_{ij} \leq |S| - 1, \quad \text{for all } S \subset V,\ 2 \leq |S| < n, \tag{8.4}$$

$$w_{ij} + w_{jk} + w_{ki} \leq 2,\ 1 \leq i,j,k \leq n-1,\ i < j,\ i < k,\ j \neq k, \tag{8.5}$$

$$x_{ij} - w_{ij} \leq 0,\ 1 \leq i,j \leq n-1,\ i \neq j, \tag{8.6}$$

$$w_{ji} + w_{ij} = 1,\ 1 \leq i < j \leq n-1, \tag{8.7}$$

$$x_{ij} \in \{0,1\},\ 0 \leq i,j \leq n-1,\ i \neq j, \tag{8.8}$$

$$w_{ij} \in \{0,1\},\ 1 \leq i,j \leq n-1,\ i \neq j. \tag{8.9}$$

We will denote the model in the following by TVP_O and its associated polytope by P^n_{OTV}. Since this model is similar to the TVP_{HP} model, we do not want to explain it in detail but make some comments on the differences. As one can see the degree constraints (8.2) and (8.3) are now equations instead of inequalities. Therefore the P^n_{OTV} polytope with reduced w_{ij}-variables is not nearly full-dimensional as the P^n_{TV}.

Due to the results of [QW93], it is clear that P^{n-1}_{TV} and P^n_{OTV} are isomorphic. This has the consequence that the dimension of the polytope P^{n+1}_{OTV} is also equal to $\frac{3n^2-3n-2}{2}$ (that means the dimension of P^n_{OTV} is equal to $\frac{3n^2-9n+4}{2}$). But when we try to explicitly prove this fact by transferring the ideas of the proof of Theorem 4.1 we suffer some difficulties because of loss of the full-dimensionality.

Another consequence of this isomorphism is that, there is an one-to-one correspondence between the facets of the polytopes. So the description of P^5_{OTV} must then correspond with the description of P^4_{TV} and consist again of 1280 facet-defining inequalities which can be sorted in 48 classes.

This also implies that for the facets f of P^{n-1}_{OTV} and g of P^{n}_{OTV} where g is the zero-lifting of f there must exist corresponding facets f' of P^{n-2}_{TV} and g' of P^{n-1}_{TV}. But to us it is not clear that g' is also the zero-lifting of f'.

For the sake of completeness we therefore show an adaption of the Theorems 4.4–4.6. Since the proofs are very similar to each other we only present a short version and focus mainly on the differences between tour and path case. Unfortunately in the tour we are case only able to prove a zero-lifting theorem (similar to Theorem 4.5) for facets which have zero as free node.

In the following let MTV_n be defined analogously as in Chapter 4 and with MTV^{i}_{n+1} we denote in the tour case the matrix of all paths where i is visited as first target after the base node.

Theorem 8.1. *Let f be a facet of P^{n}_{OTV} defined by $a_w w + a_x x \le a_0$ which contains* 0 *and some other node l as free nodes. Let $f^{(n)}$ be feasible for P^{k}_{OTV} with $k > n$. Then $f^{(n)}$ is a facet of P^{k}_{OTV} with $k \ge n$.*

Proof. We prove this theorem in a similar way as we proved Theorem 4.5. The main difference is the definition of the bijections which will be presented in the following.

Let $i \neq 0$ be any free node of $f^{(n+1)}$. Then there exists a bijection I between the rows of MTV_n and MTV^{i}_{n+1} (i.e. between S_{n-1} and a subset of S_n) such that if and only if a characteristic vector (w_r, x_r) of P^{n}_{OTV} satisfies f with equality, then $I(w_r, x_r)$ satisfies the zero-lifting $f^{(n+1)}$ with equality.

First we consider the case $i = n$ (it is obvious that n must be a free node of $f^{(n+1)}$). We insert n after the base node 0 and get the following bijection:

$$I:\ S_{n-1} \to S_n,\ \ \pi'(k) := \begin{cases} n & \text{if } k = 1, \\ 0 & \text{if } k = 0, \\ \pi(k-1) & \text{otherwise.} \end{cases}$$

Now consider the case $i < n$: At first we use the same bijection as above. Then we exchange node n and node i, which leads to the following bijection (p_i denotes the old position of i in the path):

$$I:\ S_{n-1} \to S_n,\ \ \pi'(k) := \begin{cases} i & \text{if } k = 1, \\ 0 & \text{if } k = 0, \\ n & \text{if } k = p_i + 1, \\ \pi(k-1) & \text{otherwise.} \end{cases}$$

Because 0, i and n are free nodes it is still true that $a_w w_r + a_x x_r = a_0$.

Because f contains l and 0 as free nodes it is possible to define the transformations as in the proof of Theorem 4.5 with the help of these bijection by inserting node l and n one time between the base node and the first target visited and second between the last target visited and the base node. Then we can conclude (analogous to the proof of Theorem 4.5) that no additional valid column equation does exist. □

8.2 The Symmetric Target Visitation Problem

Analogous to the traveling salesman / Hamiltonian path problem it is also possible to study a variant of the TVP where all distances are symmetric. So we now consider the case that

$$d_{ij} = d_{ji} \; \forall i, j.$$

The symmetric target visitation problem (STVP) has been studied extensively in [Ron14]. That is why we want to present in the following only main aspects and key results on this topic. Further information can then be found in the above mentioned source.

Again we first present an IP model for the STVP which is based on its two subproblems. While for this purpose the w-variables can be defined as in the asymmetric case the x-variables are now symmetric and therefore their definition has to be changed. In detail we define x_{ij}, $1 \le i < j \le n$ as follows:

$$x_{ij} := \begin{cases} 1 & \text{if } i = \pi(k) \text{ and } j = \pi(k+1) \text{ for some } k \in \{1, \dots, n-1\}, \\ 1 & \text{if } j = \pi(k) \text{ and } i = \pi(k+1) \text{ for some } k \in \{1, \dots, n-1\}, \\ 0 & \text{otherwise.} \end{cases}$$

Because of the symmetry it now becomes more difficult to combine both IP formulations. So instead of one simple constraint which makes sure that x-variables and w-variables match with each other we now need six different types of connection inequalities. So the model now looks like this:

$$\max \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} w_{ij} - \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} x_{ij} \right) \tag{8.10}$$

s. t.

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{ij} = n - 1, \tag{8.11}$$

$$\sum_{i=0}^{j-1} x_{ij} + \sum_{i=j+1}^{n} x_{ji} \leq 2, \ j \in V, \tag{8.12}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ i \leq j}} x_{ij} \leq |S| - 1, \ \forall S \subset V, \ 2 \leq |S| < n, \tag{8.13}$$

$$w_{ij} + w_{jk} + w_{ki} \leq 2, \ i, j, k \in V, \ i < j, \ i < k, j \neq k, \tag{8.14}$$

$$w_{ij} + w_{ji} = 1, \ i, j \in V, \ i \neq j, \tag{8.15}$$

$$x_{ij} + x_{jk} - w_{ij} - w_{jk} - 2w_{ki} \leq 0, \ i, j, k \in V, \ i < j < k, \tag{8.16}$$

$$x_{ij} + x_{jk} - w_{ji} - w_{kj} - 2w_{ik} \leq 0, \ i, j, k \in V, \ i < j < k, \tag{8.17}$$

$$x_{ij} + x_{ik} - w_{ij} - w_{ki} - 2w_{jk} \leq 0, \ i, j, k \in V, \ i < j, \ i < k, \tag{8.18}$$

$$x_{ij} + x_{ik} - w_{ji} - w_{ik} - 2w_{kj} \leq 0, \ i, j, k \in V, \ i < j, \ i < k, \tag{8.19}$$

$$x_{ik} + x_{jk} - w_{ik} - w_{kj} - 2w_{ji} \leq 0, \ i, j, k \in V, \ j < k, \ i < k, \tag{8.20}$$

$$x_{ik} + x_{jk} - w_{ki} - w_{jk} - 2w_{ij} \leq 0, \ i, j, k \in V, \ j < k, \ i < k, \tag{8.21}$$

$$w_{ij} \in \{0, 1\}, \ i, j \in V, \ i \neq j, \tag{8.22}$$

$$x_{ij} \in \{0, 1\}, \ i, j \in V, \ i < j. \tag{8.23}$$

Of course it is (like in the asymmetric case) possible to eliminate one half of the w-variables. Also the subtour elimination constraints (8.13) are not needed here, since they are still implied by the three cycles (8.14). We will denote this model as TVP_S and the associated polytope as P^n_{STV}.

Next we like to make some comments about the polyhedra structure of P^n_{STV}.

Theorem 8.2. *The dimension of* P^n_{STV} *is* $n^2 - n - 1$.

The proof of this theorem can be found in [Ron14].

We are able to compute P^4_{STV} which consists of 1010 facets which can be sorted by HUHFA to 38 classes. An interesting fact is that in contrary to the asymmetric variant of the TVP the three cycles (8.14) are facet-defining here. This means there exist non-trivial facets which are containing only LOP-variables (such facets do not exist in the asymmetric case). The whole description of the polytope can be found in Appendix A.4.

Finally we want to present a general zero-lifting result. The proof for this theorem can also be found in [Ron14].

Theorem 8.3. *Let f be a facet of P^n_{STV} which fulfills the following constraints:*

- *There exist two nodes k, l and two arbitrarily paths μ and ν through the nodes $\{1, \dots, n\} \setminus \{k, l\}$, so that the characteristic vectors of the paths $l \to \mu \to k$ and $k \to \nu \to l$ do fulfill f with equality.*
- *The facet contains a free node $u \in \{1, ..., n\}$.*
- *There exists an arbitrarily node h, so that for every other node $g \neq h$ and an arbitrarily path ξ through the nodes $\{1, \dots, n\} \setminus \{g, h\}$ at least one of the path $g \to \xi \to h$ or $h \to \xi \to g$ does fulfill f with equality.*

Then f is a facet of P^m_{STV} with $m \geq n$, if and only if f is a valid inequality for P^m_{STV}. Also the constraints mentioned above will then be fulfilled in dimension m.

With the help of this theorem it is possible to lift 15 facets of P^4_{STV} (see [Ron14]).

Chapter 9

Conclusion and Further Research

In this thesis, we have presented an extensive survey on the target visitation problem, a combination of the traveling salesman and the linear ordering problem.

We studied different integer programming formulations for the problem and came to the conclusion that the idea of combining the IP models of the two subproblems HP and LOP leads to the best model for practical purposes. The approaches using edge-node variables or distance variables unfortunately have both very tremendous gaps between the solution of the LP relaxation and the IP solution. Perhaps a detailed investigation of the associated polytopes could in the future lead to a more compact description based on the same type of variables.

After having a closer look at the associated polytope P^n_{TV} of the TVP_{HP} model, we found that we need way more inequalities to describe it than for the linear ordering or traveling salesman polytope. We were able to prove that the dimension of P^n_{TV} is equal to the sum of the dimension of the corresponding linear ordering polytope plus the dimension of the corresponding Hamiltonian path polytope. We also developed a zero-lifting theorem which we used to show that some facets of P^4_{TV} are general facet classes. Additionally some other general facet classes have been presented.

Because the LP/IP gap of the TVP_{HP} model is quite big, we have developed an extended formulation by extending the linear ordering variables to a three index version. This formulation then has a much smaller LP/IP gap. Unfortunately, the cubic numbers of variables makes it difficult to use this model for a branch-and-cut approach because it takes much more time to solve the linear programming relaxation with the additional variables and constraints. So we tried to work around this problem by using column generation. We also examined the connection between the P^n_{TV} and P^n_{ETV} by having a closer look at which facets are implied by the projection of P^n_{ETV} onto P^n_{TV}.

In the practical part of the thesis, we have considered several methods for solving the target visitation problem. We have shown that the standard approaches branch-and-bound as well as dynamic programming are only useful for small instances, which is not surprising since this is the case for most NP-hard problems. Also, the Lagrangean decomposition approach does not perform satisfactorily.

We achieved by far the best results with a branch-and-cut approach which uses the facet classes we discovered before as cutting planes. But, as we have seen, it is not a trivial question which facet classes should be selected in the cutting phase. Nevertheless, with an optimal choice, we were able to solve instances up to forty-five nodes, which is so far impossible with any other approach.

Unfortunately, the results of the branch-and-bound algorithm with active/non-active variables could also not compete with the branch-and-cut approach. It remains an open question whether the extension of the algorithm to a branch-and-cut algorithm would make the computation faster.

Our studies of the heuristics support the exact approaches. The combination of the best-insertion and the Kenigham-Lin heuristic is capable of quickly obtaining good bounds (in the most cases even the optimal solution). For some practical purposes this may be already sufficient.

So in the end, we conclude that the TVP is an interesting combinatorial optimization problem which possesses a nice structure and is worth further investigation. Of course this thesis can only be considered as a base work on this subject. But we think it is a good starting point for further investigation in both theoretical as well as practical directions.

Bibliography

[ABCC03] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. CONCORD: Tsp solver, 2003. `http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.html`.

[ABCC06] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem.* Princeton series in applied mathematics. Princeton Univ. Press, 2006.

[Ach09] T. Achterberg. Constraint integer programming, 2009. PhD thesis.

[ACP08] A. Arulselvan, C.W. Commander, and P.M. Pardalos. A hybrid genetic algorithm for the target visitation problem. *Technical report, University of Florida*, 2008.

[BBI$^+$06] Z. Blázsik, T. Bartók, B. Imreh, C. Imreh, and Z. Kovács. Heuristics on a common generalization of TSP and LOP. *Pure Mathematics and Applications*, 17(3-4):229–239, 2006.

[CL09] T. Christof and A. Loebel. PORTA: Polyhedron representation transformation algorithm, version 1.4, 2009. `http://comopt.ifi.uni-heidelberg.de/software/PORTA/index.html`.

[Fie12] J. Fielenbach. Lagrangian decomposition of the target visitation problem using bundle methods, 2012. Diploma thesis.

[FQ12] G. Fischer and F. Quiring. *Lernbuch Lineare Algebra und Analytische Geometrie.* Springer Spektrum, 2012.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman & Co., 1979.

[GJ04] D.A. Grundel and D.E. Jeffcoat. Formulation and solution of the target visitation problem. *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, 2004.

[GJR85] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.

[Hel64] E. Helmstädter. Die Dreiecksform der Input-Output-Matrix und ihre möglichen Wandlungen im Wachstumsprozess. *Strukturwandlungen einer wachsenden Wirtschaft*, 1964.

[HHS$^+$13] O. Heismann, A. Hildenbrandt, F. Silvestri, G. Reinelt, and R. Borndörfer. HUHFA: A framework for facet classification. Technical Report 13-45, ZIB, Takustr.7, 14195 Berlin, 2013.

[Hun14] P. Hungerländer. A semidefinite optimization approach to the target visitation problem, 2014. Technical Report.

[JT00] M. Jünger and S. Thienel. The ABACUS System for Branch-and-Cut-and-Price Algorithms in Integer Programming and Combinatorial Optimization. *Software: Practice and Experience*, 30:1325–1352, 2000.

[KV12] B. Korte and J. Vygen. *Kombinatorische Optimierung*. Springer Spektrum, 2. edition, 2012.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. Wiley Interscience Series in Discrete Mathematics. John Wiley & Sons, 1985.

[Lör14] S. Lörwald. Exact solving methods for the target visitation problem, 2014. Diploma thesis.

[MTZ60] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960.

[NR93] D. Naddef and G. Rinaldi. The graphical relaxation: A new framework for the symmetric traveling salesman polytope. *Mathematical Programming*, 58(1-3):53–88, 1993.

[QW93] M. Queyranne and Y. Wang. Hamiltonian path and symmetric travelling salesman polytopes. *Mathematical Programming*, 58:89–110, 1993.

[Rei85] G. Reinelt. *The linear ordering problem: Algorithms and Applications*. Heldermann, 1985.

[Rei08] G. Reinelt. Sublop solver library, 2008.

[Ron14] K. Ronellenfitsch. Das Symmetrische Target Visitation Problem, 2014. Diploma thesis.

[RR11] M. Rafael and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer, 2011.

[Sil12] F. Silvestri. Ein Branch-and-Bound Ansatz für das Target-Visitation-Problem, 2012. Bachelor's thesis.

[Zie95] G. M. Ziegler. *Lectures on polytopes*. Springer, 1995.

Appendix A

Description of Small TVP-Polytopes

In the following let $i, j, k, l \in \{1, 2, 3, 4\}$ and pairwise different.

A.1 Complete Description of P^4_{TV}

Cl.	Facet defining inequality
1	$w_{ij} + w_{jk} + w_{kl} + w_{lj} - x_{ij} - x_{ik} + x_{ji} + x_{jl} - x_{lj} \leq 2$
	$w_{jk} + w_{kl} + w_{li} + w_{lj} + x_{il} + x_{jl} - x_{ki} - x_{li} - x_{lj} \leq 2$
2	$w_{il} + w_{kl} + w_{lj} + x_{ji} + x_{jk} + x_{jl} + x_{li} + x_{lk} \leq 3$
	$w_{ji} + w_{jk} + w_{lj} + x_{ij} + x_{il} + x_{jl} + x_{kj} + x_{kl} \leq 3$
3	$w_{lj} + x_{il} + x_{ji} + x_{jk} + 2x_{jl} + x_{kl} - x_{lj} \leq 2$
4	$2w_{ij} + w_{jk} - x_{ij} - x_{ik} + 2x_{ji} + x_{jl} + x_{ki} + x_{kj} + x_{kl} + x_{li} \leq 3$
	$w_{il} + 2w_{lk} - x_{ik} + x_{ji} + x_{jl} + x_{ki} + x_{kj} + 2x_{kl} + x_{li} - x_{lk} \leq 3$
5	$w_{ij} + w_{il} - x_{ij} - x_{ik} - x_{il} + x_{ji} + x_{li} \leq 1$
	$w_{jk} + w_{lk} - x_{ik} - x_{jk} + x_{kj} + x_{kl} - x_{lk} \leq 1$
6	$w_{ij} + w_{lk} - x_{ij} - x_{ik} + x_{ji} + x_{jl} + x_{kl} - x_{lj} - x_{lk} \leq 1$
7	$w_{ij} + 2w_{jk} + 2w_{kl} + w_{li} + w_{lj} + x_{il} + x_{ji} + x_{jl} + x_{kj} + x_{lk} \leq 5$
8	$w_{jk} + 2w_{kl} + w_{lj} - x_{ki} - x_{kl} + x_{lk} \leq 2$
	$2w_{jk} + w_{kl} + w_{lj} - x_{ik} - x_{jk} + x_{kj} \leq 2$
9	$w_{ji} + w_{kl} + w_{lj} + x_{ij} + x_{ik} + x_{il} + x_{jk} + x_{jl} - x_{ki} + x_{lk} \leq 3$
10	$2w_{ij} + 2w_{jk} + w_{kl} + w_{li} - x_{ij} - x_{ik} + x_{ji} - x_{jk} + x_{kj} \leq 3$
11	$-x_{ki} \leq 0$
12	$w_{ij} + w_{il} + w_{jk} + w_{lk} - x_{ij} - 2x_{ik} - x_{il} + x_{ji} - x_{jk}$ $+x_{ki} + x_{kj} + x_{kl} + x_{li} - x_{lk} \leq 2$
13	$w_{il} + w_{ji} + w_{jk} + w_{kl} + w_{lj} + x_{ij} + x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 4$

14 $w_{ij} + 2w_{jk} + w_{ki} + w_{kl} + w_{lj} + x_{ji} + x_{jl} + x_{kj} \leq 4$

$w_{il} + w_{ji} + w_{jk} + w_{kl} + 2w_{lj} + x_{ij} + x_{jl} + x_{kj} \leq 4$

15 $w_{il} + w_{jk} + w_{ki} + w_{kl} + w_{lj} - x_{il} - x_{ki} - x_{kl} \leq 2$

16 $2w_{jk} + w_{kl} + 2w_{lj} - x_{ik} - x_{jk} + x_{jl} + x_{kj} - x_{li} - x_{lj} - x_{lk} \leq 2$

17 $w_{ij} + w_{il} + w_{jk} + w_{ki} - x_{il} + x_{ji} + x_{kj} + x_{li} + x_{lj} \leq 3$

$w_{ik} + w_{jk} + w_{kl} + w_{lj} - x_{ik} + x_{ji} + x_{jl} + x_{ki} + x_{kj} \leq 3$

18 $w_{ij} + 2w_{jk} + w_{lj} - x_{ik} + x_{ji} - x_{jk} + x_{jl} + x_{ki} + 2x_{kj} + x_{kl} - x_{lk} \leq 3$

$2w_{ij} + w_{jk} + w_{jl} - x_{ij} - x_{ik} - x_{il} + 2x_{ji} + x_{ki} + x_{kj} + x_{li} + x_{lj} \leq 3$

19 $w_{il} + w_{jk} + w_{lj} - x_{ik} - x_{il} - x_{jk} + x_{kj} - x_{lk} \leq 1$

$w_{ij} + w_{jk} + w_{kl} - x_{ij} - x_{ik} - x_{il} + x_{ji} - x_{kl} \leq 1$

20 $w_{ij} + w_{il} + 2w_{jk} + w_{ki} + w_{lj} + x_{ji} + x_{jl} + x_{ki} + 2x_{kj} + x_{kl} + x_{li} \leq 5$

$w_{il} + 2w_{jk} + w_{ki} + w_{kl} + w_{lj} + x_{ij} + x_{ik} + 2x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 5$

21 $2w_{il} + w_{jk} + w_{ki} + w_{lj} - x_{il} + x_{kj} + x_{li} \leq 3$

22 $w_{ij} + w_{jk} + w_{lj} - x_{ij} - x_{ik} + x_{ji} + x_{jl} + x_{ki} + x_{kj} + x_{kl} - x_{lj} - x_{lk} \leq 2$

$w_{jk} + w_{ki} + w_{kl} + x_{ij} + x_{ik} - x_{ji} - x_{jl} - x_{ki} + x_{kj} - x_{kl} + x_{lj} + x_{lk} \leq 2$

23 $w_{il} + w_{lj} - x_{ij} - x_{ik} - x_{il} - x_{lj} - x_{lk} \leq 0$

$w_{il} + w_{ji} - x_{il} - x_{ji} - x_{jl} - x_{ki} - x_{kl} \leq 0$

24 $w_{il} + w_{jk} + w_{kl} + 2w_{lj} - x_{ij} - x_{ik} - x_{il} + x_{jl} - x_{lj} \leq 2$

$2w_{jk} + w_{ki} + w_{kl} + w_{lj} - x_{ji} - x_{jk} - x_{ki} + x_{kj} - x_{li} \leq 2$

25 $w_{il} + w_{jk} + w_{ki} + w_{lj} + x_{jl} + x_{kj} + x_{kl} \leq 3$

26 $w_{ij} + w_{il} + w_{jk} + w_{lj} - x_{ij} - x_{ik} - x_{il} + x_{ji} + x_{jl} + x_{ki} + x_{kj} + x_{kl} + x_{li} \leq 3$

$w_{ji} + w_{jk} + w_{ki} + w_{lj} + x_{ij} + x_{ik} + x_{il} - x_{ji} + x_{jl} - x_{ki} + x_{kj} + x_{kl} - x_{li} \leq 3$

27 $w_{ij} + w_{il} + w_{jk} + w_{ki} + w_{lk} - x_{ij} - x_{ik} - x_{il} - x_{jk} + x_{kj} + x_{kl} - x_{lk} \leq 2$

$w_{il} + w_{ji} + w_{ki} + w_{lj} + w_{lk} - x_{ji} + x_{jl} - x_{ki} + x_{kl} - x_{li} - x_{lj} - x_{lk} \leq 2$

28 $w_{ij} + 3w_{jk} + w_{ki} + w_{kl} + w_{lj} - x_{jk} + 2x_{kj} \leq 4$

29 $w_{jk} + w_{kl} + w_{lj} + x_{kj} \leq 2$

30 $2w_{jk} + 2w_{kl} + 2w_{lj} + x_{jl} + x_{kj} + x_{lk} \leq 4$

31 $w_{ij} + 2w_{jk} + w_{ki} - x_{jk} + 2x_{kj} + x_{kl} + x_{lj} \leq 3$

32 $w_{il} + w_{kl} - x_{il} + x_{ji} + x_{jk} - x_{kl} + x_{li} + x_{lj} + x_{lk} \leq 2$

$w_{ik} + w_{il} - x_{ik} - x_{il} + x_{ji} + x_{ki} + x_{kj} + x_{li} + x_{lj} \leq 2$

33 $w_{ij} + w_{il} + w_{jl} - x_{ij} - x_{ik} - 2x_{il} + x_{ji} - x_{jl} - x_{kl} + x_{li} + x_{lj} \leq 1$

34 $w_{ij} + w_{il} + w_{jk} - x_{ij} - x_{ik} - x_{il} + x_{ji} + x_{ki} + x_{kj} + x_{li} \leq 2$

$w_{il} + w_{jk} + w_{kl} - x_{il} - x_{jl} + x_{kj} - x_{kl} + x_{li} + x_{lj} + x_{lk} \leq 2$

35 $2w_{il} - x_{ij} - x_{ik} - 2x_{il} - x_{jl} - x_{kl} + x_{li} \leq 0$

36 $w_{ij} + w_{ik} - 2x_{ij} - 2x_{ik} - x_{il} + x_{ji} + x_{ki} - x_{lj} - x_{lk} \leq 0$

$w_{jl} + w_{kl} - x_{il} - x_{ji} - 2x_{jl} - x_{ki} - 2x_{kl} + x_{lj} + x_{lk} \leq 0$

37 $w_{il} - x_{ij} - x_{ik} - x_{il} \leq 0$

$w_{ji} - x_{ji} - x_{ki} - x_{li} \leq 0$

38 $w_{ij} + w_{ik} + w_{il} - 2x_{ij} - 2x_{ik} - 2x_{il} + x_{ji} + x_{ki} + x_{li} \leq 1$

$w_{ji} + w_{ki} + w_{li} + x_{ij} + x_{ik} + x_{il} - 2x_{ji} - 2x_{ki} - 2x_{li} \leq 1$

39 $w_{ij} + 2w_{jk} + w_{kl} + x_{ji} + x_{ki} + 2x_{kj} + x_{li} + x_{lj} + x_{lk} \leq 4$

40	$w_{ij} + w_{ik} + w_{jk} + w_{kl} + w_{li} - x_{ij} - x_{ik} + x_{ji} + x_{ki} + x_{kj} \leq 3$
	$w_{il} + w_{ji} + w_{jk} + w_{ki} + w_{lj} + x_{ij} + x_{ik} - x_{ji} - x_{ki} + x_{kj} \leq 3$
41	$w_{jk} + w_{kl} + x_{kj} + x_{lj} + x_{lk} \leq 2$
42	$w_{ij} + w_{jk} + w_{jl} - x_{ij} - x_{ik} - x_{il} - x_{jk} - x_{jl} + x_{kj} + x_{lj} \leq 1$
	$w_{jk} + w_{ki} + w_{lk} - x_{ji} - x_{jk} - x_{ki} + x_{kj} + x_{kl} - x_{li} - x_{lk} \leq 1$
43	$w_{il} + w_{ki} + w_{lk} - x_{il} - x_{ki} - x_{lk} \leq 1$
44	$2w_{il} + w_{ji} + w_{ki} + w_{lj} + w_{lk} - x_{il} - x_{ji} - x_{ki} - x_{lj} - x_{lk} \leq 2$
45	$x_{ji} + x_{jk} + x_{jl} \leq 1$
	$x_{ij} + x_{kj} + x_{lj} \leq 1$
46	$w_{jk} + x_{kj} \leq 1$
47	$w_{il} + w_{kj} + w_{lk} + x_{ji} + x_{jk} + x_{jl} + x_{kl} + x_{li} \leq 3$
	$w_{ij} + w_{jk} + w_{li} + x_{il} + x_{ji} + x_{jl} + x_{kj} + x_{kl} \leq 3$
48	$w_{ik} + w_{kl} - x_{ij} - x_{ik} - x_{il} - x_{jl} - x_{kl} \leq 0$

A.2 Complete Description of P^4_{ETV}

Cl.	Facet defining inequality
1	$x_{ij} - w_{ijk} - w_{kij} \leq 0$
2	$-x_{ik} + w_ijk - w_{ilj} + w_{kij} \leq 0$
3	$\sum_{l=1,l\neq i}^{l=4} -x_{il} - w_{jki} - w_{kji} \leq -1$
	$\sum_{l=1,l\neq i}^{l=4} -x_{li} - w_{ijk} - w_{ikj} \leq -1$
4	$x_{ik} + x_{ij} + x_{jk} + w_{kji} + w_{ilk} - w_{ijk} \leq 1$
5	$-x_{ji} - x_{jl} + x_{lk} + w_{kjl} + w_{lik} + w_{lji} - w_{ljk} - w_{lkj} \leq 0$
6	$-x_{ji} - x_{jl} - x_{ki} - x_{kj} - x_{kl} + w_{kjl} + w_{lji} - w_{ljk} \leq -1$
7	$x_{ji} + x_{ki} + x_{kj} + x_{kl} + x_{li} - w_{jil} + w_{jlk} - w_{kli} - w_{lik} + w_{ljk} - w_{lki} \leq 1$

A.3 Complete Description of P^4_{ENTV}

Cl.	Facet defining inequality
1	$-w_l^{jk} \leq 0$
2	$-w_i^{ki} + w_j^{ki} \leq 0$
3	$-w_l^{jk} + w_i^{kl} - w_j^{kl} \leq 0$
4	$-w_i^{jl} + w_i^{lk} - w_j^{lk} \leq 0$
5	$-w_i^{jl} + w_k^{jl} + w_i^{lk} - w_k^{lk} \leq 0$

6	$-w_i^{jl} - w_i^{kl} + w_i^{li} - w_j^{li} \leq 0$
7	$-w_l^{jk} - w_i^{jl} + w_i^{li} - w_j^{li} \leq 0$
8	$-w_i^{jl} - w_j^{li} + w_k^{li} + w_i^{lk} - w_j^{lk} \leq 0$
9	$-w_i^{jl} + w_j^{kl} - w_l^{kl} + w_i^{li} - w_j^{li} \leq 0$
10	$-w_i^{jl} - w_i^{kl} + w_i^{li} - w_j^{li} + w_i^{lj} - w_k^{lj} \leq 0$
11	$-w_l^{jk} - w_i^{jl} + w_i^{li} - w_j^{li} + w_i^{lk} - w_j^{lk} \leq 0$
12	$w_k^{jl} + w_i^{kl} - w_j^{kl} - w_i^{li} + w_j^{li} + w_j^{lk} - w_k^{lk} \leq 0$
13	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} - w_i^{kj} - w_l^{kj} - w_j^{kl} - w_j^{li} + w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$
14	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} - w_i^{kj} - w_l^{kj} - w_j^{kl} + w_j^{li} - w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$
15	$-w_i^{jl} + w_k^{jl} - w_l^{jl} - w_j^{ki} - w_i^{kj} + w_l^{kj} - w_j^{kl} - w_j^{li} + w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$
16	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} + w_i^{kj} - w_l^{kj} + w_j^{kl} - w_j^{li} + w_k^{li} - w_j^{lj} + w_k^{lj} - w_i^{lk} \leq 0$
17	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} + w_i^{kj} - w_l^{kj} - w_j^{kl} - w_j^{li} + w_k^{li} + w_j^{lj} - w_k^{lj} - w_i^{lk} \leq 0$
18	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} + w_i^{kj} - 2w_j^{kj} + w_l^{kj} - w_j^{kl} - w_j^{li}$
	$+w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$
19	$-w_i^{jl} + w_k^{jl} - w_l^{jl} - w_j^{ki} - w_i^{kj} + w_l^{kj} - w_j^{kl} + w_j^{li} - w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$
20	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} + w_i^{kj} - w_l^{kj}$
	$-w_j^{kl} + w_j^{li} - w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk}2jw_j^{lk} \leq 0$
21	$-w_i^{jl} + w_k^{jl} - w_l^{jl} + w_j^{ki} - w_i^{kj} + w_l^{kj} + w_j^{kl} - w_j^{li} + w_k^{li} - w_j^{lj} + w_k^{lj} - w_i^{lk} \leq 0$
22	$-w_i^{jl} - w_k^{jl} + w_l^{jl} - w_j^{ki} + w_i^{kj} - 2w_j^{kj} + w_l^{kj}$
	$+w_j^{kl} - w_j^{li} + w_k^{li} - w_j^{lj} + w_k^{lj} - w_i^{lk} + 2w_j^{lk} \leq 0$
23	$w_i^{jl} - 3w_k^{jl} + w_l^{jl} - w_j^{ki} - w_i^{kj} - w_l^{kj} + w_j^{kl} - w_j^{li} + w_k^{li} - w_j^{lj} + w_k^{lj} + w_i^{lk} \leq 0$
24	$w_i^{jl} - 3w_k^{jl} + w_l^{jl} - w_j^{ki} - w_i^{kj} - w_l^{kj} - w_j^{kl} - w_j^{li} + w_k^{li} + w_j^{lj} - w_k^{lj} + w_i^{lk} \leq 0$

A.4 Complete Description of P_{STV}^4

Cl.	Facet defining inequality
1	$-x_{ik} + x_{il} - 2x_{jk} + 2x_{jl} + w_{ji} + w_{ki} - 2w_{kj} - w_{li} - w_{lj} - 2w_{lk} \leq 0$
2	$-2x_{ik} + 2x_{il} - x_{jk} + x_{jl} + w_{ji} - 2w_{ki} - w_{kj} - w_{li} + w_{lj} - 4w_{lk} \leq -1$
3	$-x_{ik} + x_{il} - 2x_{jk} + 2x_{jl} + w_{ji} + w_{ki} - 4w_{kj} - w_{li} + w_{lj} - 2w_{lk} \leq 0$
4	$-2x_{ik} - x_{il} + x_{jk} + 2x_{jl} - x_{kl} + 3w_{ji} - w_{ki} - 3w_{kj} - w_{li} - w_{lj} - w_{lk} \leq 0$
5	$-x_{ik} + x_{il} + x_{jk} - x_{jl} - 4w_{ji} + 2w_{ki} + w_{kj} + w_{li} - 4w_{lj} + 2w_{lk} \leq 1$
6	$-2x_{ik} + 2x_{il} - x_{jk} + x_{jl} + w_{ji} - 4w_{ki} - w_{kj} + w_{li} + w_{lj} - 4w_{lk} \leq -1$
7	$-x_{ik} - 3x_{il} - x_{jl} - w_{ji} - 3w_{ki} + w_{kj} + 7w_{li} - 3w_{lj} - w_{lk} \leq 0$
8	$x_{il} - x_{jk} - 2x_{jl} + x_{kl} - 3w_{ji} - w_{ki} + 3w_{kj} + 3w_{li} - 7w_{lj} + w_{lk} \leq 0$
9	$-x_{ik} \leq 0$

10	$-w_{ji} \leq 0$
11	$-w_{ji} + w_{li} - w_{lj} \leq 0$
12	$-x_{ik} - x_{il} - x_{kl} \leq -1$
13	$-x_{ik} - x_{il} - x_{kl} - w_{ji} + w_{kj} + w_{lj} \leq 0$
14	$-x_{ik} - x_{il} - 2x_{kl} + w_{kj} + w_{lj} \leq 0$
15	$-2x_{ik} + x_{jl} - w_{ji} + w_{kj} - w_{li} - w_{lk} \leq 0$
16	$-x_{ik} - x_{jl} - w_{ji} + w_{kj} + w_{li} - 2w_{lj} + w_{lk} \leq 0$
17	$-x_{ik} + w_{ji} - 3w_{ki} + w_{kj} + w_{li} - w_{lk} \leq 0$
18	$-x_{ik} - x_{il} - 2x_{kl} + w_{kj} - w_{lj} + 2w_{lk} \leq 0$
19	$-x_{ik} + x_{il} + w_{ji} - 2w_{ki} + w_{lj} - 2w_{lk} \leq 0$
20	$-2x_{ik} - x_{il} - 2x_{kl} - w_{ji} + 2w_{kj} + w_{lj} \leq 0$
21	$-2x_{ik} - x_{jk} - x_{kl} + 2w_{ki} - w_{kj} + w_{lk} \leq 0$
22	$-x_{ik} - x_{il} + x_{jk} - x_{kl} - w_{ji} + w_{ki} + w_{lj} - w_{lk} \leq 0$
23	$-x_{ik} + 2x_{jl} + w_{ji} - w_{kj} - 2w_{li} - 2w_{lk} \leq 0$
24	$-x_{ik} - x_{il} + x_{jk} + x_{jl} - 2w_{ji} + w_{ki} + w_{li} \leq 1$
25	$-2x_{ik} + x_{jl} + w_{ji} - 4w_{ki} + w_{kj} + w_{li} - w_{lk} \leq 0$
26	$-x_{jk} + x_{jl} + x_{kl} - w_{ji} - w_{ki} + 2w_{li} - 2w_{lj} - 2w_{lk} \leq 0$
27	$-x_{ik} - 2x_{il} + x_{jk} - x_{kl} - 2w_{ji} + w_{ki} + 2w_{lj} - w_{lk} \leq 0$
28	$x_{ik} - x_{il} + x_{jk} - x_{jl} - x_{kl} - w_{ki} - w_{kj} - w_{li} - w_{lj} + 2w_{lk} \leq 0$
29	$-2x_{ik} - x_{il} - 2x_{kl} + w_{ji} - 2w_{ki} + 2w_{kj} - w_{lj} + 2w_{lk} \leq 0$
30	$-x_{ik} - x_{il} + x_{jk} + x_{jl} - x_{kl} + 2w_{ji} - w_{ki} - w_{kj} - w_{li} - w_{lj} \leq 0$
31	$-x_{ik} + x_{il} - x_{kl} + 2w_{ji} - 4w_{ki} + w_{kj} + w_{lj} - 2w_{lk} \leq 0$
32	$-x_{jk} + x_{jl} + x_{kl} - w_{ji} + w_{ki} - 4w_{kj} + 2w_{lj} - 2w_{lk} \leq 0$
33	$x_{ik} - x_{jk} + x_{kl} - 2w_{ji} + w_{ki} - 4w_{kj} + 2w_{lj} - w_{lk} \leq 0$
34	$-2x_{ik} - x_{il} - w_{ji} + 4w_{ki} - 2w_{kj} - 2w_{li} + w_{lj} \leq 0$
35	$x_{ik} - x_{il} + x_{jk} - x_{jl} - x_{kl} - w_{ki} - w_{kj} + 2w_{li} + 2w_{lj} - 4w_{lk} \leq 0$
36	$x_{jl} - w_{kj} - w_{lk} \leq 0$
37	$x_{jl} - w_{ji} - w_{kj} + w_{li} - w_{lk} \leq 0$
38	$-x_{ik} - x_{jk} - x_{kl} + w_{kj} + w_{lk} \leq 0$

Appendix B

Overview of the TVP Instances

Name	N	Changes	Sol. LOP	Sol. TSP	$\frac{\text{Sol. LOP}}{\lvert\text{Sol. TSP}\rvert}$	Sol. TVP
ER_CFO_15_1	15	-	20950	-22356	0.94	-6435
LB_CFO_15_1	15	-	26150	-10093	2.59	9478
LD_CFO_15_1	15	-	131350	-15745	8.34	107620
LD_CFO_15_2	15	-	99765	-19826	5.03	71699
ER_MCO_15_1	15	$0.25N^2$	7650	-6601	1.16	-2236
ER_MCO_15_2	15	$0.25N^2$	7065	-9974	0.71	-5738
LB_MCO_15_1	15	$0.25N^2$	19260	-7452	2.58	3798
LD_MCO_15_1	15	$0.25N^2$	41850	-12441	3.36	16930
LD_BCO_15_1	15	$0.5N^2$	486045	-13029	37.30	447756
ER_BCO_15_1	15	$0.5N^2$	12735	-14706	0.87	-4986
LB_BCO_15_1	15	$0.5N^2$	25740	-11764	2.19	4206
LB_CFO_20_1	20	-	54480	-22667	2.40	16439
LB_CFO_20_2	20	-	29430	-15736	1.87	9494
LD_CFO_20_1	20	-	99505	-14584	6.82	66995
ER_CFO_20_1	20	-	15900	-15358	1.04	-7417
LB_MCO_20_1	20	$0.25N^2$	37650	-14109	2.67	14272
ER_MCO_20_1	20	$0.25N^2$	25740	-17605	1.46	905
ER_BCO_20_1	20	$0.5N^2$	27870	-26602	1.05	-7233
ER_BCO_20_2	20	$0.5N^2$	13530	-16612	0.81	-9682

Table B.1: Data for the TVP instances with 15 and 20 nodes

Name	N	Changes	Sol. LOP	Sol. TSP	$\frac{\text{Sol. LOP}}{\lvert\text{Sol. TSP}\rvert}$	Sol. TVP
ER_CFO_23_1	23	-	13920	-25960	0.54	-18453
ER_CFO_23_2	23	-	20685	-15862	1.30	-3432
LD_CFO_23_1	23	-	73745	-17357	4.25	50973
ER_CFO_23_3	23	-	23080	-23907	0.97	-8966
LD_MCO_23_1	23	$0.25N^2$	132615	-12760	10.39	95317
ER_MCO_23_1	23	$0.25N^2$	24395	-23571	1.03	-6701
ER_BCO_23_1	23	$0.5N^2$	15365	-20166	0.76	-11241
LD_BCO_23_1	23	$0.5N^2$	178500	-20495	8.71	131109
LB_CFO_26_1	26	-	55800	-22239	2.51	24774
LD_CFO_26_1	26	-	78900	-18807	4.20	43677
LB_CFO_26_2	26	-	38250	-17764	2.15	8358
LB_CFO_26_3	26	-	30600	-16642	1.84	5078
LB_MCO_26_1	26	$0.25N^2$	33180	-21419	1.55	1826
ER_MCO_26_1	26	$0.25N^2$	15090	-17553	0.86	-7639
ER_BCO_26_1	26	$0.5N^2$	20430	-19184	1.06	-5600
ER_BCO_26_2	26	$0.5N^2$	26160	-19638	1.33	-221
ER_CFO_30_1	30	-	23825	-22373	1.06	-7300
ER_CFO_30_2	30	-	16325	-20281	0.80	-11803
LB_CFO_30_1	30	-	57350	-25785	2.22	23715
LB_CFO_30_2	30	-	34150	-16988	2.01	4312
LB_MCO_30_1	30	$0.25N^2$	34300	-20399	1.68	695
ER_MCO_30_1	30	$0.25N^2$	21525	-18821	1.14	-4532
ER_MCO_30_2	30	$0.25N^2$	30625	-25365	1.21	-2314
ER_MCO_30_3	30	$0.25N^2$	19850	-22868	0.87	-8787
ER_MCO_30_4	30	$0.25N^2$	37950	-29132	1.30	-4024
LD_MCO_30_1	30	$0.25N^2$	288600	-20335	14.19	246422
LB_BCO_30_1	30	$0.5N^2$	32675	-17925	1.82	2402

Table B.2: Data for the TVP instances with 23, 26 and 30 nodes

Name	N	Changes	Sol. LOP	Sol. TSP	$\frac{\text{LOP}}{\lvert\text{TSP}\rvert}$	Sol. TVP
ER_CFO_35_1	35	-	39040	-27201	1.44	-3198
LB_CFO_35_1	35	-	30520	-19001	1.61	985
LB_CFO_35_2	35	-	31820	-20928	1.52	1932
LD_CFO_35_1	35	-	202860	-24483	8.29	163453
LB_MCO_35_1	35	$0.25N^2$	47820	-16276	2.94	16527
ER_MCO_35_1	35	$0.25N^2$	28980	-29769	0.97	-8356
LB_MCO_35_2	35	$0.25N^2$	45840	-19999	2.29	11561
LB_BCO_35_1	35	$0.5N^2$	43920	-27805	1.58	4213
ER_BCO_35_1	35	$0.5N^2$	15960	-27560	0.58	-16328
ER_BCO_35_2	35	$0.5N^2$	25080	-23565	1.06	-6224
ER_CFO_40_1	40	-	24640	-19780	1.25	-3000
ER_CFO_40_2	40	-	20370	-28990	0.70	-19420
ER_CFO_40_3	40	-	19610	-22690	0.86	-10693
ER_CFO_40_4	40	-	20940	-24753	0.85	$-14908 \ldots -13544$
ER_CFO_40_5	40	-	23060	-28657	0.80	-11827
LB_CFO_40_1	40	-	53400	-21842	2.44	21829
LD_MCO_40_1	40	$0.25N^2$	804800	-24581	32.74	743577
ER_MCO_40_1	40	$0.25N^2$	14250	-27338	0.52	-17246
LD_CFO_45_1	45	-	101880	-26172	3.89	60925
LB_CFO_45_1	45	-	70640	-28627	2.47	$19877 \ldots 22929$
LB_CFO_45_2	45	-	45720	-28403	1.61	$-1338 \ldots 3071$
LD_CFO_45_2	45	-	179780	-23890	7.53	$133647 \ldots 136410$
LB_MCO_45_1	45	$0.25N^2$	72040	-27243	2.64	$23301 \ldots 24430$
LB_MCO_45_2	45	$0.25N^2$	42640	-22365	1.91	8728
LB_MCO_45_3	45	$0.25N^2$	78560	-26879	2.92	35339
LD_BCO_45_1	45	$0.5N^2$	1140500	-28684	39.76	1070779
LB_BCO_45_1	45	$0.5N^2$	44300	-22365	1.98	3381
ER_BCO_45_1	45	$0.5N^2$	23720	-28058	0.85	9418
LB_CFO_50_1	50	-	67630	-32378	2.09	$19559 \ldots 23220$
LB_CFO_50_2	50	-	81100	-31171	2.60	$34783 \ldots 38036$
ER_CFO_50_1	50	-	39260	-27962	1.40	$-1276 \ldots 457.9$

Table B.3: Data for the TVP instances with 35–50 nodes

Appendix C

Bounds of the LP Relaxation of the Different TVP Models

Name	Opt.	TVP_{HP}	TVP_E	TVP_{EN}	TVP_D
ER_CFO_15_1	−6435	94.0	−6406.1	2370.0	7920.5
LB_CFO_15_1	9478	19219.0	13388.4	18035.3	20481.3
LD_CFO_15_1	107620	115361.5	111246.2	112091.8	124233.8
LD_CFO_15_2	71699	79834.9	77979.0	79092.3	89138.3
ER_MCO_15_1	−2236	1242.0	−1600.5	1005.5	4714.7
ER_MCO_15_2	−5738	−3099.8	−4668.6	1632.5	3256.3
LB_MCO_15_1	3798	10007.6	3861.1	7314.1	14929.6
LD_MCO_15_1	16930	28378.0	23240.3	26060.5	35965.2
LD_BCO_15_1	447756	459466.5	448043.5	450031.5	479910.8
ER_BCO_15_1	−4986	4570.0	−705.9	2984.04	10051.4
LB_BCO_15_1	4206	15699.0	8193.0	12561.8	21535.6
LB_CFO_20_1	16439	28408.1	24709.0	26836.2	42453.5
LB_CFO_20_2	9494	13743.5	9496.8	11848.8	21376.0
LD_CFO_20_1	66995	82176.7	73678.0	78910.3	87228.0
ER_CFO_20_1	−7417	2699.5	103.0	2985.8	11652.4
LB_MCO_20_1	14272	22189.7	17660.7	20036.2	34352.1
ER_MCO_20_1	905	8018.3	1208.5	7226.6	13713.7
ER_BCO_20_1	−7233	5600.3	−3452.0	5983.4	17866.9
ER_BCO_20_2	−9682	−3751.5	−6640.3	−4844.8	8190.0

Table C.1: Solution of the LP relaxation for the different TVP models for instances with 15 and 20 nodes

Name	Opt.	TVP_{HP}	TVP_E	TVP_{EN}	TVP_D
ER_CFO_23_1	-18453	-4124.8	-15426.9	-3465.5	3798.3
ER_CFO_23_2	-3432	2748.8	-804.2	947.3	16201.7
LD_CFO_23_1	50973	56883.8	52145.0	53444.1	66679.7
ER_CFO_23_3	-8966	1560.1	-6751.9	1998.7	15666.5
LD_MCO_23_1	95317	113424.4	101934.2	107364.7	132645.6
ER_MCO_23_1	-6701	8917.1	-1323.7	6203.3	21310.5
ER_BCO_23_1	-11241	-4840.9	-7972.3	-3440.2	8238.5
LD_BCO_23_1	131109	154173.0	138900.8	144670.8	180248.8
LB_CFO_26_1	24774	34948.2	27488.8	32144.9	Mem.
LD_CFO_26_1	43677	58741.3	51832.4	58201.9	Mem.
LB_CFO_26_2	8358	18201.3	12758.9	17323.6	Mem.
LB_CFO_26_3	5078	14713.5	9308.2	13041.3	Mem.
LB_MCO_26_1	1826	14497.2	9157.0	11400.4	Mem.
ER_MCO_26_1	-7639	1331.9	-3325.3	1122.7	Mem.
ER_BCO_26_1	-5600	4381.2	-4187.9	1875.8	Mem.
ER_BCO_26_2	-221	11930.3	1851.6	9840.9	Mem.

Table C.2: Solution of the LP relaxation for the different TVP models for instances with 23 and 26 nodes

Appendix D

Results of the Heuristic Approaches

D.1 Results of the Constructive Heuristics

In the following Becker X means that the number of randomly generated start tours is equal to $N \cdot X$ and BestInst X that the number of inserting operation is equal to $N \cdot X$.

Name	Opt.	Heu.
ER_CFO_15_1	-6435	-18018
LB_CFO_15_1	9478	3058
LD_CFO_15_1	107620	83077
LD_CFO_15_2	71699	39003
ER_MCO_15_1	-2236	-15660
ER_MCO_15_2	-5738	-20105
LB_MCO_15_1	3798	-14523
LD_MCO_15_1	16930	-7811
LD_BCO_15_1	447756	178897
ER_BCO_15_1	-4986	-11065
LB_BCO_15_1	4206	-4972

Name	Opt.	Heu.
LB_CFO_20_1	16439	-4248
LB_CFO_20_2	9494	-7772
LD_CFO_20_1	66995	49519
ER_CFO_20_1	-7417	-22562
LB_MCO_20_1	14272	3824
ER_MCO_20_1	905	-15623
ER_BCO_20_1	-7233	-24542
ER_BCO_20_2	-9682	-32470

Table D.1: Results of the nearest-neighbor heuristic for instances with 15 and 20 nodes

Name	Opt.	Heu.	Name	Opt.	Heu.
ER_CFO_23_1	−18453	−37488	LB_CFO_26_1	24774	−11417
ER_CFO_23_2	−3432	−23480	LD_CFO_26_1	43677	19199
LD_CFO_23_1	50973	12899	LB_CFO_26_2	8358	−15172
ER_CFO_23_3	−8966	−24253	LB_CFO_26_3	5078	−9415
LD_MCO_23_1	95317	22429	LB_MCO_26_1	1826	−8410
ER_MCO_23_1	−6701	−23296	ER_MCO_26_1	−7639	−16626
ER_BCO_23_1	−11241	−42656	ER_BCO_26_1	−5600	−29368
LD_BCO_23_1	131109	25885	ER_BCO_26_2	−221	−12427

Table D.2: Results of the nearest-neighbor heuristic for instances with 23 and 26 nodes

Name	Opt.	Becker 5	Becker 10	Becker 50	Becker 100
ER_CFO_15_1	−6435	−56016.4	−51525.3	−43891.2	−41057.9
LB_CFO_15_1	9478	−24108.4	−21690.3	−15462.1	−13299.5
LD_CFO_15_1	107620	62813.7	66293.1	73350	76317.2
LD_CFO_15_2	71699	22264.6	26373.3	34261	37230.8
ER_MCO_15_1	−2236	−38065	−35732	−29948.4	−27854.9
ER_MCO_15_2	−5738	−42167.5	−39176.4	−33443.7	−31376.9
LB_MCO_15_1	3798	−40911.8	−36847.6	−29951.4	−27658.9
LD_MCO_15_1	16930	−12842.8	−10353.3	−5406.5	−3596.8
LD_BCO_15_1	447756	354360	368370	391292	399016
ER_BCO_15_1	−4986	−54431.1	−50511.3	−41717.3	−37948.1
LB_BCO_15_1	4206	−36683.8	−33525.5	−27003.2	−24385.1

Table D.3: Average values of 500 runs of the Becker Heuristic for the instances with 15 nodes

Name	Opt.	Becker 5	Becker 10	Becker 50	Becker 100
LB_CFO_20_1	16439	−55183.5	−51936.2	−43132	−40735.8
LB_CFO_20_2	9494	−47959.3	−45174.1	−38689.6	−36676.4
LD_CFO_20_1	66995	−18787.4	−14196.2	−4719.9	−1700
ER_CFO_20_1	−7417	−67165.6	−63845.7	−55929.1	−53213.8
LB_MCO_20_1	14272	−57472.5	−53874.3	−46023.3	−42689.4
ER_MCO_20_1	905	−80211.2	−75980.6	−67835.9	−64474
ER_BCO_20_1	−7233	−77409.3	−72889	−64568.7	−61308.3
ER_BCO_20_2	-9682	−84002.1	−79334.6	−70000.3	−66006.4
ER_CFO_23_1	−18453	−113246	−108282	−98466.8	−95444.7
ER_CFO_23_2	−3432	−95460.6	−91093	−82111.6	−78665
LD_CFO_23_1	50973	−41275.2	−36750.8	−28312	−25108.7
ER_CFO_23_3	−8966	−84309.9	−80254	−73107.5	−70508.3
LD_MCO_23_1	95317	−11553.2	−7107.8	3454.5	7721
ER_MCO_23_1	−6701	−100073	−94890.4	−86052.5	−81807.9
ER_BCO_23_1	−11241	−96514.3	−91063.5	−82706.4	−79401.1
LD_BCO_23_1	131109	23831.1	28485.4	38362.4	42603.1
LB_CFO_26_1	24774	−88994.7	−84232.3	−74337.4	−71564.6
LD_CFO_26_1	43677	−53281	−48907.6	−40947.9	−37574.8
LB_CFO_26_2	8358	−94448.2	89214.4	−79327.5	−75422.2
LB_CFO_26_3	5078	−90712.6	−86015.9	−77108.2	−73229.8
LB_MCO_26_1	1826	−103233	−98459.8	−89568.5	−85993.3
ER_MCO_26_1	−7639	−102803	−98618	−90946.2	−87268.9
ER_BCO_26_1	−5600	−105033	100146	−92067.9	−88417.5
ER_BCO_26_2	−221	−88230	−84926.8	−77440.4	−74528.2

Table D.4: Average values of 500 runs of the Becker Heuristic for the instances with 20, 23 and 26 nodes

Name	Opt.	BestIns 5	BestIns 10	BestIns 50	BestIns 100
ER_CFO_15_1	−6435	−6455.7	**−6435**	**−6435**	**−6435**
LB_CFO_15_1	9478	**9478**	**9478**	**9478**	**9478**
LD_CFO_15_1	107620	105370	106329	107595	**107620**
LD_CFO_15_2	71699	71432.4	71659.5	**71699**	**71699**
ER_MCO_15_1	−2236	−2240.2	**−2236**	**−2236**	**−2236**
ER_MCO_15_2	−5738	**−5738**	**−5738**	**−5738**	**−5738**
LB_MCO_15_1	3798	**3798**	**3798**	**3798**	**3798**
LD_MCO_15_1	16930	16836.3	16922.6	**16930**	**16930**
LD_BCO_15_1	447756	447656	447738	**447756**	**447756**
ER_BCO_15_1	−4986	**−4986**	**−4986**	**−4986**	**−4986**
LB_BCO_15_1	4206	3914.4	4086.4	**4206**	**4206**
LB_CFO_20_1	16439	16060.3	16277.4	**16439**	**16439**
LB_CFO_20_2	9494	9487.6	9493.36	**9494**	**9494**
LD_CFO_20_1	66995	66178.2	66896	66938.7	66994.6
ER_CFO_20_1	−7417	−7608.4	−7545.8	−7488.8	−7417.7
LB_MCO_20_1	14272	12286.3	12656.9	13960.8	14206.4
ER_MCO_20_1	905	902.7	**905**	**905**	**905**
ER_BCO_20_1	−7233	−7588	−7391.1	−7250.18	**−7233**
ER_BCO_20_2	−9682	−9682.8	−9682.8	**−9682**	**−9682**
ER_CFO_23_1	−18453	−18580.7	−18508.1	−18453.7	**−18453**
ER_CFO_23_2	−3432	−4101	−3922	−3553	−3445.5
LD_CFO_23_1	50973	**50973**	**50973**	**50973**	**50973**
ER_CFO_23_3	−8966	−9188.4	−9051.3	−8966.3	**−8966**
LD_MCO_23_1	95317	95276.3	95312.2	**95317**	**95317**
ER_MCO_23_1	−6701	−6730.4	−6704	**−6701**	**−6701**
ER_BCO_23_1	−11241	−11377.2	−11276.8	**−11241**	**−11241**
LD_BCO_23_1	131109	126210	**131109**	**131109**	**131109**

Table D.5: Average values of 500 runs of the best-insertion heuristic for instances with 15, 20 and 23 nodes

Name	Opt.	BestIns 5	BestIns 10	BestIns 50	BestIns 100
LB_CFO_26_1	24774	24161.8	24505	24718.9	24756.9
LD_CFO_26_1	43677	42133	42985.3	43671.1	**43677**
LB_CFO_26_2	8358	7703.7	7910.9	8077.6	8167.1
LB_CFO_26_3	5078	3876.5	4536.4	5039.5	5066.2
LB_MCO_26_1	1826	203.6	849.8	1605	1762
ER_MCO_26_1	−7639	−9039.3	−8876.7	−8104.5	−8133
ER_BCO_26_1	−5600	−5876.8	−5673.8	**−5600**	**−5600**
ER_BCO_26_2	−221	−560.1	−341.5	−223.5	**−221**

Table D.6: Average values of 500 runs of the best-insertion heuristic for instances with 26 nodes

D.2 Results of the Improvement Heuristics

	ER_CFO_15_1	LB_CFO_15_1	LD_CFO_15_1	LD_CFO_15_2
Opt	−6435	9478	107620	71699
KernLin 1	**−6435**	**9478**	**107620**	**71699**
KernLin 2	**−6435**	**9478**	**107620**	**71699**
2-Opt 100	**−6435**	**9478**	**107620**	**71699**
2-Opt 1000	**−6435**	**9478**	**107620**	**71699**
3-Opt 100	**−6435**	**9478**	**107620**	**71699**
3-Opt 500	**−6435**	**9478**	**107620**	**71699**
3-Opt 1000	−6439.2	**9478**	**107620**	**71699**

	ER_MCO_15_1	ER_MCO_15_2	LB_MCO_15_1	LD_MCO_15_1
Opt	−2236	-5738	3798	16930
KernLin 1	−2240.1	**5738**	**3798**	**16930**
KernLin 2	**−2236**	**5738**	**3798**	**16930**
2-Opt 100	−3990.3	**5738**	**3798**	**16930**
2-Opt 1000	−4014.2	**5738**	**3798**	**16930**
3-Opt 100	−3976.3	**5738**	**3798**	**16930**
3-Opt 500	−3993.4	**5738**	**3798**	**16930**
3-Opt 1000	−4000.3	**5738**	**3798**	**16930**

	LD_BCO_15_1	ER_BCO_15_1	LB_BCO_15_1
Opt	447756	−4986	4206
KernLin 1	**447756**	**−4986**	**4206**
KernLin 2	**447756**	**−4986**	**4206**
2-Opt 100	447018	−4987	4090.9
2-Opt 1000	446905	**−4986**	4113.2
3-Opt 100	**447756**	**−4986**	4072.4
3-Opt 500	**447756**	−4987	4065
3-Opt 1000	**447756**	**−4986**	4090.9

Table D.7: Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 15 nodes

	LB_CFO_20_1	LB_CFO_20_2	LD_CFO_20_1	ER_CFO_20_1
Opt	16439	9494	66995	−7417
KernLin 1	16409.7	**9494**	**66995**	- −7441.9
KernLin 2	**16439**	**9494**	**66995**	**−7417**
2-Opt 100	15973.1	**9494**	66901.2	**−7417**
2-Opt 1000	16036.3	**9494**	66917.4	**−7417**
3-Opt 100	16428	**9494**	66994.6	**−7417**
3-Opt 500	16427.7	**9494**	66994.2	**−7417**
3-Opt 1000	16427.4	**9494**	66994.6	**−7417**

	LB_MCO_20_1	ER_MCO_20_1	ER_BCO_20_1	ER_BCO_20_2
Opt	14272	905	−7233	−9682
KernLin 1	14263.1	**905**	−7261.8	**−9682**
KernLin 2	**14272**	**905**	**−7233**	**−9682**
2-Opt 100	13786.8	**905**	−7238.4	−9685.9
2-Opt 1000	13742.9	**905**	−7238.6	−9686.4
3-Opt 100	13801.4	**905**	**−7233**	−9685.7
3-Opt 500	13813.2	**905**	**−7233**	−9684.8
3-Opt 1000	13819.1	**905**	**−7233**	−9685.2

	ER_CFO_23_1	ER_CFO_23_2	LD_CFO_23_1	ER_CFO_23_3
Opt	−18453	−3432	50973	−8966
KernLin 1	−18978.5	−3794.5	**50973**	−9549.1
KernLin 2	**−18453**	**−3432**	**50973**	**−8966**
2-Opt 100	−18749.8	−3729.1	**50973**	**−8966**
2-Opt 1000	−18749.5	−3736.8	**50973**	**−8966**
3-Opt 100	−18735.8	3473.3	**50973**	**−8966**
3-Opt 500	−18737.4	−3455.5	**50973**	**−8966**
3-Opt 1000	−18733.1	−3480.6	**50973**	**−8966**

Table D.8: Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 20 and 23 nodes

	LD_MCO_23_1	ER_MCO_23_1	ER_BCO_23_1	LD_BCO_23_1
Opt	95317	−6701	−11241	131109
KernLin 1	95228	−6870.2	−11669.7	129805
KernLin 2	**95317**	**−6701**	**−11241**	**131109**
2-Opt 100	94910.8	**−6701**	**−11241**	128250
2-Opt 1000	94907.6	**−6701**	**−11241**	128457
3-Opt 100	95300.8	**−6701**	**−11241**	131047
3-Opt 500	95308.5	**−6701**	**−11241**	131028
3-Opt 1000	95285.5	**−6701**	**−11241**	131044

	LB_CFO_26_1	LD_CFO_26_1	LB_CFO_26_2	LB_CFO_26_3
Opt	24774	43677	8358	5078
KernLin 1	23856.5	43221.6	7599.1	4709.4
KernLin 2	**24774**	**43677**	**8358**	**5078**
2-Opt 100	**24774**	42750.8	7976.3	4801.7
2-Opt 1000	**24774**	42789.5	7989.7	4785.9
3-Opt 100	**24774**	43648.4	8302	4870.4
3-Opt 500	**24774**	43651.4	8312	4870.7
3-Opt 1000	**24774**	43656.1	8310.3	4878.6

	LB_MCO_26_1	ER_MCO_26_1	ER_BCO_26_1	ER_BCO_26_2
Opt	1826	−7639	−5600	−221
KernLin 1	1126.1	−8416.8	−7572.8	- −1085.3
KernLin 2	**1826**	**−7639**	**−5600**	−225.2
2-Opt 100	1238	−8336.5	−5709.8	−339.2
2-Opt 1000	1289.7	−8337	−5791.4	−326.2
3-Opt 100	1768	−8351.4	−5602.3	−291.6
3-Opt 500	1767	−8344.1	**−5600**	−297.2
3-Opt 1000	1768.6	−8359	**−5600**	−289.5

Table D.9: Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 23 and 26 nodes

	LD_CFO_20_1	ER_CFO_20_1	LB_MCO_20_1	ER_CFO_23_2
Opt	66995	−7417	14272	−3432
KernLin 1	**66995**	**−7417**	14233.5	−3449.7
KernLin 2	**66995**	**−7417**	**14272**	**−3432**
2-Opt 100	**66995**	**−7417**	14248.3	−3435.3
2-Opt 1000	**66995**	**−7417**	14233.5	**−3432**
3-Opt 100	**66995**	**−7417**	**14272**	**−3432**
3-Opt 500	**66995**	**−7417**	**14272**	**−3432**
3-Opt 1000	**66995**	**−7417**	**14272**	**−3432**

	LB_CFO_26_1	LB_CFO_26_2	LB_CFO_26_3
Opt	24774	8358	5078
KernLin 1	**24774**	8318.8	5077.7
KernLin 2	**24774**	**8358**	**5078**
2-Opt 100	**24774**	8291.1	5076.2
2-Opt 1000	**24774**	8269.9	5076.3
3-Opt 100	**24774**	**8358**	5076.3
3-Opt 500	**24774**	**8358**	5076.6
3-Opt 1000	**24774**	**8358**	5076.2

	LB_MCO_26_1	ER_MCO_26_1
Opt	1826	−7639
KernLin 1	1810	−7945.9
KernLin 2	**1826**	−7729.7
2-Opt 100	1823.6	−7747.4
2-Opt 1000	1817.4	−7747.0
3-Opt 100	**1826**	−7741.3
3-Opt 500	**1826**	−7735.36
3-Opt 1000	**1826**	−7741.25

Table D.10: Average values of 500 runs of the improvement heuristics with best-insertion 100 as start heuristic (We left out instances where already the average value of the best-insertion heuristic is equal the optimal solution)

	ER_CFO_15_1	LB_CFO_15_1	LD_CFO_15_1	LD_CFO_15_2
Opt	−6435	9478	107620	71699
KernLin 1	−16342	7069	100258	39003
KernLin 2	−16342	**9478**	101910	**71699**
2-Opt 100	−16230	5891	99749	39003
2-Opt 1000	−16230	5891	99749	39003
3-Opt 100	−16230	5891	100840	50999
3-Opt 500	−16230	5891	100840	50999
3-Opt 1000	−16230	5891	100840	50999

	ER_MCO_15_1	ER_MCO_15_2	LB_MCO_15_1	LD_MCO_15_1
Opt	−2236	−5738	3798	16930
KernLin 1	−5938	−13890	−12286	4821
KernLin 2	−5655	−15378	**3798**	15583
2-Opt 100	−5938	−18189	−11035	−4522
2-Opt 1000	−5938	−18189	−11035	−4522
3-Opt 100	−5891	−18189	−11035	−3919
3-Opt 500	−5891	−18189	−11035	−3919
3-Opt 1000	−5891	−18189	−11035	−3919

	LD_BCO_15_1	ER_BCO_15_1	LB_BCO_15_1
Opt	447756	−4986	4206
KernLin 1	304661	−6462	2461
KernLin 2	444637	**−4986**	2461
2-Opt 100	183418	−7743	−3272
2-Opt 1000	183418	−7743	−3272
3-Opt 100	183529	−7743	−3272
3-Opt 500	183529	−7743	−3272
3-Opt 1000	183529	−7743	−3272

Table D.11: Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 15 nodes

	LB_CFO_20_1	LB_CFO_20_2	LD_CFO_20_1	ER_CFO_20_1
Opt	16439	9494	66995	−7417
KernLin 1	5452	−4039	53922	−12681
KernLin 2	12411	−2509	**66995**	−11695
2-Opt 100	2604	−5461	53779	−9516
2-Opt 1000	2604	−5461	53779	−9516
3-Opt 100	6922	−2865	53779	−9516
3-Opt 500	6922	−2865	53779	−9516
3-Opt 1000	6922	−2865	53779	−9516

	LB_MCO_20_1	ER_MCO_20_1	ER_BCO_20_1	ER_BCO_20_2
Opt	14272	905	−7233	−9682
KernLin 1	3824	−6405	−13342	−31259
KernLin 2	10643	−6405	−11043	−16600
2-Opt 100	3824	−9299	−19738	−17944
2-Opt 1000	3824	−9299	−15057	−17944
3-Opt 100	3824	−9299	−15057	−17944
3-Opt 500	3824	−9299	−15057	−17944
3-Opt 1000	3824	−9299	−15057	−17944

	ER_CFO_23_1	ER_CFO_23_2	LD_CFO_23_1	ER_CFO_23_3
Opt	−18453	−3432	50973	−8966
KernLin 1	−31431	−19311	27938	−22371
KernLin 2	−21957	−9031	35165	−19186
2-Opt 100	−28689	−16013	40567	−22958
2-Opt 1000	−28689	−16013	40567	−22958
3-Opt 100	−27106	−7842	39666	−18957
3-Opt 500	−27106	−7842	39666	−18957
3-Opt 1000	−27106	−7842	39666	−18957

Table D.12: Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 20 and 23 nodes

	LD_MCO_23_1	ER_MCO_23_1	ER_BCO_23_1	LD_BCO_23_1
Opt	95317	−6701	−11241	131109
KernLin 1	58389	−16390	−27325	87234
KernLin 2	85150	−10586	**−11241**	122086
2-Opt 100	28222	−15234	−23376	42461
2-Opt 1000	28222	−15234	−23376	42461
3-Opt 100	32866	−14639	−22176	42978
3-Opt 500	32866	−14639	−22176	42978
3-Opt 1000	32866	−14639	−22176	42978

	LB_CFO_26_1	LD_CFO_26_1	LB_CFO_26_2	LB_CFO_26_3
Opt	24774	43677	8358	5078
KernLin 1	2594	24540	−9457	−6415
KernLin 2	24455	41815	59	−683
2-Opt 100	4973	25695	−7342	−5711
2-Opt 1000	4973	25695	−7342	−5711
3-Opt 100	11549	31616	2128	718
3-Opt 500	11549	31616	2128	718
3-Opt 1000	11549	31616	2128	718

	LB_MCO_26_1	ER_MCO_26_1	ER_BCO_26_1	ER_BCO_26_2
Opt	1826	−7639	−5600	−221
KernLin 1	−5739	−12456	−23235	−8355
KernLin 2	−3673	−12456	−14090	−3888
2-Opt 100	−3946	−12456	−17695	−9200
2-Opt 1000	−3946	−12456	−17695	−9200
3-Opt 100	−3946	−12456	−22390	−8901
3-Opt 500	−3946	−12456	−22390	−8901
3-Opt 1000	−3946	−12456	−22390	−8901

Table D.13: Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 23 and 26 nodes

Name	Opt.	Becker 100 +Kl 2	BestIns 100 +KL 2
ER_CFO_30_1	−7300	**−7300**	**−7300**
ER_CFO_30_2	−11803	−11901.4	**−11803**
LB_CFO_30_1	23715	**23715**	**23715**
LB_CFO_30_2	4312	**4312**	**4312**
LB_MCO_30_1	695	**695**	**695**
ER_MCO_30_1	−4532	**−4532**	**−4532**
ER_MCO_30_2	−2314	−2321.5	−2327.1
ER_MCO_30_3	−8787	−8800.9	**−8787**
ER_MCO_30_4	−4024	**−4024**	**−4024**
LD_MCO_30_1	246422	**246422**	**246422**
LB_BCO_30_1	2402	2399.3	**2402**
ER_CFO_35_1	−3198	**−3198**	−3235.0
LB_CFO_35_1	985	**985**	**985**
LB_CFO_35_2	1932	1920.3	**1932**
LD_CFO_35_1	163453	**163453**	**163453**
LB_MCO_35_1	16527	**16527**	16526.8
ER_MCO_35_1	−8356	−8607.2	−8356.1
LB_MCO_35_2	11561	11558	**115610**
LB_BCO_35_1	4213	4062.62	**42130**
ER_BCO_35_1	−16328	−16568.7	−16329
ER_BCO_35_2	−6224	−6288.1	**−6224**
ER_CFO_40_1	−3000	**−3000**	**−3000**
ER_CFO_40_2	−19420	−19442.7	−19455.8
ER_CFO_40_3	−10693	−11171.7	−10809.2
ER_CFO_40_4	−14908 . . . 10662	−14954.5	−14908.3
ER_CFO_40_5	−11827	−12676.9	−11827.5
LB_CFO_40_1	21829	21816.3	21803.1
LD_MCO_40_1	743577	**743577**	**743577**
ER_MCO_40_1	−17246	−18146.1	**−17246**

Table D.14: Average values of 500 runs of the Kernighan-Lin 2 heuristic with best-insertion 100 and Becker 100 as start heuristic for the instances with 30, 35 and 40 nodes

Name	Opt.	Becker 100 +Kl 2	BestIns 100 +KL 2
LD_CFO_45_1	60925	60191.4	**60925**
LB_CFO_45_1	19877 . . . 22929	19826.8	19877
LB_CFO_45_2	1082 . . . 3071	92	1082
LD_CFO_45_2	133647. . . 136410	133320	133507
LB_MCO_45_1	23301 . . . 24430	22987.4	23270.6
LB_MCO_45_2	8728	8084.3	**8728**
LB_MCO_45_3	35339	35287.6	35335.9
LD_BCO_45_1	1070779	1070670	**1070779**
LB_BCO_45_1	3381	2756.8	**3381**
ER_BCO_45_1	−9418	−9500.7	**−9418**
LB_CFO_50_1	19559 . . . 23220	17657.4	19415.4
LB_CFO_50_2	34783 . . . 38036	34779.9	34780.3
ER_CFO_50_1	−1276 . . . −457	−2517.3	−1284.1

Table D.15: Average values of 500 runs of the Kernighan-Lin-2 heuristic with best-insertion 100 and Becker 100 as start heuristics for the instances with 45 and 50 nodes

Appendix E

Results of the Branch-and-Bound Approach

In the following tables we state the results of the hybrid dynamic programming/ branch-and-bound algorithm. Here BBS X means that the branch-and-bound starts at partial paths where $X - 1$ targets have been visited already.

Name	BBS 1	BBS 2	BBS 3	BBS 4	BBS 5	BBS 6	BBS 7
ER_CFO_15_1	0.500	**0.358**	0.476	0.448	0.413	0.544	0.817
LB_CFO_15_1	**0.276**	0.279	0.296	0.312	0.354	0.501	0.783
LD_CFO_15_1	0.228	0.226	**0.181**	0.238	0.242	0.456	0.755
LD_CFO_15_2	0.185	0.178	**0.173**	0.200	0.260	0.429	0.743
ER_MCO_15_1	0.069	**0.064**	0.065	0.102	0.159	0.354	0.704
ER_MCO_15_2	0.302	**0.245**	0.263	0.292	0.300	0.471	0.789
LB_MCO_15_1	**0.135**	0.180	0.185	0.177	0.209	0.395	0.715
LD_MCO_15_1	1.367	1.343	1.286	1.372	1.280	**1.166**	1.208
LD_BCO_15_1	0.030	**0.028**	0.032	0.056	0.137	0.341	0.696
ER_BCO_15_1	0.211	0.047	**0.036**	0.064	0.172	0.360	0.704
LB_BCO_15_1	0.412	0.412	**0.395**	0.465	0.481	0.610	0.871

Table E.1: Computation times (in seconds) of the hybrid DP/branch-and-bound algorithm for the instances with 15 nodes

	LB_CFO_20_1	LB_CFO_20_2	LD_CFO_20_1	ER_CFO_20_1
BBS 1	00:02:25	00:00:02	00:00:33	00:00:33
BBS 2	00:02:04	00:00:02	00:00:23	00:00:33
BBS 3	00:02:15	**00:00:01**	00:00:19	00:00:34
BBS 4	00:02:02	00:00:03	00:00:19	00:00:33
BBS 5	00:02:07	00:00:02	**00:00:18**	00:00:31
BBS 6	00:01:40	00:00:05	00:00:19	**00:00:30**
BBS 7	00:01:22	00:00:10	00:00:21	**00:00:30**
BBS 8	**00:01:11**	00:00:22	00:00:30	00:00:37
BBS 9	**00:01:11**	00:00:40	00:00:47	00:00:51
BBS 10	00:01:22	00:01:06	00:01:09	00:01:11
BBS 11	00:01:37	00:01:30	00:01:31	00:01:32
BBS 12	00:01:51	00:01:49	00:01:48	00:01:50

	LB_MCO_20_1	ER_MCO_20_1	ER_BCO_20_1	ER_BCO_20_2
BBS 1	**00:00:07**	00:00:51	00:03:07	00:00:04
BBS 2	**00:00:07**	00:00:43	00:02:27	00:00:04
BBS 3	**00:00:07**	00:00:28	00:03:17	00:00:04
BBS 4	00:00:08	00:00:43	00:01:59	**00:00:03**
BBS 5	00:00:09	**00:00:06**	00:01:37	00:00:04
BBS 6	00:00:09	00:00:08	00:01:17	00:00:05
BBS 7	00:00:14	00:00:12	00:01:04	00:00:11
BBS 8	00:00:24	00:00:23	**00:00:59**	00:00:22
BBS 9	00:00:42	00:00:41	00:01:03	00:00:41
BBS 10	00:01:06	00:01:06	00:01:18	00:01:05
BBS 11	00:01:29	00:01:29	00:01:35	00:01:29
BBS 12	00:01:48	00:01:47	00:01:51	00:01:47

Table E.2: Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 20 nodes

	ER_CFO_23_1	ER_CFO_23_2	LD_CFO_23_1	ER_CFO_23_3
BBS 1	00:05:05	00:01:30	00:01:41	00:05:46
BBS 2	00:04:31	00:01:36	00:00:52	00:05:52
BBS 3	00:04:03	00:01:38	00:00:13	00:06:04
BBS 4	00:04:40	00:01.41	**00:00:11**	00:08:30
BBS 5	00:03:59	00:01:30	00:00:46	00:06:13
BBS 6	**00:03:18**	**00:01:25**	00:00:28	00:06:28
BBS 7	00:03:19	00:01:34	00:00:39	**00:05:21**
BBS 8	00:03:51	00:02:17	00:01:30	00:05:24
BBS 9	00:05:20	00:03:56	00:03:17	00:06:10
BBS 10	00:07:53	00:06:49	00:06:25	00:08:33
BBS 11	00:13:16	00:11:26	00:10:37	00:11:58
BBS 12	00:17:09	00:15:33	00:15:31	00:16:08

	LD_MCO_23_1	ER_MCO_23_1	ER_BCO_23_1	LD_BCO_23_1
BBS 1	00:06:23	00:13:57	**00:01:54**	00:35:35
BBS 2	00:07:44	00:14:55	00:01:56	00:30:54
BBS 3	00:06:28	00:11:41	00:02:00	00:34:25
BBS 4	00:06:40	00:10:34	00:02:28	00:37:10
BBS 5	00:05:56	00:09:40	00:02:16	00:29:19
BBS 6	00:04:41	00:08:23	00:02:03	00:21:21
BBS 7	00:03:45	00:07:38	00:02:22	00:15:50
BBS 8	**00:03:37**	**00:06:53**	00:02:57	00:12:33
BBS 9	00:04:37	00:07:32	00:04:30	00:10:48
BBS 10	00:07:06	00:09:18	00:07:19	**00:10:30**
BBS 11	00:10:50	00:12:32	00:11:13	00:12:38
BBS 12	00:15:09	00:16:22	00:15:43	00:16:18

Table E.3: Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 23 nodes

	LB_CFO_26_1	LD_CFO_26_1	LB_CFO_26_2	LB_CFO_26_3
BBS 1	00:16:13	08:49:11	05:20:11	00:32:39
BBS 2	00:21:22	07:38:28	04:21:25	00:34:11
BBS 3	**00:10:20**	07:02:02	04:21:39	00:31:55
BBS 4	00:11:19	07:02:27	04:29:26	00:36:02
BBS 5	00:11:04	06:24:51	04:12:27	00:32:00
BBS 6	00:13:07	05:37:12	03:37:34	00:27:35
BBS 7	00:10:49	04:47:50	03:09:25	**00:24:11**
BBS 8	00:12:57	03:53:56	02:45:38	00:25:14
BBS 9	00:19:47	03:09:50	02:24:41	00:34:17
BBS 10	00:36:46	**02:40:29**	**02:19:49**	01:00:41

	LB_MCO_26_1	ER_MCO_26_1	ER_BCO_26_1	ER_BCO_26_2
BBS 1	02:46:00	01:50:07	00:51:30	03:32:29
BBS 2	02:20:26	01:46:54	00:43:13	03:39:11
BBS 3	01:55:44	01:49:21	00:37:06	03:04:47
BBS 4	01:27:09	00:40:55	00:43:23	03:22:06
BBS 5	01:56:00	00:39:15	00:42:02	02:52:25
BBS 6	01:26:39	00:41:58	00:34:43	02:36:24
BBS 7	01:06:33	00:32:24	00:31:41	01:48:07
BBS 8	00:55:15	**00:29:48**	**00:30:19**	01:24:35
BBS 9	**00:50:16**	00:35:42	00:36:00	**01:20:07**
BBS 10	01:11:20	00:50:08	00:52:33	01:22:38

Table E.4: Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 26 nodes

Appendix F

Results of the Branch-and-Cut Approach

F.1 Test of different cutting planes

In the following tables we state the results of the tests for using different facet classes as cutting planes. Here w.c. means TVP_{HP} model without cuts, TX = TVP_{HP} model with facet class X as cutting plane. In bold we mark the three fastest times. Note that in case of class 29 the facet replace the normal three cycles.

	ER_CFO_15_1	LB_CFO_15_1	LD_CFO_15_1	LD_CFO_15_2
w.c.	00.00.03	00:04:53	00:00:08	00:00:10
T2	00:00:03	**00:00:09**	00:00:04	00:00:06
T7	00:00:03	00:01:16	00:00:03	00:00:20
T13	00:00:02	00:00:27	00:00:07	00:00:04
T14	00:00:03	00:01:18	00:00:04	00:00:16
T20	00:00:10	00:01:31	00:00:11	00:00:11
T25	$<$ **1**	**00:00:15**	**00:00:01**	**00:00:03**
T29	00:00:01	00:00:19	**00:00:01**	00:00:05
T30	$<$ **1**	00:00:37	00:00:02	00:00:05
T39	00:00:01	00:00:17	00:00:02	**00:00:03**
T41	$<$ **1**	**00:00:08**	**00:00:01**	**00:00:03**

Table F.1: Computation times with different cutting planes for the instances with 15 nodes

	LB_CFO_20_1	LB_CFO_20_2	LD_CFO_20_1	ER_CFO_20_1
w.c.	Mem.	00:00:07	Mem.	Mem.
T2	00:16:06	00:00:02	00:05:18	Mem.
T7	00:34:42	00:00:03	00:05:36	Mem.
T13	00:35:36	00:00:04	00:09:25	Mem.
T14	**00:05:18**	00:00:05	00:06:17	Mem.
T20	00:40:32	00:00:19	00:16:27	Mem.
T25	**00:01:30**	**< 1**	**00:00:38**	**05:12:14**
T29	**00:02:40**	**00:00:01**	**00:00:36**	Mem.
T30	01:16:03	**00:00:01**	00:12:20	Mem.
T39	00:06:35	00:00:02	**00:02:07**	**06:18:59**
T41	00:11:34	**00:00:01**	00:03:35	Mem.

	LB_MCO_20_1	ER_MCO_20_1	ER_BCO_20_1	ER_BCO_20_2
w.c.	00:06:45	00:02:10	Mem.	00:21:01
T2	00:01:05	00:00:10	00:16:00	00:02:17.
T7	00:00:55	00:00:33	00:47:49	00:01:40
T13	00:01:20	00:00:30	00:45:13	00:01:40
T14	**00:00:39**	00:00:26	00:23:54	00:01:33
T20	00:04:57	00:00:46	01:03:03	00:05:29.
T25	**00:00:12**	**< 1**	**00:00:56**	**00:00:32**
T29	**00:00:05**	**00:00:02**	**00:03:36**	**00:00:19**
T30	00:01:16	**00:00:05**	00:06:07	00:02:58
T39	00:00:54	00:00:17	00:10:07	00:00:49
T41	00:00:51	00:00:06	**00:02:51**	**00:00:37**

Table F.2: Computation times with different cutting planes for the instances with 20 nodes

	ER_CFO_23_1	ER_CFO_23_2	LD_CFO_23_1	ER_CFO_23_3
w.c.	Mem.	00:05:50	00:00:16	Mem.
T2	Mem.	00:02:10	00:00:22	00:22:51
T7	Mem.	00:02:28	00:00:15	00:29:15
T13	Mem.	00:03:31	00:00:14	00:10:47
T14	Mem.	00:02:05	00:00:15	00:08:38
T20	Mem.	00:09:34	**00:00:02**	00:43:01
T25	**00:01:33**	**00:01:02**	00:00:03	**00:01:01**
T29	Mem.	**00:00:17**	**00:00:02**	**00:01:27**
T30	Mem.	00:01:15	00:00:03	00:06:00
T39	**07:16:32**	00:01:45	00:00:05	00:04:50
T41	**00:01:51**	**00:01:11**	**00:00:01**	**00:01:11**

	LD_MCO_23_1	ER_MCO_23_1	ER_BCO_23_1	LD_BCO_23_1
w.c.	Mem.	Mem.	01:10:42	Mem.
T2	Mem.	Mem.	00:06:37	Mem.
T7	00:44:13	17:11:59	00:09:13	Mem.
T13	Mem.	Mem.	00:07:58	Mem.
T14	**00:18:36**	09:53:52	00:07:36	Mem.
T20	08:12:15	41:30:00	00:20:30	Mem.
T25	**00:03:49**	**00:02:35**	**00:02:12**	**00:03:07**
T29	**00:02:04**	**00:39:10**	**00:00:44**	Mem.
T30	03:10:49	**00:04:13**	00:06:34	Mem.
T39	05:40:54	02:10:25	00:07:17	Mem.
T41	Mem.	00:48:11	**00:02:46**	Mem.

Table F.3: Computation times with different cutting planes for the instances with 23 nodes

F.2 Test of combinations of different cutting planes

In the following tables we state the results of the test for using different combinations of facet classes as cutting planes. Hereby means:

- TA = TVP_{HP} model + facet classes 25 and 29
- TB = TVP_{HP} model + facet classes 25 and 41
- TC = TVP_{HP} model + facet classes 29 and 41
- TD = TVP_{HP} model + facet classes 25, 29 and 41
- TX = TVP_{HP} model with facet class X.

In bold we mark the three fastest times.

	LB_CFO_20_1	LD_CFO_20_1	ER_CFO_20_1	ER_BCO_20_1
T25	00:01:30	00:00:38	05:12:14	00:00:56
T29	00:02:40	00:00:36	Mem.	00:03:36
T41	00:11:34	00:03:35	Mem.	00:02:51
TA	00:00:31	00:00:18	02:11:42	00:00:30
TB	00:01:13	00:00:15	01:53:55	00:00:25
TC	**00:00:23**	00:00:16	01:18:58	00:00:23
TC/T30	**00:00:23**	00:00:16	01:12:44	< **1**
TC/T39	00:00:35	**00:00:09**	**00:44:10**	< **1**
TC/T30/T39	00:00:37	**00:00:09**	**00:44:34**	< **1**
TD	00:00:26	00:00:12	00:58:30	00:00:12
TD/T30	**00:00:24**	00:00:13	01:00:00	< **1**
TD/T39	00:00:53	**00:00:07**	**00:51:07**	< **1**
TD/T30/T39	00:00:54	**00:00:07**	00:53:44	< **1**

Table F.4: Test of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 20 nodes

	ER_CFO_23_1	ER_CFO_23_2	LD_CFO_23_1	ER_CFO_23_3
T25	00:01:33	00:01:02	00:00:03	00:01:01
T29	Mem.	00:00:17	00:00:02	00:01:27
T41	00:01:51	00:01:11	00:00:01	00:01:11
TA	00:00:26	00:00:16	00:00:01	00:00:23
TB	00:00:46	00:00:31	00:00:01	00:00:20
TC	00:00:20	00:00:15	< **1**	00:00:11
TC/T30	00:00:19	**00:00:12**	< **1**	00:00:11
TC/T39	**00:00:12**	00:00:17	< **1**	**00:00:07**
TC/T30/T39	**00:00:12**	00:00:17	< **1**	**00:00:07**
TD	00:00:15	**00:00:13**	< **1**	00:00:10
TD/T30	00:00:15	**00:00:13**	< **1**	00:00:09.
TD/T39	**00:00:13**	00:00:22	< **1**	**00:00:08**
TD/T30/T39	**00:00:13**	00:00:21	< **1**	00:00:09

	LD_MCO_23_1	ER_MCO_23_1	ER_BCO_23_1	LD_BCO_23_1
T25	00:03:49	00:02:35	00:02:12	00:03:07
T29	00:02:04	00:39:10	00:00:44	Mem.
T41	Mem.	00:48:11	00:02:46	Mem.
TA	00:01:06	00:00:55	00:00:29	00:01:15
TB	00:01:08	00:01:37	00:00:59	00:01:51
TC	00:00:34	00:00:59	00:00:25	00:00:53
TC/T30	00:00:49	00:01:02	00:00:28	00:00:54
TC/T39	**00:00:19**	**00:00:18**	00:00:26	**00:00:39**
TC/T30T39	**00:00:19**	00:00:19	00:00:29	00:00:40
TD	00:00:34	00:00:45	**00:00:21**	**00:00:39**
TD/T30	00:00:34	00:00:40	00:00:27	00:00:41
TD/T39	**00:00:21**	**00:00:07**	**00:00:21**	00:00:40
TD/T30T39	**00:00:21**	**00:00:07**	**00:00:21**	**00:00:39**

Table F.5: Test of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 23 nodes

Name	TC	TC/T39	TD	TD/T30	TD/T30/T39
LB_CFO_26_1	00:00:19	00:00:07	00:00:24	00:00:25	**00:00:06**
LD_CFO_26_1	00:13:55	00:03:52	00:04:27	00:04:33	**00:03:13**
LB_CFO_26_2	00:13:45	00:12:45	00:07:10	**00:07:01**	00:08:28
LB_CFO_26_3	00:04:11	00:03:50	00:02:54	00:02:55	**00:02:00**
LB_MCO_26_1	00:02:40	**00:01:15**	00:02:16	00:02:16	00:01:18
ER_MCO_26_1	00:03:21	**00:03:14**	00:03:54	00:03:55	00:04:39
ER_BCO_26_1	00:00:09	00:00:10	00:00:08	**00:00:07**	**00:00:07**
ER_BCO_26_2	00:00:24	**00:00:14**	00:00:19	00:00:21	**00:00:14**
ER_CFO_30_1	**00:00:55**	00:01:08	00:01:25	00:01:25	00:01:22
ER_CFO_30_2	00:32:34	00:28:59	00:46:26	00:52:48	**00:23:49**
LB_CFO_30_1	00:04:53	00:01:26	00:00:56	00:00:55	**00:00:46**
LB_CFO_30_2	00:05:41	00:07:57	**00:04:05**	00:04:16	00:05:42
LB_MCO_30_1	00:05:27	00:05:43	00:01:35	**00:01:18**	00:03:08
ER_MCO_30_1	00:32:59	00:24:08	00:16:47	00:18:07	**00:16:00**
ER_MCO_30_2	01:00:08	00:34:11	00:27:45	00:28:24	**00:26:08**
ER_MCO_30_3	00:00:36	**00:00:28**	00:00:38	00:00:35	00:00:33
ER_MCO_30_4	00:24:18	00:21:23	00:08:23	00:08:25	**00:07:20**
LD_MCO_30_1	**00:00:11**	**00:00:11**	00:00:15	00:00:14	00:00:19
LB_BCO_30_1	00:03:42	00:02:00	00:02:15	00:02:50	**00:01:13**
ER_CFO_35_1	01:28:56	00:24:45	00:36:06	00:36:35	**00:11:53**
LB_CFO_35_1	00:43:21	00:30:57	00:28:21	00:30:23	**00:17:37**
LB_CFO_35_2	00:03:32	00:02:34	00:02:26	00:02:31	**00:01:37**
LD_CFO_35_1	04:09:06	01:37:44	01:21:46	01:30:40	**00:24:19**
LB_MCO_35_1	01:44:57	01:51:06	**01:18:52**	01:31:37	01:39:43
ER_MCO_35_1	00:55:58	00:31:08	**00:22:39**	00:25:44	00:25:51
LB_MCO_35_2	00:03:53	00:04:02	00:01:30	00:01:39	**00:01:26**
LB_BCO_35_1	00:49:07	00:39:45	00:33:34	**00:30:36**	00:42:08
ER_BCO_35_1	00:16:46	00:14:32	00:12:45	**00:12:34**	00:15:20
ER_BCO_35_2	00:17:50	**00:10:49**	00:18:48	00:19:06	00:14:10

Table F.6: Results of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 26–35 nodes

Name	TC	TC/T39	TD	TD/T30	TD/T30/T39
ER_CFO_40_1	**00:05:20**	00:06:00	00:06:32	00:06:00	00:06:21
ER_CFO_40_2	Mem.	Mem.	03:48:17	**03:44:44**	04:48:41
ER_CFO_40_3	00:17:59	00:11:54	**00:09:42**	00:10:08	00:10:09
ER_CFO_40_4	Mem.	Mem.	Mem.	Mem.	Mem.
ER_CFO_40_5	00:06:20	00:03:28	00:03:16.	00:03:10	**00:02:13**
LB_CFO_40_1	01:05:43	00:56:39	00:53:27	**00:53:03**	00:55:46
LD_MCO_40_1	Mem.	04:42:02	03:31:26	04:02:05	**03:24:02**
ER_MCO_40_1	00:02:03	**00:01:12**	00:02:16	00:02:29	00:01:39
LD_CFO_45_1	Mem.	Mem.	Mem.	Mem.	**06:59:24**
LB_CFO_45_1	Mem.	Mem.	Mem.	Mem.	Mem.
LB_CFO_45_2	Mem.	Mem.	Mem.	Mem.	Mem.
LD_CFO_45_2	Mem.	Mem.	Mem.	Mem.	Mem.
LB_MCO_45_1	Mem.	Mem.	Mem.	Mem.	Mem.
LB_MCO_45_2	Mem.	Mem.	Mem.	Mem.	**123:28:09**
LB_MCO_45_3	Mem.	Mem.	Mem.	Mem.	**30:56:43**
LD_BCO_45_1	Mem.	Mem.	Mem.	Mem.	**06:48:56**
LB_BCO_45_1	Mem.	23:11:35	08:42:35	09:46:53	**02:39:49**
ER_BCO_45_1	Mem.	Mem.	Mem.	Mem.	**60:01:40**
LB_CFO_50_1	Mem.	Mem.	Mem.	Mem.	Mem.
LB_CFO_50_2	Mem.	Mem.	Mem.	Mem.	Mem.
ER_CFO_50_1	Mem.	Mem.	Mem.	Mem.	Mem.

Table F.7: Results of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 40 or more nodes

F.3 Detailed Information for the Runs with different facet combinations

Name	Root-B.	# sub	# LPs	Level
LB_CFO_26_1	27488.7	245	4694	14
LD_CFO_26_1	51832.3	6733	147068	28
LB_CFO_26_2	12758.9	6301	162813	30
LB_CFO_26_3	9319.1	2803	49582	23
LB_MCO_26_1	9157.0	1379	33558	24
ER_MCO_26_1	−3320.2	1409	40809	29
ER_BCO_26_1	−4186.7	49	2041	9
ER_BCO_26_2	1855.9	205	5367	13
ER_CFO_30_1	−4686.7	337	14231	15
ER_CFO_30_2	−4678.2	7335	283966	32
LB_CFO_30_1	28518.9	2209	49569	23
LB_CFO_30_2	8654.1	1619	51669	21
LB_MCO_30_1	4468.2	653	50729	17
ER_MCO_30_1	−749.2	6861	285838	27
ER_MCO_30_2	3656.8	13325	443931	27
ER_MCO_30_3	−6639.0	151	6039	13
ER_MCO_30_4	1185.7	4193	204334	25
LD_MCO_30_1	248297.2	37	1838	7
LB_BCO_30_1	5460.0	719	34341	17
ER_CFO_35_1	3422.0	7535	453056	30
LB_CFO_35_1	4811.0	4607	253980	26
LB_CFO_35_2	4592.8	469	30823	15
LD_CFO_35_1	169349.0	12313	1185571	32
LB_MCO_35_1	21618.3	10897	595203	31
ER_MCO_35_1	−4616.9	4601	388274	23
LB_MCO_35_2	14749.5	367	27052	17
LB_BCO_35_1	9463.8	4961	296800	24
ER_BCO_35_1	−12726.5	1213	117225	20
ER_BCO_35_2	−1279.1	1751	116587	23

Table F.8: Detailed information for the runs with facet classes 29 and 41 as cutting planes for instances with 26, 30 and 35 nodes

Name	Root-B.	# sub	# LPs	Level
ER_CFO_40_1	−373.7	509	32852	18
ER_CFO_40_3	−8059.9	1935	89192	20
ER_CFO_40_5	−8842.0	767	38973	19
LB_CFO_40_1	25560.5	7091	314076	26
ER_MCO_40_1	−15583.1	129	12311	9

Table F.9: Detailed information for the runs with facet classes 29 and 41 as cutting planes for instances with 40 nodes

Name	Root-B.	# sub	# LPs	Level
LB_CFO_26_1	26823.8	45	1578	10
LD_CFO_26_1	49805.6	1337	41416	21
LB_CFO_26_2	12712.8	3687	143359	26
LB_CFO_26_3	9046.7	1737	43418	18
LB_MCO_26_1	5544.9	361	14377	13
ER_MCO_26_1	−3352.2	765	37277	20
ER_BCO_26_1	−4282.8	31	2018	6
ER_BCO_26_2	1241.8	69	2905	9
ER_CFO_30_1	−5099.1	211	10847	12
ER_CFO_30_2	−6173.8	4099	237201	25
LB_CFO_30_1	26552.4	351	13361	15
LB_CFO_30_2	8368.1	1107	69037	18
LB_MCO_30_1	4360.0	363	49408	16
ER_MCO_30_1	−855.7	2931	205345	20
ER_MCO_30_2	2481.9	5433	269346	22
ER_MCO_30_3	−6848.2	81	4127	10
ER_MCO_30_4	549.6	2223	177710	21
LD_MCO_30_1	247996.5	21	1511	6
LB_BCO_30_1	4979.5	269	17300	13

Table F.10: Detailed information for the runs with facet classes 29, 39 and 41 as cutting planes for instances with 26 and 30 nodes

Name	Root-B.	# sub	# LPs	Level
ER_CFO_35_1	1254.2	1597	114996	20
LB_CFO_35_1	4430.8	1897	158162	18
LB_CFO_35_2	4374.3	205	18307	12
LD_CFO_35_1	168334.7	4381	463448	29
LB_MCO_35_1	21276.8	6563	641487	30
ER_MCO_35_1	−5115.8	2329	189164	19
LB_MCO_35_2	14412.6	205	26810	15
LB_BCO_35_1	8598.9	2885	230791	21
ER_BCO_35_1	−13490.6	779	96040	22
ER_BCO_35_2	−2236.7	575	68366	19
ER_CFO_40_1	−811.9	231	32637	11
ER_CFO_40_3	−8406.1	869	60210	16
ER_CFO_40_5	−9680.0	161	19226	13
LB_CFO_40_1	25256.9	3513	255798	28
LD_MCO_40_1	751596.9	13137	1051956	30
ER_MCO_40_1	−16290	67	6503	8
LB_BCO_45_1	7441.0	7589	3998484	24

Table F.11: Detailed information for the runs with the facets 29, 39 and 41 as cutting planes for instances with 35, 40 and 45 nodes

Name	Root-B.	# sub	# LPs	Level
LB_CFO_26_1	26831.3	197	4986	16
LD_CFO_26_1	50240.8	1193	46602	24
LB_CFO_26_2	11887.0	2235	80195	22
LB_CFO_26_3	8353.8	1275	32938	19
LB_MCO_26_1	8687.6	729	26112	24
ER_MCO_26_1	−3381.2	1177	43303	27
ER_BCO_26_1	−4529.6	19	1664	5
ER_BCO_26_2	1546.1	137	3669	12

Table F.12: Detailed information for the runs with facet classes 25, 29 and 41 as cutting planes for instances with 26 nodes

Name	Root-B.	# sub	# LPs	Level
ER_CFO_30_1	−4784.4	263	14260	14
ER_CFO_30_2	−4902.1	6451	385644	37
LB_CFO_30_1	28198	179	9398	18
LB_CFO_30_2	8089.9	509	36681	18
LB_MCO_30_1	3149.3	139	13832	13
ER_MCO_30_1	−1296.1	2711	136548	25
ER_MCO_30_2	3030.7	4873	219888	25
ER_MCO_30_3	−7019.4	107	5954	10
ER_MCO_30_4	128.5	999	67155	21
LD_MCO_30_1	248235.9	33	2216	7
LB_BCO_30_1	4533.4	331	19901	15
ER_CFO_35_1	3192.9	2163	162274	25
LB_CFO_35_1	4383.3	1973	136063	23
LB_CFO_35_2	4389.4	275	17050	15
LD_CFO_35_1	168641.4	3463	373888	26
LB_MCO_35_1	20762.2	5367	443410	30
ER_MCO_35_1	−5252.9	1393	141260	21
LB_MCO_35_2	13514.6	59	10012	12
LB_BCO_35_1	8787.5	1897	202256	22
ER_BCO_35_1	−12848.7	757	82912	18
ER_BCO_35_2	−1811.3	1035	115999	21
ER_CFO_40_1	−703.1	381	35554	20
ER_CFO_40_2	−15212.4	11719	936485	28
ER_CFO_40_3	−8738.5	539	51493	15
ER_CFO_40_5	−9310.4	195	17784	15
LB_CFO_40_1	25358.4	3157	249978	28
LD_MCO_40_1	751813.6	10599	716117	34
ER_MCO_40_1	−15801.3	53	12857	12
LB_BCO_45_1	6846.7	4171	1451019	24

Table F.13: Detailed information for the runs with facet classes 25, 29 and 41 as cutting planes for instances with 30–45 nodes

Name	Root-B.	# sub	# LPs	Level
LB_CFO_26_1	26831.3	189	5105	15
LD_CFO_26_1	50240.8	1201	47658	23
LB_CFO_26_2	11887.0	2233	79120	22
LB_CFO_26_3	8353.8	1229	33305	19
LB_MCO_26_1	8687.6	737	26302	22
ER_MCO_26_1	−3381.2	1159	44171	27
ER_BCO_26_1	−4529.6	19	1460	5
ER_BCO_26_2	1546.1	139	4230	12
ER_CFO_30_1	−4784.4	265	14068	14
ER_CFO_30_2	−4902.1	6469	444057	38
LB_CFO_30_1	28198	175	9318	18
LB_CFO_30_2	8089.9	511	38304	18
LB_MCO_30_1	3149.3	139	10936	13
ER_MCO_30_1	−1296.1	2811	150279	25
ER_MCO_30_2	3030.6	4955	225824	25
ER_MCO_30_3	−7019.4	107	5310	11
ER_MCO_30_4	128.5	995	67370	21
LD_MCO_30_1	248235.9	33	2129	7
LB_BCO_30_1	4533.4	295	25013	15
ER_CFO_35_1	3192.9	2195	166302	25
LB_CFO_35_1	4383.3	2067	146406	20
LB_CFO_35_2	4389.4	275	17446	15
LD_CFO_35_1	168641.4	3779	409978	26
LB_MCO_35_1	20762.2	2195	166302	25
ER_MCO_35_1	−5252.9	1423	167865	21
LB_MCO_35_2	13514.6	59	11184	12
LB_BCO_35_1	8787.5	1869	184074	22
ER_BCO_35_1	−12848.7	781	82151	18
ER_BCO_35_2	−1811.3	1073	119233	21

Table F.14: Detailed information for the runs with facet classes 25, 29, 30 and 41 as cutting planes for instances with 26,30 and 35 nodes

Name	Root-B.	# sub	# LPs	Level
ER_CFO_40_1	−707.8	345	32442	20
ER_CFO_40_2	−15212.4	11731	947315	28
ER_CFO_40_3	−8738.5	575	54232	16
ER_CFO_40_5	−9310.4	183	17694	15
LB_CFO_40_1	25358.4	3103	250121	25
LD_MCO_40_1	751813.6	12183	782004	34
ER_MCO_40_1	−15801.3	63	14275	15
LB_BCO_45_1	6846.7	4159	1659277	24

Table F.15: Detailed information for the runs with facet classes 25, 29, 30 and 41 as cutting planes for instances with 40 and 45 nodes

Name	Root-B.	# sub	# LPs	Level
LB_CFO_26_1	26285.2	13	117	6
LD_CFO_26_1	48892.7	13	1171	6
LB_CFO_26_2	11853.5	1453	86405	21
LB_CFO_26_3	8095.3	529	20167	16
LB_MCO_26_1	4818.8	161	13255	13
ER_MCO_26_1	−3406.0	751	49288	24
ER_BCO_26_1	−4561.8	13	1378	4
ER_BCO_26_2	1037.4	47	2640	10
ER_CFO_30_1	−5136.9	163	11508	12
ER_CFO_30_2	−6706.0	1469	175085	21
LB_CFO_30_1	25857.0	65	6241	9
LB_CFO_30_2	7846.3	439	43586	19
LB_MCO_30_1	3090.7	97	24766	11
ER_MCO_30_1	−1417.5	1227	123020	18
ER_MCO_30_2	1873.4	2089	187553	21
ER_MCO_30_3	−7209.4	61	4578	9
ER_MCO_30_4	−809.3	433	53313	16
LD_MCO_30_1	247943.8	11	2550	5
LB_BCO_30_1	4074.2	79	9530	10

Table F.16: Detailed information for the runs with facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 26 and 30 nodes

Name	Root-B.	# sub	# LPs	Level
ER_CFO_35_1	651.6	487	47386	15
LB_CFO_35_1	3982.1	709	72819	16
LB_CFO_35_2	4054.1	95	9479	10
LD_CFO_35_1	167350.2	893	94978	21
LB_MCO_35_1	20523.8	3049	537980	29
ER_MCO_35_1	−5847.8	721	144844	16
LB_MCO_35_2	13091.5	35	8415	11
LB_BCO_35_1	8063.4	1173	233690	21
ER_BCO_35_1	−13674.3	573	90650	18
ER_BCO_35_2	−2893.5	369	80107	17
ER_CFO_40_1	−1233.6	129	30102	12
ER_CFO_40_2	−15568.6	7309	1218332	26
ER_CFO_40_3	−8985.7	317	44900	14
ER_CFO_40_5	−10340.5	55	10254	12
LB_CFO_40_1	25065.6	1523	215904	22
LD_MCO_40_1	750172.0	6385	677171	29
ER_MCO_40_1	−16590.7	17	8677	6
LD_CFO_45_1	65440.9	3543	974978	23
LB_MCO_45_2	12551.4	10907	12819858	28
LB_MCO_45_3	41045.5	7083	5591037	23
LD_BCO_45_1	1075516.8	4769	1044165	25
LB_BCO_45_1	6123.1	1241	432959	18
ER_BCO_45_1	−4236.9	10145	11059998	32

Table F.17: Detailed information for the runs with facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 35, 40 and 45 nodes

Appendix G

Results of the Branch-and-Bound with Active/Nonactive variables

Name	Time. BaB	Time BaC
ER_CFO_15_1	00:00:22	< 1
LB_CFO_15_1	00:00:32	00:00:01
LD_CFO_15_1	00:00:29	< 1
LD_CFO_15_2	00:00:36	00:00:02
ER_MCO_15_1	00:00:22	< 1
ER_MCO_15_2	00:00:26	< 1
LB_MCO_15_1	00:00:04	< 1
LD_MCO_15_1	00:01:57	00:00:29
LD_BCO_15_1	00:00:01	< 1
ER_BCO_15_1	00:00:18	00:00:01
LB_BCO_15_1	00:00:32	00:00:01
LB_CFO_20_1	00:04:01	00:00:54
LB_CFO_20_2	00:00:12	< 1
LD_CFO_20_1	00:05:10	00:00:07
ER_CFO_20_1	00:38:51	00:53:44
LB_MCO_20_1	00:01:50	00:00:01

Table G.1: Results of the branch-and-bound algorithm using active/nonactive variables compared with the best results of the branch-and-cut approach for the instances with 15 and 20 nodes

Name	Time. BaB	Time BaC
ER_MCO_20_1	00:00:08	< 1
ER_BCO_20_1	00:02:48	< 1
ER_BCO_20_2	00:03:14	00:00:06
ER_CFO_23_1	00:11:46	00:00:13
ER_CFO_23_2	00:12:32	00:00:21
LD_CFO_23_1	00:01:43	< 1
ER_CFO_23_3	00:05:24	00:00:09
LD_MCO_23_1	00:11:47	00:00:21
ER_MCO_23_1	00:04:41	00:00:07
ER_BCO_23_1	00:22:31	00:00:21
LD_BCO_23_1	00:18:06	00:00:39
LB_CFO_26_1	00:06:17	00:00:06
LD_CFO_26_1	06:30:34.	00:03:13
LB_CFO_26_2	08:34:56	00:08:28
LB_CFO_26_3	07:12:15	00:02:00
LB_MCO_26_1	02:03:11	00:01:18
ER_MCO_26_1	03:36:03	00:04:39
ER_BCO_26_1	01:21:02	00:00:07
ER_BCO_26_2	01:30:07	00:00:14
ER_CFO_30_1	16:31:06	00:01:22
ER_CFO_30_2	166:57:11	00:23:49
LB_CFO_30_1	00:50:30	00:00:46
LB_CFO_30_2	10:10:31	00:05:42
LB_MCO_30_1	02:57:30	00:03:08
ER_MCO_30_1	64:28:11	00:16:00
ER_MCO_30_2	11:37:49	00:26:08
ER_MCO_30_3	02:01:05	00:00:33
ER_MCO_30_4	12:08:42	00:07:20
LD_MCO_30_1	02:03:54	00:00:19
LB_BCO_30_1	01:45:55	00:01:13

Table G.2: Results of the branch-and-bound algorithm using active/nonactive variables compared with the best results of the branch-and-cut approach for the instances with 20, 23, 26 and 30 nodes

Appendix H

CPLEX Computation Times

In the following (*) means that CPLEX was only able to compute a value which is about 1% close to the optimum due to numerical problems.

Name	Time TVP_{HP}	Time TVP_E
ER_CFO_15_1	00:00:02	00:00:02
LB_CFO_15_1	00:00:05	00:00:22
LD_CFO_15_1	00:00:04	00:00:20
LD_CFO_15_2	00:00:02	00:00:28
ER_MCO_15_1	00:00:01	00:00:05
ER_MCO_15_2	00:00:02	00:00:05
LB_MCO_15_1	00:00:02	00:00:03
LD_MCO_15_1	00:00:12	00:03:22
LD_BCO_15_1	00:00:01	00:00:01
ER_BCO_15_1	00:00:04	00:00:24
LB_BCO_15_1	00:00:05	00:00:35
LB_CFO_20_1	00:00:51	00:04:48
LB_CFO_20_2	00:00:04	00:00:07
LD_CFO_20_1	00:00:50	00:05:08
ER_CFO_20_1	00:01:55	01:11:56
LB_MCO_20_1	00:00:16	00:03:17
ER_MCO_20_1	00:00:07	00:00:16
ER_BCO_20_1	00:01:00	00:05:33
ER_BCO_20_2	00:00:25	00:05:40

Table H.1: Computation times with CPLEX for the instances with 15 and 20 nodes

Name	Time TVP_{HP}	Time TVP_E
ER_CFO_23_1	00:00:43	00:13:20
ER_CFO_23_2	00:00:25	00:08:49
LD_CFO_23_1	00:00:07	00:00:33
ER_CFO_23_3	00:00:34	00:05:59
LD_MCO_23_1	00:02:35 (*)	00:07:45
ER_MCO_23_1	00:02:44	00:12:09
ER_BCO_23_1	00:00:51	00:13:27
LD_BCO_23_1	00:04:21 (*)	00:29:13

Table H.2: Computation times with CPLEX for the instances with 23 nodes

Name	Time TVP_{HP}
LB_CFO_26_1	00:01:05 (*)
LD_CFO_26_1	00:26:28 (*)
LB_CFO_26_2	00:07:47
LB_CFO_26_3	00:09:41
LB_MCO_26_1	Mem.
ER_MCO_26_1	00:07:52
ER_BCO_26_1	00:02:10
ER_BCO_26_2	00:01:37
ER_CFO_30_1	00:23:24
ER_CFO_30_2	05:03:13
LB_CFO_30_1	02:26:37 (*)
LB_CFO_30_2	02:16:38
LB_MCO_30_1	Mem.
ER_MCO_30_1	Mem.
ER_MCO_30_2	Mem.
ER_MCO_30_3	01:31:52
ER_MCO_30_4	Mem.
LD_MCO_30_1	01:25:00 (*)
LB_BCO_30_1	05:30:28

Name	Time TVP_{HP}
ER_CFO_35_1	Mem.
LB_CFO_35_1	06:43:18
LB_CFO_35_2	Mem.
LD_CFO_35_1	Mem.
LB_MCO_35_1	Mem.
ER_MCO_35_1	Mem.
LB_MCO_35_2	Mem.
LB_BCO_35_1	Mem.
ER_BCO_35_1	11:37:10
ER_BCO_35_2	08:23:39
ER_CFO_40_1	Mem.
ER_CFO_40_2	Mem.
ER_CFO_40_3	Mem.
ER_CFO_40_4	Mem.
ER_CFO_40_5	05:28:17
LB_CFO_40_1	Mem.
LD_MCO_40_1	Mem.
ER_MCO_40_1	16:02:05(*)

Table H.3: Computation times with CPLEX for the instances with 26–40 nodes

Name	Time TVP_{HP}
LD_CFO_45_1	Mem.
LB_CFO_45_1	Mem.
LB_CFO_45_2	Mem.
LD_CFO_45_2	Mem.
LB_MCO_45_1	Mem.
LB_MCO_45_2	Mem.
LB_MCO_45_3	Mem.
LD_BCO_45_1	Mem.
LB_BCO_45_1	Mem.
ER_BCO_45_1	Mem.

Name	Time TVP_{HP}
LB_CFO_50_1	Mem.
LB_CFO_50_2	Mem.
ER_CFO_50_1	Mem.

Table H.4: Computation times with CPLEX for the instances with 45 and 50 nodes

Appendix I

Results of the Lagrangean Decomposition

Name	Opt.	Sub-Gradient	Time
ER_CFO_15_1	-6435	-2584.34	00:00:44
LB_CFO_15_1	9478	14899.37	00:01:30
LD_CFO_15_1	107620	113958.56	00:05:09
LD_CFO_15_2	71699	79106.11	00:10:11
ER_MCO_15_1	-2236	-147.95	00:00:50
ER_MCO_15_2	-5738	-3579.11	00:00:51
LB_MCO_15_1	3798	9989.59	00:00:36
LD_MCO_15_1	16930	28150.93	00:05:09
LD_BCO_15_1	447756	470651.01	00:00:41
ER_BCO_15_1	-4986	-2939.17	00:00:38
LB_BCO_15_1	4206	12224.60	00:02:29
LB_CFO_20_1	16439	29591.80	00:01:07
LB_CFO_20_2	9494	12302.46	00:10:02
LD_CFO_20_1	66995	82380.33	00:01:03
ER_CFO_20_1	-7417	-1050.85	00:12:01
LB_MCO_20_1	14272	21558.75	00:02:57
ER_MCO_20_1	905	5835.71	00:01:05
ER_BCO_20_1	-7233	-368.29	00:07:54
ER_BCO_20_2	-9682	-4923.61	00:02:03

Table I.1: Results of the sub-gradient method

Name	Opt.	Prox. Bundle	Time
ER_CFO_15_1	-6435	−3761.45	00:00:50
LB_CFO_15_1	9478	13681.63	00:00:23
LD_CFO_15_1	107620	109768.92	00:01:39
LD_CFO_15_2	71699	77676.90	00:02:06
ER_MCO_15_1	-2236	−599.81	00:00:37
ER_MCO_15_2	-5738	−3939.28	00:00:43
LB_MCO_15_1	-3798	8019.76	00:00:53
LD_MCO_15_1	16930	25737.66	00:01:09
LD_BCO_15_1	447756	456329.55	00:05:35
ER_BCO_15_1	-4986	−3635.63	00:00:47
LB_BCO_15_1	4206	10041.04	00:00:30
LB_CFO_20_1	16439	26766.83	00:01:51
LB_CFO_20_2	9494	11354.76	00:01:16
LD_CFO_20_1	66995	77731.36	00:00:57
ER_CFO_20_1	-7417	−1932.72	00:00:52
LB_MCO_20_1	14272	19733.93	00:01:05
ER_MCO_20_1	905	4796.29	00:04:31
ER_BCO_20_1	-7233	−856.92	00:42:32
ER_BCO_20_2	-9682	−5869.82	00:01:12

Table I.2: Results of the proximal bundle method

Name	Opt.	Conic Bundle	Time
ER_CFO_15_1	-6435	-3761.42	00:00:17
LB_CFO_15_1	9478	13662.88	00:02:43
LD_CFO_15_1	107620	109769.74	00:02:21
LD_CFO_15_2	71699	77677.31	00:03:57
ER_MCO_15_1	-2236	-599.79	00:00:13
ER_MCO_15_2	-5738	-3940.00	00:00:16
LB_MCO_15_1	-3798	8019.78	00:01:40
LD_MCO_15_1	16930	25737.54	00:01:32
LD_BCO_15_1	447756	473013.00	00:00:01
ER_BCO_15_1	-4986	-3635.60	00:00:23
LB_BCO_15_1	4206	10041.06	00:00:37
LB_CFO_20_1	16439	26766.82	00:02:03
LB_CFO_20_2	9494	11354.82	00:02:51
LD_CFO_20_1	66995	77731.73	00:02:03
ER_CFO_20_1	-7417	-1932.63	00:03:28
LB_MCO_20_1	14272	19733.90	00:04:49
ER_MCO_20_1	905	4796.35	00:01:27
ER_BCO_20_1	-7233	-856.83	00:07:26
ER_BCO_20_2	-9682	-5869.69	00:03:04

Table I.3: Results of the conic bundle method

List of Algorithms

1 Branch-and-Bound 11
2 Cutting Plane 12
3 Branch-and-Cut 13

4 Sub-gradient method 56
5 Dynamic-Programming 58

6 Nearest-Neighbor Method 62
7 Best-Insertion Heuristic 63
8 Becker Heuristic 64
9 2-opt Heuristic 64
10 3-opt Heuristic 64
11 Kernighan-Lin-1 Heuristic 65
12 Kernighan-Lin-2 Heuristic 66

13 Reduced costs 74

List of Tables

3.1 Example of an LP solution which is only violated by the subtour elimination constraint 22
3.2 Sketch for the proof of Lemma 3.3 23

4.1 Solution of the LP relaxation of the different TVP models for instances with 15–23 nodes 32
4.2 Key facts of different TVP polytopes 34
4.3 Facet-defining inequality classes of P^4_{TV} for which zero-lifting is possible ($i, j, k, l \in \{1, 2, 3, 4\}$ and pairwise different) 43

7.1 Computation times of the dynamic programming approach 75

B.1 Data for the TVP instances with 15 and 20 nodes 97
B.2 Data for the TVP instances with 23, 26 and 30 nodes 98
B.3 Data for the TVP instances with 35–50 nodes 99

C.1 Solution of the LP relaxation for the different TVP models for instances with 15 and 20 nodes 101
C.2 Solution of the LP relaxation for the different TVP models for instances with 23 and 26 nodes 102

D.1 Results of the nearest-neighbor heuristic for instances with 15 and 20 nodes 103
D.2 Results of the nearest-neighbor heuristic for instances with 23 and 26 nodes 104
D.3 Average values of 500 runs of the Becker Heuristic for the instances with 15 nodes 104
D.4 Average values of 500 runs of the Becker Heuristic for the instances with 20, 23 and 26 nodes 105
D.5 Average values of 500 runs of the best-insertion heuristic for instances with 15, 20 and 23 nodes 106
D.6 Average values of 500 runs of the best-insertion heuristic for instances with 26 nodes 107

D.7 Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 15 nodes 108
D.8 Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 20 and 23 nodes 109
D.9 Average values of 500 runs of the improvement heuristics with Becker 100 as start heuristic for instances with 23 and 26 nodes 110
D.10 Average values of 500 runs of the improvement heuristics with best-insertion 100 as start heuristic (We left out instances where already the average value of the best-insertion heuristic is equal the optimal solution) 111
D.11 Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 15 nodes 112
D.12 Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 20 and 23 nodes . . . 113
D.13 Average values of 500 runs of the improvement heuristics with nearest-neighbor as start heuristic for instances with 23 and 26 nodes . . . 114
D.14 Average values of 500 runs of the Kernighan-Lin 2 heuristic with best-insertion 100 and Becker 100 as start heuristic for the instances with 30, 35 and 40 nodes 115
D.15 Average values of 500 runs of the Kernighan-Lin-2 heuristic with best-insertion 100 and Becker 100 as start heuristics for the instances with 45 and 50 nodes 116

E.1 Computation times (in seconds) of the hybrid DP/branch-and-bound algorithm for the instances with 15 nodes 117
E.2 Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 20 nodes 118
E.3 Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 23 nodes 119
E.4 Computation times of the hybrid DP/branch-and-bound algorithm for the instances with 26 nodes 120

F.1 Computation times with different cutting planes for the instances with 15 nodes 121
F.2 Computation times with different cutting planes for the instances with 20 nodes 122
F.3 Computation times with different cutting planes for the instances with 23 nodes 123
F.4 Test of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 20 nodes 124
F.5 Test of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 23 nodes 125

F.6 Results of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 26–35 nodes 126
F.7 Results of the combinations of facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 40 or more nodes 127
F.8 Detailed information for the runs with facet classes 29 and 41 as cutting planes for instances with 26, 30 and 35 nodes 128
F.9 Detailed information for the runs with facet classes 29 and 41 as cutting planes for instances with 40 nodes 129
F.10 Detailed information for the runs with facet classes 29, 39 and 41 as cutting planes for instances with 26 and 30 nodes 129
F.11 Detailed information for the runs with the facets 29, 39 and 41 as cutting planes for instances with 35, 40 and 45 nodes 130
F.12 Detailed information for the runs with facet classes 25, 29 and 41 as cutting planes for instances with 26 nodes 130
F.13 Detailed information for the runs with facet classes 25, 29 and 41 as cutting planes for instances with 30–45 nodes 131
F.14 Detailed information for the runs with facet classes 25, 29, 30 and 41 as cutting planes for instances with 26,30 and 35 nodes 132
F.15 Detailed information for the runs with facet classes 25, 29, 30 and 41 as cutting planes for instances with 40 and 45 nodes 133
F.16 Detailed information for the runs with facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 26 and 30 nodes 133
F.17 Detailed information for the runs with facet classes 25, 29, 30, 39 and 41 as cutting planes for instances with 35, 40 and 45 nodes . . 134

G.1 Results of the branch-and-bound algorithm using active/nonactive variables compared with the best results of the branch-and-cut approach for the instances with 15 and 20 nodes 135
G.2 Results of the branch-and-bound algorithm using active/nonactive variables compared with the best results of the branch-and-cut approach for the instances with 20, 23, 26 and 30 nodes 136

H.1 Computation times with CPLEX for the instances with 15 and 20 nodes 137
H.2 Computation times with CPLEX for the instances with 23 nodes . . 138
H.3 Computation times with CPLEX for the instances with 26–40 nodes 138
H.4 Computation times with CPLEX for the instances with 45 and 50 nodes 139

I.1 Results of the sub-gradient method 141
I.2 Results of the proximal bundle method 142
I.3 Results of the conic bundle method 143

Nomenclature

TVP_{HP}	TVP model based on combination of LOP and HP models
TVP_D	TVP model based on distance variables
TVP_E	Extended formulation of the TVP_{HP} model
TVP_{EN}	TVP model based on edge-node variables
TVP_O	TVP tour model based on combination of LOP and ATSP models
TVP_S	Symmetric TVP model based on combination of LOP and TSP models
$\mathbf{0}$	Vector with all entries equal to zero
$\mathbf{1}$	Vector with all entries equal to one
$MTVEq_n(f)$	Matrix of feasible characteristic vectors that fulfill facet f with equality
$MTVEq_n^i(f)$	Matrix of feasible characteristic vectors that fulfill facet f with equality and encode paths starting with target i
MTV_n^i	Matrix of feasible characteristic vectors of paths starting with target i
MTV_n	Matrix of feasible characteristic vectors
$NTVEq_n^i(f)$	Matrix of feasible characteristic vectors that fulfill facet f with equality and encode paths ending with target i
P_{DTV}^n	Associated polytope of the TVP_D model
P_{ENTV}^n	Associated polytope of the TVP_{EN} model
P_{ETV}^n	Associated polytope of the TVP_E model
P_{OTV}^n	Associated polytope of the TVP_O model
P_{STV}^n	Associated polytope of the TVP_S model
P_{TV}^n	Associated polytope of the TVP_{HP} model
d_{ij}	Distance cost of getting from i to j
p_{ij}	Preference value of visiting i before j
#	Number of items of a set
ATSP	Asymmetric traveling salesman problem
DP	Dynamic programming
HP	Hamiltonian path problem
IP	Integer program
LOP	Linear ordering problem

LP	Linear program
Mem.	Memory overflow
N	Number of targets of a certain instance
TSP	Traveling salesman problem
TVP	Target visitation problem
V	Set of all targets

Index

adjacent, 2
affine rank, 2
arcs, 2

benchmark instances, 67
binary program, 6
branch-and-bound, 4, 12, 57, 75
branch-and-cut, 15, 71
branch-and-price, 15, 73
bundle methods, 56

characteristic vector, 3
column vector, 1
combination
  affine, 1
  conic, 1
  convex, 1
  linear, 1
combinatorial optimization problem, 4
cutting plane method, 14
cycle, 2
  Hamiltonian, 2

dynamic programming, 56, 74

edge, 2
  weighted, 2

face, 7
face-defining, 7
facet, 7
facet-defining, 7
feasible set, 5
Fourier-Motzkin elimination, 53
free node, 43

graph, 2
  cardinality, 2
  complete, 2
  directed, 2
  simple, 2

halfspace, 6
Hamiltonian path problem, 8
Helmstädter condition, 58
heuristic
  k-opt, 64
  Becker, 64
  best-insertion, 62
  constructive, 61
  improvement, 61
  Kernighan-Lin, 65
  nearest-neighbor, 62
HUHFA, 17, 41
hull
  affine, 1
  conic, 1
  convex, 1
  linear, 1
hyperplane, 6

incident, 2
integer program, 6

Lagrangean decomposition, 55, 76
lagrangean relaxation, 10
lifting, 42
  zero, 43
linear ordering problem, 9
linear program, 5
linear relaxation, 6

matrix, 1
minimal aborescence, 58

NP, 3
NP-complete, 4
NP-hard, 4

objective function, 5
optimization problem, 3
outer description, 6

P, 3
path, 2
    Hamiltonian, 2
    length, 2
polyhedron, 6
polynomially reducible, 3
projection, 52

row vector, 1

separation, 14
simplex method, 5
sub-gradient method, 11, 56
subgraph, 2
submatrix, 1
subtour elimination constraints, 25

target visitation problem
    distance variable formulation, 34
    edge-node formulation, 31
    extended formulation, 29, 48
    LOP/TSP formulation, 23
    path formulation, 22
    symmetric, 82
    tour formulation, 21, 79
tournament equations, 28
traveling salesman problem, 7
    asymmetric, 7

V-representation, 6
vertice, 2
    degree, 2

weight function, 2