Universidad de La Laguna

Beatriz Santos Hernández

The split-demand one-commodity pickup-and-delivery travelling salesman problem

El problema del viajante de comercio con recogida y entrega de una mercancía con demanda dividida

Directores:
Juan José Salazar González
Hipólito Hernández Pérez

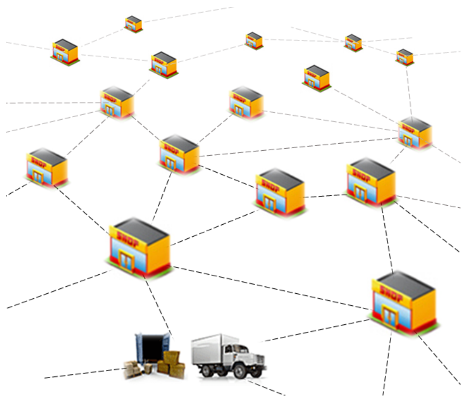
2016

Departamento de Matemáticas, Estadística
e Investigación Operativa

Beatriz Santos Hernández

The split-demand one-commodity pickup-and-delivery travelling salesman problem

El problema del viajante de comercio con recogida y entrega de una mercancía con demanda dividida



Under the guidance of:
Juan José Salazar González
Hipólito Hernández Pérez

2016

D. Juan José Salazar González, catedrático de universidad del Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna y D. Hipólito Hernández Pérez contratado doctor con régimen de interinidad del Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna,

CERTIFICAN:

Que la presente memoria, titulada "El problema del viajante de comercio con recogida y entrega de una mercancía con demanda dividida" (en inglés "The split-demand one-commodity pickup-and-delivery travelling salesman problem"), ha sido realizada bajo nuestra dirección por Dña Beatriz Santos Hernández y constituye su Tesis para optar al grado de Doctor en Ingeniería Informática por la Universidad de La Laguna.

Y para que conste, en cumplimiento de la legislación vigente, y a efectos que hayan lugar, firmamos la presente, en La Laguna, a 16 de Julio de 2016

Juan José Salazar González

Hipólito Hernández Pérez

To my family

Contents

Acknowledgments

This memory has been done under the supervision of professors Juan José Salazar González and Hipólito Hernández Pérez, to whom I want to express my most sincere gratitude. They proposed this subject for my PhD-research and guided me in the scientific research. I would also like to thank all my colleagues at the *Department of Mathematics, Statistics and Operational Research* (ULL) which somehow, they have helped me in developing the PhD-research.

I appreciate the support by the Spanish Government through the Research Project MTM2012-36163-C06-01 for the grant 'Subprograma de Formación de Personal Investigador del Ministerio de Ciencia e Innovación (Subprograma FPI-MICINN)', for their financial support, which has made this PhD-research possible. I also want to express my gratitude to the 'Universidad de La Laguna' for the grant 'Obra Social La Caixa-Fundación CajaCanarias para Posgraduados de la Universidad de La Laguna. Convocatoria 2014'.

It was a great honour for me to work with the group of the Management Science in the DTU Management Engineering department in the Technical University of Denmark (Lyngby, Denmark, July 24-October 24, 2014) which for two weeks got improved my knowledge in Operations Research. Apart, from the profitable lectures, this research group introduced me to a group of friends. I would also like to thank to the 'Secretaría de Estado de Investigación, Desarrollo e Innovación, for the grant 'movilidad predoctoral para la realización de estancias breves en Centros de I+D, convocatoria 2013', which has made this work in Denmark possible.

I cannot forget to thank Emilio for always believing in me. To my parents and my sister for their support in all my future plans and to my friends, specially to Sara, for their advices.

Preface

Since the existence of the first organized societies, optimization issues have been proposed for the use of resources and planning of tasks. Nowadays, these issues are more difficult to solve as we are in a competitive world where big and small business have to fight to get a place. In many cases, this fight is based in a good management of the available resources and a good operation planning. It is in this point where the mathematical modeling and the combinatorial optimization are useful. Thus, a lot of algorithms that help to solve diversity of problems in logistic, repositioning, genetics, etc., have been arising.

This thesis focuses on a kind of problems thoroughly studied in operational research, in particular in *Vehicle Routing Problems.* These arise for the first time in 1930 when a mathematical model for the *Travelling Salesman Problem* (TSP) was proposed. In this, it is set a group of cities that have to be visited taking the shortest distance as be possible, starting and finishing in the same city and going through them exactly once. This is one of the most famous problems in the combinatorial optimization field, and although it has a simple proposal, is an NP-hard problem as it is still looking for an algorithm to solve it polynomially.

The importance of these kind of problems is not due to the computational complexity only but also the variety of practical applications. In fact, the idea of *Operational Research* appears formally in 1938 in the Second World War, in the frame in collaborative *researches* between soldiers and scientists about planning of flight military *operations.* A part of the most obvious examples in logistic (distribution of commodities, scholar routes,etc.), there are more examples as operational control of traffic light, those that can be found in robotic systems that allow to solve production problems, in genetic, etc. Another kind of practical applications arise due to the growing worry about environment. This thesis studies the case of the management of bicycle sharing systems. These are increasingly in demand due to the need of space, the high traffic density and the high emission of noise and CO_2.

Despite there are a lot of studies about vehicle routing problems, the most basic are still difficult to solve optimally, even for small size and the lacking practical use. Because of that, this thesis develops algorithms to solve a particular problem known as the *The split-demand one-commodity pickup-and-delivery travelling salesman problem* (SD1PDTSP).

The research in this PhD thesis has been prepared at the Department of Statistic, Operational Research and Computer Science of the University of La Laguna, and this work has been supervised by professor Juan José Salazar González and professor Hipólito Hernández Pérez.

The dissertation is organized in six chapters. Chapter 1 describes some mathematical concepts in Operational Research. It also shows some important tools for this work: Benders decomposition, a branch-and-cut algorithm and some interesting concepts about heuristic techniques. Chapter 2 describes several related problems as the *One-Commodity Pickup-and-Delivery Travelling Salesman Problem* (1-PDTSP), or the *Split Delivery Vehicle Routing Problem* (SDVRP), as well as the SD1PDTSP. Chapter 3 presents a mathematical model for the SD1PDTSP and theoretical results of the problem. Chapter 4 describes a branch-and-cut algorithm to solve the SD1PDTSP exactly and some computational results. Finally chapter 5 presents a heuristic algorithm for the same problem based on well-known heuristic approaches and a math-based approach.

The main results of this work have been presented in several national and international conferences. Some of them are enumerated below:

- *Seminario de Doctorandos en Matemáticas.* San Cristóbal de La Laguna (Spain). November 10, 2011.
- *Combinatorial Optimization, Routing and Location (CORAL2012).* Benicassim, Castellón (Spain). May 2-5, 2012.
- *The first meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VEROLOG2012).* Bologna (Italy), 18-20 June 2012.
- *Cuarto Encuentro de Jóvenes Investigadores en Matemáticas (PEJIM2014).* San Cristóbal de La Laguna (Spain). November 19-22, 2014.
- *The fourth meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VEROLOG2014).* Vienna (Austria). June 8-10, 2015.
- *Encuentro de Jóvenes Investigadores/Doctorandos del IUDR.* San Cristóbal de La Laguna. September 28, 2015

In addition, I have been able to attend to some courses, seminars, etc., described below:

- *Winter School on Network Optimization.* Estoril (Portugal). January 17-21, 2011.
- *Reunión científica miembros del MTM2009-140390C06.* April 11-12, 2011.

- *Primer Encuentro de Jóvenes Investigadores de La Laguna (PEJIM2011).* San Cristóbal de La Laguna. September 28-30, 2011.
- *Summer School: Stochastic Programming: Extensions to Integer Programming.* Barcelona (Spain). June 11-15, 2012.
- *Summer School: Course on Distribution Management.* Barcelona (Spain). June 4-8, 2012.
- *Segundo Encuentro de Jóvenes Investigadores en Matemáticas (PEJIM2012).* San Cristóbal de La Laguna. September 27-29, 2012
- *International Network Optimization Conference (INOC2013).* Adeje (Spain). May 20-22, 2013.

Also, I have been able to work for three months with an excellent research group of the Management Science, DTU Management Engineering department in the Technical University of Denmark.

Finally the main results of this thesis have been published in Salazar-González and Santos-Hernández (2015). With this one I have obtained the II award to the new research's best paper from IUDR 2016. Also, Hernández-Pérez et al. (2016) has been submitted to European Journal of Operational Research.

Chapter 1

Basic Concepts, Models and Approaches

This paper introduces some mathematical concepts in Operational Research, and presents some closely related problem as the travelling salesman problem, the vehicle routing problem and the pickup and delivery problem. The Benders' decomposition and the branch-and-cut method are also shown as basic tools in order to solve the combinatorial optimization problems to optimality. Finally, some interesting concepts about heuristic algorithm are presented.

1.1 Graph Theory

Graph Theory is an important tool used to solve problems in mathematics and computational science. In particular, it is used to model and analyze *routing problems.* In this section some elementary concepts and properties on Graph Theory are given. The reader is encouraged to read textbooks by Berge (1973), Christofides (1975) and Bondy and Murty (2007) for further information.

A *directed graph* G is a pair (V, A), where V is a finite set of *vertices* and $A \subseteq \{(i, j) : i, j \in V, i \neq j\}$, i.e. A is a set of pairs of ordered elements of V called *arcs.* Each arc $a \in A$ is represented by $a = (i, j)$, where i denotes the *tail* and j the *head.* When the links between the vertices are not oriented, the graph is called *undirected graph* and is denoted by $G = (V, E)$, where E is a set of pairs of non-ordered elements of V called *edges.* Each edge $e \in E$ is represented by $e = [i, j]$, where i and j denote the *incident vertices* of e. Note that $e = [i, j] = [j, i]$. Moreover, a subgraph of G is a new graph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$. A *complete graph* (directed or undirected graph) is a graph where all pairs of distinct vertices are connected between them. If $|V| = n$, the complete graph is denoted by K_n.

Given S and T subsets of V, the following subsets are associated to the graph G:

$$\begin{aligned}\delta^+(S) &= \{(i,j) \in A : i \in S, j \notin S\}\\ \delta^-(S) &= \{(i,j) \in A : i \notin S, j \in S\}\\ A(S) &= \{(i,j) \in A : i \in S, j \in S\}\\ \delta(S) &= \{[i,j] \in E : i \in S, j \in V \setminus S\}\\ E(S) &= \{[i,j] \in E : i \in S, j \in S\}\end{aligned}$$

For the sake of simplicity, if $S = \{i\}$, it writes $\delta^+(i)$, $\delta^-(i)$ and $\delta(i)$ instead of $\delta^+(\{i\})$, $\delta^-(\{i\})$ and $\delta(\{i\})$.

A set $P = \{(i_1, i_2), (i_2, i_3), \dots (i_{t-1}, i_t)\}$ of a directed graph $G = (V, A)$ is the *path* between i_1 and i_t. It is said the path starts in i_1 and finishes in i_t. The *length* of a path is the number of their arcs and is denoted by $|P|$. A path C where the first and the last vertex are the same is called *closed path*, and besides, it is a *cycle* if $i_k \neq i_l$ for all $k, l \in \{2, \dots, t-1\}$, $k \neq l$. On the one hand, if $|C| = |V|$, i.e. it is going through every vertex only once, C is a *Hamiltonian cycle.* On the other hand, C is called *Eulerian cycle* if the cycle includes all the arcs of the graph.

1.2 Mathematical Programming

Mathematical Programming is the area of Operational Research where decision problems that need optimizing a fixed goal are solved, satisfying some restrictions in the available resources. Hence, its aim is to solve a kind of problem as the following

$$min\,\{f(x) : x \in S\}$$

where $S \subseteq \mathbb{R}^n$, $f : S \longrightarrow \mathbb{R}$, and it seeks to find (if exist) an element $x \in S$ in which the function f reaches its minimum (or maximum) value. Each element x is a *solution*, thus S represents the set of *feasible solutions*, and f is the *objective function.* Depending on the kind of variables, constrains and objective function that are described in the mathematical model of the problems, this area is divided in different parts. When f is a linear function, and the constrains can be defined by linear inequations, it is called *Linear Programming.* Moreover, if the variables have to be integers, it is called *Integer Linear Programming* and makes it much difficult to solve in general. Other kind of problems can be found when f is a non-linear function, and the constrains can be defined by non-linear inequations. This is called *Non-linear Programming.* If f is a vectorial function, it is said *Multiobjective Programming.* When the parameters that describe the problem are fixed, it deals with *Deterministic Programming*, whereas in *Stochastic Programming* some parameters are random variables.

Another part of Mathematical Programming is the *Combinatorial Optimization* which seeks the resolution of optimization problems with a finite number of feasible solutions (although in general very large). Integer Linear Programming problems and Combinatorial Optimization problems are closely related because the latter can be modelled as the former. Examples of this kind of problems are given in Section 1.4.

1.3 Polyhedral Theory

Combinatorial Optimization problems can be solved making use of the *Polyhedral Theory.* Thus, some important concepts about it are presented below. However, this area is studied with more detail in Bachem and Grötschel (1980) and Nemhauser and Wolsey (1988).

Given $x_1, \ldots, x_t \in \mathbb{R}^n$, it is said that a vector $x \in \mathbb{R}$ is a *affine combination* of $x_1, \ldots, x_t$ if there are $\lambda_1, \ldots, \lambda_t \in \mathbb{R}$ such that $x = \sum_{i=1}^{t} \lambda_i x_i$ and $\sum_{i=1}^{t} \lambda_i = 1$. If $\lambda_i \geq 0$ for $i \in \{1, \ldots, t\}$, then x is a *convex combination.* Let S be a subset of $\mathbb{R}^n$, it is said that S is a *cone* if and only if for all $x_1, x_2 \in S$ and for all $\lambda_1, \lambda_2 \geq 0$, then $\lambda_1 x_1 + \lambda_2 x_2 \in S$. The *convex hull* of S, denoted by $\text{conv}(S)$, is the set of all convex combinations of a finite number of vectors in S.

Let a set $H \subseteq \mathbb{R}^n$ be a *hyperplane* if there is a vector $a \in \mathbb{R}^n \backslash \{0\}$ and a scalar $a_0 \in \mathbb{R}$ such that $H = \{x : a^T x = a_0\}$. Besides, a set $B \subseteq \mathbb{R}^n$ is a *halfspace* if $B = \{x : a^T x \leq a_0\}$. It is said that B is the halfspace defined by the inequality $a^T x \leq a_0$, and H is the hyperplane defined by $a^T x \leq a_0$. An inequality $a^T x \leq a_0$ is said *valid* for a set $S \subseteq \mathbb{R}^n$ if $S \subset \{x \in \mathbb{R}^n : a^T x \leq a_0\}$. Moreover, the inequality is said *supporting* for S if it is valid and $S \cap \{x \in \mathbb{R}^n : a^T x = a_0\} \neq \emptyset$.

A *polyhedron* P is the intersection of a finite number of halfspaces, i.e. P can be defined as a set $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ where A is a matrix of m rows and n columns, and $b \in \mathbb{R}^m$. Moreover, if the polyhedron is bounded, it is called *polytope.* In fact, polytopes are precisely those sets in $\mathbb{R}^n$ which are the convex hulls of finitely many points, i.e. every polytope P can be written as $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and as $P = conv(S)$ $(S \subset \mathbb{R}^n)$. A set C is a *polyhedral cone* if and only if there is a finite system of equations $Ax = 0$ such that $C = \{x \in \mathbb{R}^n : Ax \leq 0\}$.

Given $S \subseteq \mathbb{R}^n$, x_0 is an *extreme point* if and only if $x_0 \in S$ and when $x_0 = \lambda x_1 + (1 - \lambda) x_2$ with $x_1, x_2 \in S$ and $\lambda \in (0, 1)$, then $x_1 = x_2$. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a polyhedron, it is said that a point $x_0 \in P$ is an *extreme point* of P if and only if x_0 is the intersection of n linearly independent hyperplanes from the set defining P.

Let $d \in \mathbb{R}^n$ be a *direction* (ray) of S if and only if there is a point $x_0 \in S$ such that $x_0 + \lambda d \in S$, for all $\lambda \geq 0$. d is an *extreme direction* of S if and only if d is a direction of S, and when $d = \lambda_1 d_1 + \lambda_2 d_2$ with d_1, d_2 directions of S and $\lambda_1, \lambda_2 \geq 0$, then there exits

$\mu \in \mathbb{R}$ such that $d_1 = \mu d_2$. Thus, a vector d' is a extreme direction of the polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ if and only if $d' \in \{d \in \mathbb{R}^n : Ad \leq 0\}$.

A valid inequality $a^T x \leq a_0$ for P defines a subset F of P, called *face*, if $F = \{x \in P : a^T x = a_0\}$. A face F is called *proper* if $F \neq P \neq \emptyset$. If a inequality $a^T x \leq a_0$ defines a face F, and another inequality $a'^T x \leq a'_0$ defines a face F' such that $F \subset F'$ and $F \neq F'$, then it is said that $a^T x \leq a_0$ is *dominated* by $a'^T x \leq a'_0$.

The dimension, $dim(P)$ of a polyhedron P is the maximum number of affinely independent points in P minus 1.Thus, faces of dimension 0 are called extreme points, faces of dimension 1 are called edges, and faces of dimension $dim(P)1$ are called *facets.*

In particular, a linear programming problem can be modeled as follows

$$\min c^T x$$

subject to:

$$Ax \leq b$$

where vector $c \in \mathbb{R}^n$ is called the vector of *costs*, A is a matrix of m rows and n columns, $b \in \mathbb{R}^m$ and vector $x \in \mathbb{R}^n$ is the vector of *variables* (see Chvátal (1983) and Bazaraa et al. (2010) for further information). Thus, the set of feasible solutions is a face of the polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, and if c_0 is the optimum value of $\max \{c^T x : x \in P\}$, then $c^T x \leq c_0$ is a supporting inequality for P, and the set $F = \{x \in P : c^T x = c_0\}$ of the optimal solutions is a face of P. If P is non-empty, then every face contains a vertex and this implies that every linear program over a polytope has at least one optimum vertex solution.

1.4 Some Combinatorial Optimization Problems

The polyhedral description of Combinatorial Optimization problems is usually given as the convex hull of a finite set of points, yet in order to apply Linear Programming techniques, polyhedral description have to be given in the form $\{x \in \mathbb{R}^n : Ax \leq b\}$. In this way, it is necessary to explain some concepts which relate Graph Theory to Polyhedral Theory

Let E be a finite set (e.g. the edge set of the graph $G = (V, E)$) where each element $e \in E$ has associated a cost c_e and a component x_e of a vector $x \in \mathbb{R}^{|E|}$ indexed by e. Let $\mathcal{H}$ be a collection of subsets in E, called collection of feasible solutions. For each subset $F \subset E$, it is defined its *incidence vector* $x^F \in \mathbb{R}^{|E|}$ as follows

$$x_e^F = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{if } e \notin F \end{cases}$$

Thus, it is possible to associate the collection of subsets $\mathcal{H}$ with the polytope $P_{\mathcal{H}}$, which is the convex hull of all incidence vectors of $\mathcal{H}$, i.e. $P_{\mathcal{H}} = conv\left\{x^F \in \mathbb{R}^{|E|} : F \in \mathcal{H}\right\}$. Now, a combinatorial optimization problem is to find a feasible solution of $\mathcal{H}$ with minimum cost. Clearly, each feasible solution $F \in \mathcal{H}$ corresponds to a vertex of $P_{\mathcal{H}}$ and vice versa. Therefore, it is possible to solve this computational problem as the following linear programming problem

$$\min\left\{c^T x : x \in P_{\mathcal{H}}\right\}$$

In order to use linear programming techniques, it is necessary to know the system of linear inequalities described for the polyhedron $P_{\mathcal{H}}$. For almost all combinatorial optimization problems, finding the complete description of $P_{\mathcal{H}}$ is a difficult task, but it can be a great computational help to have a partial description of those inequalities and add new ones when they are necessary.

1.4.1 The Travelling Salesman Problem

The *Travelling Salesman Problem* (TSP) arises in 1930, and it is the most extended combinatorial optimization problem. It is the benchmark problem for new algorithmic ideas as well as the basis for new problems on this field. The TSP has a very simple approach, and although it is difficult to solve, there are many exact and heuristic algorithms that helps us to get good solutions. The most popular formulation was presented by Dantzig et al. (1954) providing a solution for a problem with 49 cities. Besides, many studies put in place tools to solve large instances.

Let a set of n cities where the travel cost from one city to another is assumed to be known. The aim of the TSP is to find the route of minimum cost such that, starting and finishing in the same city, visits every city exactly once. The TSP can be modeled as a undirected capacitated graph (The *symmetric TSP*), where each city is a vertex, and the travel costs are the edge lengths between the vertices. Thus, in graph theory, the problem is to find a Hamiltonian cycle, called *tour*. Without loss of generality, it is supposed that the graph is a complete graph, otherwise it could be replaced the missing edges with edges of large cost. Although in general, it studies the symmetric case, also, it is possible to study the asymmetric case, where the travel cost going from a city to another and the travel cost in the opposite direction are different. This is due to, in real life problems, there are one-way streets, streets closed because of works, etc., hence the symmetric problem is insufficient.

Let $K_n = (V, E)$ be a undirected complete graph, where V is the set of cities with $|V| = n$, and E is the set of edges between cities with $|E| = \frac{1}{2}n(n-1)$. For each edge $e \in E$, $e = [i, j]$, $\forall i, j \in V$, where i is the *origin* city and j is the *destination* city of this edge. Let $T \subset E$ be a *tour* if it is a cycle of length n in K_n. A cycle of length $n_1 < n$ is called a *subtour*. As it is seen above, given $\mathcal{H}$ a collection of subsets in E, if a tour

$T \in \mathcal{H}$, then it has associated a incidence vector x^T and the TSP polytope is the convex hull of incidence vectors of the all tours in K_n. That is

$$P_{\mathcal{H}} = conv\left\{x^T \in \mathbb{R}^{|E|} : T \in \mathcal{H}\right\}$$

A complete description of the polytope $P_{\mathcal{H}}$ as a system linear inequalities is not known, which limits the application of linear programming techniques to solve the TSP. However, this description is a classical topic of polyhedral combinatorics (see Nemhauser and Wolsey (1988) for further information) which has been extensively studied. Indeed, in the last years, many new valid inequalities and facets for $P_{\mathcal{H}}$ have been introduced. See Lawler et al. (1985), Jünger et al. (1995), Naddef and Rinaldi (1993) and Naddef and Rinaldi (2007) for some references.

The most popular formulation as a system linear inequalities was given by Dantzig et al. (1954). It was described as a $0-1$ linear programming model associating each edge e with a travel cost c_e and a binary variable x_e as follows

$$x_e = \begin{cases} 1 & \text{if } e \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

The model proposed is the following

$$\min \sum_{e \in E} c_e x_e \tag{1.1}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V \tag{1.2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \forall S \subset V \tag{1.3}$$

$$0 \leq x_e \leq 1 \qquad \forall e \in E \tag{1.4}$$

$$x_e \text{ integer} \qquad \forall e \in E \tag{1.5}$$

Equations (1.2) are the *degree equations*, ensuring that each city is visited only once. Inequalities (1.3) are the *subtour elimination constraints*, ensuring that the solution contains no Hamiltonian cycles over less than n cities.

Owing to the constraints (1.2), the following equality holds

$$2 \sum_{e \in E(S)} x_e + \sum_{e \in \delta(S)} x_e = 2|S| \qquad \forall S \subset V \tag{1.6}$$

Therefore an alternative formulation can be obtained, as it is easy to conclude that subtour elimination constraints (1.3) are equivalent to the following connectivity constraints

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V \tag{1.7}$$

Connectivity constraints simply ensure that any set of at least two nodes has to be linked. In theory, using the constraints (1.7) instend of (1.3) is irrelevant, but the formulation with (1.7) has been shown stronger than other formulations. Moreover, the constraints (1.3) (or (1.7)) defined by S are equivalents to the same constraints defined by $V \setminus S$. Therefore, constraints (1.7) can be used to reduce the number of variables.

The family constraints has a cardinality growing exponentially with n, hence it is difficult to directly solve it. A possible way of avoiding this is to use cutting-plane procedures which consider a limited subset of constraints, and tries to find violated inequalities to be introduced as new constraints. The faster and the more efficient the identification of this inequalities is, the faster the resolution is.

Valid inequalities for the TSP have been widely studied. The trivial inequalities $x_e \geq 0$ and $x_e \leq 1$ define facet for $n \geq 5$ and $n \geq 4$, respectively. These are not important for the solution procedure, but they are for the complete description of the TSP polytope. Besides, as it was seen, for every $S \subset V$, the connectivity constraints (1.7) define a facet for $n \geq 4$.

Another family of known facets defining inequalities are the *comb inequalities*. These were the first inequalities discovered, in a more restrictive form, by Chvátal (1973) who derived them from the 2-matching constraints of Edmonds (1965). They were generalized to the current form and shown to be facet inducing for the TSP by Grötschel and Padberg (1979). A comb inequality is usually defined by a set $H \subset V$, called *handle*, and an odd number $t \geq 3$ of vertex subsets $T_1, T_2, \ldots, T_t$, called *teeth*, such that

$$\begin{aligned} H \cap T_i &\neq \emptyset & \forall\, 1 \leq i \leq t \\ T_i \setminus H &\neq \emptyset & \forall\, 1 \leq i \leq t \\ T_i \cap T_j &= \emptyset & \forall\, 1 \leq i < j \leq t \end{aligned}$$

The corresponding comb inequality is

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{t} \sum_{e \in \delta(T_i)} x_e \geq 3t + 1 \tag{1.8}$$

The comb inequalities are generalized to several handles producing well-known valid inequalities as *star inequalities*, *path inequalities*, *clique tree inequalities*, *bipartition inequalities*, etc.

1.4.2 The Vehicle Routing Problem

The *Vehicle Routing Problems* (VRP) are another very extended combinatorial problems. A vehicle routing problem was presented by Dantzig and Ramser (1959) as a real application about petrol supply to service stations. A first mathematical formulation and a resolution algorithm were proposed. The VRP are a natural generalization of the TSP, where the customers demand of an amount of products that has to be deliver by a fleet of vehicles. For this reason, the VRP are also a difficult problems to solve, and there are many studies about exact and heuristic algorithms that help us to get good solutions. See e.g. Toth and Vigo (2014) for a textbook on VRP.

Although the *Capacitated Vehicle Routing Problem* (CVRP) has the simplest description, it is the most studied. This consists in finding the minimum cost route that has to make a fleet of identical vehicles to serve the demand of a set of customers, where each vehicle has a limited capacity and has to start and end in a single central depot.It is assumed that the customer demands cannot be split. As in almost all combinatorial problems, the CVRP can be described through a graph, whose edges represent the road sections, and whose vertices correspond to the depot and the customer locations. The graph can be directed or undirected, depending on the features of the real problems to be applied (for instance, if there are one-way streets, it is used a directed graph). For the sake of simplicity, it considers the symmetrical case, where the length to go from one customer to another is the same that in the opposite direction.

Let $G = (V, E)$ be a undirected complete graph, where E is the set of edges between the customers, and V is the set of customers with $|V| = n$, being 0 the depot. For each edge $e = [i, j] \in E$, it is denoted by c_e the length going from i to j. Each customer i is associated with a known nonnegative demand d_i to be delivered, and the depot has associated the demand $d_0 = 0$. Besides, given a set $S \subseteq V$, the total demand of this set is denoted by $d(S) = \sum_{i \in S} d_i$.

The CVRP has a fleet of k vehicles with capacities Q. To ensure the feasibility, it is assumed that $d_i \leq Q$ for all customers. Each vehicle can carry out at most one route, and it is assumed that $k \leq k_{min}$, where k_{min} is the minimum number of necessary vehicles to serve all the customers. Given $S \subseteq V \setminus \{0\}$, it is denoted by $r(S)$ the minimum number of necessary vehicles to serve all customers in S. Clearly, if $S = V \setminus \{0\}$, then $r(S) = k_{min}$. This number can be obtained by solving the *Bin Packing Problem* (BPP), yet this is an NP-hard problem and k_{min} is usually defined by the trivial BPP lower bound

$$k_{min} = \left\lceil \frac{d(S)}{Q} \right\rceil$$

Thus, the aim of the CVRP is to find k simple vehicle routes with minimum cost such that each vehicle starts and ends at the depot, each customer is visited exactly once by an only vehicle, and the capacity Q of each vehicle is not exceed.

Several different formulations have been proposed for the basic version of the VRP in the literature. The most used is the so-called *vehicle flow formulation* that uses $O(n^2)$ binary variables to point out if a vehicle traverses an edge in the optimal solution. There are also formulations based on the so-called *commodity flow formulation* where additional integer variables are associated with the edges representing the flow of the commodities along the paths crossed by the vehicles. Another formulation arises with an exponential number of binary variables, each one associated with a different feasible circuit. This is formulated as a *Set-Partitioning Problem* (SPP) calling for the determination of the collection of circuits with minimum cost, which serves each customer once and satisfies, possibly, additional constraints. However, these models generally require to deal with a very large number of variables.

In the two-index vehicle flow formulation, each edge $e \in E$ has associated a variable defined as follows

$$x_e = \begin{cases} 1 & \text{if } e \text{ is in the route} \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the mathematical model is

$$\min \sum_{e \in E} c_e x_e \tag{1.9}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V \setminus \{0\} \tag{1.10}$$

$$\sum_{e \in \delta(0)} x_e = 2k \tag{1.11}$$

$$\sum_{e \in E(S)} x_e \leq |S| - r(S) \qquad \forall S \subseteq V \setminus \{0\} \tag{1.12}$$

$$x_e \in \{0,1\} \qquad \forall e \notin \delta(0) \tag{1.13}$$

$$x_e \in \{0,1,2\} \qquad \forall e \in \delta(0) \tag{1.14}$$

The *degree equations* (1.10) and (1.11) impose that each customer is visited only once by a single vehicle, and each vehicle moves in and out from the depot exactly once. Inequalities (1.12) are the *subtour elimination constraints*, where $r(S)$ could be replaced by the trivial BPP lower bound. Moreover, these inequalities avoid to exceed the capacity of the vehicle. Note that, when single-routes are not allowed, the edges incident to the depot can be traversed at most once, i.e. $x_e \in \{0,1\}$ with $e = [i,0], \forall i \in V \setminus \{0\}$. On the other hand, when single-customer routes are allowed, the edges incident to the depot can be traversed at most twice, i.e. $x_e \in \{0,1,2\}$ with $e = [i,0], \forall i \in V \setminus \{0\}$. Note that the above described model is taken allowing single-customer routes.

As it was seen in the previous section 1.4.1, it is possible to take an alternative formulation by transforming the constraints (1.12), using the dregree equations (1.10) and

(1.11), into this other version

$$\sum_{e\in\delta(S)} x_e \geq 2r(S) \qquad \forall S \subset V \tag{1.15}$$

These, so-called *capacity constraints*, impose the connectivity of the solution as well as the vehicle requirements. Besides, the constraints (1.12) and (1.15) defined by S are equivalents if it is used $V \setminus S$ instead of S. Likewise these constraints for the CVRP polytope play, in some sense, the same role as the subtour elimination constraints (1.3) for the TSP polytope, as there is an exponential number of them, and all are necessary to define the integer liner programming model. However, while all subtour elimination constraints define facets of the TSP polytope, the same does not always hold for the CVRP polytope.

There is a hierarchy of capacity inequalities for the capacity constraints, all sharing the same left-hand side but with a different right-hand side. The higher the right-hand side value is, the stronger the inequalities are, but its separation procedure is much more difficult. As it was seen above, the right-hand side value can be defined as follows

$$2\frac{d(S)}{Q} \tag{1.16}$$

Therefore, the family of inequalities resultant is called the *fractional capacity constraints*, and it is the weakest family. The separation procedure is easy, and it is solvable in polynomial time. However, they are almost never supported for the CVRP polytope. On the other hand, its linear programming relaxation, obtained by dropping the integrability requirement, yields a bound that can be computed in polynomial time. A *rounded capacity constraint* is obtained by rounding (1.16) to the nearest larger integer. The resulting inequality is

$$\sum_{e\in\delta(S)} x_e \geq 2\left\lceil\frac{d(S)}{Q}\right\rceil$$

The separation procedure is much difficult than the one of the (1.16), but it is still computationally accessible. As it was seen above, a better lower bound on the left-hand side of the inequality is given by twice the solution of the BPP needed to pack the demands of customers in S. This is called the *weak capacity inequality.* Since computing this lower bound is an NP-hard problem, the separation problem is difficult. However, these constraints cannot support the CVRP polytope because it does not take into account the demands outside the set S. A *capacity constraint* is the one taking the right-had side of (1.15) for a set S and is the tightest constraint. Let $\mathcal{P}$ the set of all possible k-partition of $V \setminus \{0\}$. For any set $S \subseteq V \setminus \{0\}$ and for any k-partition $P = \{S_1, \dots, S_k\}$, $r(S)$ can be defined by

$$r(S) = \min_{P\in\mathcal{P}} \beta(P, S)$$

where $\beta(P,S)$ is the number of nonempty intersection of S, and S_i, for all $i = 1,\dots,k$. In fact, it is the number of vehicles needed to satisfy the demands of all customers in S in the k-partition P. This inequality supports the CVRP polytope (by definition), but whether it defines a facet still depends on the demands of the customers d_i, on the number of the vehicles k, and on the capacity of the vehicles Q. Since weak capacity inequalities are a special case of the capacity inequalities, the separation for the last ones is also difficult.

All the families of constraints have a cardinality growing exponentially, and hence it is practically impossible to solve the CVRP directly. A possible way of avoiding this is to use an appropriate *separate procedure* (e.g. cuting-plane procedures), considering only a limited set of these constraints and adding the remaining ones if needed.

The CVRP is known to be NP-hard and generalizes the well-known TSP, calling for the determination of a minimum-cost simple tour visiting all the customers and arising when $Q > d(V)$ and $k = 1$. A k-route and a Hamiltonian cycle have very close structures: they are both connected subgraphs of G, where all nodes have degree 2 except for node 0, whose degree is different in both subgraphs. Indeed, a k-route is a special case of a *tour*. Of course, any inequality valid for all tours of G is also valid for all its k-routes. Therefore, all the inequalities proposed for the TSP are valid for the CVRP. A good example is the well-known family of comb inequalities (1.8), however they can be rather weak for the CVRP polytope. Naddef and Rinaldi (2001) describe them by taking into account whether the depot is in a tooth or not. Given a handle $H \subset V$ and an odd number $t \geq 3$ of teeth $T_1, T_2, \dots, T_r, T_{r+1}, \dots, T_t$ with $T_i \subset V$ for all $i = 1,\dots,t$ satisfying

$$\begin{aligned}
T_i \setminus H &\neq \emptyset && \forall\, 1 \leq i \leq t\\
T_i \cap H &\neq \emptyset && \forall\, 1 \leq i \leq t\\
T_i \cap T_j &= \emptyset && \forall\, 1 \leq i < j \leq r\\
T_i \cap T_j &= \{0\} && \forall\, r+1 \leq i < j \leq t
\end{aligned}$$

where r may be any value between 0 (all teeth intersect) and t (no teeth intersect nor contain the depot). The teeth from T_{r+1} to T_t intersect in the depot. Moreover, it is assumed that $r(V \setminus T_i) = k$, for all $i = r+1,\dots,t$. Then the following comb inequality is valid

$$\sum_{e\in\delta(H)} x_e + \sum_{i=1}^{t} \sum_{e\in\delta(T_i)} x_e \geq t + 1 + 2r + 2k(t-r)$$

Not only do valid inequalities for the TSP are valid for the CVRP, but also there are more valid inequalities taking from other problem. The *path-bin inequalities* are one example that combines structures from the BPP and the TSP by taking into account the demands and the capacity inequalities. The *clique cluster inequalities* are other examples derived from the *Stable Set Problem* (or *Set Packing Problem*). First presented by Augerat

(1995) , are defined by $W_1, \ldots, W_w$ subset of $V \setminus \{0\}$ such that

$$\begin{aligned} W_i \cap W_j = \{v\} &\qquad \forall\, 1 \leq i < j \leq w \\ \sum_{l \in W_i} d_l \leq Q &\qquad \forall\, 1 \leq i \leq w \\ \sum_{l \in W_i \cup W_j} d_l > Q &\qquad \forall\, 1 \leq i < j \leq w \end{aligned}$$

Thus, it can be built a graph for the stable set problem with a vertex, for each subset of customers and with an edge, for each incompatible subsets of customers. Then, the following inequality is valid

$$\sum_{i=1}^{w} \sum_{e \in \delta(W_i)} x_e \geq 4w - 2$$

More well-known inequalities are the *odd hole inequality*, which says that at most t nodes of a chordless cycle of lenght $2t + 1$ can belong to a stable set; the *multistar inequalities*, studied for example by Fisher (1994), Gouveia (1995) and Letchford et al. (2002); and some variants called *homogeneous multistar inequalities* (see Letchford et al. (2002) for further information)

1.4.3 The General Pickup and Delivery Problem

The *General Pickup and Delivery Problems* are an important class of routing problems where a set of products (or customers) have to be transported from an origin to a destination by a fleet of capacitated vehicles. Each vehicle has a start location and an end location. Each product has to be transported by one vehicle from its origin to its destination without any transshipment at other location. Savelsbergh and Sol (1995) presented a survey on pickup and delivery problems until 1995. Berbeglia et al. (2007), Parragh et al. (2008a) and Parragh et al. (2008b) extend this study by presenting general frameworks to model it as well as a classification schemes for these problems.

Three well-known routing problems are cases of the GPDP. The first is the *Pickup and Delivery Problem* (PDP) where each product has only a pickup location and only a delivery location, and all vehicles start and finish in a central depot. The second problem is the *Dial-a-Ride Problem* (DARP) where the quantity transported of each product is one unit (typically the different products are peoples). The last problem is the vehicle routing problem (seen in Section 1.4.2) where all origins or all destinations are located at the depot.

Most existing pickup and delivery problem can be defined within the following general framework. Let $G = (V, A)$ be a directed complete graph, where A is the set of arcs between the locations, and V is the set of locations with $|V| = n$, being 0 the depot. For each arc $a = (i, j) \in A$ it is denoted by c_a the length going from i to j satisfying the triangle inequality. Let H be the set of p products. Each vertex, including the depot, can either need or supply a non-negative amount of each product. Let $D = (d_{ih})$ denote a product matrix where a positive d_{ih} is the quantity of h supplied by vertex i, and $-d_{ih}$ is the quantity of h required by vertex i if $d_{ih} < 0$. It is assumed that $\sum_{i \in V} d_{ih} = 0$ for each product $h \in H$. That is, for each product the total supply and the total demand are balanced. Let H_h^+ and H_h^- be the set of all origins and destinations of the product h, respectively. Then, it defines $d_h = \sum_{i \in H_h^+, d_{ih} > 0} d_{ih} = -\sum_{i \in H_h^-, d_{ih} < 0} d_{ih}$. Moreover, H^+ and H^- define the set of all origins and all destinations, respectively, and $V = H^+ \cup H^-$. Let M be the set of m vehicles where each vehicle k has a capacity Q_k, a start location k^+, and an end location k^-. Let W be the set of start and end locations of the vehicles. A vehicle can either pick up or deliver the entire amount of a product. A route is a circuit over some vertices, starting and finishing at the depot. The general pickup and delivery problem consist of building at most m vehicle routes such that

- all pickup and delivery requests are satisfied;
- all origins of a product are visited before their destinations.
- no transshipments of commodities are made;
- the load of a vehicle never exceeds its capacity;
- the sum of route costs is minimized.

For all $h \in H$, taking $|W| = 1$ and $|H_h^+| = |H_h^-| = 1$, it has the pickup and delivery problem. In this case it defines h^+ as the unique element of H_h^+, and h^- as the unique element of H_h^-. The dial-a-ride problem is addressed when $|W| = 1$, $|H_h^+| = |H_h^-| = 1$ and $d_h = 1$, for all $h \in H$. Finally, the VRP is the case when $|W| = 1$, $|H_h^+| = |H_h^-| = 1$, for all $h \in H$, and $N^+ = W$ or $N^- = W$.

Different objective functions have been referenced in the literature such as minimize duration, completion time, travel time, route length, customer inconvenience, the number of vehicles and/or maximize profit.

1.5 Benders' Decomposition

The *Benders' Decomposition* is a popular approach in solving large linear programming problems. Since computational difficulty of these problems that increases exponentially with the number of variables and constraints, instead of considering all of them, Benders' decomposition divides the problem into several smaller problems. This can be more efficient than solving a single large problem.

In Benders' decomposition, iteratively, a master problem is solved for a subset of variables, and using the values obtained, a subproblem with the remaining variables is solved.

Then, one or more constraints are generated and added to the master problem, which is then re-solved.

Given the following linear problem

$$\begin{aligned} \min\quad & c^T x + f^T y \\ \text{s.t.}\quad & Ax + By \geq b \\ & y \in Y \\ & x \geq 0 \end{aligned} \tag{1.17}$$

such that x and y are vectors, Y is a polyhedron, A and B are matrices, and b, c and f are vectors having appropriate dimensions. Supposing that variables y are 'complicated variables' in the sense that the problem becomes significantly easier to solve, if these are fixed, then the problem can be written in term of variable y

$$\begin{aligned} \min\quad & f^T y + q(y) \\ \text{s.t.}\quad & y \in Y \end{aligned} \tag{1.18}$$

where $q(y)$ is defined to be the optimal value of

$$\begin{aligned} \min\quad & c^T x \\ \text{s.t.}\quad & Ax \geq b - By \\ & x \geq 0 \end{aligned} \tag{1.19}$$

Note that if (1.19) is unbounded for some y, then (1.18) is also unbounded, as well as the original problem (1.17). Therefore, if this does not happen, it is possible to use its dual problem to solve it. Thus, (1.19) can be written as follows

$$\begin{aligned} \max\quad & \alpha^T (b - By) \\ \text{s.t.}\quad & A^T \alpha \leq c \\ & \alpha \text{ unrestricted} \end{aligned} \tag{1.20}$$

Note that the feasible region of the dual model does not depend on y. Thus, if this region is empty, either the primal problem is unbounded for some $y \in Y$ and then, the original problem (1.17) is unbounded, or the primal problem is also empty for all $y \in Y$ and then, (1.17) is infeasible. Considering then that feasible region is not empty, it can be taken all extreme points $(\delta^1, \dots, \delta^p)$ and all extreme rays $(\lambda^1, \dots, \lambda^q)$, where p and q are the number of extreme points and extreme rays, respectively. Because of the Farkas' Lemma, for a given vector $\overline{y}$, the dual problem (1.20) can be solved by checking whether $(\lambda^j)^T (b - B\overline{y}) \geq 0$. As the dual problem is assumed bounded, this is equivalent to solve the following problem

$$\max_{i=1,\dots,p} \ (\delta^i)^T (b - B\overline{y})$$

that can be reformulated as follows

$$\begin{aligned} \min \quad & u \\ \text{s.t.} \quad & (\delta^i)^T(b - B\overline{y}) \leq u \quad \forall i = 1, \ldots, p \\ & (\lambda^j)^T(b - B\overline{y}) \leq 0 \quad \forall j = 1, \ldots, q \\ & u \text{ unrestricted} \end{aligned}$$

Now this can replace $q(y)$ in (1.18), obtaining a reformulation of the original problem (1.17) called the *Master Problem*

$$\begin{aligned} \min \quad & f^T y + u \\ \text{s.t.} \quad & (\delta^i)^T(b - B\overline{y}) \leq u \qquad \forall i = 1, \ldots, p \\ & (\lambda^j)^T(b - B\overline{y}) \leq 0 \qquad \forall j = 1, \ldots, q \\ & y \in Y, u \text{ unrestricted} \end{aligned} \tag{1.21}$$

The advantage of this reformulation is that the variables x have been changed by only one u, but the number of constraints have greatly increased. However, using a linear programing method only useful constraints may be considered. Thus, Benders' decomposition starts with a subset of these constraints and solves a *relaxed master problem*. Without loss of generality, the problem can be assumed bounded by introducing a dummy constraint with a large enough right-hand side. If the optimal solution depend on the dummy constraints, then (1.18) is unbounded. If it is not feasible, then (1.21) (and also (1.17)) is not feasible. Otherwise, it yields a candidate optimal solution (y^*, u^*). Now the dual problem (1.20) solves to obtain $q(y^*)$:

- If it is unbounded, there is a extreme ray λ^j such that $(\lambda^j)^T(b - By^*) > 0$. Hence, this corresponding constraints associated with λ^j is added to the relaxed master problem, which is then re-solved.
- If it is not feasible, then (1.18) is not feasible or unbounded.
- If there is an optimal solution $q^*(u^*)$ such that
  - $q(u^*) > u^*$, there is a extreme point δ^i such that $(\delta^i)^T(b - B\overline{y}) > u$. Hence, this corresponding constraints associated with δ^i is added to the relaxed master problem.
  - $q(u^*) \leq u^*$ then the optimal solution of (1.18) has been found and the algorithm stops.

As p and q are finite, and new feasibility or optimality cuts are generated in each iteration, this method converges to an optimal solution in a finite number of iterations.

1.6 The branch-and-cut method

The *branch-and-cut algorithm* is the most popular approach to solve Integer Linear Programming problems. In fact, it is a *branch-and-bound method* based on the resolution of a linear relaxation, appropriately strengthened with *cutting-planes*. Use of facets defining cutting-planes and the automatic generation of these, in conjunction with the branch-and-bound method, were formulated and successfully applied for the first time by Grötschel et al. (1984) for the Linear Ordering Problem. The term *branch-and-cut* was introduced by Padberg and Rinaldi (1991) as an algorithm for the TSP. This was defined to add only facet from the polytope, although in most cases this is not required.

The algorithm starts using a *linear relaxation* (denoted by LP) of an integer linear program (denoted by ILP) in which the condition that all variables have to be integers has been eliminated. For example, in the TSP model (1.1)-(1.5), the constraints (1.5) are removed. If z_{ILP} is the optimal solution of the integer linear program, and z_{LP} is the optimal solution of its linear relaxation (in the minimization case), then it holds $z_{LP} \leq z_{ILP}$, i.e. z_{LP} is the lower bound of the optimal solution z_{ILP}.

The cutting-plane algorithm works as follows. For each iteration h ($h \geq 0$), let $LP(h)$ be the relaxation problem. Solving $LP(h)$ yields a solution $x^*_{LP(h)}$. If it is integral, it corresponds to a feasible solution representing an optimal solution; otherwise, it is run a so-called *separation algorithm* that seeks a violated valid inequality by $x^*_{LP(h)}$ to be added in $LP(h)$. Then, if one violated inequality is found, it is added to the $LP(h)$ and solved a new relaxation problem $LP(h+1)$. Thus, the separation algorithm stops when all the violated valid inequalities are found, and ends up with an optimal solution. Note that, if $z_{LP(h)}$ is the optimal value of $LP(h)$, it holds $z_{LP(h)} \leq z_{LP(h+1)} \leq z_{LP(\infty)} \leq z_{ILP}$, where $LP(\infty)$ is the strengthened relaxation problem with additional valid inequalities. Therefore, if $z_{LP(\infty)}$ is integral, then it is an optimal solution of the ILP, i.e. $z_{PL(\infty)} = z_{ILP}$; otherwise the *branch* starts, i.e. the problem is decomposed into two new problems, for example, by adding upper and lower bounds to a variable whose current value is fractional.

The branching process is done as in a branch-and-bound method. That is, it is chosen a variable x_e with fractional value x^*_e in the relaxed linear problem to build two new problems (the integer variables are indexed by a set E where $e \in E$). Thus, both problems are solved by adding the constraints $x_e \geq \lceil x^*_e \rceil$ and $x_e \leq \lfloor x^*_e \rfloor$. Now, the algorithm builds a tree structure in which each node corresponds with a new linear programming problem. The branching process finishes when a integer solution (improving the current optimal integer solution) is found or if the value of the optimal solution is greater than the current upper bound.

Note that the branch-and-bound algorithm and the cutting-plane method produce benefits. Branch-and-bound is a divide and conquer approach to solve a problem by dividing into smaller problems. The local solution of a subproblem gives a lower bound of the solution for the original problem, thus the highest such a local solution obtained so far is a global lower bound for the original problem, and it is used to avoid parts of the tree that cannot produce the optimal value. On the other hand, a pure branch-and-bound approach can be sped up considerably by using a cutting-plane method, because this carries to a considerable reduction in the size of the tree.

1.7 Background on Heuristic Methods

Heuristic algorithms for the TSP can be divided into two clasees: *construction procedures*, which build a tour by successively adding a new node at each step; and *improvement procedures*, which start from an initial tour and seek a better one by iteratively moving from one solution to another, according to a given neighbourhood structure. Combined, such approaches yield *composite procedures* that attempt to obtain better solutions by applying an improvement procedure to a solution given by a construction procedure. Often, the success of these algorithms depends heavily on the quality of the initial solution, and a powerful design of neighbourhood structures. See Rego and Glover (2007) for further information.

1.7.1 The k-exchange Neighbourhood

Fundamental neighbourhood structures for the TSP (and for several other classes of graph-based permutation problems) are based on *edges-exchanges* procedures. A classical procedure of this type is the k-exchange. This terminology derives from methods initially proposed by Lin (1965) to find so-called k-opt TSP tours.

The 2-exchange (2-opt) procedure is the simplest method and frequently used in combinatorial problems that seek to obtain optimal circuits (or cycles) in graphs. This includes the TSP and its extensions to the wider classes of assignment, routing and scheduling problems. This procedure is a local search improvement method thus, a initial feasible solution is required to initiate the approach. It consists of removing two arcs (v_i, v_{i+1}) and (v_j, v_{j+1}) from the tour and reconnecting the two paths created by adding (v_i, v_j) and (v_{i+1}, v_{j+1}). In order to keep a consistent orientation of the tour, one of the two subpaths remaining after dropping the first two edges must be reversed. The cost difference produced by a 2-opt move is defined as $\Delta_{ij} = d(v_i, v_j) + d(v_{i+1}, v_{j+1}) - d(v_i, v_{i+1}) - d(v_j, v_{j+1})$. Finally, a 2-opt solution is obtained by iteratively applying 2-opt moves until any possible movement yields a negative value of the difference Δ.

Another movement typically used is the 3-opt. By direct analogy to the 2-opt move, this consists of removing three arcs from the solution and reconnecting the paths by adding three arcs in the best possible way. As in 2-opt move, the cost difference of a 3-opt move can be computed as the sum of the costs of the added edges minus the sum of the costs of the deleted edges, where a negative difference is an improvement move. However, unlike 2-opt move, there are several ways to reconnect the three subpaths.

The 2-opt and 3-opt moves can be generalized to perform k-opt moves that drop some k edges in a tour and add k new edges. For small values of k, with respecto to n, the verification of k-optimality yields an $O(n^k)$ time complexity, and therefore k-opt moves for $k > 4$ is not used, except some cases.

1.7.2 The Nearest Neighbourhood Heuristic

The *Nearest Neighbourhood Heuristic* is the most simple constructive method to build a tour and consists of successively adding a new node at each step. Given a set of n nodes, the algorithm stars selecting randomly a node x_1. Inductively, suppose $i < n$ and $T_i = (x_1, x_2, \ldots, x_{i-1})$ being the current partial tour. Now it is chosen, from all nodes not selected yet, the next node x_i to be the nearest node to x_{i-1}. If $i = n$ then it adds the edge (x_n, x_1), and the algorithm stops; otherwise, it adds the edge (x_{i-1}, x_i) and looks for the next node x_{i+1}.

1.8 The Variable Neighbourhood Search

The *Variable Neighbourhood Search*(VNS) (see Mladenović and Hansen (1997)) is a local search-based metaheuristic which exploits many different neighbourhoods to escape from bad local optima. Contrary to most other local search methods, VNS does not follow a trajectory, but explores increasingly distant neighbourhoods of the current incumbent solution, and jumps from them to a new one if and only if an improvement was made. Moreover, a local search routine is applied repeatedly to get from these neighbourhoods a local optima.

Let N_k be a finite set of pre-selected neighbourhood structures for $k = 1, \ldots, k_{max}$ and with $N_k(x)$ being the set of solutions in the k^{th} neighbourhood of x. The scheme of the basic VNS algorithm is:

- *Step 1*: Select the set of neighbourhood structures (often nested) N_k, for $k = 1, \ldots, k_{max}$, that will be used in the search
- *Step 2*: Select a initial solution x as input data.
- *Step 3*: Set $k = 1$
- *Step 4*: Choose a random solution x' from $N_k(x)$.

- *Step 5*: Apply some local search method with x' as a initial solution; denote with x'' the obtained local optimum.
  - If x'' is better than x, the new incumbent has been obtained. Then set $x = x''$ and go to step 4.
  - Otherwise, set $k = k + 1$. If $k = k_{max}$ the algorithm stops; otherwise, go to step 4.

Other stopping condition that it can be applied in the algorithm are e.g. maximum number of iterations, maximum CPU time allowed, or maximum number of iterations between two improvements. Different neighbourhood structures can be exploited in both deterministic and stochastic ways. In order to avoid cycling, stochastic approach is used in the step 4 where the point x' is generated at random. A deterministic approach can be used in the local search. In general, the neighbourhood structures used for this local search are independent of the neighbourhood structures N_k, for $k = 1, \ldots, k_{max}$, that is, it does not necessarily belong to them. In addition, an extended version of VNS could contain more than one neighbourhood in the local search. Note that each local search could use *first improvement* (a move made when an improvement in the neighbourhood is found) or *best improvement* (a move to the best solution in the neighbourhood). The latter is also known as *steepest descent*.

The *Variable Neighbourhood Descent* (VND) is a well-known variant of the basic VNS. The difference with respect to the basic VNS is that the VND takes the best solution x' from $N_k(x)$ instead of a random one in the step 4. Another variant is the *Reduce Variable Neighbourhood Search* (RVNS) where random points are selected from $N_k(x)$, no local search is applied, and the values of these new points are compared with that of the incumbent and an update takes place in case of improvement. RVNS is useful for very large instances for which local search is costly.

Unlike many other metaheuristics, the basic schemes of VNS and its extensions are simple, and they require few, and sometimes no parameters. Therefore, in addition to providing very good solutions, often in simpler ways than other methods, VNS can lead to more efficient and sophisticated implementations as well as be applied to a variety of problems.

Chapter 2

Related Problems

This chapter focuses on the *Split-Demand One-Commodity Pickup-and-Delivery Travelling Salesman Problem* (SD1PDTSP) and some related problem. This problem is a combination of three well-known vehicle routing problems of the literature. The CVRP (described in Section 1.4.2), aiming at designing the routes for a vehicle to deliver a commodity from the depot to a set of customers. The *Split Delivery Vehicle Routing Problem* (SDVRP), presented in Section 2.3, which allows a customer to be visited more than once if convenient. And the *One-Commodity Pickup-and-Delivery Travelling Salesman Problem* (1-PDTSP), described in Section 2.2, which allows more than one pickup location although it still moves one commodity. Thus, using elements of these three known problems, the SD1PDTSP is defined in Section 2.1. Besides, Section 2.4 presents other related problems.

2.1 The Split-Demand One-Commodity Pickup-and-Delivery TSP

The SD1PDTSP is an original problem in which a finite set of locations is given, and the travel distance (or cost) from one location to another location is assumed to be known. One specific location is considered to be a depot, and the other locations are identified as customers. The latter are divided into two types of customers depending on what kind of service is required. *Pickup customers* are those provide a given demand of a *single commodity* (the product), and *delivery customers* are those require a given demand of the product. Thus, a product unit collected from a pickup location can be supplied to any delivery location. It assumes that there is *one* vehicle with a given capacity, originally at the depot, that must visit each location through a route to move the commodity, and satisfy all the customers demands. A route consists of paths, starting from and ending at the depot. Each path is called here *trip*. Hence, a route is a set of trips that cover each customer *at least* once. While following the route, the vehicle can either deliver or collect product in each location. All the visits to a customer must end up with exactly its required demand. Although it may be convenient to serve a customer with several visits, it is also reasonable to limit the maximum number of allowed visits with a parameter. When this parameter is one for all customer, it talks about the *split-forbidden* variant. As any customer, the depot, in the problem definition, is allowed to be visited several times by the vehicle through the route. The number of visits to the depot is the number of trips in the route and it is limited to at most a given parameter. The SD1PDTSP is the problem of finding a minimum cost route for the capacitated vehicle such that it satisfies the demand of all customers. Note that such a route may not exist, and in general, checking whether a feasible solution exists is an NP-complete problem due to the limitation in the number of visits to each location and the vehicle capacity.

This thesis deals with the single-vehicle case, although the results could be adapted to the multi-vehicle case. Moreover, the initial load of the vehicle when leaving the depot is a decision that must be computed within the optimization problem. The depot is also treated as a dummy customer, providing (or absorbing) the sum of the customers' demands so the balance of the commodity in the system is zero. As it is discussed in Section 3.2, the model and algorithm proposed in this dissertation for the SD1PDTSP can easily be adapted to several variants, including the one where the vehicle is required to leave the depot with full (or empty) load, the one where customers with zero demand may be not visited, and the one where not all customers must be visited and the demand of the depot is a decision variable.

Although it could make sense to require that product collected in a trip must be delivered in the same trip, it does not impose this requirement in the SD1PDTSP. Therefore all trips are considered to be performed by a single vehicle sequentially. The problem name contains the words “Travelling Salesman Problem” (TSP) to emphasize this assumption. When the depot is the only pickup location, as on VRP instances, or when the vehicle is forced to leave the depot with full load, then the trips may be executed in parallel when a fleet of identical vehicles are available. In general, however, product collected from a customer may go through the depot before being delivered in another customer and therefore, the trips cannot be performed in parallel by different vehicles. It is discussed in Section 3.2.4 how to adapt the mathematical model to ensure that products collected in a trip are also delivered in the same trip.

In the SD1PDTSP each location is assumed to have a known inventory of the product before starting the vehicle service. Also, each location is associated with a desired inventory that it must have after the last vehicle service. The difference between the inventories is the demand of the location. Inventories in a location can be reduced or increased during intermediate visits of the vehicle with the only constraint of considering a given capacity associated with the location. In other words, it is allowed *preemption* in the SD1PDTSP, i.e. product units collected in a location can be unloaded (fully or partially) at any intermediate location to be picked up later and delivered in another location. In Section 3.2.2 it will be observed how to adapt the model and the algorithm described in this dissertation for the *non-preemptive* variant. It does not consider inventory holding cost in the SD1PDTSP. See e.g. Coelho et al. (2014) for a recent survey on the Inventory-Routing Problem.

A practical application of SD1PDTSP arises in the context of a *self-service bike-sharing system*, where every night a capacitated vehicle has to reallocate the bicycles between the stations in the district of a city. The logistic problem aims at finding a route for the vehicle with a minimum travel cost to restore the initial configuration of the system (see e.g. Chemla et al. (2013), Raviv et al. (2013) and Dell'Amico et al. (2014) for further information). Another application occurs in the context of inventory repositioning, where a set of retailers is geographically dispersed in a region. Often, due to the random nature of the demand, some retailers have an excess on inventory of a product while others need additional stock. In many cases, the company may decide to transfer inventory from retailers with low sales to those with high sales. Determining the cheapest way to transfer a given stock with a single vehicle is the SD1PDTSP.

As far as it is known no research has been conducted previously on the SD1PDTSP, however in the literature there are other problems very closely related to it. Further down some of them are shown, starting with the closely related problem involving the split-demand forbidden, then the problem with split-demand, an only one pickup location (the depot) and several vehicles, and finally related problems involving uncapacitated vehicle, several commodities, preemption, etc.

An example

In order to visualize the effect of allowing or not *split demand* and *preemption*, three problems are taken based on a benchmark instance introduced in Mosheiov (1994). This instance has 25 customers and demands between -7 and 7. Note that the definition of the SD1PDTSP does not require the vehicle to leave (or enter) the depot with a pre-specified load (e.g. empty or full initial load). In fact, it assumes that the depot can supply the vehicle with any extra initial load, which is unknown value in the problem (see Section 3.2.1 for details). Figures 2.1, 2.2 and 2.3 show optimal routes for these three problems. Each circle represents a customer and is located using given coordinates. The travel cost between two customers is defined as the Euclidean distance. The demand of a customer is the value near the circle representing that customer. Positive values are associated with pickup customers and negative values with delivery customers. The vehicle capacity is assumed to be 7. This is the smallest value for the capacity where it can be obtained an optimal solution without split-demand or preemption. The load of the vehicle when leaving customer 1 (depot) is 6. Figures 2.1 and 2.3 correspond to the SD1PDTSP where the maximum number of visits allowed to serve a customer is 1 and 2, respectively. Figure 2.2 corresponds to the SD1PDTSP where the maximum

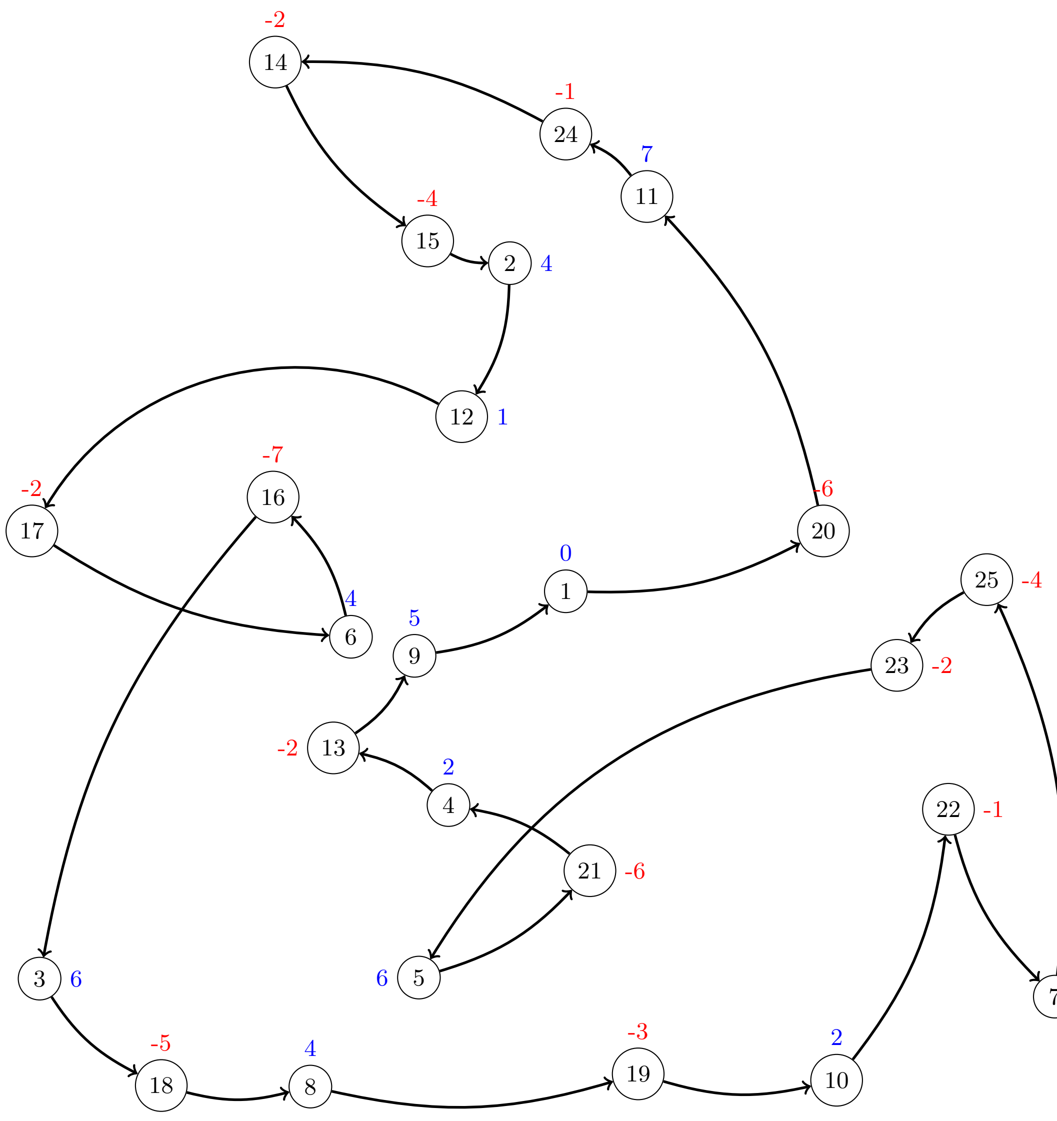
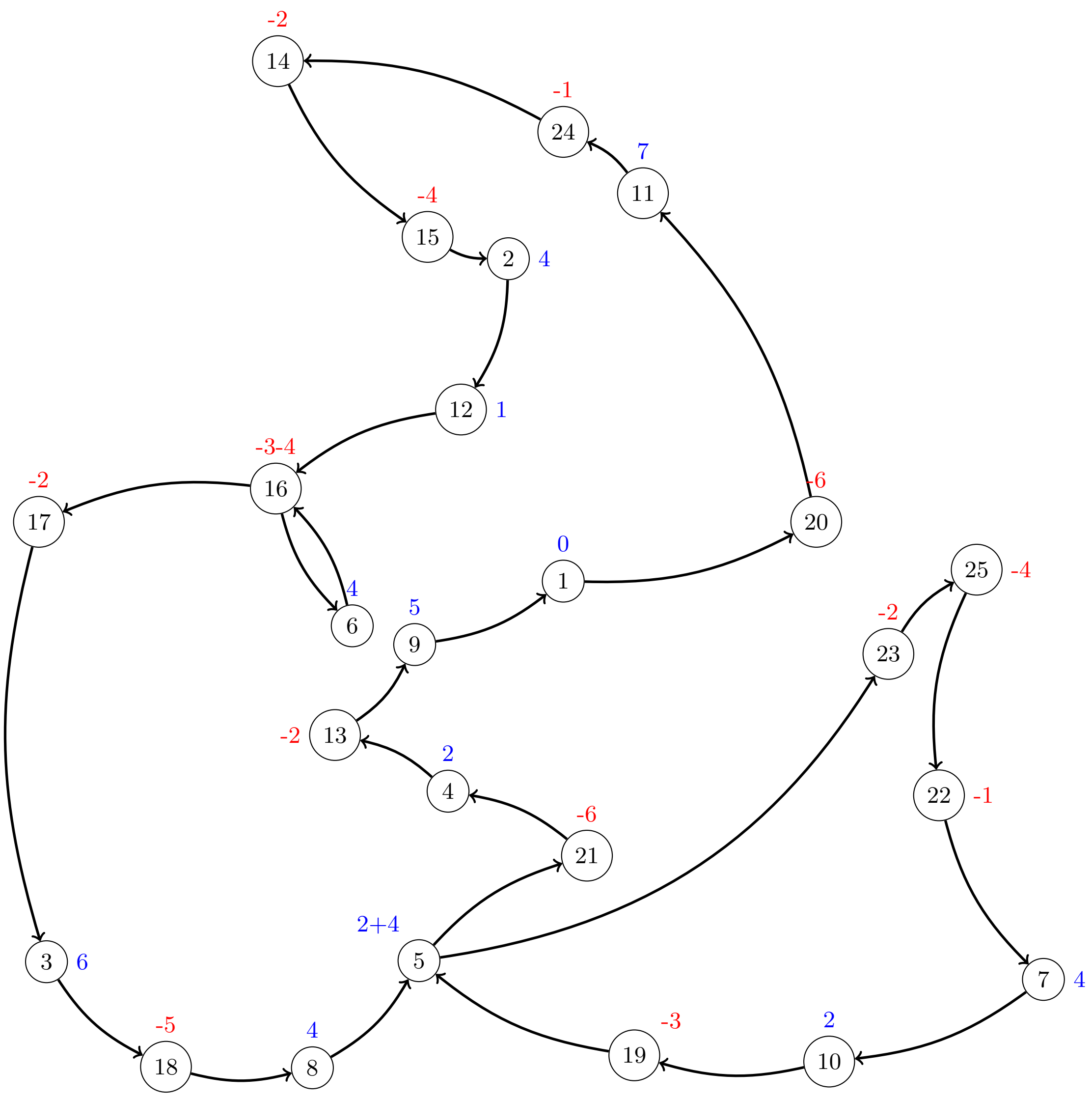


FIGURE 2.1: SD1PDTSP solutions when the vehicle capacity is 7 and split-demand is forbidden. The cost is 5741.

number of visits allowed to serve a customer is 2, and with the additional constraint that preemption of the product in a customer is not allowed. These figures show that the three problems on the same data may have different optimal objective values, thus the travel cost may be reduced by allowing split-demand and preemption.

2.2 The One-Commodity Pickup-and-Delivery Travelling Salesman Problem

The SD1PDTSP is an immediate generalization of the so-called *One-Commodity Pickup-and-Delivery Travelling Salesman Problem* (1-PDTSP) considered in e.g. Hernández-Pérez and Salazar-González (2004b), and where each location must be visited exactly

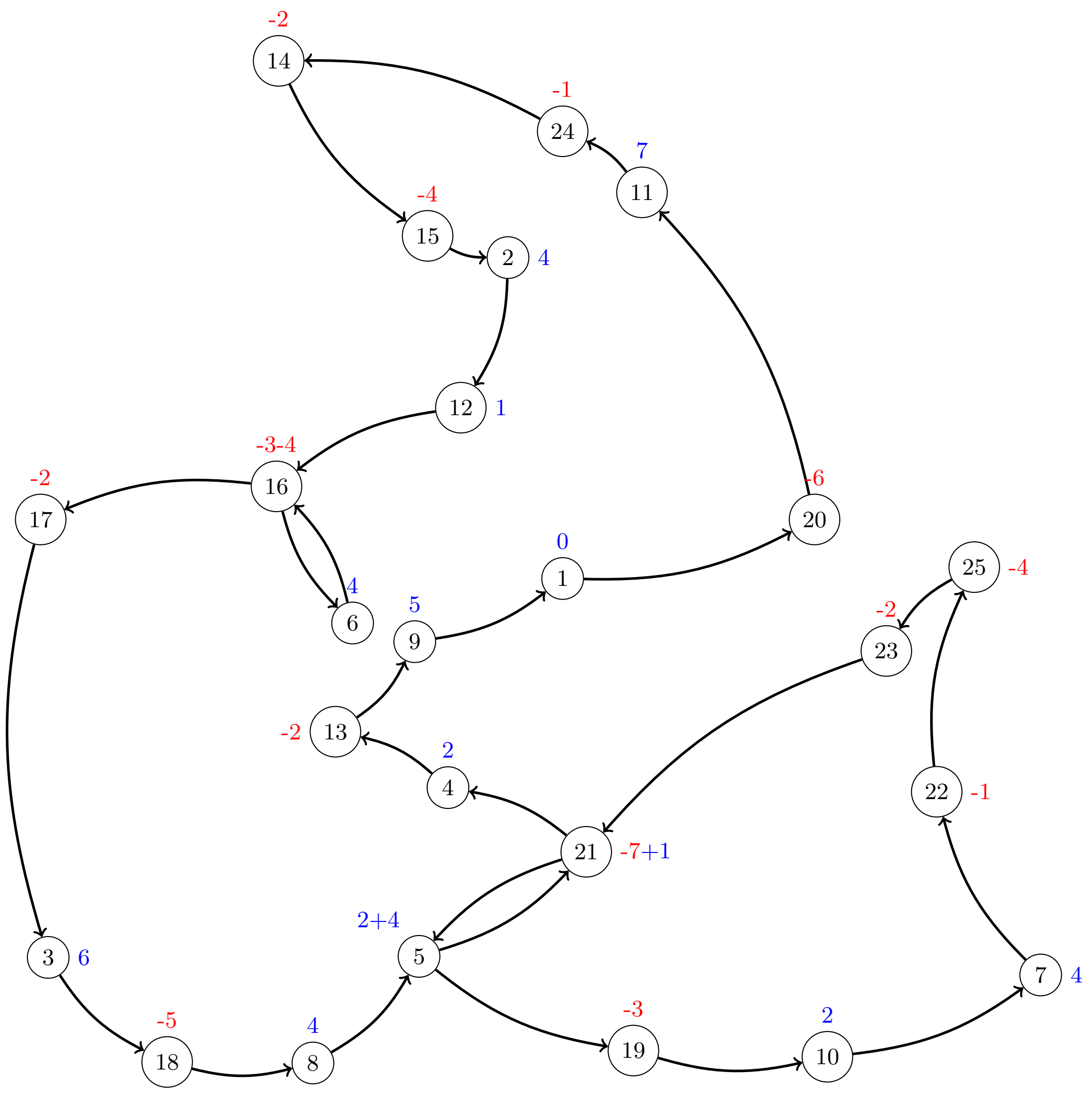


(b) Optimal route when split-demand is allowed and preemption is forbidden. The cost is 5338.

Figure 2.2: SD1PDTSP solutions when the vehicle capacity is 7, split-demand is allowed and preemption is forbidden. The cost is 5338.

once. It is worth mentioning that, when assuming an unlimited number of trips, finding a feasible VRP solution is trivial (e.g. trips consist of single customers). However, for the 1-PDTSP, even checking whether a feasible solution exists is a strongly NP-complete problem (see Hernández-Pérez and Salazar-González (2004b)). Therefore, allowing split-demand on the 1-PDTSP (and so, a location is allowed to be visited several times) has not only the advantage of reducing the travel cost of the route, but it may also help to find routes with respect to the split-forbidden variant.

Hernández-Pérez and Salazar-González (2004a) introduce the problem as a $0-1$ integer linear programming model, both for the asymmetrical and symmetrical case. It is also described a branch-and-cut algorithm for finding an optimal solution, where an initial heuristic based on the TSP nearest insertion is implemented and capacity constraints



(c) Optimal route when split-demand and preemption are allowed. The cost is 5337.

Figure 2.3: SD1PDTSP solutions when the vehicle capacity is 7, split-demand and preemption are allowed. The cost is 5337.

are introduced using a cutting-plane procedure. Later, Hernández-Pérez and Salazar-González (2007) improve this algorithm adapting some valid inequalities from CVRP as comb inequalities, multi-star inequalities, clique cluster inequalities, and the so-called rounded Benders' cut, obtained from the inequalities capacity constraints and the degree constraints.

Several heuristic approaches have been also studied for the 1-PDTSP. Hernández-Pérez and Salazar-González (2004b) propose two heuristic approaches. One is based on a greedy algorithm where it is defined an infeasiblility function that it allows to extend different TSP greedy procedures to (possibly) build an initial 1-PDTSP solution, and it is improved with a k-optimality criterion. The other one is based on a simple local search procedure developed to provide an initial upper bound for the branch-and-cut algorithm.

Other heuristic algorithms for the 1-PDTSP are in Hernández-Pérez et al. (2009) and Zhao et al. (2009). The first article proposes a hybrid algorithm that combines the GRASP and variable neighbourhood descent metaheuristics. The second article presents a genetic algorithm which uses several local searches and a pheromone-based crossover operator. More recently, Mladenović et al. (2012) describe a *general variable neighbourhood search* which combines extended neighbourhood structures from the TSP such as k-opt, double bridge and insertion operators, and a binary indexed tree data structure. This procedure is currently the best approach for large-sized 1-PDTSP instances, both in computational time and solution quality.

2.3 The Split Delivery Vehicle Routing Problem

The SD1PDTSP assumes that the demand at a location can be split (i.e. served by different visits of the vehicle). The new aspect makes the problem harder as the demand served by the vehicle in each visit is now unknown. This has been addressed on the SDVRP by several authors (see e.g. Archetti and Speranza (2012)). The SDVRP is the particular case of the SD1PDTSP where the depot is the pickup location only, the customers are delivery locations, and the parameter limiting the number of visits to a location is unlimited. An interesting difference of the SDVRP with respect to the VRP is that there exists always a SDVRP route with a number of trips equal to the total demand divided by the vehicle capacity, rounded up to the nearest integer. The problem was introduced by Dror and Trudeau (1990) showing that when the number of trips is unlimited and the travel cost satisfies the triangular inequality, there exists an optimal SDVRP solution where two trips share at most one customer. Moreover, they show empirically that allowing split deliveries can lead to substantial cost savings, while in Archetti et al. (2008a) this saving depends on the characteristic of the instance. Archetti et al. (2006a) show that there exists an optimal SDVRP solution where the sum of the number of splits over all customers is less than the number of trips, and Archetti and Speranza (2008) show that the optimal value of the VRP may be twice the optimal value of the SDVRP. There are many articles in the SDVRP literature, especially on heuristic techniques.

Regarding exact methods, Dror et al. (1994) present an integer formulation and different classes of valid inequalities, and they develop a branch-and-cut algorithm. Belenguer et al. (2000) and Moreno et al. (2010) describe lower bound procedures where the customers' demand is lower than the capacity of the vehicle, and the quantity delivered by a vehicle when visiting a customer is an integer. Lee et al. (2006) develop a dynamic programming approach, Jin et al. (2007) propose an iterative procedure where a sequence of TSPs are solved, and Archetti et al. (2011) describe a column generation technique. More recently, Archetti et al. (2014) describe a branch-and-cut algorithm.

With respect to heuristic techniques, Archetti et al. (2006b) and Aleman and Hill (2010) propose tabu search heuristics, whereas Aleman et al. (2009) and Aleman et al. (2010) put forth constructive and local search procedures. Boudia et al. (2007) combine a genetic algorithm with a local search procedure for intensification and a distance measure to control population diversity. Chen et al. (2007) and Archetti et al. (2008b) develop different kinds of hybrid algorithms. More recently, Wilck IV and Cavalier (2012a) describe a two-phase constructive procedure, and also a genetic algorithm in Wilck IV and Cavalier (2012b) and Silva et al. (2015) implement a multi-star iterative local search. See Archetti and Speranza (2008) and Archetti and Speranza (2012) for a survey on SDVRP and related problems.

The literature is also quite extensive on related problems. The SDVRP with time windows which for example, a branch-and-price algorithm is designed in Gendreau et al. (2006), a branch-and-price-and-cut algorithm in Desaulniers (2010), and more recently McNabb et al. (2015) implement a max-min ant system constructive heuristic along with Or-opt or 2-opt operators.

In practice, visiting a customer is costly to both the transportation company and the customer. As observed in Gulczynski et al. (2010), it takes time, involves paperwork and data processing, and often distracts the customer from primary activities. It is especially undesirable for the customer to be interrupted and distracted too many times. As a result both parties may impose a maximum number m of visits to a location. In addition, other conditions can be required. For example, as analyzed in Gulczynski et al. (2010), only SDVRP solutions with a minimum delivery amount in each visit could be accepted.

Although the SD1PDTSP algorithm is not intended to compete with specific exact SDVRP methods on benchmark SDVRP instances, Section 4.4 shows that the SD1PDTSP implementation found optimal solutions to benchmark instances that the recent technique in Archetti et al. (2011) was not able to solve.

2.4 Other related problems

When the number of visits to a location is unbounded, Naddef and Rinaldi (1993) study the uncapacitated variant of the SD1PDTSP. This problem is called the *Graphical Travelling Salesman Problem* (GTSP), where the aim is to find a min-cost route visiting each location at least once. The study of the *Graphical TSP* has been initiated by Cornuéjols et al. (1985) and Fleischmann (1985). The term *graphical* may not be the most suitable, but at the time it was defined, the idea was to reflect the fact that this version of the TSP has an optimal solution as long as the graph is connected. Later on, Naddef and Rinaldi (1993) suggested to view the GTSP not only as an optimization problem in its own right, but also as a relaxation of the symmetric TSP which can actually be used to solve the symmetric TSP. Thus, they develop a method to deal with the polyhedral combinatorics of symmetric TSP by studying Graphical TSP. Some other references on this problem are Fonlupt and Naddef (1992), Naddef and Rinaldi (2007), Naddef and Rinaldi (1991), Oswald et al. (2006) and more recently Theis (2010) and Theis (2014).

If the nodes are partitioned into clusters, and the problem calls for a minimum cost cycle visiting at least one node for each cluster, the problem is the *Generalized TSP*. This calls for a minimum cost cycle visiting at least one node for each cluster. Another version, called E-Generalized TSP (where E stands for Equality), arises when imposing the additional constraint that exactly one node of each cluster must be visited. The Generalized TSP was first simultaneously mentioned by Henry-Labordere (1969) and Srivastava et al. (1969) who addressed the asymmetrical and the symmetrical version of the problem, respectively. They proposed a dynamic programming approach for its solution. Later, Laporte and Nobert (1983) and Laporte et al. (1987) also study the symmetrical and the asymmetrical case, formulating the problem as an integer program, and developing a branch and bound algorithm for its solution. Fischetti et al. (1995) study the facial structure of the corresponding polytopes. ALso Fischetti et al. (1997) present a particular case in which the number of visits to a location is bounded and the vehicle capacity is unlimited. In this, each location is represented by a cluster of nodes (visits to a customer) in a graph. When the maximum number of visits to a location is one, the uncapacitated variant is the TSP. Not only exact approaches have been studied, but also heuristic methods have been introduced by e.g. Renaud and Boctor (1998) and more recently by e.g. Karapetyan and Gutin (2011), Pintea et al. (2013) and Silberholz and Golden (2007).

There are few publications on optimization problems including vehicle routing, split demands and pickup-and-deliveries. Anily and Bramel (1999) address the SD1PDTSP when the demand of each customer is plus-minus one unit. They propose two heuristic algorithms and analyze their worst-case performance. Nowack et al. (2008) consider a related problem where several commodities must be transported. Each commodity goes from one origin to one destination, a location may simultaneously serve as both an origin and a destination, and split demands are allowed. The authors show that the maximum travel-cost reduction of the split-allowed problem versus the split-forbidden problem is obtained when all the customer demands are just above one half of vehicle capacity. They also made the conjecture that this reduction is at most 50%, and they describe a heuristic approach to solve the split-allowed problem starting from a feasible split-forbidden solution. Hernández-Pérez and Salazar-González (2009) and Gouveia and Ruthmair (2015) address the split-forbidden problem. Kerivin et al. (2008) and Kerivin et al. (2012) extend the problem by allowing preemption, i.e. a commodity collected in a location can be unloaded (fully or partially) at any intermediate location to be picked up later by the same or another vehicle. This unloading/picking-up process can be repeated several times for a demand until its destination is reached. There is no restriction on the number of times that a location may be visited, and the vehicle capacity is not on each vehicle routing an arc but on all vehicles traversing the same arc. Nowack et al. (2012) extends the problem with the additional constraint that, on each trip, all pickup customers must be visited before any delivery customer, and describes a dynamic programming algorithm that exploits the precedence constraints between origins and destinations.

Chapter 3

The SD1PDTSP: Mathematical Model

This chapter presents a mathematical model and properties for the SD1PDTSP. Section 3.1 introduces two mixed integer linear programming formulation for both the asymmetric and symmetric SD1PDTSP. Section 3.2 discusses several minor modifications to also model other related problem. Section 3.3 shows valid inequalities for the model of the asymmetric SD1PDTSP such as fractional capacity inequalities, 2-matching inequalities, etc.

3.1 Mathematical Model

This section presents an integer linear programming formulation for the asymmetric and symmetric SD1PDTSP. First, it will begin setting up the notation. Let n be the number of locations (i.e. a depot and customers), and $I = \{1, \dots, n\}$ be the set of locations being 1 the depot. Let m_i be the maximum number of visits allowed to serve customer i, and the maximum number of visits allowed to the depot is denoted by k. For simplicity of the exposition, given a constant m it assumes that $m_i = m$ $\forall i \in I$ and $k = m$, although in some applications the values m_i and k could be desired to be different than m. Let V_i be a set of m nodes representing potential visits to location i. V_i is an ordered set, so (e.g.) i_1 and i_m represent the first and the last visit, respectively, by the vehicle to location i. The set $V = \cup_{i \in I} V_i$ is the node-set of a graph $G = (V, A)$, where A contains the arcs connecting nodes associated with different locations. Given a node $v \in V$, the location associated with this node is denoted by $i(v)$. For a given subset S of nodes, it writes $\delta^+(S) = \{(v, w) \in A : v \in S, w \notin S\}$ and $\delta^-(S) = \{(v, w) \in A : v \notin S, w \in S\}$.

Given an arc $a = (v, w)$ the travel cost from v to w is denoted by c_a. For each location $i \in I$, let p_i be the units of product in i before starting the service, and p'_i the desired units of product in i after the end of the service. Let $d_i = p'_i - p_i$ be the demand of location i. When $d_i > 0$ the location i is a pickup customer, which means that it provides product from the system. When $d_i < 0$ the location i is a delivery customer, which means that it needs product to the system. In case of a customer with demand zero, it assumes that is a pickup customer. It also assumes that $\sum_{i \in I} d_i = 0$, so the number of product units in the system remains equal before and after performing the vehicle service. Moreover it supposes that all locations must be served by the vehicle, including those customers with zero demand. In addition, a capacity q_i associated with location i is given, meaning that this location can store between 0 and q_i units of the commodity. The capacity of the vehicle is Q and is also assumed to be known.

In the bike-sharing application, I represents the stations, p_i and p'_i are the initial and final (respectively) numbers of bikes at station i, q_i is the number of slots for bikes at station i, and Q is the maximum number of bikes that the vehicle can transport.

To present a mathematical formulation, for each arc $a \in A$, a binary variable

$$x_a = \begin{cases} 1 & \text{if } a \text{ is routed} \\ 0 & \text{otherwise} \end{cases}$$

and a continuous variable f_a being the number of units of product in the vehicle when going through arc a are introduced. For each node $v \in V$, a binary variable

$$y_v = \begin{cases} 1 & \text{if the visit } v \text{ is performed by the vehicle} \\ 0 & \text{otherwise} \end{cases}$$

and a continuous variable g_v being the number of units collected (if positive) or delivered (if negative) when performing the visit v are also defined. Note that, given a solution, the final number of units of product at customer i is given by $p_i + \sum_{v \in V_i} g_v$, which must be equal to p'_i.

Then a mathematical formulation of the asymmetric SD1PDTSP is given by

$$\min \sum_{a \in A} c_a x_a \tag{3.1}$$

subject to

$$y_{i_1} = 1 \qquad \forall i \in I \tag{3.2}$$

$$y_{i_l} \geq y_{i_{l+1}} \qquad \forall i \in I, \forall l = 1, \dots, m-1 \tag{3.3}$$

$$\sum_{a \in \delta^+(v)} x_a = \sum_{a \in \delta^-(v)} x_a = y_v \qquad \forall v \in V \tag{3.4}$$

$$\sum_{a \in \delta^+(S)} x_a \geq y_v + y_w - 1 \qquad \forall S \subseteq V, \forall v \in S, \forall w \in V \setminus S \tag{3.5}$$

$$\sum_{a \in \delta^+(v)} f_a - \sum_{a \in \delta^-(v)} f_a = g_v \qquad \forall v \in V \tag{3.6}$$

$$0 \leq f_a \leq Q x_a \qquad \forall a \in A \tag{3.7}$$

$$\sum_{l=1}^{m} g_{i_l} = d_i \qquad \forall i \in I \tag{3.8}$$

$$0 \leq p_i + \sum_{1 \leq k \leq l} g_{i_k} \leq q_i \qquad \forall i \in I, \forall l = 1, \dots, m-1 \tag{3.9}$$

$$-q_i y_{i_l} \leq g_{i_l} \leq q_i y_{i_l} \qquad \forall i \in I, \forall l = 2, \dots, m \tag{3.10}$$

$$y_v, x_a \in \{0, 1\} \qquad \forall v \in V, \forall a \in A. \tag{3.11}$$

Equations (3.2) force to visit once each location. Inequalities (3.3) impose an order to use the nodes of a location. These constraints avoid symmetrical solutions representing the same route. Equations (3.4) are the degree equations, ensuring that the vehicle enters and leaves each node v with $y_v = 1$. Inequalities (3.5) are the subtour elimination constraints. Constraints (3.6)–(3.8) ensure that the load of the vehicle is able to satisfy the demand of each location. Constraints (3.9) guarantee that the storage of product in a location is always between 0 and its capacity. Note that case $l = m$ is useless because (3.8) is in the model, $0 \leq p'_i \leq q_i$ and $0 \leq p_i \leq q_i$ is assumed. Inequalities (3.10) impose that product can be delivered to or collected from a location in each visit. Note that case $l = 1$ is useless because (3.2) and (3.9).

Trivially, for a given S, inequality (3.5) is dominated by

$$\sum_{a \in \delta^+(S)} x_a \geq 1 \tag{3.12}$$

when there are locations i and j such that $i_1 \in S$ and $j_1 \in V \setminus S$. Otherwise, it is dominated by

$$\sum_{a \in \delta^+(S)} x_a \geq y_v \tag{3.13}$$

for all $v \in V \setminus S$ when $i_1 \in S$ for each $i \in I$, or for all $v \in S$ when $i_1 \in V \setminus S$ for each $i \in I$.

Given a feasible SD1PDTSP solution x^1, then there is a vector x^2 that defines a feasible SD1PDTSP solution in the opposite direction. For all $a = (v, w) \in A$ with $v, w \in V$ it holds

$$x^2_{(v,w)} = \begin{cases} 1 & \text{if } x^1_{(w,v)} = 1 \\ 0 & \text{otherwise} \end{cases}$$

Indeed, its load is defined by

$$f^2_{(v,w)} = Q - f^1_{(w,v)} \qquad \forall (v,w) \in A$$

Hence, if $c_{(v,w)} = c_{(w,v)}$ $\forall v, w \in V$, a symmetric model can be considered. In fact, if there is an oriented cycle $x' \in \mathbb{R}^{|A|}$, and a continuous variable $f' \in \mathbb{R}^{|A|}$ satisfying the restrictions in the asymmetric model, then there also is an unoriented cycle $x'' \in \mathbb{R}^{|E|}$, and a continuous variable $s'' \in \mathbb{R}^{|A|}$ satisfying the restrictions in the symmetric model and vice versa. Thus, if it is defined

$$f'_{(v,w)} = s''_{(v,w)} + (\frac{Q}{2} - s''_{(w,v)})$$

for each arc (v,w) in the oriented tour, the solution of the symmetric model is defined by

$$x''_{[v,w]} = x'_{(v,w)} + x'_{(w,v)} \qquad \forall [v,w] \in E$$

and

$$s''_{(v,w)} = \begin{cases} f'_{(v,w)}/2 & \text{if } x'_{(v,w)} = 1 \\ (Q - f'_{(w,v)})/2 & \text{if } x'_{(w,v)} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall (v,w) \in A$$

Therefore, for each $e \in E$ it is considered the new variable

$$x_e = \begin{cases} 1 & \text{if and only if } e \text{ is routed} \\ 0 & \text{otherwise} \end{cases}$$

and a continuous variable s_a for each $a \in A$ such that it does not represent the load of the vehicle going through arc a, as happen with the variable f_a in the asymmetric model, but this guarantees the existence of a feasible SD1PDTSP solution. Then, the symmetric SD1PDTSP formulation is as follows

$$\min \sum_{e \in E} c_e x_e \tag{3.14}$$

subject to

$$
\begin{aligned}
y_{i_1} &= 1 && \forall i \in I && (3.15)\\
y_{i_l} &\geq y_{i_{l+1}} && \forall i \in I, \forall l = 1, \dots, m-1 && (3.16)\\
\sum_{e \in \delta(v)} x_e &= 2y_v && \forall v \in V && (3.17)\\
\sum_{e \in \delta(S)} x_e &\geq 2(y_v + y_w - 1) && \forall S \subseteq V, \forall v \in S, \forall w \in V \setminus S && (3.18)\\
\sum_{a \in \delta^+(v)} s_a - \sum_{a \in \delta^-(v)} s_a &= g_v && \forall v \in V && (3.19)\\
0 \leq s_{(u,v)} &\leq \frac{Q}{2} x_{[u,v]} && \forall u, v \in V && (3.20)\\
\sum_{l=1}^{m} g_{i_l} &= d_i && \forall i \in I && (3.21)\\
0 \leq p_i + \sum_{1 \leq k \leq l} g_{i_k} &\leq q_i && \forall i \in I, \forall l = 1, \dots, m-1 && (3.22)\\
-q_i y_{i_l} \leq g_{i_l} &\leq q_i y_{i_l} && \forall i \in I, \forall l = 2, \dots, m && (3.23)\\
y_v, x_e &\in \{0, 1\} && \forall v \in V, \forall e \in E. && (3.24)
\end{aligned}
$$

Without the above observation, it would be a mistake thinking that it is possible to use variables f_a instead of s_a and replace the constraints (3.19)–(3.20) by

$$
\begin{aligned}
\sum_{a \in \delta^+(v)} f_a - \sum_{a \in \delta^-(v)} f_a = g_v \quad \forall v \in V\\
0 \leq f_{(v,w)} \leq Q x_{[v,w]} \quad \forall a \in A
\end{aligned}
\qquad (3.25)
$$

Considering a instance with a depot and two customers with demands $d_2 = +5$ and $d_3 = -2$ and $m = 2$. If the vehicle capacity is $Q = 2$, then the SD1PDTSP is not feasible, but the mathematical model taking (3.25) instead of (3.19)–(3.20) has the integer solution $x_{[1_1,2_1]} = x_{[2_1,3_1]} = x_{[2_2,3_1]} = x_{[2_2,3_2]} = x_{[1_1,3_2]} = 1$ and 0 otherwise with $f_{(1_1,2_1)} = f_{(3_2,2_2)} = f_{(3_1,2_1)} = 0$, $f_{(1_1,3_2)} = f_{(2_2,3_1)} = 1$, $f_{(2_1,3_1)} = f_{(3_1,2_2)} = f_{(2_2,3_2)} = f_{(3_2,1_1)} = f_{(2_1,1_1)} = 2$, and $g_{1_1} = -3$, $g_{2_1} = +4$, $g_{2_2} = +1$, $g_{3_1} = -1$, $g_{3_2} = -1$. Therefore this model is not valid.

3.2 SD1PDTSP Variants

As pointed out in Chapter 2, model (3.1)–(3.11) is also valid for special cases of the SD1PDTSP like the VRP and the SDVRP. In these problems, the depot is the only pickup location that can be visited at most k times, while the customers are delivery locations. On the VRP the number of visits to a delivery location is limited to 1, while on the SDVRP it is limited to m. Chapter 4 proposes a technique to solve the

SD1PDTSP to optimality based on the formulation (3.1)–(3.11), and therefore this is a valid approach to solve VRP and SDVRP. As it was seen in Chapter 2, there are a lot of problems closely related to the SD1PDTSP. In fact, they can be regarded as variants of the SD1PDTSP, and easily modeled by slightly modifying the formulation (3.1)–(3.11).

3.2.1 Load vehicle fixed

In the definition of the SD1PDTSP, the vehicle is not required to leave the depot with full or empty load. In fact, it assumes that the depot can supply the vehicle with any extra initial load, which is an unknown value in the problem. However one can be interested in fixing this load. For example, if j represents the depot, adding $f_a = Qx_a$ for all $a \in \delta^+(V_j)$ forces the vehicle to leave the depot with full load. These equations are useful to solve VRP and SDVRP instances with the model and the exact algorithm described in this thesis.

3.2.2 Avoid preemption

By replacing (3.10) with

$$\begin{aligned} 0 \leq g_{i_l} \leq d_i y_{i_l} \quad & \forall i \in I : d_i \geq 0 \quad \forall l = 1, \ldots, m \\ d_i y_{i_l} \leq g_{i_l} \leq 0 \quad & \forall i \in I : d_i < 0 \quad \forall l = 1, \ldots, m \end{aligned}$$

one can force that a vehicle never stores a unit of product in an intermediate location between it was collected and it will be delivered. In other words, these inequalities avoid preemption during the route which may be considered as disturbing the customer in some applications. Note that, when preemption is forbidden, the inequalities (3.9) are redundant, and the notation in the problem description can be simplified (e.g. the model uses d_i but not p_i, p'_i, q_i).

3.2.3 Minimum amount of product served to a customer

Archetti et al. (2011) assumes that, if a customer is visited, it has also to be served. This implies that the customers visited in any trip receive or deliver at least a fix amount of product (say, one unit). Then preemption is forbidden and customers with demand smaller than this minimum amount do not exist. The variant can be modeled with

$$\begin{aligned} 1 \leq g_{i_l} \leq d_i y_{i_l} \quad & \forall i \in I : d_i \geq 1 \quad \forall l = 1, \ldots, m \\ d_i y_{i_l} \leq g_{i_l} \leq -1 \quad & \forall i \in I : d_i \leq -1 \quad \forall l = 1, \ldots, m \end{aligned}$$

instead of (3.10) and again inequalities (3.9) are unnecessary. In addition, under some conditions many variables could be removed from the model for this variant. For example, a customer with $d_i < 2$ will be visited only once, thus $y_{i_l} = 0$ for all $l = 2, \dots, m$.

Gulczynski et al. (2010) also impose a minimum amount of product served to a customer in each visit, but in this case the minimum amount is proportional to the demand of the customer rather than a fix number of units. Again, a customer is interrupted with a visit only when the delivery or pickup is substantial in amount or value each time. To model this variant, (3.10) should be replaced by:

$$
\begin{aligned}
r d_i y_{i_l} \le g_{i_l} \le d_i y_{i_l} \quad & \forall i \in I : d_i \ge 0 \quad \forall l = 1, \dots, m \\
d_i y_{i_l} \le g_{i_l} \le r d_i y_{i_l} \quad & \forall i \in I : d_i < 0 \quad \forall l = 1, \dots, m
\end{aligned}
$$

where r is the percentage of the customer demand defining the minimum amount (e.g. 35%). Clearly this parameter has a high impact on the maximum number of visits (e.g. $r = 0.35$ implies $m = 2$). To our knowledge, this is the first mathematical formulation known for the SDVRP with minimum delivery amounts and the solution approach described in 4 is the first exact method to solve it.

3.2.4 Other variants

A variant concerns relaxing the constraints that customers with zero-demand must be visited. This is easily obtained by simply removing equations (3.2) from the formulation. In this case inequalities (3.12) and (3.13) are valid only under some conditions. For example, inequalities (3.12) is valid when there are customers i and j with non-zero demand such that $V_i \subseteq S$ and $V_j \subseteq V \setminus S$.

Other variants relax the constraint that all customers must be visited. In addition to apply the modifications mentioned in the previous variant, equations (3.8) must be replaced by inequalities, and potentially, a term should be added to the objective function to consider penalties when customers are partially served or not visited by the vehicle.

On some applications it is desired that any product collected from a customer must be delivered by the vehicle before returning to the depot. This constraint is important when the whole route will not be performed by a single vehicle, but by a fleet of vehicles. If j represents the depot with zero demand, the constraint can be modeled by simply adding $f_a = 0$ for all $a \in \delta^+(V_j) \cup \delta^-(V_j)$.

3.3 Valid inequalities

As for almost all combinatorial optimization problems, the SD1PDTSP can be described as the polytope formed for the convex hull of all the tours that verify the problem requirements, i.e. the convex hull of the feasible solutions (3.1)–(3.11). This is described by a finite number of inequalities. Nevertheless, unlike the TSP polytope, the dimension of the SD1PDTSP polytope does not only depend on the number of locations n, but also depends, in a complex way, on the a large set of variables.

Finding the complete description of the polytope as a system linear inequalities it is a difficult task. Moreover, given a inequality, it is also difficult to determinate if it supports the polytope. However, it can be a great computational help to have a partial description of that inequalities.

3.3.1 Rounded Fractional Capacity Cuts

The large set of variables is the major drawback, but by Benders' decomposition (see Section 1.5 for further information), it is possible to project out the variables f_a in the model (3.1)–(3.11). Acording to Farkas' Lemma, given the polytope described by (3.6)–(3.7), there is a fixed vector x in the polytope if, and only if, for all extremes directions $(\alpha, \beta) \in \mathbb{R}^{|V|+|A|}$ the inequalities

$$\sum_{v \in V} \alpha_v g_v - \sum_{a \in A} \beta_a Q x_a \leq 0$$

are valid for the cone

$$\begin{aligned} \alpha_v - \alpha_w - \beta_{(v,w)} &\leq 0 \quad \forall (v,w) \in A \\ \beta_{(v,w)} &\geq 0 \quad \forall (v,w) \in A \end{aligned} \tag{3.26}$$

Since the total demand of product in the system is zero, the equations (3.6) can all be replaced by smaller-or-equal inequalities, and the inequalities (3.7) can be replaced by greater-or-equal inequalities if $d_i > 0$, and by smaller-or-equal inequalities otherwise, without adding new solutions. Moreover, clearly (3.26) is a 1-dimensional linear space generated by the vector defined by $\widetilde{\alpha_v} = 1$, $\forall v \in V$ and $\widetilde{\beta_a} = 0$, $\forall a \in A$. Therefore, it is possible to assume $\alpha_v \geq 0$, $\forall v \in V$ in (3.26) in order to simplify the characterization of the extreme rays, thus (3.26) can be replaced by

$$\begin{aligned} \alpha_v - \alpha_w - \beta_{(v,w)} &\leq 0 \quad \forall (v,w) \in A \\ \beta_{(v,w)} &\geq 0 \quad \forall (v,w) \in A \\ \alpha_v &\geq 0 \quad \quad \forall v \in V \end{aligned} \tag{3.27}$$

It was seen by , for each $S \subset V$, all extremes directions (α, β) are defined by $\alpha_v = 1$ $\forall v \in S$, $\alpha_v = 0$ $\forall v \in V \setminus S$, $\beta_a = 1$ $\forall a \in \delta^+(S)$ and $\beta_a = 0$ $\forall a \in A \setminus \delta^+(S)$. Indeed, the linear system defined by (3.6) and (3.7) can be replaced by

$$\sum_{a \in \delta^+(S)} x_a \geq \frac{1}{Q} \sum_{v \in S} g_v \qquad \forall S \subset V \tag{3.28}$$

These inequalities are similar to the so-called *fractional capacity cuts*, known in the vehicle routing literature. The major difference is that the right-hand side is not a constant for us, and therefore rounding up would create a non-linear inequality. Still this is not always the case, and now it is presented a particular situation where rounding up keeps the linearity of the inequality, while at the same time it strengthens the linear programming relaxation of the formulation.

Consider S defined by the union of a collection of V_i. Let C be a subset of customers, and it assumes that $S = \cup_{i \in C} V_i$. Then the previous inequality for S is dominated by the following one

$$\sum_{a \in \delta^+(S)} x_a \geq \left\lceil \frac{1}{Q} \left| \sum_{i \in C} d_i \right| \right\rceil \tag{3.29}$$

Even if there is an exponential number of linear inequalities in (3.29), today's state-of-the-art cutting-plane approaches allow to manage them all in very effective way. These inequalities can be heuristically separated by first finding the most violated inequality without the rounded up operation. To this end, one must shrink each set V_i of the graph G in a single node, and then solve a max-flow problem on a properly defined network with an artificial node. To be more precise it is discussed in further detail in Section 4.2.2.

In order to simply the model, the inequalities (3.5) and (3.29) can be replaced by the following inequalities

$$\sum_{a \in \delta^+(S)} x_a \geq r(S) \qquad \forall S \subset V$$

where

$$r(S) = \max \left\{ 1, \left\lceil \frac{1}{Q} \left| \sum_{i \in C} d_i \right| \right\rceil \right\}$$

As happen in other optimization problems, owing to the constraints (3.4), the above constraints are equivalent to the following

$$\sum_{a \in A(S)} x_a \geq |S| - r(S) \tag{3.30}$$

Another improvement of the constraints (3.5) and (3.29) in the SD1PDTSP model can be got by defining $r'(S)$ as the smallest number of times the vehicle with capacity Q must go inside S to meet the demand d_i of the customers in S. Then, the following inequality is valid

$$\sum_{a \in \delta^+(S)} x_a \geq r'(S) \qquad \forall S \subset V \tag{3.31}$$

Note that $r'(S)$ is not the solution of a Bin Packing Problem since d_i can be a negative value. Moreover $r(S) \leq r'(S)$.

Nevertheless, both improvement (3.30) and (3.31) are not considered in the algorithm described in Chapter 4 because, in the first case the right-hand side is not a constant, and therefore its use would create a non-linear inequality. And in the second case, the computation of $r'(S)$, even for a fixed subset S, is an NP-hard problem in the strong sense.

3.3.2 2-matching inequalities

Further strengthening arises by using valid inequalities for the Generalized TSP polytope. This is the case, for example, of the *comb inequality* that is defined by a family of customer subsets $T_1, \ldots, T_t, H$ such that,

$$\begin{aligned} H \cap T_i &\neq \emptyset \qquad \forall\, 1 \leq i \leq t \\ T_i \setminus H &\neq \emptyset \qquad \forall\, 1 \leq i \leq t \\ T_i \cap T_j &= \emptyset \qquad \forall\, 1 \leq i < j \leq t \end{aligned}$$

Then the comb inequality is as follows (see Section 1.4.1 for an equivalent form of the comb inequality)

$$\sum_{e \in E(H)} x_e + \sum_{j=1}^{t} \sum_{e \in E(T_j)} \leq |H| + \sum_{j=1}^{t} |T_j| - 1 - (t+1)/2$$

This inequality is the base for yielding other inequalities, e.g. when a handle is given, and the number of teeth is zero, the subtour elimination constraints are obtained. In particular, it uses the 2-matching inequality. This arises when a handle $H \subset V$ and an number $t \geq 3$ of teeth $T_1, T_2, \ldots, T_t$ with $T_i \subset V$, for all $i = 1, \ldots, t$ are given, such that

$$\begin{aligned} |H \cap T_i| &= 1 \qquad \forall\, 1 \leq i \leq t \\ |T_i \setminus H| &= 1 \qquad \forall\, 1 \leq i \leq t \end{aligned}$$

More concretely, it uses the generalized 2-matching inequality

$$\sum_{a \in \delta^+(H) \setminus T} x_a \geq \sum_{a \in T} x_a - \sum_{v \in H} y_v + 1 \tag{3.32}$$

for all $H \subset V$ and $T \subset \delta^+(H)$ with $|T|$ odd. Although these inequalities can be separated in polynomial time, it uses the heuristic procedure described in Fischetti et al. (1997) to incorporate them in the branch-and-cut approach.

3.3.3 Benders' Cut

Using Benders' Decomposition enables to eliminate the g_v variables from the model. Let α_v be the dual variable of equation (3.6), β_a of the right-hand side inequality in (3.7), γ_i of equation (3.8), $\delta^1_{i_l}$ of the left-hand side inequality in (3.9), $\delta^2_{i_l}$ of the right-hand side inequality in (3.9), $\lambda^1_{i_l}$ of the left-hand side inequality in (3.10), and $\lambda^2_{i_l}$ of the right-hand side inequality in (3.10). If (x^*, y^*) is the solution from the master problem, the dual problem maximizes

$$\sum_{i\in I}\gamma_i d_i + \sum_{a\in A}\beta_a Q x^*_a + \sum_{i\in I}\sum_{l=1}^{m-1}\left(\delta^2_{i_l}(q_i - p_i) - \delta^1_{i_l}p_i\right) + \sum_{i\in I}\sum_{l=2}^{m}(\lambda^2_{i_l} - \lambda^1_{i_l})q_i y^*_{i_l}$$

subject to

$$\begin{aligned}
\alpha_u - \alpha_v + \beta_{(u,v)} &\le 0 && \forall (u,v)\in A\\
-\alpha_{i_1} + \gamma_i + \sum_{k=1}^{m-1}(\delta^1_{i_k} + \delta^2_{i_k}) &= 0 && \forall i\in I\\
-\alpha_{i_l} + \gamma_i + \sum_{k=l}^{m-1}(\delta^1_{i_k} + \delta^2_{i_k}) + \lambda^1_{i_l} + \lambda^2_{i_l} &= 0 && \forall i\in I, \forall l = 2,\dots,m\\
\beta_a &\le 0 && \forall a\in A\\
\delta^1_v, \lambda^1_v &\ge 0 && \forall v\in V\\
\delta^2_v, \lambda^2_v &\le 0 && \forall v\in V\\
\alpha_v, \gamma_i \text{ unsigned} & && \forall v\in V, \forall i\in I
\end{aligned}$$

Note that $\forall i\in I$, for all $l = 2,\dots,m$

$$\lambda^1_{i_l} + \lambda^2_{i_l} = \alpha_{i_l} - \gamma_i - \sum_{k=l}^{m-1}(\delta^1_{i_k} + \delta^2_{i_k})$$

is unsigned. Since $\lambda^1_{i_l} \ge 0$ and $\lambda^2_{i_l} \le 0$ then

$$\lambda^1_{i_l} - \lambda^2_{i_l} = |\alpha_{i_l} - \gamma_i - \sum_{k=l}^{m-1}(\delta^1_{i_k} + \delta^2_{i_k})|$$

Observe also that

$$\gamma_i = \alpha_{i_1} - \sum_{k=1}^{m-1}(\delta^1_{i_k} + \delta^2_{i_k})$$

Since the total demand of product in the system is zero, equations (3.6) can all be replaced by smaller-or-equal inequalities, and equations (3.7) can all be replaced without adding new solutions by greater-or-equal inequalities if $d_i > 0$, and by smaller-or-equal inequalities otherwise. Then β_a must be non-negative for all a, and γ_i must be non-negative if $d_i > 0$ and non-positive otherwise. Introducing also a new variable β'_a to replace $-\beta_a$, the above dual problem can be reformulated as maximizing

$$\sum_{i\in I}\big(\alpha_{i_1}-\sum_{k=1}^{m-1}(\delta^1_{i_k}+\delta^2_{i_k})\big)d_i-\sum_{a\in A}\beta'_a Q x^*_a+$$
$$\sum_{i\in I}\sum_{l=1}^{m-1}\big(\delta^2_{i_l}(q_i-p_i)-\delta^1_{i_l}p_i\big)-\sum_{i\in I}\sum_{l=2}^{m}\big|\alpha_{i_l}-\alpha_{i_1}+\sum_{k=1}^{l-1}(\delta^1_{i_k}+\delta^2_{i_k})\big|q_i y^*_{i_l}$$

subject to

$$\begin{aligned}\alpha_u-\alpha_v-\beta'_{(u,v)}&\le 0 && \forall (u,v)\in A\\ \alpha_v,\delta^1_v&\ge 0 && \forall v\in V\\ \beta'_a&\ge 0 && \forall a\in A\\ \delta^2_v&\le 0 && \forall v\in V\end{aligned}$$

The extreme rays of the polyhedral cone on the (α,β') space are known. Hernández-Pérez and Salazar-González (2003) shows that each extreme ray is associated with a set $S\subset V$ and defined by $\alpha_v=1$ if and only if $v\in S$, and by $\beta'_a=1$ if and only if $a\in\delta^+(S)$. This allows us to avoid the unboundness of the dual problems by strengthening the master problem with the following set of inequalities for each subset $S\subset V$

$$Q\sum_{a\in\delta^+(S)}x_a+$$
$$\sum_{i\in I}\sum_{l=2}^{m}\big|\alpha_{i_l}-\alpha_{i_1}+\sum_{k=1}^{l-1}(\delta^1_{i_k}+\delta^2_{i_k})\big|q_i y_{i_l}$$
$$\ge\sum_{i\in I}\left(\alpha_{i_1}d_i-\sum_{k=1}^{m-1}\big((\delta^1_{i_k}+\delta^2_{i_k})d_i+\delta^1_{i_k}p_i-\delta^2_{i_k}(q_i-p_i)\big)\right)\tag{3.33}$$

where $\delta^1_v\ge 0$ and $\delta^2_v\le 0$ for all $v\in V$, and $\alpha_v=1$ if and only if $v\in S$. Although the extreme rays of the cone defined by (δ^1,δ^2) can be characterized by subsets of nodes, a simpler expression for these inequalities has not been found, and the separation procedure still requires solving a linear program. However, the analysis made it suggests a flow-based procedure to cut off infeasible solutions, which is the separation procedure described in Section 4.2.4. Inequalities (3.33) are the *Benders' cuts.*

3.3.4 Other inequalities

Another interesting family of inequalities is

$$g_u x_{(u,v)}\le f_{(u,v)}\le (Q-g_v)x_{(u,v)}\qquad \forall (u,v)\in A$$

They can be linearized as follows

$$g_u x_{(u,v)} \geq g_u - q_{i(u)}(1 - x_{(u,v)}) \qquad \forall (u,v) \in A$$

The system defined by (3.6) and these linear bounds on the continuous variables can be replaced by

$$Q \sum_{a \in \delta^+(S)} x_a \geq \sum_{u \in S} \big(g_u + \sum_{v \notin S} (2g_v - q_{i(u)} - q_{i(v)} + q_{i(u)} x_{(u,v)} + q_{i(v)} x_{(v,u)}) \big) \qquad \forall S \subset V \tag{3.34}$$

Inequalities (3.28) and (3.34) ensure the existence of a load for the vehicle on a route such that $\max\{0, g_u\} x_{(u,v)} \leq f_{(u,v)} \leq \min\{Q - g_v, Q\} x_{(u,v)}$.

Chapter 4

The SD1PDTSP: The Branch-and-Cut Algorithm

This chapter describes an exact algorithm for the SD1PDTSP. The branch-and-cut algorithm is based on the theoretical results of Chapter 3. A general scheme of the approach is given in Section 4.1. In order to introduce in the model the valid inequalities presented in Section 3.3, it is necessary to implement good separation procedures. These are described in Section 4.2. Section 4.3 is dedicated to the branching phase of the branch-and-cut algorithm. Finally, Section 4.4 analyzes computational results obtained by applying the algorithm to solve SD1PDTSD, SDVRP and VRP instances.

4.1 A Branch-and-Cut Approach

The formulation given in Chapter 3 suggests a branch-and-cut algorithm to solve the SD1PDTSP. This is a very effective tool for solving some combinatorial optimization problems as the SD1PDTSP.

Considering that the large set of variables is the major drawback, using Benders' Decomposition enables to eliminate the f and g variables from the model, and iteratively, work with a master model based only on the binary variables (x, y), and with a subproblem defined by the continuous variables (f, g). Although, the classical Benders' Decomposition would suggest to keep the integrability condition in the master problem solved at each iteration, a more efficient approach is to replace such requirement by its linear-programming relaxation, and apply a branching scheme when the iterative procedure stops and (x^*, y^*) is not integer.

Thus, as it was seen in Section 1.6 the branch-and-cut algorithm starts with the following *linear relaxation* of the SD1PDTSP in which, the continuous variables (f, g) and the condition that all variables have to be integers have been eliminated.

$$\min \sum_{a\in A} c_a x_a \tag{4.1}$$

subject to

$$y_{i_1} = 1 \qquad \forall i \in I \tag{4.2}$$

$$y_{i_l} \geq y_{i_{l+1}} \qquad \forall i \in I, \forall l = 1,\dots,m-1 \tag{4.3}$$

$$\sum_{a\in\delta^+(v)} x_a = \sum_{a\in\delta^-(v)} x_a = y_v \qquad \forall v \in V \tag{4.4}$$

$$0 \leq y_v \leq 1 \qquad \forall v \in V \tag{4.5}$$

$$0 \leq x_a \leq 1 \qquad \forall a \in A. \tag{4.6}$$

The subproblem defined by the continuous variables (f, g) is constructed by fixing the binary variables (x, y) to an optimal solution (x^*, y^*) of the master problem. That subproblem is given below

$$\min 0 \tag{4.7}$$

$$\sum_{a\in\delta^+(v)} f_a - \sum_{a\in\delta^-(v)} f_a = g_v \qquad \forall v \in V \tag{4.8}$$

$$\sum_{v\in V_i} g_v = d_i \qquad \forall i \in I \tag{4.9}$$

$$0 \leq f_a \leq Q x_a^* \qquad \forall a \in A \tag{4.10}$$

$$0 \leq p_i - \sum_{1\leq k\leq l} g_{i_k} \leq q_i \qquad \forall i \in I, \forall l = 1,\dots,m-1 \tag{4.11}$$

$$-q_{i(v)} y_v^* \leq g_v \leq q_{i(v)} y_v^* \qquad \forall v \in V \tag{4.12}$$

At each iteration, the linear relaxation (4.1)–(4.6) is solved. If it is infeasible, then the original SD1PDTSP is infeasible (m or Q are too small). Otherwise, let (x^*, y^*) be the optimal solution of the linear relaxation. Then, the subproblem checks the feasibility of the linear system (4.8)–(4.12) on the variables (f, g). If this system is feasible, then the linear relaxation includes all the necessary constraints, and (x^*, y^*) defines an optimal SD1PDTSP route. Otherwise, a separation method works dynamically seeking violated inequalities by (x^*, y^*). That is, the algorithm looks for a first family of inequalities adding them if any violated is found, and solves the strengthened linear relaxation problem until it does not find violated inequalities of this family. Then, the algorithm searches the next family of inequalities. If any violated is found, it is inserted, it solves the strengthened relaxation problem until any more is found, and goes back to find first family of inequalities; otherwise it looks for other family of inequalities. This procedure continues until any violated inequalities are not found, then the branching starts.

The best performances are obtained when the separation procedures are applied in the following order. Firstly, subtour elimination inequalities (3.12) and (3.13), secondly the rounding fractional capacity inequalities (3.29). Then the generalized 2-matching inequalities (3.32). And the last ones are the Benders' cut (3.33). The strengthened linear relaxation problem can be considered as the master problem, and the infeasible system (4.8)–(4.12) can be considered as a primal problem whose dual problem is unbounded. The ray proving this problem status defines a new Benders' cut that must be included in the master problem. Inequalities (3.28) and (3.34) are not included in the master problem because they involve g variables.

Computing a good upper bound on the optimal objective value would be great for a successful branch-and-cut approach. A initial constructive heuristic procedure, described in Section 5.1.3, gives a first upper bound to the branch-and-cut algorithm, and this can cut off the linear program (subproblems) avoiding unnecessary time consuming. This is the first step of our branch-and-bound algorithm. However, it also executes when the linear-programming relaxation gives a solution (x^*, y^*), and no violated cut is found, i.e. at each node of search. In this case, the solution (x^*, y^*) is used to affect the TSP's through the iterative approach. This is done by changing the cost c_a of an arc a with $(1 - x^*_a)c_a$.

4.2 Separation Procedures

In order to use a class of valid linear inequalities as cutting planes, it is necessary to solve the so-called *separation problem.* For a given family of constraints $\mathcal{F}$, and given a solution (x^*, y^*) of a linear relaxation of the model (3.1)–(3.11), the separation problem is the decision problem whether all constraints of $\mathcal{F}$ are satisfied by (x^*, y^*). In case this does not happen, it should return a valid constraint in $\mathcal{F}$ which is violated by (x^*, y^*). The aim of this section is to describe efficient procedures to solve the separation problems associated with the constraints introduced in Section 3.3.

4.2.1 Separation Algorithm for Subtour Elimination Constraints

Given a solution (x^*, y^*) of the linear relaxation of the SD1PDTSP, it builds a capacitated graph $G^* = (V^*, A^*)$, where $V^* = V$ is the set of vertices, $A^* = \{a \in A : x^*_a > 0\}$ is the set of arcs, and for each arc $a \in A$, x^*_a is its capacity. Given a set $S \subseteq V$, in order to find a violated inequality, the algorithm must search a minimum capacity cut $(S, V \setminus S)$ in the graph G^*.

For the family of subtour elimination constraints (3.5)

$$\sum_{a \in \delta^+(S)} x_a \geq y_v + y_w - 1 \qquad \forall v \in S, \forall w \in V \setminus S$$

this can be done in $O(n^3)$ time, as it is equivalent to find the maximum flow from v to w (see, e.g. Ahuja et al. (1989)). If the maximum flow is not less than $y_v^* + y_w^* - 1$, then all the inequalities (3.5) are satisfied; otherwise the maximum flow is less than $y_v^* + y_w^* - 1$, and a violated inequality (3.5) has been discovered. Trying all possible pairs (v, w) produces an $O(n^5)$ time overall separate algorithm. Actually, a better algorithm having overall $O(n^4)$ time complexity can be obtained, in analogy with the TSP case (see Padberg and Grötschel (1985)). However, these inequalities are not added in the model.

For the family of inequalities (3.13)

$$\sum_{a \in \delta^+(S)} x_a \geq y_v$$

where there are locations i such that $V_i \subset V \setminus S$ and $v \in S$, as before, the algorithm must search a minimum capacity cut $(S, V \setminus S)$ in the graph G^*. Hence it can be discovered by finding the maximum flow from v to t, where t is an additional node connected with each $w \in V_i$ through an edge having very large capacity (t is a node representing the nodes of V_i). Trying all possible pairs (v, t) yields an $O(mn^4)$ time overall separate algorithm. Clearly, nodes v with $y_v^* = 0$ need not be considered.

Finally for the family of inequalities (3.12)

$$\sum_{a \in \delta^+(S)} x_a \geq 1$$

where there are locations i and j such that $V_i \subset S$ and $V_j \subset V \setminus S$, following the same scheme, the algorithm have to find the maximum flow from s to t, where s and t are additional nodes connected with each $v \in V_i$ and $w \in V_j$, respectively, through an edge having very large capacity (s and t are nodes representing the nodes of V_i and V_j, respectively). Thus, the overall time complexity in this case is $O(m^2n^3)$.

According to the above scheme the separation algorithm for the families of inequalities (3.12) and (3.13) requires $O(mn^4)$ in the worst case. However, in the practice, the computing time required is much smaller as the capacitated graph G^* is very sparse, and it has many isolated nodes. Moreover, several max-flow computations can be avoided because some values of y^* are small.

4.2.2 Separation Algorithm for Rounded Fractional Capacity Inequalities

The rounded fractional capacity inequalities (3.29) are separated using the same idea as in Section 4.2.1. Let (x^*, y^*) be a given solution of the linear relaxation of the SD1PDTSP. In order to check if any of these inequalities are violated, let us write the constraints in a different form. Let I be the set of all locations, A'' the set of arcs between locations, and C a subset of customers of I. For each $S = \cup_{i \in C} V_i$

$$\sum_{a \in \delta^+(S)} x_a^* \geq \sum_{i \in C} \frac{+d_i}{Q}$$

is algebraically equivalent to:

$$\sum_{a \in \delta^+(S)} x_a^* + \sum_{i \in I \setminus C : d_i > 0} \frac{d_i}{Q} - \sum_{i \in C : d_i < 0} \frac{d_i}{Q} \geq \sum_{i \in I : d_i > 0} \frac{d_i}{Q}$$

The right-hand side of the inequality is the positive constant K/Q where

$$K = \max \left\{ \sum_{i \in I : d_i > 0} d_i, - \sum_{i \in I : d_i < 0} d_i \right\}$$

and the coefficients in the left-hand side are also positives. Therefore, in order to solve the separation problem of these inequalities, one must shrink each set V_i of the graph in a single node, and then solve a max-flow problem on a properly defined network.

Hence, it builds a capacitated graph $G' = (I', A')$ from the fractional solution (x^*, y^*), where $I' = I \cup \{s, t\}$ with two dummy nodes s and t, $A' = A_1 \cup A_2$ with $A_1 = \{a \in A'' : x_a^* > 0\}$ and capacities x_a^*, and A_2 is the set of arcs (s, i) $\forall i \in I$ such that $d_i > 0$ with capacities $\frac{d_i}{Q}$, and (i, t) $\forall i \in I$ such that $d_i < 0$ with capacities $\frac{-d_i}{Q}$. Thus, in order to find a violated inequality, the algorithm must search a minimum capacity cut $(S', I \setminus S')$ in the graph G' with $s \in S'$ and $t \in I' \setminus S'$. This is equivalent to find the maximum flow from s to t. If the maximum flow is greater than K/Q, then all inequalities are satisfied, otherwise the maximum flow is less than K/Q, and a violated inequality (3.29) is discovered. Thus, $S \setminus \{s\}$ defines the cut to be added in the model.

4.2.3 Separation Algorithm for Generalized 2-matching Inequalities

To incorporate the generalized 2-matching inequalities to the branch-and-cut algorithm it is used the heuristic procedure described in Fischetti et al. (1997). This derives from similar procedures proposed for the TSP (see, e.g. Padberg and Grötschel (1985)), and it is based on the result obtained by Padberg and Rao (1982). Employing a construction similar to that proposed by them for the b-matching problem, one can transform the separation problem into a minimum capacity odd cut problem. Hence, the generalized 2-matching inequalities can be separated in polynomial time.

Given a fractional solution (x^*, y^*), it is defined the subgraph $G' = (V', A')$ where $A' = \{a \in A : x_a^* > 0\}$. The handles of violated 2-matching inequalities are considered, successively, as each connected component H of G', such that their two-node teeth correspond to the arcs $a \in \delta^+(H)$ with $x_a^* = 1$ (if the number of these arcs is even, the

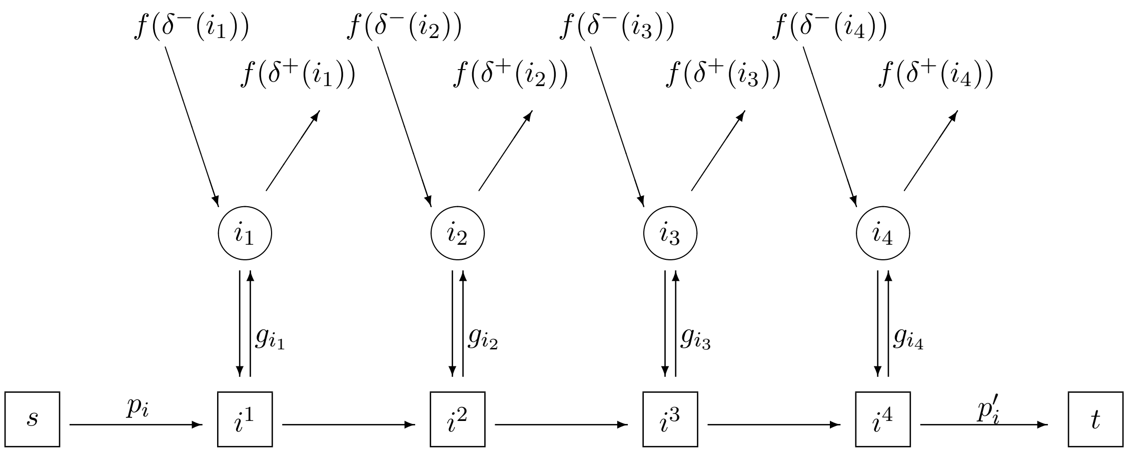


Figure 4.1: Graph representation to check feasibility of a route at location i with $m = 4$.

inequality is clearly rejected). The procedure takes $O(n + |A'|)$ time, if it is properly implemented.

4.2.4 Flow-based Separation

Unlike other cases, it has not been found a simpler expression for the Benders' cuts, and hence the separation procedure still requires solving a linear program. In this manner, this section describes an alternative procedure to check whether a route defined by a given (x^*, y^*) is feasible for the SD1PDTSP. Instead of applying a linear programming solver to find a vector $(\alpha, \beta, \gamma, \delta)$ that may define a violated Benders' cut, it transforms the system (3.6)–(3.10) into a max-flow problem on a specific graph, next described.

Let us consider a graph G^* where the node set contains two dummy nodes (s and t) and two nodes (i_l and i^l) for each $i_l \in V$. See figure 4.1, where it is illustrated the graph associated with a location $i \in I$ on an SD1PDTSP instance with $m = 4$. The node i_l, represented by a circle in the figure, controls the product loaded or unloaded from the vehicle to the customer in the l-th visit. The node i^l, represented by a box in the figure, controls the inventory of the customer before and after the l-th visit. The arcs of G^* join the nodes i_l exactly as in A, each one a with capacity Qx^*_a. G^* also has arcs connecting i_l and i^l for all $l = 1, \ldots, m$ to model whether the service during the l-th visit to i is a delivery (flow through arc (i_l, i^l)) or a pickup (flow through arc (i^l, i_l)). These arcs represent the product transferred between the vehicle and the inventory of customer i, and they both have capacity equal to $q_i y^*_{i_l}$. In addition, the arc set of G^* contains the arcs (s, i^1), (i^m, t) and (i^l, i^{l+1}) for each $l = 1, \ldots, m-1$ and each customer $i \in I$. The capacity of (s, i^1) is set to p_i, which is the inventory at i before the first service, and the capacity of (i^m, t) is set to p'_i, which is the inventory at i after the last service. The capacity of arc (i^l, i^{l+1}) is q_i.

Clearly, an integer solution (x^*, y^*) defines a feasible route for the SD1PDTSP if and only if it is possible to send $\sum_{i\in I} p_i$ units of product in a flow from s to t through the capacitated graph G^*. Hence, the first step to check the feasibility of the linear system (3.6)–(3.10) is to build G^* and compute the maximum flow from s to t in G^*. If the capacity of the optimal flow is $\sum_{i\in I} p_i$ then (x^*, y^*) is a feasible SD1PDTSP solution (thus it satisfies all Benders' cuts). Otherwise, there exists a valid inequality violated by (x^*, y^*). It is important to design an efficient procedure to identify such inequality. In this dissertation now proposes such a procedure which can also be applied when (x^*, y^*) is not integer, and therefore embedded in a branch-and-cut approach.

A well-known result in graph theory ensures that the flow in G^* from s to t with maximum capacity is associated with a cut separating t from s in G^* with minimum capacity, and both capacities coincide. The desired inequality can be constructed by imposing that the capacity of this cut must be at least $\sum_{i\in I} p_i$ on the graph associated with a generic vector (x, y). Let S and S' be the nodes of type i_l and i^l, respectively, in the group of s in the cut. The capacity of the cut is given by $Q\sum_{a\in\delta^+(S)} x^*_a$, plus $q_i y^*_{i_l}$ for each $i \in I$ and $l = 1, \dots, m$ such that $i_l \in S, i^l \notin S'$ or $i_l \notin S, i^l \in S'$, plus q_i for each $i \in I$ and $l = 1, \dots, m-1$ such that $i^l \in S', i^{l+1} \notin S'$, plus p_i if $i^1 \notin S'$, plus p'_i if $i^m \in S'$. Replacing x^* by x and y^* by y in the obtained expression, one gets the capacity of the cut on the graph G^* of a generic vector (x, y). The inequality is then

$$Q \sum_{a\in\delta^+(S)} x_a + \sum_{\substack{i_l\in S, i^l\notin S' \\ or \\ i_l\notin S, i^l\notin S'}} q_i y_{i_l} \geq \sum_{i^1\notin S'} p_i - \sum_{i^m\in S'} p'_i - \sum_{i^l\in S', i^{l+1}\notin S'} q_i \tag{4.13}$$

Observe that it is possible to adapt the decomposition approach in Section 3.3.3, and the graph G^* in the current subsection to work also with the variants annotated at Section 3.2. For example, the requirement on the minimum amount of product served to a customer in each visit can be addressed in G^* with one arc between i_l and i^l, which is (i_l, i^l) if $d_i > 0$ and (i^l, i_l) otherwise, with lower capacity $r|d_i|$.

4.3 Branching

When the separation procedures end because no violated inequality has been generated from a fractional solution (x^*, y^*), and the heuristic procedure has been executed on this solution, the branching step creates two problems. To this end, it selects a binary variable y_v with fractional value y^*_v. When all values y^*_v are integer, then it selects a binary variable x_a with fractional value x^*_a. In the implementation of the SD1PDTSP, the branching variable is automatically selected by CPLEX.

4.4 Computational Experiments

The branch-and-cut approach described in this chapter has been implemented in C++, using IBM ILOG CPLEX 12.5 as a framework. The code was executed on a computer with 4 GB of RAM and an Intel Core 2 Duo CPU E8600 @ 3.33 Ghz running Microsoft Windows 7 (64 bits). To evaluate the performance of our implementation it has considered five classes of instances.

Class I is based on the benchmark instance introduced in Mosheiov (1994). It has 25 locations, Euclidean distances, and demands d_i between -7 and $+7$. It has defined $p_i = 7 - d_i$, $p'_i = 7$ and $q_i = 14$ for all locations. Different SD1PDTSP instances have been created by varying Q and m. When $m = 1$ and $Q \geq 16$ then the optimal SD1PDVRP route is the optimal TSP tour. On these instances $\max\{|d_i| : i \in I\} = 7$, thus $m > 1$ when $Q < 7$.

Class II is based on the randomly-generated instances described in Hernández-Pérez and Salazar-González (2004a) for the 1-PDTSP. They use Euclidean distances and demands d_i between -10 and $+10$. For this implementation it has defined $p_i = 10 - d_i$, $p'_i = 10$ and $q_i = 20$ for each location i and created ten SD1PDTSP instances with $n = 30$ following their description. Then, for $m \in \{1, 2, 3\}$, it has considered $Q \in \{10, 12, 15\}$ to have instances where split may not be necessary, and $Q \in \{5, 6, 7\}$ to have instances where split is necessary.

Class III is based on benchmark instances from the VRP library Toth and Vigo (2014). Five instances were selected from ones considered by Belenguer et al. (2000) with less than 75 customers, used also in Archetti et al. (2011) and in Moreno et al. (2010) and named `eil22`, `eil23`, `eil30`, `eil33` and `eil51`. They include Euclidean distances and give customer demands d'_i, vehicle capacity Q' and fleet size k'. To define the SD1PDTSP instances it has used the given distances. The demands for the customers are the given VRP demands, but with positive sign on the first half of customers and negative sign on the second half of customers. More precisely, if d'_i is the given VRP demand then it has used $p_i = 0$, $p'_i = d'_i$ and $q_i = 2p'_i$ when $i < n/2$, and it has used $p_i = d'_i$, $p'_i = 0$ and $q_i = 2p_i$ when $i \geq n/2$. The depot is considered as a customer such that the total demand is zero. Each customer (including the depot) is allowed to be visited at most m times. It has considered different values for Q, being the given VRP capacity Q' one of them, and $m \in \{1, 2, 3\}$.

Class IV is also based on the same five instances from the VRP library as before, but now the SD1PDTSP instances are defined to have the same optimal routes of the VRP and SDVRP instances. To this end, it has used the given distances and demands d'_i. Let Q' and k' the given vehicle capacity and fleet size, respectively. It has used $p_i = 0$ and $q_i = p'_i = d'_i$ for each customer i. If j represents the depot, it uses $p_j = -\sum_i d'_i$, $p'_j = 0$ and $q_j = -2p_j$. The depot was allowed to have at most k' visits, and each customer was allowed to be visited at most m times. No preemption of the product in a customer is allowed, so the depot is the only pickup location. Two values for Q have considered, being Q' one of them, and $m \in \{1, 2, 3\}$. Note that in the SDVRP literature, the fleet size is typically ignored, thus allowing an unlimited number of return trips to the depot. Instead, it is used the fleet size k' to impose a maximum number of visits to the depot.

Class V is based on six SDVRP instances with 50 customers from Belenguer et al. (2000) and nine SDVRP instances with up to 48 customers from Chen et al. (2007). No preemption of the product in a customer is allowed. According to Belenguer et al. (2000), the customers in the first six instances (`S51D1`,...,`S51D6`) were placed randomly around a central depot, and each customer demand was generated randomly based on a high and low threshold, with a vehicle capacity of 160. According to Chen et al. (2007), the customers in the last nine instances (`SD1`,...,`SD9`) were placed on rings surrounding a central depot, and each customer demand was either 60 or 90, with a vehicle capacity of 100. The parameters p_i, p'_i and q_i have been defined as in Class IV. When adding the constraint on the minimum delivery amount to the problem, the nine instances from Chen et al. (2007) tend to be infeasible and Gulczynski et al. (2010) have slightly modified the customer demands to obtain feasible instances (`MD1`,...,`MD9`). In all cases, the distances have been considered rounded to integer numbers. Similar performances of the implementation are observed when these numbers are not rounded.

Tables 4.1 and 4.2 refer to Class I, i.e. SD1PDTSP instances, all identical except with different vehicle capacities. Table 4.1 shows results when $Q \geq 7$, i.e. when the vehicle capacity is enough to serve each customer individually. The table shows the optimal value LB of the linear programming relaxation before the first branching operation, the travel cost UB of the optimal SD1PDTSP solution, the computational time *rtime* when LB was computed, and the total computational time *ttime*. Times are in seconds. The table shows that all instances were solved to optimality in a few seconds, although the computational time increases with m due to the larger size of the mathematical formulation. Another observation is that there is benefit of splitting when $Q < 10$ but not when $m > 2$. Some UB values are displayed in bold to remark the instances and the m values where there is such benefit. Table 4.2 shows results when $Q < 7$, i.e. when some customers must be visited more than once. In this situation it has tried several values of m such that $m \cdot Q \geq 7$, as reported in the table. It is worth to observe that the best routes for these instances were found with $m = \max\{\lceil |d_i|/Q \rceil : i \in I\}$, as it happened also when $Q \geq 10$.

Tables 4.3 and 4.4 refer to instances in Class II. As it happens with Class I, splitting helps to reduce the travel cost when the vehicle capacity is slightly over the maximum customer demand. On these instances the gain is obtained by allowing two visits, while when $m > 2$ the mathematical formulation is simply larger, but with no better solution than $m = 2$. Indeed, it achieved the time limit (2 hours) on some instances when $m = 3$. Some UB values are displayed in bold to remark the instances and the m values where there is a travel cost reduction. Table 4.5 allows to measure the benefit of using the implemented cut-generation approach versus using a model with the flow variables explicitly. The instances in Table 4.3 have used with $m = 2$. Since the cut-generation approach is a Benders' decomposition algorithm strengthened with additional inequalities, the lower bounds computed during the branch-and-bound process may be larger. For the two approaches, the table shows the lower bound *LB* before the first branching and the number of nodes *Nnode* explored in the branch-and-bound tree. It also shows the numbers of inequalities generated before the first branching during the cutting-generation approach: *bend* is the number of (4.13), *sec* is the number of (3.12)–(3.13), *round* is the number of (3.29), and *2m* is the number of (3.32). Subtour elimination constraints (3.12)–(3.13) are also necessary in the model with the flow variables. Comparing the lower bounds and computational times, there is a clear benefit of using the cut-generation approach versus using the flow variables explicitly.

Table 4.6 refers to instances in Class III, showing performances of the algorithm similar to the ones observed on the SD1PDTSP instances in Classes I and II. The empty cells in the table correspond to instances where $m = 1$, and the vehicle capacity is smaller than a customer demand, i.e. infeasible instances. As before, some *UB* values are in bold to remark the instances, and the m values where there is a travel cost reduction. Increasing m implies weakening the lower bound at the root node and increasing the computational time due to the size of the model. On the instances, the split-demand variant allows to find a shorter route than the split-forbidden variant only on one instance (`eil33` with $Q = 7000$) and with $m = 2$. It is worth mentioning that these results for $m = 1$ are similar to the ones in Hernández-Pérez and Salazar-González (2003), and therefore one could expect to solve to proven optimality instances with a larger number of customers. However, the scope of this study is oriented to $m > 1$ and, under this condition, solving larger instances requires higher computational times.

Table 4.7 shows the performance of the implementation of SD1PDTSP to solve VRP instances ($m = 1$) and SDVRP instances ($m > 1$). Once again, using data from the VRP library, the split-demand variant found routes with smaller travel costs than the split-forbidden variant on a few instances, but no advantage was found on allowing a customer to be visited more than twice. Column *#s* shows the number of customers needing more than one visit in the optimal route. It is interesting to note that this number is small. Column *%a* shows the minimum percentage of demand served to a customer in a visit. For example, the demand of customer 21 in `eil30` is 1500 units; when $Q = 4500$, the optimal route visits this customer twice, serving 1000 units in one visit and 500 in the other; thus %a shows 33.3. When comparing the performance of the SD1PDTSP implementation on VRP instances with the performance of a modern VRP algorithm, the first is far to be competitive. For instance, Pecin et al. (2014) solved to optimality all instances in the VRP library with up to 199 customers, and also some larger VRP instances with up to 360 customers. The performance of the implementation of SD1PDTSP is competitive with the best approach in the SDVRP literature, which according to our knowledge is the column-generation algorithm in Archetti et al. (2011). Observe that this algorithm achieved the time limit of 6 hours in a similar computer when solving `eil22` with $Q = 6000$, `eil23` with $Q = 4500$, `eil30` with $Q = 4500$ and `eil33` with $Q = 8000$. The implementation of this dissertation, instead, solved each of these instances to optimality in 17.9, 3.2, 39.9 and 130.9 seconds, respectively, when each customer is allowed to be visited at most twice. Investigating the performance of the SD1PDTSP implementation to solve VRP and SDVRP instances with more than 50 customers is an interesting task but it goes out the main scope of this paper.

Table 4.8 shows the performance of the SD1PDTSP implementation to solve SDVRP instances from the literature. These instances have been introduced by other authors to evaluate lower bounds and heuristic approaches. This table shows the number of customers $n - 1$, the fleet size k (i.e. maximum number of visits to the depot), and the vehicle capacity Q given in these instances. Column *UB'* shows the value of the best known solution, taken from Moreno et al. (2010) for the first six instances and from Archetti et al. (2011) for the other nine instances. For this thesis it performed experiments with $m = 2$ and 3, but the table reports the results with $m = 2$ because (as it occurred with other instances) no better solution was found with $m = 3$. Even if the implementation is intended to solve a more general and complex problem, it succeeded in finding heuristic solutions with similar quality than the best-known solutions. These best-known solutions were found by the column-generation algorithm in Archetti et al. (2011), tested on SDVRP instances with up to 288 customers. As mentioned before, investigating the performance of the SD1PDTSP implementation on these larger SDVRP instances is of high interest, but it would deviate the main scope of our paper.

Table 4.9 shows the results when a minimum amount of product must be pickup from or delivery to the customer in each visit. This amount is measured as a percentage shown in the Column $\%r$. It does not report the best-known value *UB'* because Gulczynski et al. (2010) consider distances with two decimals, while the distances in this study are integer numbers. Instance `S51D1` does not appear in the table because the optimal route visits each customer once, thus it serves 100% of the customer demand in each visit. As expected, the problem becomes more complicated to solve since the gap at the root node is larger and the objective value increase.

	$m = 1$				$m = 2$				$m = 3$			
Q	LB	UB	rtime	ttime	LB	UB	rtime	ttime	LB	UB	rtime	ttime
7	5387.7	**5741**	4.0	60.7	5198.1	**5337**	7.3	8.2	5085.5	5337	18.9	274.6
8	5269.9	**5355**	2.2	9.3	5021.1	**5274**	6.0	19.5	4976.8	5274	14.8	380.6
9	4993.5	**5052**	0.6	0.7	4927.0	**5044**	4.8	18.7	4824.8	5044	12.8	82.4
10	4981.0	4993	0.6	0.7	4799.7	4993	2.1	3.7	4701.9	4993	9.7	26.0
11	4822.1	4828	0.6	0.8	4735.0	4828	4.0	6.3	4729.1	4828	9.4	25.3
12	4828.0	4828	0.6	1.5	4474.5	4828	4.2	36.9	4466.3	4828	11.4	80.5
13	4641.0	4641	0.4	0.5	4444.2	4641	1.0	5.1	4441.0	4641	8.6	20.7
14	4490.0	4490	0.1	0.1	4465.4	4490	1.1	1.3	4462.7	4490	1.0	3.7
15	4490.0	4490	0.0	0.0	4483.6	4490	1.2	1.6	4480.4	4490	2.2	3.6
16	4445.0	4445	0.0	0.0	4423.0	4445	0.1	1.0	4423.0	4445	3.1	7.0

Table 4.1: Class I. SD1PDTSP instances based on data in Mosheiov (1994) with large Q.

Q	m	LB	UB	rtime	ttime
1	7	21879 .0	21879	62.6	199.3
2	4	11329.5	12093	92.9	3743.2
3	3	8605.6	9221	10.5	1644.1
4	2	7303.8	**7807**	7.7	1599.5
4	3	6978.3	**7804**	25.3	3152.6
5	2	6584.9	6773	3.9	100.7
5	3	6269.5	6773	20.1	596.5
6	2	5719.6	5788	5.7	63.9
6	3	5437.5	5788	19.6	140.1

Table 4.2: Class I. SD1PDTSP instances based on data in Mosheiov (1994) with small Q.

		$m = 1$				$m = 2$				$m = 3$			
Name	Q	LB	UB	rtime	ttime	LB	UB	rtime	ttime	LB	UB	rtime	ttime
n30A	10	5724.8	**6727**	12.6	1253.9	5654.1	**6256**	12.1	2266.4	5597.4	6256	22.8	4716.4
n30A	12	5583.8	5782	7.0	131.6	5111.6	5782	10.7	281.7	5008.6	5782	18.6	1937.2
n30A	15	5105.2	**5595**	5.8	91.2	5024.8	**5465**	7.5	137.6	5007.8	5465	15.1	399.4
n30B	10	6193.2	6603	7.3	165.8	5478.6	6603	6.8	591.1	5304.5	6603	14.8	1527.0
n30B	12	5522.0	**6229**	6.1	96.3	5003.3	**6152**	8.3	416.2	4869.7	6152	8.3	619.7
n30B	15	5094.0	5631	3.8	53.5	4632.0	5631	4.2	119.1	4454.7	5631	6.9	407.3
n30C	10	5215.5	**6486**	14.9	1197.9	5149.3	**6348**	13.4	2278.5	5005.1	6348	30.8	4289.7
n30C	12	5334.3	**5456**	5.3	69.1	5245.3	**5367**	5.3	92.9	5069.6	5367	12.7	293.4
n30C	15	4876.5	5181	1.9	14.2	4788.8	5181	2.7	42.8	4715.2	5181	7.7	176.8
n30D	10	5450.0	**6577**	22.9	2698.4	5279.2	**6380**	20.4	3811.8	5125.6	6380	26.9	4900.9
n30D	12	5368.5	**6256**	8.0	201.9	5095.9	**6025**	23.7	1907.1	5046.1	6025	23.5	3501.8
n30D	15	5285.5	**5577**	4.2	33.6	5170.6	**5568**	8.2	95.2	5029.4	5568	9.4	518.1
n30E	10	5691.2	**6070**	6.8	588.1	5402.5	**6052**	8.1	942.6	5253.1	6052	15.5	1433.8
n30E	12	5293.0	**5876**	1.5	18.3	5213.0	**5762**	3.9	71.7	5107.8	5762	11.7	234.9
n30E	15	5326.0	5416	1.7	7.8	4905.3	5416	3.1	42.8	4745.6	5416	5.1	97.6
n30F	10	5392.5	**5737**	11.5	600.4	5225.0	**5727**	14.7	1008.4	5006.9	5727	22.5	2284.3
n30F	12	5189.6	**5260**	0.7	3.2	5176.5	**5259**	4.3	62.1	4901.7	5259	10.9	188.2
n30F	15	4692.0	4893	0.1	1.5	4591.0	4893	2.7	30.8	4349.5	4893	7.1	145.7
n30G	10	8705.8	**9305**	16.0	2135.1	8641.7	**9005**	26.9	3994.7	8526.1	9005	28.1	7200.0
n30G	12	7438.0	**8497**	10.5	692.2	6066.8	**8264**	24.1	3049.5	5973.5	8264	21.9	6433.7
n30G	15	6530.0	**7424**	6.3	211.4	6453.3	**7226**	6.4	691.7	6324.5	7226	27.2	4659.7
n30H	10	5191.5	**6433**	10.7	457.9	5020.4	**6164**	30.5	4598.6	4913.7	6164	26.3	6973.4
n30H	12	5481.9	**6025**	5.3	219.5	5367.5	**5947**	20.3	1533.5	5239.3	5947	24.7	4217.6
n30H	15	5244.0	**5613**	7.6	82.7	5143.5	**5447**	7.0	241.9	4997.5	5447	21.7	1815.3
n30I	10	5156.2	**5864**	8.1	310.4	4969.5	**5596**	26.6	4839.8	4705.4	5596	25.1	7200.0
n30I	12	4646.0	**5154**	5.7	59.7	4572.6	**4991**	6.8	140.6	4228.6	4991	19.8	1683.1
n30I	15	4671.5	4762	0.9	1.9	4547.9	4762	4.0	11.4	4399.1	4762	3.5	415.1
n30J	10	5865.7	**6192**	7.3	224.8	5601.4	**6090**	19.6	1712.5	5389.5	6090	22.9	3556.4
n30J	12	5322.8	**5874**	4.6	99.7	5241.8	**5647**	8.1	179.3	5165.8	5647	10.2	654.7
n30J	15	4894.0	5349	3.7	19.2	4476.3	5349	3.7	141.7	4301.6	5349	6.0	362.7

Table 4.3: Class II. SD1PDTSP instances based on Hernández-Pérez and Salazar-González (2004b) with large Q.

		$m = 2$				$m = 3$			
Name	Q	LB	UB	rtime	ttime	LB	UB	rtime	ttime
n30A	5	10096.5	10157	21.9	3376.1	9984.2	10157	27.9	6472.6
n30A	6	8841.9	8957	18.2	3203.8	8764.1	8957	24.4	5997.5
n30A	7	7941.5	8048	14.7	2152.7	7793.8	8048	18.9	4734.1
n30B	5	10114.0	10515	25.5	3614.9	10060.7	10515	26.1	6916.3
n30B	6	9155.3	9321	19.5	3004.7	9003.5	9321	22.3	6209.7
n30B	7	8200.7	8358	17.3	2971.3	8150.6	8358	20.8	5004.6
n30C	5	9058.8	9337	18.2	3519.5	8909.9	9337	27.8	6705.2
n30C	6	8125.9	8234	8.3	2386.4	8033.7	8234	25.3	4596.1
n30C	7	7287.8	7487	12.9	1246.9	7164.8	7487	24.6	4120.7
n30D	5	8749.5	8981	18.8	3839.6	8669.1	8981	28.0	7200.0
n30D	6	7904.6	8022	16.1	2477.9	7882.7	8022	27.3	6191.4
n30D	7	7213.4	7332	11.8	329.1	7148.3	7332	22.1	2530.6
n30E	5	9066.2	**9273**	22.4	4335.7	8994.5	9733	28.6	7200.0
n30E	6	7794.6	7936	19.1	3792.5	7646.5	7936	26.9	7200.0
n30E	7	7800.3	7936	18.7	3309.6	7754.9	7936	24.5	6008.4
n30F	5	8947.1	9121	21.5	4681.8	8788.6	9567	26.1	7200.0
n30F	6	8035.7	8127	10.6	2341.8	7998.2	8127	23.3	6413.5
n30F	7	7262.4	7449	7.7	490.3	7191.8	7449	12.7	1344.8
n30G	5	14893.2	15074	20.4	4375.3	14601.5	15601	29.9	7200.0
n30G	6	12789.2	12978	16.5	3661.9	12661.2	12978	29.0	6628.9
n30G	7	11149.7	11605	12.7	946.6	11063.7	11605	19.1	2671.2
n30H	5	9261.2	9463	26.1	6505.7	9210.6	9506	22.4	7200.0
n30H	6	8292.4	8399	20.9	5016.4	8201.1	8462	20.6	7200.0
n30H	7	7776.3	7813	19.9	3958.3	7619.5	7813	20.1	6193.7
n30I	5	7800.6	7983	23.7	6867.1	7768.1	8012	24.3	7200.0
n30I	6	7107.3	7249	20.6	3816.7	7081.4	7249	23.3	6843.6
n30I	7	6398.5	6510	8.1	1268.6	6322.7	6510	17.5	2997.9
n30J	5	8564.2	8793	11.3	2497.2	8474.8	8793	25.5	6846.1
n30J	6	7896.4	8017	6.9	957.5	7743.9	8017	25.0	5944.8
n30J	7	7268.4	7406	4.5	688.6	7167.8	7406	24.1	5041.9

Table 4.4: Class II. SD1PDTSP instances based on Hernández-Pérez and Salazar-González (2004b) with small Q.

		without Benders					with Benders					
Name	Q	LB	rtime	ttime	sec	Nnode	LB	bend	sec	round	2m	Nnode
n30A	10	5422.6	36.1	6794.1	175	4799	5654.1	101	95	75	9	2179
n30A	12	5082.0	14.9	595.3	125	3931	5111.6	84	72	48	5	1871
n30A	15	4989.9	14.9	351.3	95	2865	5024.8	51	31	31	2	1101
n30B	10	5397.3	15.3	1764.4	234	1543	5478.6	110	85	46	21	757
n30B	12	4748.0	15.3	1201.7	193	1147	5003.3	92	49	35	13	406
n30B	15	4534.6	10.5	396.1	104	1002	4632.0	65	51	46	5	299
n30C	10	4953.4	35.7	6009.7	181	7868	5149.3	162	105	84	14	3246
n30C	12	5018.1	12.4	304.9	148	3429	5245.3	98	75	55	10	979
n30C	15	4676.7	12.1	209.5	94	979	4788.8	84	78	31	5	297
n30D	10	5134.8	31.1	6998.3	219	7999	5279.2	116	63	64	16	4543
n30D	12	4828.6	25.9	4741.5	131	5225	5095.9	81	59	48	6	2334
n30D	15	5001.9	16.2	669.3	81	2954	5170.6	54	58	35	2	1272
n30E	10	5390.1	25.4	3050.2	141	1806	5402.5	254	96	69	7	595
n30E	12	5121.5	12.3	322.7	111	1035	5213.0	138	82	45	2	410
n30E	15	4857.5	11.7	211.8	99	752	4905.3	90	79	31	5	100
n30F	10	5035.3	26.6	3551.9	193	6571	5225.0	85	55	43	10	3065
n30F	12	4640.5	9.3	197.1	182	4256	5176.5	96	62	41	7	725
n30F	15	4446.5	9.1	115.1	121	165	4591.0	52	48	25	8	14
n30G	10	8409.7	34.2	6977.5	135	8656	8641.7	171	89	69	20	4659
n30G	12	5918.0	32.7	6032.4	105	6899	6066.8	93	57	60	19	2345
n30G	15	6300.9	22.0	1673.9	96	5119	6453.3	99	66	29	3	1987
n30H	10	4901.5	36.9	7200.0	174	3975	5020.4	156	112	65	13	1663
n30H	12	5148.5	23.8	4147.9	112	2178	5367.5	194	84	61	4	986
n30H	15	5001.4	16.1	809.4	89	1543	5143.5	102	51	36	4	764
n30I	10	4811.2	37.4	7200.0	188	4710	4969.5	69	69	57	8	2003
n30I	12	4435.3	10.3	359.9	123	1387	4572.6	78	48	42	7	712
n30I	15	4336.8	8.9	101.7	91	867	4547.9	63	32	18	9	122
n30J	10	5321.9	28.4	5908.6	146	10589	5601.4	151	91	47	6	5416
n30J	12	5030.5	12.6	595.2	135	9939	5241.8	73	80	35	5	4977
n30J	15	4268.7	7.4	395.6	94	9108	4476.3	110	43	38	5	5101

Table 4.5: Class II. SD1PDTSP instances based on Hernández-Pérez and Salazar-González (2004b) with large Q and $m = 2$.

		$m = 1$				$m = 2$				$m = 3$			
Name	Q	LB	UB	rtime	ttime	LB	UB	rtime	ttime	LB	UB	rtime	ttime
eil22	2500					552.6	574	1.2	3.9	550.9	574	3.3	15.2
eil22	3000					490.5	509	1.1	3.9	487.1	509	3.0	13.6
eil22	3300	466.8	486	0.1	0.1	460.4	486	1.0	3.3	455.8	486	2.7	12.9
eil22	4000	419.3	435	0.0	0.1	409.1	435	0.2	1.8	401.0	435	1.7	9.3
eil22	5000	357.1	374	0.0	0.1	348.7	374	0.2	1.8	343.1	374	1.5	8.4
eil22	6000	336.4	344	0.0	0.0	331.0	344	0.1	0.5	328.2	344	0.4	2.1
eil23	3500					450.9	527	1.1	3.5	435.1	527	4.1	16.6
eil23	4100					461.5	527	0.7	2.9	458.0	527	4.2	14.0
eil23	4200	526.0	527	0.1	0.1	517.3	527	0.9	3.2	499.2	527	4.9	11.3
eil23	4300	505.2	525	0.1	0.1	497.5	525	0.2	2.8	462.6	525	4.7	8.7
eil23	4400	505.2	525	0.1	0.2	497.5	525	0.9	2.9	462.6	525	2.4	8.6
eil23	4500	502.2	506	0.1	0.2	491.2	506	1.0	2.2	473.9	506	1.2	7.1
eil23	4700	493.0	496	0.0	0.1	446.0	496	0.3	2.0	422.8	496	1.8	6.9
eil23	5000	496.0	496	0.0	0.1	446.0	496	0.1	1.9	422.8	496	1.0	3.4
eil30	2900					401.8	417	2.1	43.9	389.1	417	2.0	72.5
eil30	3000					401.8	417	1.5	32.4	389.1	417	1.6	62.9
eil30	3100	402.0	411	0.2	0.7	400.1	411	0.9	11.5	381.6	411	1.4	50.4
eil30	3200	402.0	411	0.2	0.9	400.1	411	1.1	11.0	381.6	411	1.9	53.1
eil30	3300	400.6	403	0.2	0.5	399.4	403	0.7	12.8	380.5	403	1.8	48.3
eil30	3500	398.0	399	0.2	0.3	398.0	399	0.6	13.1	378.1	399	2.1	47.0
eil30	3800	390.6	391	0.1	0.4	390.6	391	0.4	7.2	375.6	391	1.1	39.7
eil30	4100	381.0	381	0.0	0.1	381.0	381	0.3	4.7	372.2	381	0.8	38.9
eil33	4900					550.7	570	3.0	1052.2	545.0	570	7.2	3011.7
eil33	5100					546.3	570	2.7	701.0	539.2	570	6.7	1958.3
eil33	5600	525.8	555	0.1	0.2	522.1	555	2.1	368.8	517.4	555	5.4	1207.1
eil33	6000	530.8	543	0.1	0.2	524.1	543	1.8	92.0	518.6	543	5.3	1099.9
eil33	7000	487.7	**497**	0.1	0.2	479.7	**492**	1.2	56.2	471.4	492	5.3	837.6
eil33	8000	469.6	475	0.0	0.1	459.0	475	1.2	53.5	452.6	475	4.5	611.0
eil51	80	424.7	435	0.2	0.5	423.8	435	3.0	3100.7	421.3	435	7.9	4106.7
eil51	100	423.7	432	0.1	0.4	418.9	432	2.6	2641.0	418.1	432	7.4	2997.5
eil51	160	426.0	426	0.1	0.1	422.5	426	0.8	97.3	420.9	426	5.1	548.1

TABLE 4.6: Class III. SD1PDTSP instances based on data in the VRP library.

		$m = 1$				$m = 2$						$m = 3$			
Name	Q	LB	UB	rtime	ttime	LB	UB	rtime	ttime	#s	%a	LB	UB	rtime	ttime
eil22	6000	326.2	375	0.3	2.0	318.7	375	1.0	17.9	0	100.0	317.0	375	3.0	56.4
eil22	7000	348.5	370	0.1	1.6	330.8	370	0.5	14.2	0	100.0	298.5	370	2.6	59.7
eil23	4500	567.2	569	0.1	0.7	562.4	569	0.5	3.2	0	100.0	549.7	569	2.3	8.1
eil23	5000	528.6	544	0.0	0.9	475.6	544	1.1	2.7	0	100.0	460.2	544	1.2	7.2
eil30	4500	483.3	**534**	1.9	31.9	482.8	**510**	0.8	39.9	1	33.3	389.4	510	1.9	75.9
eil30	9000	379.7	387	0.1	1.3	355.1	387	0.5	5.3	0	100.0	341.1	510	1.7	30.5
eil33	8000	823.8	835	1.1	3.2	814.9	835	2.7	130.9	0	100.0	801.9	835	3.1	604.6
eil33	9000	781.1	805	0.8	2.3	771.3	805	1.6	75.4	0	100.0	757.7	805	2.9	355.9
eil51	160	509.3	521	1.0	7.8	501.9	521	1.9	903.5	0	100.0	492.5	521	7.2	2103.1
eil51	200	480.5	498	0.0	6.4	471.6	498	0.9	711.4	0	100.0	450.8	498	5.4	1788.3

TABLE 4.7: Class IV. VRP and SDVRP instances from the VRP library.

Name	$n-1$	k	Q	LB	UB	rtime	ttime	#s	%a	UB'
S51D1	50	3	160	453.1	458	0.1	312.4	0	100.0	458
S51D2	50	9	160	649.9	726	2.1	7200.0	2	20.1	726
S51D3	50	15	160	926.0	**969**	1.5	7200.0	1	25.4	**972**
S51D4	50	27	160	1522.7	**1670**	1.3	7200.0	3	29.9	**1677**
S51D5	50	23	160	1271.5	1440	2.7	7200.0	3	27.6	1440
S51D6	50	41	160	2115.3	2327	0.8	7200.0	1	15.5	2327
SD1	8	6	100	21419.2	22828	0.1	1.0	4	11.1	22828
SD2	16	12	100	70801.9	70828	1.0	46.3	10	11.1	70828
SD3	16	12	100	41084.7	43040	1.1	57.1	8	11.1	43040
SD4	24	18	100	56155.8	63062	2.1	680.0	12	11.1	63062
SD5	32	24	100	133792.3	138994	4.6	7200.0	8	11.1	138994
SD6	32	24	100	78236.0	83086	4.7	4010.6	16	11.1	83086
SD7	40	30	100	363782.9	364000	6.5	7200.0	11	11.1	364000
SD8	48	36	100	496677.6	506828	9.1	4998.1	4	11.1	506828
SD9	48	36	100	203098.0	204293	9.3	7200.0	9	11.1	204288

Table 4.8: Class V. SDVRP instances from Belenguer et al. (2000) and Chen et al. (2007), with $m = 2$.

Name	$\%r$	LB	UB	rtime	ttime	#s	%a
S51D2	30	632.4	752	1.3	7200.0	1	30.5
S51D3	30	910.7	1008	0.8	7200.0	1	30.4
S51D4	40	1548.5	1755	1.1	7200.0	2	40.7
S51D5	30	1259.0	1473	2.6	7200.0	1	30.5
S51D6	20	2076.7	2369	1.4	7200.0	1	20.6
MD1	40	22828.0	22828	0.0	1.6	4	40.4
MD2	40	70824.0	72000	1.5	59.7	11	40.4
MD3	40	43060.0	43060	0.0	43.2	8	40.4
MD4	40	62907.4	63108	5.6	429.9	12	40.4
MD5	40	131187.9	140244	12.1	3417.5	22	40.4
MD6	40	82457.5	83404	21.4	7200.0	16	40.4
MD7	40	335631.7	358830	22.3	7200.0	21	40.4
MD8	40	503539.2	504000	22.7	7200.0	21	40.4
MD9	40	204850.9	206378	22.9	7200.0	14	40.4

Table 4.9: Class V. SDVRP instances from Belenguer et al. (2000) and Gulczynski et al. (2010), with $m = 2$ and minimum delivery amount.

Chapter 5

The SD1PDTSP: The heuristic approach

The effectiveness of any branch-and-bound algorithm depends not only on the quality of the lower bound, but also on the procedure to find good upper bounds during the enumeration scheme. This latter requirement implies generating heuristic solutions for the SD1PDTSP, which is an NP-hard problem. This is a major difference between SD1PDTSP and other related problems like SDVRP. For these other problems, heuristic approaches typically adapt solutions of the split-forbidden variant by using swaps and insertions to reduce the travel cost. On the SD1PDTSP, instead, there may not exist any solution to the split-forbidden problem. Therefore, finding a heuristic solution for SD1PDTSP is a challenging optimization problem in itself. On the other hand, large instances cannot be solved with the branch-and-cut algorithm described in Chapter 4. Therefore, it is necessary a good heuristic approach to obtain good solution for large instances.

Section 5.1 describes a basic heuristic approach for the 1-PDTSP and shows how to adapt it to the SD1PDTSP. Section 5.2 presents an improvement procedure based on the branch-and-cut algorithm in Chapter 4 to obtain better results than in the previous section. Finally, Section 5.3 shows several computational results.

5.1 Basic heuristic approach for the SD1DPTSP

This section exploits the idea that the SD1PDTSP could be solved if it is applied a 1-PDTSP approach on a graph where each customer is replaced by an appropriated number of nodes, each one associated with a partial demand of the customer served in a potential visit. Clearly, finding the decomposition of the customer demands and solving the 1-PDTSP are complex problems. It proposes a heuristic approach for large-sized

1-PDTSP instances and two strategies for decomposing the customer demands. The performance of the approach on the two strategies is compared in Section 5.3 with a third strategy (Strategy 0) which simply forces each customer to be visited once. In addition, the heuristic approach and the strategies will be later used in an optimization-based procedure. While the strategies and approach described in this section do not generate solutions with the preemption characteristic, the optimization-based procedure (described in Section 5.2) is suitable to produce solutions with preemption if convenient and desired.

5.1.1 General variable neighbourhood search

In this section, it assumes that each SD1PDTSP customer i has been replaced by a set of s_i nodes $i_1, \dots, i_{s_i}$ and that each node i_l for $l = 1, \dots, s_i$ is associated with part of the customer demand d_{i_l} (where $d_i = \sum_{l=1}^{s_i} d_{i_l}$). Let $n' = \sum_{i \in I} s_i$ be the number of nodes in the 1-PDTSP instance. This section describes a procedure to construct a 1-PDTSP route among the n' nodes. Currently the best 1-PDTSP heuristic approach is a *general variable neighbourhood search* (GVNS) proposed in Mladenović et al. (2012). The *variable neighbourhood search* (VNS) (see Section 1.8) consists of a meta-heuristic approach that exploits systematically the concept of neighbourhood change within the local search algorithm. The GVNS is an adaptation of the VNS where the local search is applied to several neighbourhood structures.

Let N_k be a finite set of pre-selected neighbourhood structures for $k = 1, \dots, k_{max}$, and with $N_k(T)$ being the set of solutions in the k^{th} neighbourhood of T (where $T = (v_1, \dots, v_{n'})$ denotes a Hamiltonian cycle of n' nodes). The explorations of these neighbourhoods are used for diversification purpose (known as *shaking step*). Hence, the GVNS works as follows.

- *Initialization*: Select a initial solution T as input data.
- *Shaking step*: For each $k = 1, \dots, k_{max}$, it is chosen a random solution T' from $N_k(T)$ by performing a combination of k 3-opt and double-bridge moves. These are performed so the feasibility is kept.
- *Local search*: A new local optimal solution T'' is got by applying a VND procedure over T' as a initial solution.
  - If T'' is better than T, the new incumbent has been obtained. Then it sets $T = T''$ and the search is repeated around it.
  - Otherwise, a new random solution T' is generated from the neighbourhood $N_{k+1}(T)$.

If no better solution is found with any neighbourhood N_k, the whole procedure is repeated until a number of iterations (or the time limit) is reached.

An important ingredient in a GVNS algorithm are the operators used in a stochastic way (to escape of a local minimum) and in a deterministic way (within the VND). Fundamental operators structures for the TSP, based on edge exchanges and node-insertion moves, are extended to the 1-PDTSP. This operators are 2-opt, 3-opt, insertion and double-bridge. The 2-opt move is the simplest k-opt edge-exchange method. It consists of removing two arcs (v_i, v_{i+1}) and (v_j, v_{j+1}) from the tour and reconnecting the two paths created by adding (v_i, v_j) and (v_{i+1}, v_{j+1}). The 3-opt move consists of removing three arcs from the solution and reconnecting the paths by adding three arcs. There are various ways to reconnect the three paths. Insertion is a particular case of a 3-opt move where two of the three removed arcs are consecutive. Then, the effect of reconnecting the tour is that of moving a vertex forward or backward in a sequence. Thus, the insertion neighbourhood is split in forward insertion and backward insertion. Finally, double-bridge is a particular case of 4-opt moves keeping the orientation of the four paths obtained by removing the four arcs.

The implementation of the GVNS in Mladenović et al. (2012) was set to $k_{max} = 8$. The maximum number of iterations was set to 200 and the time limit (in seconds) was set to the number of customers of the 1-PDTSP instance. Mladenović et al. (2012) describe two kinds of VND procedures: a sequential VND and a mixed-nested VND. In the first one, three operators (2-opt, forward insertion and backward insertions) are explored one by one in a sequence. In the second one, two operators (2-opt and forward insertion) are explored in a nested way, and the other operator (backward insertions) is explored in a sequential way. Both variants of VND use a *first improvement* approach, which means that a move is made when an improvement in the neighbourhood is found for the first time.

A constructive procedure is used to generate an initial solution T. It starts at the depot (i.e. $v_1 = 1$) and by randomly choosing a first customer v_2 to be add in the initial tour T. Then, the next customer v_i is iteratively selected from a structure within of 20 closest customers to the last customer inserted v_{i-1}. From this structure, it is taken the customer with the greatest demand which has not visited yet and keep the solution feasible, i.e. the capacity constraint is not violated if that customer is added to tour T. If such a customer does not exist, for all customers not visited in the partial tour, the procedure adds the nearest customer with probability 0.9 (or a random customer with probability 0.1) that makes the route feasible. If it is not possible to insert a customer that keeps the route feasible, it adds a random customer even if the route is now infeasible. These steps are repeated until all customers are inserted. This constructive procedure was first proposed in Zhao et al. (2009) but it allows infeasible solutions in Mladenović et al. (2012).

Contrary to the TSP, not all tours are 1-PDTSP solutions because the vehicle capacity constraint. Then, before making a move in a neighbourhood, it is necessary check the feasibility. As shown in Hernández-Pérez and Salazar-González (2004b), the check can be done in $O(n')$ time. Let $T = (v_1, \ldots, v_{n'})$ denote the TSP tour to be checked. Let $l_i(T)$ the load of the vehicle after visiting the i-th node in the tour T. Hence $l_i(T) = l_{i-1}(T) - d_{v_i}$. Let $l_1 = -d_{v_1}$. Then T is 1-PDTSP feasible if and only if

$$\max_{0 \le i \le n'} \{l_i(T)\} - \min_{0 \le i \le n'} \{l_i(T)\} \le Q \tag{5.1}$$

However, the check can also be implemented in a efficient way using *binary indexed tree* (BIT) data structures as proposed in Mladenović et al. (2012). The BIT structure can be used to calculate the maximum (or minimum) of an array of n' elements in a time $O(\log n')$. This allows to reduce the complexity of the feasibility checking in the 2-opt and insertion operators. In addition, the GVNS restricts the 3-opt and double-bridge moves to those where the load through the removed arcs are the same than through the added arcs (so the feasibility of the resultant tour is warranted).

Computational results of two implementations, named sequential VND and mixed-nested VND, are shown in Mladenović et al. (2012). The solution values given by the mixed-nested VND are slightly better than those given by the sequential VND, but sequential VND is slightly faster than mixed-nested VND. Taken into account that in this dissertation it uses the 1-PDTSP algorithm several times inside of the matheuristic algorithm, it has decided to apply the sequential VND instead of the mixed-nested VND for the experiments. It has implemented the sequential VND using the description in their article. However, it did not get exactly the same results with our implementation (see Section 5.3) and it is thought that this is due to the random nature of the algorithm (and potentially non-documented details in their implementation). In this implementation, it found a better performance if it uses the best improvement approach for the VND instead of first improvement approach.

5.1.2 Strategy 1

This section describes a way of representing each SD1PDTSP customer by a set of 1-PDTSP nodes, each one associated with part of the customer demand. This first strategy exploits the empirical observation that typically customers are not visited more than twice, and preemption is unnecessary. Then one can decompose each customer demand into a small set of non-splitable pieces. Note that, when a customer is visited twice, at least $d_i/2$ units of its demand will be served together in a visit. To be precise, let $|x|$ denote the absolute value of x, $\log(x)$ the logarithm in base 2 of x, $\lfloor x \rfloor$ the integer part of real number x and $\text{sign}(x)$ be $+1$ if $x \ge 0$ and -1 otherwise. It defines $s_i = \lfloor \log(|d_i|) \rfloor + 1$ if $d_i \neq 0$, $s_i = 1$ otherwise, $d_{i_l} = \text{sign}(d_i) 2^{(l-1)}$ for $l = 1, \ldots, s_i - 1$ and $d_{s_i} = d_i - \sum_{l=1}^{s_i - 1} d_{i_l}$. The value s_i represents the number of 1-PDTSP nodes associated

d_i	s_i	d_{i_1}	d_{i_2}	d_{i_3}
$+7$	3	$+1$	$+2$	$+4$
$+6$	3	$+1$	$+2$	$+3$
$+5$	3	$+1$	$+2$	$+2$
$+4$	3	$+1$	$+2$	$+1$
$+3$	2	$+1$	$+2$	
$+2$	2	$+1$	$+1$	
$+1$	1	$+1$		
0	1	0		
-1	1	-1		
-2	2	-1	-1	
-3	2	-1	-2	
-4	3	-1	-2	-1
-5	3	-1	-2	-2
-6	3	-1	-2	-3
-7	3	-1	-2	-4

FIGURE 5.1: Transforming SD1PDTSP demands into 1PDTSP demands.

with each SD1PDTSP customer i and each term d_{i_l} represents a non-splitable piece of the customer demands. Figure 5.1 shows the values of s_i and d_{i_l} for each demand d_i in $[-7, 7]$.

Consider now a 1-PDTSP instance with a customer i_l with demand d_{i_l} for each $i \in I$ and $l = 1, \dots, s_i$. The travel cost $c_{i_k j_l}$ between nodes i_k and j_l if $c_i j$ is customers i and j are different, and 0 otherwise. It solves this instance with the approach described in Section 5.1.1. It is worth noting that each solution of the SD1PDTSP instance with at most two visits per customer and no preemption corresponds to a solution of the constructed 1-PDTSP instance. A customer i visited once in the SD1PDTSP instance corresponds to visiting the nodes $i_1, \dots, i_{s_i}$ consecutively in the 1-PDTSP instance. Moreover, if a customer is visited twice in the SD1PDTSP instance, the amount of demand served in the first visit can be obtained by adding some values d_{i_l}. Other 1-PDTSP solutions can yield SD1PDTSP solutions with some customers visited more than twice. Another observation of practical use is that quantities d_i for $i \in I$ and the vehicle capacity Q can be divided by a common integer divisor and the optimization problems does not change. Since the magnitude of d_i values has a direct impact in the number of customers in the 1-PDTSP instance, it divides the parameters d_i and Q by the greatest common divisor in a preprocessing phase.

5.1.3 Strategy 2

Strategy 1 has the disadvantage of splitting most of the SD1PDTSP customers into several 1-PDTSP nodes, thus creating large 1-PDTSP instances. An alternative way to reduce the total number of nodes is selecting only a few customers to be split several times in the 1-PDTSP and reducing the number of splits of the other customers. Equation (5.1) allows checking the feasibility of a partial 1-PDTSP route $P = (v_1, \ldots, v_t)$ in lineal time of its number of nodes. The demand values d_{v_i} are a-priori known in the 1-PDTSP. Instead, the demand served at each visit to a customer i is a-priori unknown in the SD1PDTSP, hence it is not possible to evaluate the infeasibility on a partial route P. It proposes the following adaptation to evaluate the infeasibility of these partial routes.

The procedure consists of at most m iterations, where m is the maximum number of allowed visits to a customer. Iteration 1 searches for a feasible TSP solution on a graph with $|I|$ nodes using a nearest neighbourhood algorithm. Using this TSP route the vehicle can serve a demand d_{i_k} (with $k = 1$) of the customer i. Iteration l with $l \geq 2$ solves again the nearest neighbourhood algorithm on a TSP problem with a node for each SD1PDTSP customer i such that $\sum_{k=1}^{l-1} d_{i_k} \neq d_i$.The iterative approach stops after iteration l if $\sum_{k=1}^{l} d_{i_k} = d_i$ for all $i \in I$. In such case, a feasible SD1PDTSP route has been obtained by merging the tours $P^1, \ldots, P^l$ in a single route. Tours P^l and P^{l+1} are combined into a route by selecting a customer i visiting P^l and P^{l+1} with the largest value $|d_{i_l} + d_{i_{l+1}}|$, and then the values d_v in the new tour are recomputed to make the tour feasible. This procedure may conclude with a SD1PDTSP solution that suggest a reasonable way of splitting the SD1PDTSP customer demands. Still, based on preliminary experiments, it found more convenient to further replace each of the resulting node by two nodes, one with demand $\lceil d_{i_k}/2 \rceil$ and the other with demand $d_{i_k} - \lceil d_{i_k}/2 \rceil$. If one of these numbers is zero then the resulting node remains without between replaced by two. The motivation of this enlargement of the 1-PDTSP size is because it allows more feasible SD1PDTSP solutions while keep the size of the 1-PDTSP reasonable for this algorithm. Strategy 2 consists of solving a new 1-PDTSP instance with the approach in Section 5.1.1. If a better 1PDSTP solution is found, a better SD1PDTSP has been found. In addition, the initial solution for the GVNS described in Section 5.1.1 is given by this constructive procedure.

It illustrates here the procedure on a SD1PDTSP instance with 25 customers, demands between -7 and 7, and $Q = 6$. This instance is based on benchmark data introduced in Mosheiov (1994). Figure 5.2 shows the two tours P^1 (solid lines and P^2 (dashed lines) created in the two iterations of the procedure. Each circle represents a customer and is located using the given coordinates. The demand of a customer is the value near its circle. Route P^1 was created starting from customer 1 (the depot). Customers 7, 9, 15, 16 and 18 require additional visits in order to serve their demand completely. Hence, the iteration 2 solves a new TSP problem and the solution is P^2 in the figure. Since no customer needs another visit, the algorithm stops at iteration 2. The two routes are combined as follow. The vehicle starts the tour in the depot with empty load, following route P^1 until customer 18. Then the vehicle goes through the route P^2 and it returns to customer 18. Finally, goes through P^1 from customer 18 until the depot. During this combined tour the served demands at each visit is recomputed to keep the feasibility.

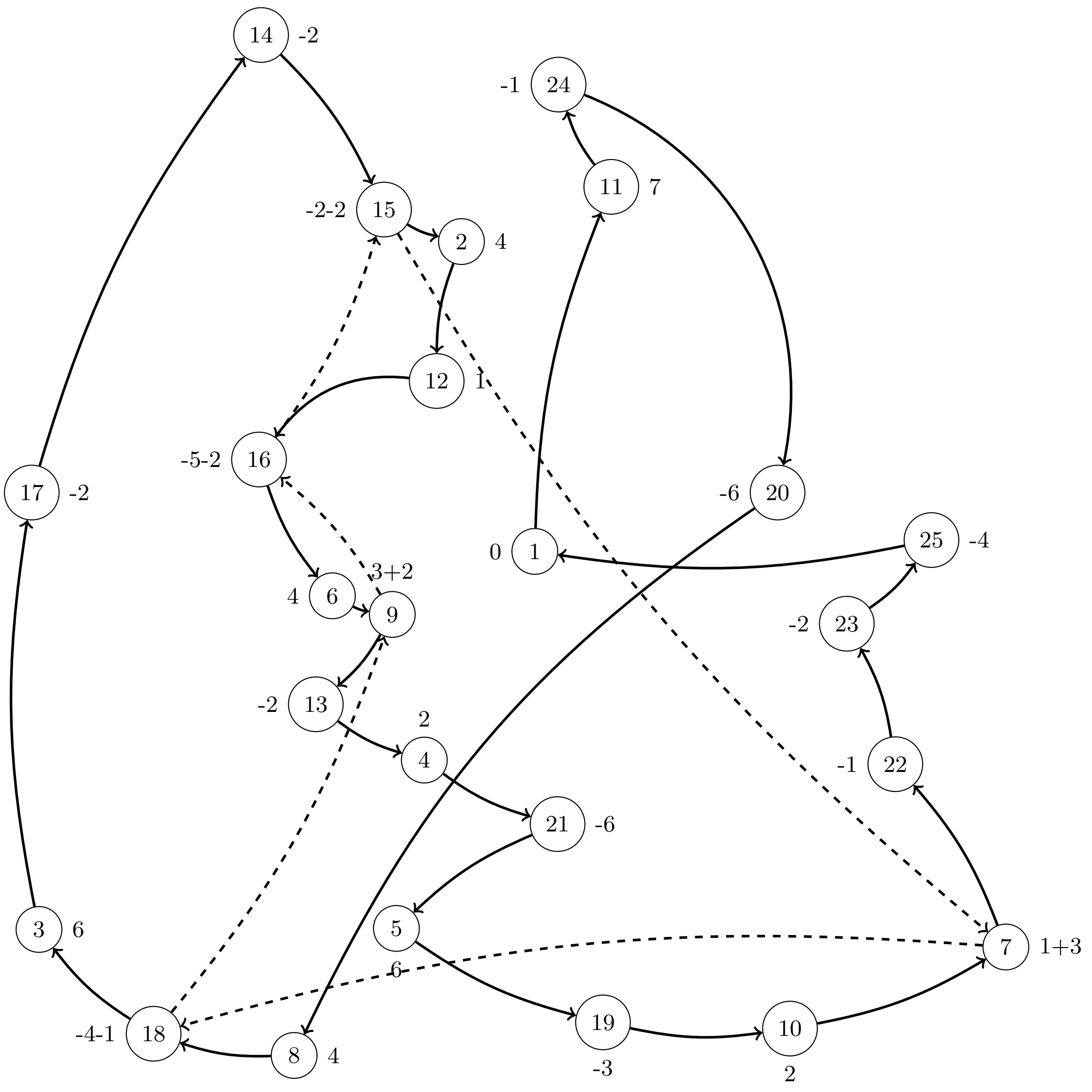


FIGURE 5.2: Initial solution by merging P^1 and P^2. The travel cost is 8327.

The travel cost of P^1 and P^2 are 5633 and 2694 respectively. Then, the travel cost of the combined tour is 8324. This solution suggests a decomposition of the SD1PDTSP customers into 1-PDTSP nodes. For example, customer 11 (which is visited once in this solution) is replaced by two nodes with demands 4 and 3; and customer 18 (which is viited twice, splitting its demands in -4 and -1) is replaced by 3 nodes with demands -2, -2 and -1. The split demand -4 generates two nodes of demands -2, and split demand -1 generates only one node with demand -1.

Again the approach in Section 5.1.1 will be applied on the new 1-PDTSP instance and a better SD1PDTSP solution may be obtained. Figure 5.3 shows the SD1PDTSP solution obtaines after apply the GVNS with this strategy on this example.

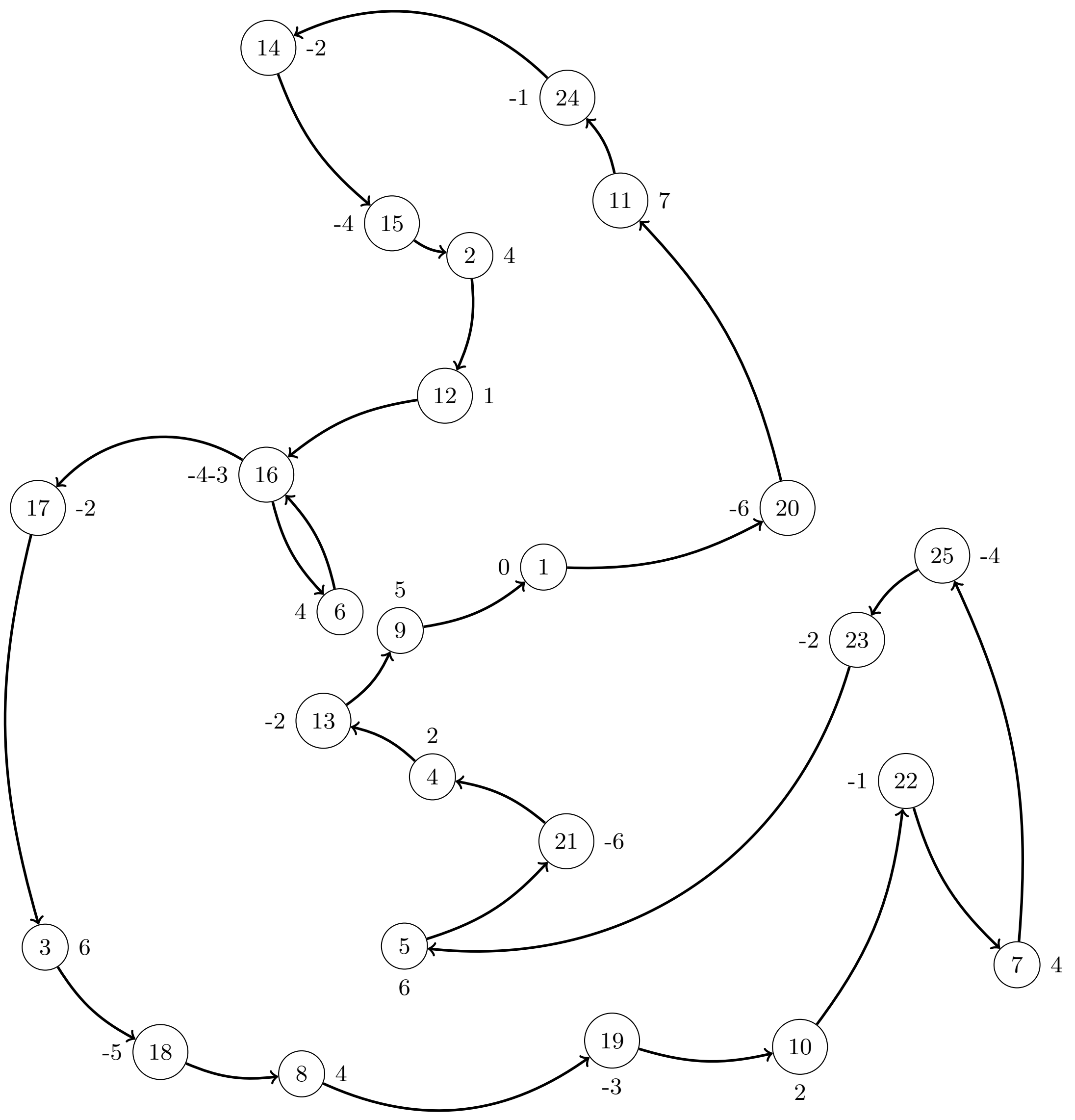


FIGURE 5.3: SD1PDTSP solution after GVNS algorithm. The travel cost is 5509.

5.2 Matheuristic algorithm for the SD1PDTSP

A matheuristic approach is an approximation algorithm that makes use of mathematical programming techniques. It is typically more sophisticated and time consuming than other classical heuristic approaches, but in some cases they may succeed in generating better solutions than basic heuristic algorithms. This section describes a matheuristic approach using a simplified variant of the branch-and-cut algorithm in Chapter 4. The motivation for using a variant and not the whole procedure is due to the heavy limitations of the branch-and-cut procedure, that hardly solves instances with more than 30 nodes. This variant adapts the mathematical model and the branch-and-cut procedure to work on a partial route with fixed initial and final locations, and where the load of the vehicle when entering the initial location is also fixed. The preemption characteristic may be allowed or forbidden within the branch-and-cut procedure as desired. In other words, this section describes a procedure to find SD1PDTSP solutions where a branch-and-cut approach is used to optimize some partial routes. It details first the mathematical model and later the overall approach.

5.2.1 The MILP subproblem

As pointed out in Chapter 3, the mathematical model (3.1)–(3.11) for SD1PDTSP produces a complete route, and it now adapts to define optimal partial routes. The adapted model will be used in our matheuristic procedure. Let us consider a subset of locations $J \subset I$. Let s be the first location, t be the last location, and l_s be the load of the vehicle when entering s in the first visit. It introduces a dummy node z with demand $d_z = -\sum_{i \in J} d_i$ and travel cost $c_{zs} = c_{tz} = 0$. Then, this adaptation is based on the model (3.1)–(3.11) where the set I is replaced by $J \cup \{z\}$, and with the additional constraints:

$$y_z = 1 \tag{5.2}$$

$$x_{zs_1} = 1 \tag{5.3}$$

$$f_{zs_1} = l_s \tag{5.4}$$

$$\sum_{l=1}^{m} x_{t_l z} = 1. \tag{5.5}$$

Equations (5.2) force z to be visited once. Equation (5.3) forces the arc from z to s_1 to be in the route, and equation (5.4) fixes its load. Equation (5.5) forces the partial route to end at customer t.

Note that the preemption characteristic can be forbidden by simply replacing equations (3.10) with:

$$\begin{aligned} 0 \le g_{i_l} \le q_i y_{i_l} \qquad & \forall i \in I : d_i \ge 0, \forall l = 1, \dots, m_i \\ -q_i y_{i_l} \le g_{i_l} \le 0 \qquad & \forall i \in I : d_i < 0, \forall l = 1, \dots, m_i \end{aligned}$$

5.2.2 Improvement phase

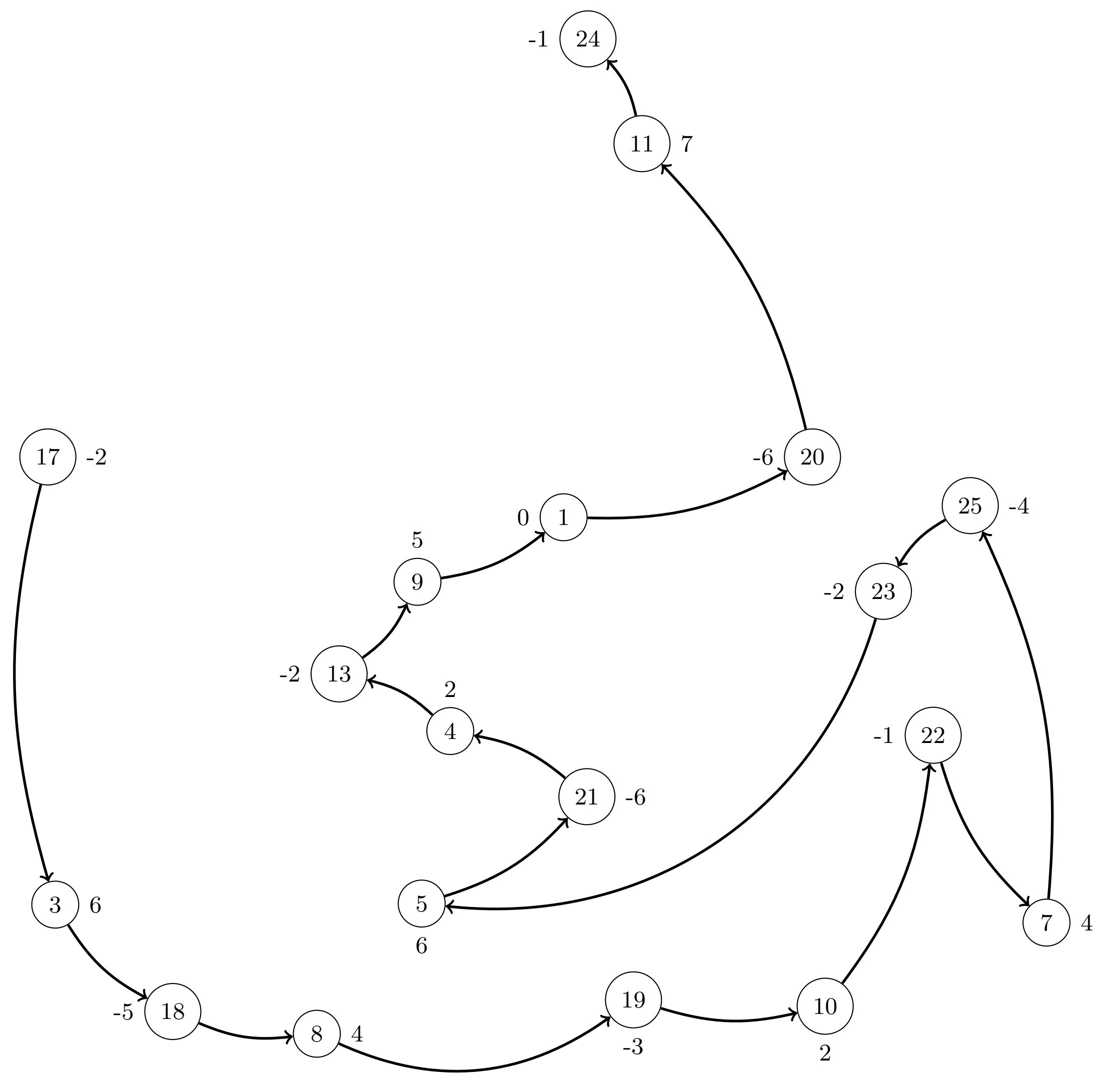
It is now described an improvement phase to potentially generate a better solution from an initial one. An SD1PDTSP solution is a route through I, and can be seen as a sequence of locations, some of which may appear several times in the sequence. Let k be the length of this sequence, and denote by $T = (v_1, \dots, v_k)$ the sequence itself and by $l_j(T)$ the load of the vehicle outgoing v_j. The matheuristic procedure iteratively analyzes partial routes and uses three additional parameters (say, p_1, p_2 and p_3). Parameter p_1 is the number of nodes in the current partial route, parameter p_2 is the number of nodes to skip for the next partial route when the MILP subproblem is not applied, and parameter p_3 is the number of nodes to skip when the MILP subproblem is applied.

The procedure starts selecting the partial route $P = (v_1, \ldots, v_{p_1})$ and checks if all locations in that route are not visited in the current route outside P. If so, the MILP subproblem is applied on P; otherwise, the partial route is moved forward p_2 positions to get another subsequence, i.e. a new $P = (v_{1+p_2}, \ldots, v_{p_1+p_2})$ is considered. When the MILP subproblem gives a new partial route that improves the cost of P, the current route T is update; otherwise, a new partial route P is selected by moving forward p_3 positions from the current P. Note that T is a circuit, thus the algorithm will go on selecting partial routes until it gets $(v_k, v_1, \ldots, v_{p_1-1})$. When all partial routes have been checked and the MILP subproblems do not improve the current solution, the algorithm stops. Each MILP subproblem is solved within a given time limit. After preliminary computational results, we set parameters $p_1 = 20$, $p_2 = 1$ and $p_3 = 5$. The time limit for solving each MILP subproblem was set to 10 seconds.

Figure 5.4 shows the initial partial route of the MILP subproblem. It corresponds a partial path of the solution shown in figure 5.3. The load of the vehicle entering the partial route through location 17 is five, and the travel cost of this partial route is 4102. When the MILP subproblem is applied, it returns the partial solution shown in figure 5.5 with travel cost 3930. It can see that all customers in the input path are visited once while the output path has two customers (customers 5 and 21) visited twice. The final partial route has a smaller travel cost than the initial one (it is 172 units lower), so the MILP subproblem was worth to be applied. It can see that the matheuristic algorithm allows generating a solution with the preemption characteristic. Customer 21 is visited twice. In the first visit (entering from customer 5), seven units of product are delivered by the customer to the vehicle. In the second visit (entering from customer 23) one unit of product is picked up by the customer from the vehicle. Moreover, on this example, the obtained route is the optimal solution of the SD1PDTSP and its cost is 5337. This route can be seen by replacing in figure 5.3 the path in figure 5.4 by the path in figure 5.5.

5.2.3 Matheuristic framework

A first SD1PDTSP solution is constructed with the basic approach described in Section 5.1.1, applying one of the three mentioned strategies. The improvement phase described in Section 5.2.2 is applied on this solution, which means that a sequence of partial routes are examined. Each time the MILP subproblem improves a partial route, the solution is replaced and the basic approach is reapplied using the decomposition of the customer demands. Another iteration is performed using the resulting new SD1PDTSP solution. The iterative procedure go on until the improvement phase is not able to generate a better solution. There are random decisions in the constructive phase that uses the



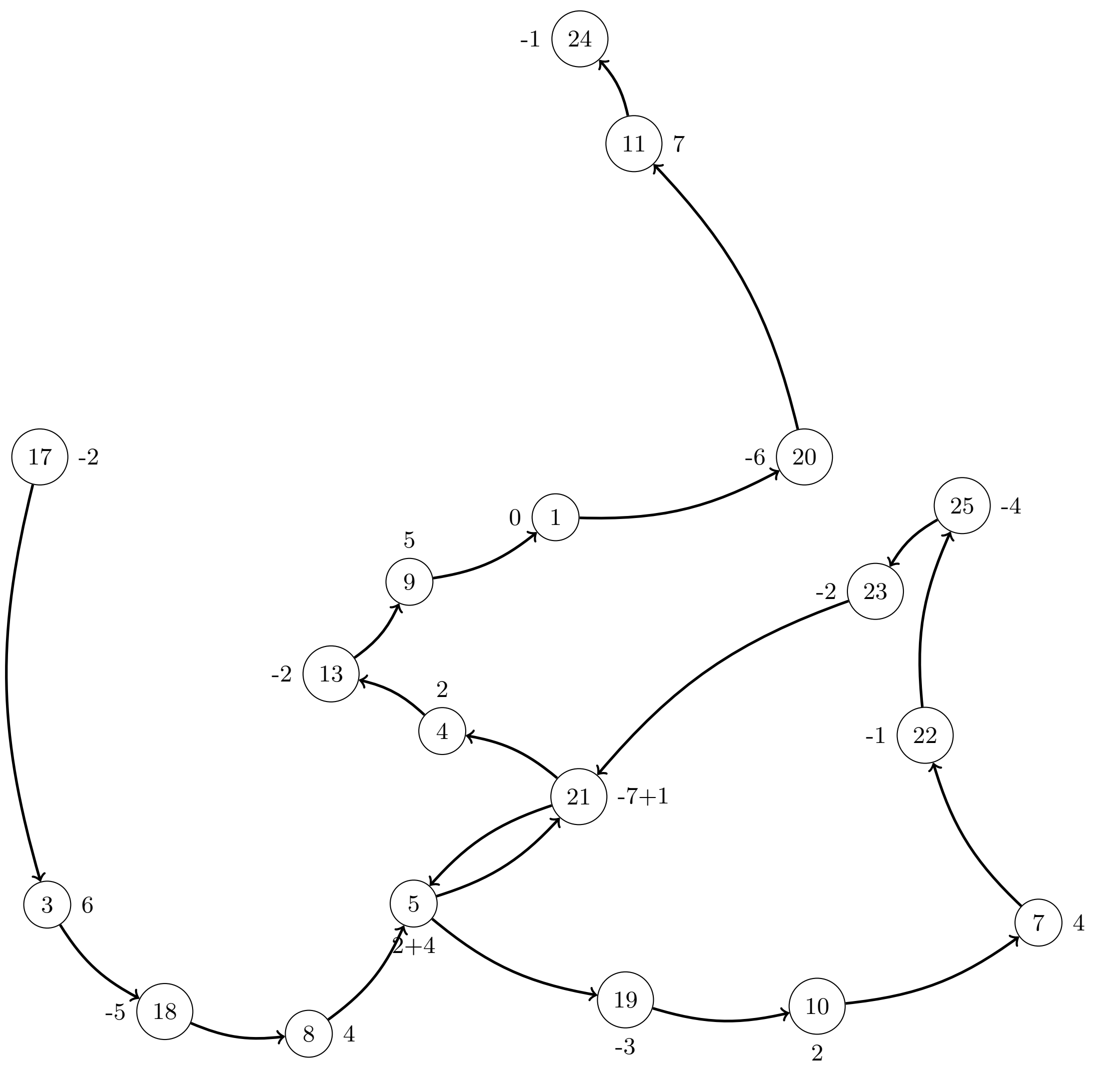
(a) Before the MILP subproblem. The travel cost is 4102.

FIGURE 5.4: Partial routes starting at location 17 with load 2.

basic approach, and it uses a math-based approach to solve the improvement phase. For that reason the whole framework can be classified as a matheuristic technique. Three implementations are possible depending on the strategy used in the first constructive step.

5.3 Computational results

The heuristic algorithm described in this chapter has been implemented in C++, and executed on a personal computer with a Intel Core 2 Duo CPU E8600 3.3 Ghz running Microsoft Windows 7, using CPLEX 12.5 to solve the MILP subproblems. To evaluate the performance of our implementation it has created SD1PDTSP instances



(b) After the MILP subproblem. The travel cost is 3930.

FIGURE 5.5: Partial routes starting at location 17 with load 2.

based on the benchmark 1-PDTSP instances proposed in Hernández-Pérez and Salazar-González (2004b). These instances were generated in the following way. The customers $2, \ldots, n$ were randomly located in the square $[-500, 500]$ and have integer demands d_i randomly chosen in the interval $[-10, 10]$. Customer 1 is located in the point $(0, 0)$ with a demand value d_1 such that the sum of all customer demands is zero. The travel costs are computed as the Euclidean distances, rounded to the closest integer numbers. The vehicle capacity is $Q = 10$. The instances are publicly available from `hhperez.webs.ull.es/PDsite/`. When necessary, it has defined $p_i = 10 - d_i$, $p'_i = 10$ and $q_i = 20$ for each customer i. Regarding the size, it has considered three classes: small instances with $n = 30$, medium instances with n in $\{40, 50, 60\}$, and large instances with n in $\{100, 200, 300, 400, 500\}$. It reports computational results on 10 instances for each value of n, and for each one it considers the case $m_i = 1$ for all i (i.e. a 1-PDTSP instance) and the cases $m_i = 2$ for all i with and without the preemption characteristic.

The algorithm in Hernández-Pérez and Salazar-González (2004a) was able to solve to optimality the small and medium 1-PDTSP instances, and the algorithm in Salazar-González and Santos-Hernández (2015) was able to solve to optimality only the small instances with $m_i = 2$, with and without preemption. In addition, the algorithm was also applied with $Q = 5$ on the large instances (i.e., $n \in \{100, 200, 300, 400, 500\}$). This forces customers with demands $|d_i| > 5$ to be visited more than once.

The results obtained from our computational results are distributed in seven tables with the following headings:

Name: name of the instance.

Best: travel cost of the best solution after having applied all the methods.

NS: number of customers visited twice in the best solution.

NP: number of customers where preemption is performed in the best solution.

Min: gap between the smallest cost z in 10 executions of a specific method respect to $Best$; it is computed as $100(z - Best)/Best$.

Avg: gap between the average cost z' in 10 executions of a specific method respect to $Best$; it is computed as $100(z' - Best)/Best$.

Time: average computational time in seconds of each execution of the specific method.

When the method is an exact approach, it is executed once instead of 10 times, and **Gap** is $100(z^* - Best)/Best$ with z^* being the optimal cost. Tables 5.1–5.3 refer to the 1-PDTSP ($m_i = 1$) and the SD1PDTSP ($m_i = 2$) without preemption, while Tables 5.4–5.6 refer to the SD1PDTSP with preemption (i.e., when refinement phase is applied).

Table 5.1 shows the results on the small instances. The best solutions on the small instances were computed solving the case $m_i = 2$ with preemption using the exact approach in Salazar-González and Santos-Hernández (2015), and no one of these solutions make use of the preemption characteristic (i.e. *NP=0*). Each line also shows the gap of the optimal values of the split-forbidden problem, and of the split-allowed problem without the preemption characteristic. The time to compute them are reported in the table, making clear that the problem with $m_i = 2$ is much harder to solve than the problem with $m_i = 1$. The computational time consumed by the basic approach is very small in both cases. Note that the number of customers visited twice by the vehicle is small. From the table, Strategy 1 shows the best performances.

Table 5.2 shows the results on the medium instances. Again, the exact algorithm for 1-PDTSP found an optimal solution for each instance in reasonable computing time, while the exact algorithm in Salazar-González and Santos-Hernández (2015) was not able to solve the instances with $m_i = 2$. The values *Best* were obtained in all cases by our matheuristic approach, and in some cases also by the basic approach described in Section 5.1. The basic approach found a very good solution to each 1-PDTSP instance in about 2 seconds, so again the results confirm the good quality of the GVNS procedure. The basic approach shows a similar performance also on the instances with $m_i = 2$. The best routes of the 30 instances with $m_i = 2$ and preemption use the split characteristic (see NS) while only on 3 over the 30 instances also use the preemption characteristic (see NP). The split characteristic allows reducing the cost in about 3%, while the reduction for the preemption characteristic is negligible. Now, the performances of Strategy 1 and Strategy 2 are similar.

Table 5.3 shows the results on the large instances, where no optimal solution is known. Value *Best* was computed by the best execution of our math-based approach on the three problem variants for each line in the table. The table shows the results reported by Mladenović et al. (2012), so their and our GVNS implementations can be compared in computational time and solution quality. Their computer uses the Intel Core 2 Duo CPU T5800 2.0 GHz running Linux, which is very similar to our computer. Although our implementation loss a bit of quality in the solution cost, it is faster than their implementation. Taking into account the average gap of the ten runs on the ten instances of each size n, it observes Strategy 2 won twice ($n = 100$ and $n = 300$), Strategy 1 won twice ($n = 400$ and $n = 500$) and the Strategy 0 with the implementation reported by Mladenović et al. (2012) won when $n = 200$ (but it spent more time). it can also see that the computational time of Strategy 1 is higher than the computational time of Strategy 2. It may be because the number of nodes in the 1-PDTSP instance when using Strategy 1 is greater than the number of nodes when using Strategy 2.

Tables 5.4, 5.5 and 5.6 shows details of three variants of the matheuristic approach on the SD1PDTSP with the preemption characteristic. For each instance the tables show:

Math St0: The matheuristic algorithm described in Section 5.2 is applied starting from the 1-PDTSP solution generated by the approach in Section 5.1.1 on Strategy 0;

Math St1: As in *Math St0*, but Strategy 1 in Section 5.1.2 is used instead of Strategy 0;

Math St2: As in *Math St0*, but Strategy 2 in Section 5.1.3 is used instead of Strategy 0.

It is interesting to observe that very few of the best routes make use of the preemption characteristic. It occurs on 3 instances over the 30 median-sized problems, and on 7 instances over the 50 large-sized problems. This explains why the matheuristic approach showed very similar results when applied to the problem without the preemption characteristic on these instances. The improvement in solution quality of the math-based approach on the GVNS route is around 1%, and it shows very similar performances when

				$m_i = 1$					$m_i = 2$ without preemption								
				Optimal		Strategy 0			Optimal		Strategy 1			Strategy 2			
Name	Best	NS	NP	Gap	Time	Min	Avg	Time	Gap	Time	Min	Avg	Time	Min	Avg	Time
n30q10A	6256	1	0	2.35	1.0	2.35	2.35	0.6	0.00	2266.4	0.00	0.00	1.9	0.00	0.00	1.5
n30q10B	6603	0	0	0.00	0.1	0.00	0.00	0.5	0.00	591.1	0.00	0.00	1.8	0.00	0.00	1.3
n30q10C	6348	3	0	2.17	0.4	2.17	2.42	0.5	0.00	2278.5	0.00	0.00	2.5	0.57	0.57	1.7
n30q10D	6380	5	0	4.26	0.3	4.26	4.26	0.5	0.00	3811.8	0.00	0.14	2.4	0.27	0.28	1.6
n30q10E	6052	4	0	0.30	0.1	0.30	0.30	0.4	0.00	942.6	0.00	0.16	1.6	0.26	0.26	1.0
n30q10F	5727	1	0	0.17	0.1	0.17	0.17	0.4	0.00	1008.4	0.00	0.00	1.5	0.00	0.00	1.0
n30q10G	9005	4	0	4.06	1.3	4.06	4.06	0.8	0.00	3994.7	0.00	0.04	3.6	0.43	0.63	3.0
n30q10H	6164	2	0	4.33	0.2	4.33	4.33	0.5	0.00	4598.6	0.00	0.00	1.8	0.39	0.39	1.1
n30q10I	5596	3	0	4.02	0.4	4.02	4.10	0.5	0.00	4839.8	0.00	0.51	1.9	0.00	0.00	1.3
n30q10J	6090	3	0	1.59	0.5	1.59	1.59	0.4	0.00	1712.5	0.00	0.36	2.3	0.00	0.00	1.4
Average	6422.1	2.6	0	2.33	0.4	2.33	2.36	0.5	0.00	2604.4	0.00	0.12	2.1	0.19	0.21	1.5

TABLE 5.1: Results of the basic approach on small instances

using each of the three strategies. In other words, the impact of the selected strategy for the matheuristic approach is not relevant on these instances. The number of MILP subproblems solved is around 10 and the number of MILP subproblems generating a better partial route is around 2.

Finally, it performed other experiments by solving the same benchmark instances with $Q = 5$ instead of $Q = 10$. Since the customer demands are in the interval $[-10, +10]$, the routes for the instances with $Q = 5$ contain a larger number of customers visited twice, and Strategy 0 can not be applied. Still the number of customers making use of the preemption characteristic was negligible. Table 5.7 shows the computational results and gaps from these experiments. Based on these results, Strategy 2 provides better results than Strategy 1, specially on the instances with $n = 300$.

				$m_i = 1$					$m_i = 2$ without preemption					
				Optimal		Strategy 0			Strategy 1			Strategy 2		
Name	Best	NS	NP	Gap	Time	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time
n40q10A	7035	3	0	1.96	5.1	1.96	2.00	1.0	0.00	0.86	3.6	0.33	1.31	2.6
n40q10B	6138	2	0	6.83	16.2	6.83	6.92	1.9	0.00	0.04	2.3	0.00	0.00	2.2
n40q10C	7501	1	0	0.36	0.3	0.36	0.36	1.0	0.00	0.00	3.2	0.00	0.00	2.6
n40q10D	7781	3	0	3.57	7.3	3.57	3.57	1.2	0.00	0.00	3.9	0.00	0.21	2.5
n40q10E	6728	3	0	2.97	17.8	2.97	2.97	0.6	0.00	0.14	2.9	0.00	0.10	1.7
n40q10F	7300	3	0	2.82	51.2	2.82	2.97	1.0	0.00	0.11	4.3	0.16	0.35	2.9
n40q10G	7457	3	0	2.24	2.5	2.24	2.36	0.9	0.00	0.01	3.8	0.08	0.09	3.1
n40q10H	6638	3	0	2.30	2.6	2.30	2.30	0.8	0.00	0.15	3.7	0.05	0.93	3.0
n40q10I	7018	3	0	2.81	6.8	2.81	2.81	0.9	0.00	0.19	3.8	0.48	0.48	2.6
n40q10J	6457	3	0	0.85	2.0	0.85	0.85	0.6	0.00	0.14	3.0	0.12	0.20	2.2
Average	7005.3	2.7	0	2.67	11.2	2.67	2.71	1.0	0.00	0.16	3.5	0.12	0.37	2.5
n50q10A	6795	2	1	2.83	2.6	2.83	2.83	1.1	0.04	0.04	3.7	0.04	0.04	2.6
n50q10B	9170	4	0	3.47	26.2	3.47	3.50	1.6	0.00	0.79	6.4	0.34	1.11	4.9
n50q10C	8842	2	0	3.03	99.6	3.03	3.07	1.6	0.00	0.17	7.6	0.78	0.98	4.8
n50q10D	9940	4	0	3.22	21.8	3.22	4.09	2.1	0.11	0.66	6.7	1.23	1.30	5.5
n50q10E	9238	4	1	2.75	9.4	2.75	2.75	2.1	0.08	0.59	6.5	0.10	0.22	5.9
n50q10F	7722	5	0	12.46	119.8	12.46	12.46	1.8	0.00	0.55	5.6	0.09	2.33	4.7
n50q10G	7067	2	0	0.83	1.4	0.83	1.09	1.1	0.01	0.08	4.4	0.07	0.12	3.0
n50q10H	8653	4	0	2.68	67.6	2.72	3.02	2.1	0.00	0.19	6.0	0.00	0.41	4.6
n50q10I	8056	5	0	3.39	32.0	3.39	3.77	1.4	0.24	1.07	6.3	0.07	0.64	4.4
n50q10J	8230	4	0	2.75	6.8	2.75	2.75	1.2	0.00	1.62	5.0	0.30	0.85	4.5
Average	8371.3	3.6	0.2	3.74	38.7	3.74	3.93	1.6	0.05	0.58	5.8	0.30	0.80	4.5
n60q10A	8340	4	0	3.14	882.1	3.14	3.37	1.8	0.00	0.43	5.7	0.35	0.48	5.5
n60q10B	8389	5	0	1.49	23.3	1.49	1.49	1.8	0.00	1.39	4.3	1.11	1.29	4.2
n60q10C	9153	4	0	3.28	119.6	3.28	3.50	2.6	0.00	0.42	7.4	0.02	0.32	5.9
n60q10D	10625	4	1	4.10	287.6	4.10	5.08	3.1	0.08	0.80	8.2	0.17	0.55	6.5
n60q10E	9345	3	0	1.52	23.0	1.52	2.31	1.8	0.00	1.20	6.9	0.00	1.13	6.0
n60q10F	8325	5	0	8.86	227.9	8.97	9.89	2.2	0.00	1.35	4.2	1.15	1.33	4.7
n60q10G	8736	3	0	2.01	32.3	2.01	2.32	1.9	0.00	0.49	8.3	0.48	0.99	6.1
n60q10H	8211	4	0	2.59	28.6	2.59	2.68	1.5	0.00	0.69	5.5	0.00	0.53	4.2
n60q10I	9232	4	0	1.75	251.5	2.53	2.82	2.3	0.05	0.92	5.6	0.05	0.81	6.8
n60q10J	8226	6	0	6.37	228.7	6.37	7.01	1.8	1.30	3.19	3.8	0.71	1.95	4.8
Average	8858.2	4.2	0.1	3.51	210.4	3.60	4.05	2.1	0.14	1.09	6.0	0.40	0.94	5.5

TABLE 5.2: Results of the basic approach on medium instances

				$m_i = 1$						$m_i = 2$ without preemption					
				Mladenovic et al.			Strategy 0			Strategy 1			Strategy 2		
Name	Best	NS	NP	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time
n100q10A	11458	5	0	1.84	3.42	9.7	2.23	3.07	6.3	0.34	1.88	7.0	0.28	1.96	6.8
n100q10B	12729	7	0	1.64	2.00	25.3	2.39	3.42	8.0	0.00	3.31	12.2	1.54	2.87	9.1
n100q10C	13457	7	0	3.24	3.30	23.4	3.49	4.60	6.9	1.14	2.84	9.9	0.00	1.71	7.1
n100q10D	14156	7	0	0.63	1.01	25.4	0.63	2.43	9.0	0.00	1.86	10.1	0.14	1.42	10.1
n100q10E	11140	7	0	2.36	4.74	7.5	2.43	3.50	4.9	0.20	2.44	8.3	1.26	2.22	5.1
n100q10F	11514	8	0	0.83	0.93	19.7	0.83	1.37	7.6	0.22	1.88	8.3	1.00	1.98	5.9
n100q10G	11321	8	0	4.81	5.88	9.1	5.57	5.99	6.7	1.57	3.28	8.6	0.66	2.78	5.5
n100q10H	12483	8	0	1.31	1.73	30.3	1.39	2.70	7.4	0.95	2.20	10.3	0.63	1.87	6.7
n100q10I	13608	5	0	1.05	1.88	20.3	1.37	2.44	7.3	0.24	1.62	11.2	0.00	2.44	6.8
n100q10J	12936	6	0	1.77	2.99	12.2	2.86	3.78	8.2	0.74	2.20	6.7	0.93	1.74	6.9
Average	12480.2	6.8	0	1.95	2.79	18.3	2.32	3.33	7.2	0.54	2.35	9.3	0.64	2.10	7.0
n200q10A	17005	0	0	0.14	1.95	73.0	0.30	2.45	22.0	1.21	3.01	37.2	1.01	3.07	32.0
n200q10B	17291	7	0	0.86	1.31	79.5	1.89	4.24	11.5	0.35	3.19	50.9	0.91	2.57	27.2
n200q10C	15869	11	0	1.63	2.72	71.8	2.79	3.59	13.2	0.01	1.71	35.8	0.33	2.15	25.5
n200q10D	20769	15	0	0.24	1.25	81.7	1.11	2.39	25.1	0.26	2.02	54.6	0.71	1.73	46.0
n200q10E	18766	6	1	0.47	1.75	70.3	1.74	2.79	29.7	0.04	1.71	58.6	0.26	2.31	32.1
n200q10F	21104	10	0	0.69	1.49	75.0	0.60	1.73	50.3	0.88	1.40	57.4	0.59	1.34	50.6
n200q10G	17004	12	0	0.31	1.44	71.5	1.27	2.83	12.3	0.00	1.52	40.1	0.77	2.06	35.5
n200q10H	20291	0	0	2.11	3.33	73.2	0.00	3.22	33.1	2.34	3.46	56.9	2.66	3.59	40.3
n200q10I	17660	16	1	1.05	1.95	80.9	2.26	3.08	23.6	0.47	2.75	48.4	0.45	1.50	36.0
n200q10J	18556	8	0	0.37	2.56	50.5	2.86	4.20	22.4	0.84	2.62	49.2	2.23	3.17	36.9
Average	18431.5	8.5	0.2	0.79	1.98	72.7	1.48	3.05	24.3	0.64	2.34	48.9	0.99	2.35	36.2
n300q10A	21826	22	0	1.41	3.60	133.3	3.44	4.47	25.3	0.00	2.21	156.7	2.03	2.74	101.3
n300q10B	21966	26	0	0.94	2.00	118.9	1.71	3.31	35.0	0.27	1.77	158.0	0.03	2.15	109.0
n300q10C	20782	9	0	0.66	1.97	118.9	2.77	3.85	20.7	0.46	2.50	127.3	1.02	2.49	85.2
n300q10D	24607	22	0	0.46	2.00	117.8	1.66	2.73	35.6	0.25	1.49	164.3	1.03	1.88	146.6
n300q10E	25682	30	0	1.48	3.62	117.9	3.46	4.41	40.2	0.00	2.85	175.7	0.75	2.26	146.5
n300q10F	23563	15	0	0.71	2.47	125.9	1.93	3.48	27.9	0.00	1.96	196.0	0.20	2.53	114.3
n300q10G	22930	19	0	1.22	2.45	118.9	1.82	3.75	28.3	0.65	1.63	179.4	0.42	2.10	113.2
n300q10H	21152	14	0	0.15	1.27	128.2	1.91	3.43	25.9	1.23	2.77	140.5	1.48	2.11	113.7
n300q10I	23369	15	0	0.57	1.91	119.9	2.62	3.81	32.8	1.44	2.63	174.2	0.99	1.83	125.7
n300q10J	21583	22	0	0.60	2.26	132.0	3.28	4.33	25.4	0.71	2.60	121.7	0.00	1.83	96.6
Average	22746	19.4	0	0.82	2.36	123.2	2.46	3.76	29.7	0.50	2.24	159.4	0.80	2.19	115.2
n400q10A	29374	41	0	1.80	3.06	174.0	3.03	4.22	75.6	0.11	2.04	509.2	0.00	2.31	289.9
n400q10B	23394	25	1	1.13	2.24	162.8	2.05	4.08	48.7	1.69	3.52	359.1	1.08	2.52	222.8
n400q10C	27323	27	1	0.83	2.03	159.5	1.59	2.79	55.6	0.29	1.69	431.3	0.91	1.43	208.1
n400q10D	22697	20	0	2.19	3.21	160.6	3.63	4.91	42.0	1.20	3.01	302.6	1.38	2.44	204.5
n400q10E	23820	22	0	1.67	3.55	180.2	2.97	4.47	44.0	0.46	2.21	352.8	1.41	2.75	203.4
n400q10F	25765	24	0	0.52	1.86	174.1	1.79	3.02	50.3	0.29	1.11	401.8	0.00	1.63	221.7
n400q10G	23034	23	1	0.67	2.41	178.1	1.55	3.42	39.2	0.60	2.05	277.1	1.37	2.05	193.9
n400q10H	24368	38	0	0.10	1.38	161.3	1.42	2.71	47.6	0.94	1.66	311.1	1.29	2.56	231.6
n400q10I	27421	26	0	0.80	2.04	174.8	2.19	3.39	52.8	1.01	1.78	387.8	0.69	2.06	234.8
n400q10J	24530	29	0	0.97	3.05	81.3	2.26	3.55	47.2	0.00	2.09	353.4	0.08	1.98	241.0
Average	25172.6	27.5	0.3	1.07	2.48	160.7	2.25	3.66	50.3	0.66	2.12	368.6	0.82	2.17	225.2
n500q10A	26877	27	1	1.05	2.53	222.2	1.66	3.41	74.4	0.73	1.97	949.5	0.92	2.06	477.9
n500q10B	25397	28	0	0.84	2.22	213.2	0.99	2.72	67.5	0.01	1.74	804.9	0.98	1.87	384.4
n500q10C	29127	0	0	0.13	2.20	134.4	0.00	2.10	102.5	1.03	1.75	960.0	0.63	2.09	480.2
n500q10D	28842	34	0	1.46	2.80	127.2	1.60	3.25	90.7	0.77	2.20	818.4	0.70	2.08	409.4
n500q10E	28770	26	0	1.34	2.63	139.4	2.29	3.19	95.1	0.92	1.92	934.3	0.68	1.90	476.0
n500q10F	27172	23	0	1.25	2.35	215.3	2.23	2.70	104.2	0.17	2.32	908.4	1.03	2.89	372.8
n500q10G	25489	38	0	0.99	1.93	212.6	1.97	3.27	67.1	0.00	1.36	679.4	1.39	2.34	281.7
n500q10H	34776	36	0	0.63	2.33	160.9	0.75	2.78	141.4	0.09	1.49	1046.3	0.95	1.76	605.8
n500q10I	29039	35	1	0.80	3.00	124.9	1.15	2.67	94.3	0.85	2.06	880.9	1.83	2.34	484.9
n500q10J	29122	38	0	1.35	3.16	127.4	2.90	4.13	120.8	0.00	2.29	1011.1	1.68	2.55	499.9
Average	28461.1	28.5	0.2	0.98	2.52	167.8	1.55	3.02	95.8	0.46	1.91	899.3	1.08	2.19	447.3

TABLE 5.3: Results of the basic approach on large instances

				$m_i = 2$ with preemption								
				Math St0			Math St1			Math St2		
Name	Best	NS	NP	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time
n30q10A	6256	1	0	0.00	0.00	212.0	0.00	0.00	184.5	0.00	0.00	128.8
n30q10B	6603	0	0	0.00	0.00	210.2	0.00	0.00	85.4	0.00	0.00	128.7
n30q10C	6348	3	0	0.57	0.57	230.1	0.00	0.00	144.4	0.57	0.57	122.0
n30q10D	6380	5	0	0.00	0.22	229.5	0.00	0.05	115.6	0.00	0.12	107.7
n30q10E	6052	4	0	0.26	0.26	225.7	0.00	0.16	91.3	0.26	0.26	116.4
n30q10F	5727	1	0	0.00	0.00	226.2	0.00	0.00	117.2	0.00	0.00	115.8
n30q10G	9005	4	0	0.27	0.63	225.7	0.00	0.04	143.4	0.27	0.42	115.4
n30q10H	6164	2	0	0.00	0.00	233.9	0.00	0.00	135.6	0.00	0.00	133.3
n30q10I	5596	3	0	0.00	0.00	239.6	0.00	0.00	53.6	0.00	0.00	122.4
n30q10J	6090	3	0	0.00	0.00	211.4	0.00	0.00	35.3	0.00	0.00	111.9
Average	6422.1	2.6	0	0.11	0.17	224.4	0.00	0.03	110.6	0.11	0.14	120.2

TABLE 5.4: Results of the matheuristic on small instances

				$m_i = 2$ with preemption								
				Math St0			Math St1			Math St2		
Name	Best	NS	NP	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time
n40q10A	7035	3	0	0.00	0.97	238.9	0.00	0.86	119.4	0.33	1.23	112.9
n40q10B	6138	2	0	0.00	0.04	238.6	0.00	0.03	94.1	0.00	0.00	120.2
n40q10C	7501	1	0	0.00	0.00	230.4	0.00	0.00	199.3	0.00	0.00	137.0
n40q10D	7781	3	0	0.00	0.26	240.8	0.00	0.00	118.6	0.00	0.17	143.8
n40q10E	6728	3	0	0.00	0.00	228.2	0.00	0.00	143.4	0.00	0.07	125.8
n40q10F	7300	3	0	0.00	0.10	249.5	0.00	0.04	122.9	0.00	0.13	141.3
n40q10G	7457	3	0	0.00	0.01	251.0	0.00	0.01	116.4	0.00	0.00	88.8
n40q10H	6638	3	0	0.00	0.13	233.4	0.00	0.12	74.0	0.05	0.25	99.8
n40q10I	7018	3	0	0.00	0.21	242.4	0.00	0.11	91.0	0.48	0.48	80.3
n40q10J	6457	3	0	0.00	0.07	214.6	0.00	0.04	97.0	0.12	0.13	119.7
Average	7005.3	2.7	0	0.00	0.18	236.8	0.00	0.12	117.6	0.10	0.25	117.0
n50q10A	6795	2	1	0.00	0.03	198.0	0.00	0.03	98.0	0.00	0.04	129.7
n50q10B	9170	4	0	0.00	0.13	284.0	0.00	0.09	169.9	0.00	0.46	160.7
n50q10C	8842	2	0	0.00	0.14	235.2	0.00	0.12	93.6	0.00	0.55	138.3
n50q10D	9940	4	0	0.00	0.30	256.2	0.00	0.19	152.0	0.03	0.39	165.6
n50q10E	9238	4	1	0.00	0.33	233.1	0.00	0.09	163.0	0.00	0.07	172.0
n50q10F	7722	5	0	0.00	0.74	228.7	0.00	0.54	119.8	0.09	0.88	153.0
n50q10G	7067	2	0	0.00	0.04	226.6	0.00	0.05	118.7	0.07	0.11	174.2
n50q10H	8653	4	0	0.00	0.05	253.8	0.00	0.03	181.2	0.00	0.00	152.3
n50q10I	8056	5	0	0.07	0.34	234.1	0.07	0.26	120.8	0.00	0.23	156.7
n50q10J	8230	4	0	0.00	0.13	276.7	0.00	0.00	242.5	0.00	0.58	172.3
Average	8371.3	3.6	0.2	0.01	0.22	242.6	0.01	0.14	145.9	0.02	0.33	157.5
n60q10A	8340	4	0	0.00	0.12	295.7	0.00	0.09	185.3	0.00	0.12	194.5
n60q10B	8389	5	0	0.00	0.96	287.6	0.00	0.96	183.5	0.69	0.92	207.1
n60q10C	9153	4	0	0.00	0.18	303.7	0.00	0.14	256.2	0.00	0.16	247.9
n60q10D	10625	4	1	0.01	0.11	331.8	0.01	0.06	273.6	0.00	0.19	239.4
n60q10E	9345	3	0	0.00	0.29	275.6	0.00	0.39	187.4	0.00	0.41	211.9
n60q10F	8325	5	0	0.00	0.65	233.2	0.00	0.42	154.2	0.37	0.82	186.7
n60q10G	8736	3	0	0.00	0.49	234.3	0.00	0.30	126.3	0.00	0.67	129.0
n60q10H	8211	4	0	0.00	0.27	266.5	0.00	0.19	160.1	0.00	0.25	208.9
n60q10I	9232	4	0	0.24	0.88	261.0	0.00	0.54	190.8	0.05	0.55	221.6
n60q10J	8226	6	0	0.00	1.25	232.9	0.00	1.16	131.4	0.28	0.82	176.8
Average	8858.2	4.2	0.1	0.02	0.52	272.2	0.00	0.43	184.9	0.14	0.49	202.4

TABLE 5.5: Results of the matheuristic on medium instances

				$m_i = 2$ with preemption								
				Math St0			Math St1			Math St2		
Name	Best	NS	NP	Min.	Ave.	Time	Min.	Ave.	Time	Min.	Ave.	Time
n100q10A	11458	5	0	0.00	1.33	211.9	0.34	1.26	234.6	0.00	1.25	247.8
n100q10B	12729	7	0	0.06	1.64	249.4	0.00	2.03	255.6	0.06	1.65	213.3
n100q10C	13457	7	0	0.61	1.12	202.4	0.66	1.42	229.7	0.00	1.04	227.5
n100q10D	14156	7	0	0.45	0.88	215.0	0.00	0.97	236.1	0.14	0.93	239.2
n100q10E	11140	7	0	0.00	1.31	195.3	0.13	1.35	250.0	0.00	0.94	236.1
n100q10F	11514	8	0	0.42	0.79	199.0	0.00	0.97	251.4	0.42	0.87	261.4
n100q10G	11321	8	0	0.00	1.48	211.3	0.02	1.82	256.7	0.00	1.27	230.7
n100q10H	12483	8	0	0.17	0.72	199.2	0.00	0.68	227.4	0.24	0.92	239.5
n100q10I	13608	5	0	0.79	1.19	240.1	0.24	1.08	256.9	0.00	1.51	268.8
n100q10J	12936	6	0	0.52	1.32	200.0	0.00	0.91	257.8	0.52	1.14	249.0
Average	12480.2	6.8	0	0.30	1.18	212.4	0.14	1.25	245.6	0.14	1.15	241.3
n200q10A	17005	0	0	0.00	1.64	189.4	0.98	2.00	294.1	0.24	1.97	262.8
n200q10B	17291	7	0	0.74	1.44	274.3	0.35	2.09	256.7	0.00	1.54	273.3
n200q10C	15869	11	0	0.00	1.23	324.0	0.01	1.28	268.1	0.00	0.93	251.8
n200q10D	20769	15	0	0.00	1.05	289.9	0.12	1.00	262.3	0.00	1.06	257.1
n200q10E	18766	6	1	0.37	1.27	298.7	0.04	1.13	263.3	0.00	1.36	312.7
n200q10F	21104	10	0	0.50	0.87	263.5	0.00	0.89	265.8	0.50	0.97	323.7
n200q10G	17004	12	0	0.15	1.21	253.3	0.00	1.18	268.1	0.15	1.23	246.6
n200q10H	20291	0	0	0.00	2.58	282.9	2.34	2.68	286.1	2.32	3.26	292.8
n200q10I	17660	16	1	0.45	1.18	287.6	0.00	1.02	273.0	0.45	1.18	256.3
n200q10J	18556	8	0	0.00	1.88	314.7	0.84	1.77	264.6	0.00	1.88	278.6
Average	18431.5	8.5	0.2	0.22	1.44	277.8	0.47	1.50	270.2	0.37	1.54	275.6
n300q10A	21826	22	0	0.00	1.77	325.3	0.00	1.87	425.9	1.31	2.13	391.9
n300q10B	21966	26	0	0.39	1.23	400.8	0.27	1.47	370.0	0.00	1.43	373.7
n300q10C	20782	9	0	1.00	1.70	457.6	0.46	1.65	422.5	0.00	1.53	394.3
n300q10D	24607	22	0	0.93	1.47	384.9	0.00	1.04	428.3	1.02	1.42	340.4
n300q10E	25682	30	0	0.35	1.55	449.0	0.00	2.06	480.6	0.35	1.76	366.0
n300q10F	23563	15	0	0.00	1.48	387.9	0.00	1.70	447.6	0.19	1.71	376.7
n300q10G	22930	19	0	0.56	1.08	388.3	0.00	1.22	413.1	0.42	1.21	405.4
n300q10H	21152	14	0	1.17	1.83	403.6	1.15	1.51	433.0	0.00	1.84	366.2
n300q10I	23369	15	0	0.00	0.96	372.8	0.63	1.60	458.6	0.00	1.18	368.9
n300q10J	21583	22	0	0.00	1.86	375.6	0.26	1.86	406.7	0.00	1.47	411.4
Average	22746	19.4	0	0.44	1.49	394.6	0.28	1.60	428.6	0.33	1.57	379.5
n400q10A	29374	41	0	0.00	1.82	434.1	0.11	1.81	705.9	0.00	1.82	573.3
n400q10B	23394	25	1	0.00	1.75	479.3	1.15	2.18	692.6	0.00	1.78	653.2
n400q10C	27323	27	1	0.06	0.86	486.3	0.10	0.85	771.5	0.00	0.74	629.5
n400q10D	22697	20	0	1.38	2.10	422.0	0.00	1.80	701.8	1.38	2.04	631.5
n400q10E	23820	22	0	0.00	1.61	464.0	0.46	1.50	731.3	0.00	1.61	621.4
n400q10F	25765	24	0	0.00	0.80	463.6	0.23	0.81	793.2	0.00	0.88	685.5
n400q10G	23034	23	1	0.00	1.43	438.9	0.24	1.33	669.4	0.00	1.48	587.7
n400q10H	24368	38	0	0.27	1.00	469.6	0.00	1.14	703.8	0.27	1.05	523.2
n400q10I	27421	26	0	0.58	1.46	502.4	0.00	1.44	726.6	0.38	1.31	633.0
n400q10J	24530	29	0	0.04	1.23	417.2	0.00	1.36	724.9	0.04	1.29	642.1
Average	25172.6	27.5	0.3	0.23	1.41	457.7	0.23	1.42	722.1	0.21	1.40	618.0
n500q10A	26877	27	1	0.00	1.16	525.0	0.00	1.16	1206.0	0.57	1.41	966.8
n500q10B	25397	28	0	0.35	0.71	558.9	0.00	0.73	1072.6	0.35	0.92	879.5
n500q10C	29127	0	0	0.00	1.04	517.8	0.79	1.30	1220.1	0.28	1.52	894.1
n500q10D	28842	34	0	0.00	1.00	491.7	0.52	1.40	1167.8	0.00	1.00	923.0
n500q10E	28770	26	0	0.00	0.93	538.8	0.54	1.30	2004.9	0.00	1.06	966.2
n500q10F	27172	23	0	0.17	1.82	524.0	0.00	1.48	1221.7	1.03	2.53	915.8
n500q10G	25489	38	0	0.20	1.81	536.8	0.00	0.88	1044.6	0.20	1.53	897.3
n500q10H	34776	36	0	0.25	1.40	460.5	0.09	0.86	1410.5	0.00	0.90	900.8
n500q10I	29039	35	1	0.61	1.35	524.3	0.00	0.89	1161.3	0.61	1.47	965.7
n500q10J	29122	38	0	0.54	2.00	553.6	0.00	1.85	1346.1	0.54	1.95	955.2
Average	28461.1	28.5	0.2	0.21	1.32	523.1	0.19	1.19	1285.6	0.36	1.43	926.4

TABLE 5.6: Results of the matheuristic on large instances

				$m_i = 2$ without preemption						$m_i = 2$ with preemption					
				Strategy 1			Strategy 2			Math St1			Math St2		
Name	Best	NS	NP	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time	Min	Avg	Time
n100q5A	17195	39	2	0.79	1.68	15.4	0.85	1.39	25.3	0.00	1.10	218.9	0.58	1.03	258.5
n100q5B	20609	45	0	0.49	1.53	22.1	0.00	1.14	35.6	0.03	0.89	241.5	0.00	0.64	187.9
n100q5C	22069	41	0	0.16	0.84	30.5	0.22	1.37	44.3	0.00	0.55	162.5	0.22	0.76	194.5
n100q5D	23456	51	0	0.28	1.67	27.1	0.00	1.10	48.4	0.28	1.67	216.7	0.00	0.55	181.8
n100q5E	17561	54	0	1.28	3.06	17.5	0.00	2.62	26.2	0.88	2.86	219.1	0.00	0.88	226.9
n100q5F	17641	42	0	0.33	2.39	16.9	0.22	1.48	29.8	0.00	1.45	221.3	0.22	1.02	216.7
n100q5G	17496	41	1	1.22	1.97	18.1	0.95	1.77	24.6	0.00	1.55	232.5	0.00	1.00	263.3
n100q5H	20117	47	1	0.15	2.28	29.1	0.66	1.72	31.6	0.15	1.63	217.0	0.00	1.01	187.8
n100q5I	22120	54	0	0.13	2.07	25.8	0.73	2.65	40.6	0.00	1.35	238.0	0.62	1.63	169.3
n100q5J	21116	41	0	0.00	0.73	27.8	0.18	1.00	51.8	0.00	0.73	215.3	0.18	0.59	193.5
Average	19938	44.9	0.4	0.48	1.82	23.0	0.38	1.62	35.8	0.13	1.38	218.3	0.18	0.91	208.0
n200q5A	27141	94	0	0.28	2.26	79.4	0.38	1.83	120.2	0.00	1.74	274.4	0.00	0.93	310.5
n200q5B	28244	106	0	0.69	2.00	89.1	0.34	1.44	139.9	0.55	1.85	265.7	0.00	0.66	297.6
n200q5C	24324	84	0	0.53	1.80	72.0	0.00	2.00	99.0	0.53	1.65	236.1	0.00	1.47	310.6
n200q5D	34275	102	0	0.46	1.62	114.6	0.00	0.88	184.0	0.40	1.61	296.4	0.00	0.56	300.9
n200q5E	31117	100	0	0.91	2.17	113.7	0.04	2.09	142.3	0.91	2.17	270.5	0.00	1.60	324.5
n200q5F	34539	95	0	0.00	1.56	137.0	0.92	2.02	221.0	0.00	1.25	300.4	0.92	1.65	354.8
n200q5G	27197	91	0	0.61	1.51	85.7	0.00	1.17	117.6	0.61	1.44	270.1	0.00	1.09	313.3
n200q5H	34195	97	0	0.01	1.21	113.4	0.54	1.60	213.4	0.01	1.08	278.3	0.00	0.80	297.5
n200q5I	28325	94	0	0.06	2.44	79.8	0.58	2.01	157.5	0.00	2.20	255.1	0.58	1.44	328.5
n200q5J	30130	100	0	0.97	2.25	97.0	0.70	1.74	131.1	0.97	2.11	249.0	0.00	1.08	286.1
Average	29948.7	95.6	0	0.45	1.88	98.2	0.35	1.68	152.6	0.40	1.71	269.6	0.15	1.13	312.4
n300q5A	35224	147	0	0.02	0.82	253.9	0.00	0.78	255.4	0.02	0.81	378.1	0.00	0.75	378.4
n300q5B	35814	156	0	0.16	1.36	270.6	0.00	1.11	273.6	0.16	1.33	377.4	0.00	1.11	417.6
n300q5C	33831	147	0	0.33	1.39	228.1	0.00	0.66	250.3	0.33	1.39	366.9	0.00	0.65	416.6
n300q5D	40920	140	0	0.10	0.96	287.6	0.00	0.83	336.3	0.10	0.95	364.0	0.00	0.82	417.0
n300q5E	43654	166	1	0.28	0.83	357.1	0.18	0.95	327.4	0.00	0.77	478.5	0.18	0.95	443.0
n300q5F	39098	158	0	0.00	1.53	278.4	0.10	1.18	303.6	0.00	1.49	380.8	0.10	1.18	422.8
n300q5G	37388	142	1	0.63	1.38	271.6	0.00	0.84	263.2	0.63	1.36	374.2	0.00	0.82	404.2
n300q5H	33515	146	0	0.89	2.06	263.5	0.00	1.67	274.6	0.89	2.06	373.3	0.00	1.66	419.6
n300q5I	38464	152	0	0.00	0.97	259.3	0.01	0.70	287.4	0.00	0.96	381.8	0.01	0.70	423.0
n300q5J	34908	149	0	0.00	1.10	218.6	0.02	1.08	226.7	0.00	1.08	340.4	0.02	1.08	364.8
Average	37281.6	150.3	0.2	0.24	1.24	268.9	0.03	0.98	279.9	0.21	1.22	381.5	0.03	0.97	410.7
n400q5A	49087	212	0	0.35	0.90	794.0	0.00	0.72	477.2	0.35	0.90	910.1	0.00	0.72	777.5
n400q5B	37642	188	0	0.00	1.32	539.2	0.46	1.32	498.3	0.00	1.32	819.6	0.46	1.32	771.8
n400q5C	44276	185	1	0.06	1.39	653.6	0.93	2.07	522.2	0.06	1.37	850.7	0.00	1.26	799.3
n400q5D	35213	189	0	0.36	1.56	563.8	0.00	1.28	483.4	0.36	1.55	799.1	0.00	0.74	772.4
n400q5E	37689	201	2	0.73	1.72	621.4	1.13	2.24	450.6	0.73	1.70	801.8	0.00	1.94	778.9
n400q5F	41097	210	1	0.54	1.48	563.6	0.33	1.32	559.5	0.54	1.48	813.0	0.00	0.69	773.6
n400q5G	36240	185	0	0.12	0.70	516.5	0.43	0.75	445.1	0.12	0.70	755.8	0.00	0.65	779.4
n400q5H	38325	204	0	0.52	1.23	607.5	0.00	1.17	646.9	0.52	1.20	794.4	0.00	1.17	865.5
n400q5I	45448	221	0	0.00	1.00	705.1	0.24	0.96	707.5	0.00	1.00	837.7	0.24	0.96	827.9
n400q5J	39238	195	0	0.51	2.12	580.0	0.00	1.54	515.5	0.51	2.12	810.6	0.00	1.53	770.3
Average	40425.5	199	0.4	0.32	1.34	614.5	0.35	1.34	530.6	0.32	1.33	819.3	0.07	1.10	791.7
n500q5A	42359	258	0	0.54	1.90	949.5	0.16	1.59	477.9	0.14	1.17	1166.0	0.00	0.96	966.8
n500q5B	38628	226	0	1.22	2.04	804.9	0.82	1.68	384.4	1.00	1.52	1062.6	0.00	0.97	879.5
n500q5C	46956	245	0	0.15	1.55	910.9	0.49	1.78	474.7	0.01	0.91	1200.7	0.00	0.82	894.1
n500q5D	46505	239	0	0.72	1.45	818.4	0.00	0.89	409.4	0.64	1.16	1167.8	0.00	0.66	923.0
n500q5E	46376	238	0	0.00	1.35	934.3	0.24	1.31	476.0	0.00	0.78	2004.9	0.16	0.91	966.2
n500q5F	43203	242	0	0.75	1.76	908.4	0.03	1.13	372.8	0.00	1.13	1221.7	0.01	0.74	915.8
n500q5G	38906	223	0	0.93	2.36	679.4	1.54	2.82	281.7	0.72	1.77	1044.6	0.00	2.13	1763,5
n500q5H	58123	251	0	0.00	1.46	1046.3	1.30	2.00	605.8	0.00	0.99	1410.5	1.01	1.41	898.8
n500q5I	47007	255	0	0.65	1.72	880.9	0.00	1.26	484.9	0.35	1.47	1161.3	0.00	1.12	965.7
n500q5J	47224	244	0	0.28	1.52	1011.1	0.00	1.00	499.9	0.08	1.03	1346.1	0.00	0.88	955.2
Average	45528.7	242.1	0	0.52	1.71	894.4	0.46	1.55	446.8	0.29	1.19	1278.6	0.12	1.06	1012.9

TABLE 5.7: Results of the basic approach and the matheuristic approach on large instances for $Q = 5$

Chapter 6

Conclusion

The literature on vehicle routing problems has traditionally assumed that customers can only be visited once. In recent years several research articles have been motivating and analyzing models and algorithms for problems without this assumption. This thesis contributes to the literature by introducing a new routing problem in which a capacitated vehicle is used to transport a single commodity from pickup locations to delivery locations, and multiple visits to the same location are allowed. Several assumptions have been presented to properly define the problem, but it has also shown how other variants of this problem can be addressed with the results in this thesis. Some examples of these assumptions are: the vehicle must visit each customer at least once; the initial load of the vehicle must be computed; preemption is allowed, i.e. collecting or delivering temporarily commodity in a customer.

This thesis describes a single-commodity flow formulation, theoretical results, and a exact and a heuristic approach for the SD1PDTSP. The exact algorithm is based on a branch-and-cut approach and also a Benders' decomposition on the flow variables. The Benders' cuts have been analyzed, and a separation procedure based on solving a max-flow problem has been described. It has also described other inequalities to strengthen the linear programming relaxation. The branch-and-cut approach has been implemented and evaluated on five classes of benchmark instances. As it occurs in other split-demand routing problems in the literature, our experiments confirm that splitting the demand may reduce the travel cost. The approach in this thesis can be applied to solve both the classical Capacitated Vehicle Routing Problem and also its split-demand variant. While our implementation is not competitive with the last developments on the first problem, we have shown that it gives good results on the second.

The heuristic approach is based on a sophisticated technique for the 1-PDTSP which is adapted to the SD1PDTSP, and a math-based technique based on the branch-and-cut algorithm. We address two variants depending whether it is allowed or forbidden a preemption characteristic that consists of collecting or delivering temporarily commodity in a customer. It has described a basic approach that first replaces each customer by a set of nodes, and then applies a meta-heuristic algorithm for the non-split variant on these nodes. Three strategies are analyzed for the replacement step. Although this basic approach do not generate routes with the preemption characteristic, another math-based technique is used later to potentially improve the route. The technique uses a MILP formulation that is solved within a branch-and-cut approach to optimize a partial route. The procedures were implemented and analyzed to solve benchmark instances with up to 500 customers. Our experiments show that, while the split-demand characteristic helps to find better routes, the preemption characteristic does not help on most of our instances.

This thesis also opens interesting questions. One concerns a worst-case analysis, trying to find instances where the travel cost of a SD1PDTSP route is smaller than half of the travel cost of an optimal 1-PDTSP route, or proves that no one exists under some conditions (e.g. costs satisfying the triangular inequality). Similar investigations have been done on related problems by other authors (e.g. Archetti et al. (2006a), Xiong et al. (2013), Wang et al. (2014)). Another question could be to extend the results to the multi-commodity case in the line of the splittable pickup and delivery problem with reloads studied in Kerivin et al. (2008).

Addressing stochastic demands, multicriteria optimization, several vehicles, several depots, and/or time-windows when visiting customers are also other topics that deserve future investigations.

Bibliography

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows. In G. Nemhauser, A. Kan, and M. Todd, editors, *Optimization*, pages 211–369. Elsevier North-Holland, Inc. US, 1989.

R.E. Aleman and R.R. Hill. A tabu search with vocabulary building approach for the vehicle routing problem with split demands. *Int. J. Metaheuristics*, 1(1):55–80, May 2010.

R.E. Aleman, X. Zhang, and R.R. Hill. A ring-based diversification scheme for routing problems. *Int. J. Math. Oper. Res.*, 1(1-2):163–190, 2009.

R.E. Aleman, X. Zhang, and R.R. Hill. An adaptive memory algorithm for the split delivery vehicle routing problem. *Journal of Heuristics*, 16(3):441–473, 2010.

S. Anily and J. Bramel. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics*, 46(6):654–670, 1999.

C. Archetti, N. Bianchessi, and M. G. Speranza. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58(4):241–254, 2011.

C. Archetti, N. Bianchessi, and M.G. Speranza. Branch-and-cut algorithms for the split delivery vehicle routing problem. *European Journal of Operational Research*, 238(3): 685 – 698, 2014.

C. Archetti, M.W.P. Savelsbergh, and M.G. Speranza. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, 40(2):226–234, 2006a.

C. Archetti, M.W.P. Savelsbergh, and M.G. Speranza. To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):114–123, 2008a.

C. Archetti and M. G. Speranza. The split delivery vehicle routing problem: A survey. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 103–122. Springer US, 2008.

C. Archetti and M. G. Speranza. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1-2):3–22, 2012.

C. Archetti, M. G. Speranza, and A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64—73, 2006b.

C Archetti, M.G. Speranza, and M.W.P. Savelsbergh. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42(1):22–31, 2008b.

P. Augerat. *Approche Polyèdrale du Problème de Tournées de Véhicules.* Phd thesis, Institut National Polytechnique de Grenoble, Jun 1995.

A. Bachem and M. Grötschel. *New aspects of polyhedral theory.* Institut für Ökonometrie und Operations-Research Bonn: Report. Inst. für Ökonometrie und Operations Research, 1980.

M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows.* Wiley, 2010.

J. M. Belenguer, M. C. Martínez, and E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48:801–810, 2000.

G. Berbeglia, J. F. Cordeau, I. Gribkovskaia, and L. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31, 2007.

C. Berge. *Graphs and Hypergraphs.* North-Holland mathematical library. Amsterdam, 1973.

J.A. Bondy and U.S.R. Murty. *Graph Theory.* Graduate Texts in Mathematics. Springer London, 2007.

M. Boudia, C. Prins, and M. Reghioui. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In T. Bartz-Beielstein, M. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 16–30. Springer Berlin Heidelberg, 2007.

D. Chemla, F. Meunier, and R. Wolfler-Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.

S. Chen, B. Golden, and E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4):318–329, 2007.

N. Christofides. *Graph Theory: An Algorithmic Approach (Computer Science and Applied Mathematics).* Academic Press, Inc., New York, 1975.

V. Chvátal. Edmonds polytope and weakly Hammiltonian graphs. *Mathematical Programming*, 5:29–40, 1973.

V. Chvátal. *Linear Programming.* Series of books in the mathematical sciences. Freeman, W. H., 1983.

L. C. Coelho, J. F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48:1–19, 2014.

G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, 1985.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

G. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6: 80–91, 10 1959.

M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45: 7–19, 2014.

G. Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192, 2010.

M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Appl. Math.*, 50:239–254, May 1994.

M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37(3): 383–402, 1990.

J. Edmonds. Maximum matching and a polyhedron with $0, 1$ vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

M. Fischetti, J.J. Salazar-González, and P. Toth. The symmetric generalized traveling salesman polytope. *Networks*, 26(2):113–123, 1995.

M. Fischetti, J.J. Salazar-González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994.

B. Fleischmann. A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research*, 21(3):307 – 317, 1985.

J. Fonlupt and D. Naddef. The traveling salesman problem in graphs with some excluded minors. *Math. Program.*, 53(2):147–172, January 1992. ISSN 0025-5610.

M. Gendreau, P. Dejax, D. Feillet, and C. Gueguen. Vehicle routing with time windows and split deliveries. *Technical Report 2006-851, Laboratoire Informatique dAvignon*, 2006.

L. Gouveia. A result on projection for the vehicle routing problem. *European Journal of Operational Research*, 85(3):610 – 624, 1995.

L. Gouveia and M. Ruthmair. Load-dependent and precedence-based models for pickup and delivery problems. *Computers and Operations Research*, 63:56–71, 2015.

M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195 – 1220, 1984.

M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem i: Inequalities. *Mathematical Programming*, 16(1):265–280, 1979.

D. Gulczynski, B. Golden, and E. Wasil. The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E*, 46(5):612–626, 2010.

A.L. Henry-Labordere. The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. *Revue Francaise d'Informatique De Recherche Operationnelle B2*, pages 43–49, 1969.

H. Hernández-Pérez, I. Rodríguez-Martín, and J.J. Salazar-González. A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research*, 36(5):1639 – 1645, 2009. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

H Hernández-Pérez, J. Salazar-González, and B. Santos-Hernández. Heuristic for the split-demand one-commodity pickup-and-delivery travelling salesman problem. Submitted to European Journal of Operation Research, 2016.

H. Hernández-Pérez and J. J. Salazar-González. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, 196(3):987–995, 2009.

H. Hernández-Pérez and J.J. Salazar-González. The one-commodity pickup-and-delivery travelling salesman problem. *Lecture Notes in Computer Science 2570*, pages 89–104, 2003.

H. Hernández-Pérez and J.J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145: 126–139, 2004a.

H. Hernández-Pérez and J.J. Salazar-González. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, 38:245–255, 2004b.

H. Hernández-Pérez and J.J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks*, 50:258–272, 2007.

M. Jin, K. Liu, and R.O. Bowden. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics*, 105(1):228 – 242, 2007.

M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M.O. Ball et al., editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 225–330. Elsevier Publisher B.V., Amsterdam, 1995.

D. Karapetyan and G. Gutin. Lin-kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research*, 208(3):221 – 232, 2011.

H. Kerivin, M. Lacroix, and A. Mahjoub. Models for the single-vehicle preemptive pickup and delivery problem. *Journal of Combinatorial Optimization*, 23(2):196–223, 2012.

H. Kerivin, M. Lacroix, A. Mahjoub, and A. Quilliot. The splittable pickup and delivery problem with reloads. *European Journal of Industrial Engineering*, 2(2):112–133, 2008.

G. Laporte, H. Mercure, and Y. Nobert. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, 18(2): 185 – 197, 1987.

G. Laporte and Y. Nobert. Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR Journal*, 21(1):61–75, 1983.

E.L. Lawler, D.B. Shmoys, A.H.G.R. Kan, and J.K. Lenstra. *The Traveling Salesman Problem*. John Wiley & Sons, Incorporated, 1985.

C. Lee, White C. Epelman, M., and Y. Bozer. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B: Methodological*, 40(4):265 – 284, 2006.

N. Letchford, W. Eglese, and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94(1):21–40, 2002.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

M. McNabb, J. Weir, R. Hill, and S. Hall. Testing local search move operators on the vehicle routing problem with split deliveries and time windows. *Computers and Operations Research*, 56:93–109, 2015.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

N. Mladenović, D. Urosević, S. Hanafi, and A. Ilić. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*, 220(1):270 – 285, 2012.

L. Moreno, M. Poggi, and E. Uchoa. Improved lower bounds for the split delivery vehicle routing problem. *Operations Research Letters*, 38:302–306, 2010.

G. Mosheiov. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79(2):299–310, 1994.

D. Naddef and G. Rinaldi. The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities. *Mathematical Programming*, 51(1):359–400, 1991.

D. Naddef and G. Rinaldi. The graphical relaxation: a new framework for the symmetric travelling salesman polytope. *Mathematical Programming*, 58:53–88, 1993.

D. Naddef and G. Rinaldi. *The Vehicle Routing Problem*, chapter Branch-and-cut Algorithms for the Capacitated VRP, pages 53–84. Society for industrial ans Applied mathematics, USA, 2001.

D. Naddef and G. Rinaldi. The symmetric traveling salesman polytope: New facets from the graphical relaxation. *Mathematics of Operations Research*, 32:233–256, 2007.

G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1988.

M. Nowack, O. Ergun, and C. White. Pickup and delivery with split loads. *Transportation Science*, 42:32–43, 2008.

M. Nowack, O. Ergun, and C. White. Precedence constrained pickup and delivery with split loads. *International Journal of Logistics-research and Applications*, 12(1):1–14, 2012.

M. Oswald, G. Reinelt, and D.O. Theis. On the graphical relaxation of the symmetric traveling salesman polytope. *Mathematical Programming*, 110(1):175–193, 2006.

M. Padberg and M. Grötschel. Polyhedral computations. In E. Lawler, J. Lenstra, A. Kan, and D. Shmoys, editors, *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, chapter 9, pages 307–360. Wiley, 1985.

M. Padberg and M. Rao. Odd minimum cut-sets and *b*-matchings. *Mathematics of Operations Research*, 1982.

M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, 33:60–100, February 1991.

S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems. part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58 (1):21–51, 2008a.

S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008b.

D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. *Integer Programming and Combinatorial Optimization: 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, chapter Improved Branch-Cut-and-Price for Capacitated Vehicle Routing, pages 393–403. Springer International Publishing, Cham, 2014.

C. Pintea, P. Pop, and C. Chira. The generalized traveling salesman problem solved with ant algorithms. *CoRR*, abs/1310.2350, 2013.

T. Raviv, M. Tzur, and I.A. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Trasnportation and Logistics*, 2: 187–229, 2013.

C. Rego and F. Glover. *The Traveling Salesman Problem and Its Variations*, chapter Local Search and Metaheuristics, pages 309–368. Springer US, Boston, MA, 2007.

J. Renaud and F.F. Boctor. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research*, 108(3): 571 – 584, 1998.

J.J. Salazar-González and B. Santos-Hernández. The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Research Part B: Methodological*, 75(0):58 – 73, 2015.

M. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation science*, 29:17–29, 1995.

J. Silberholz and B. Golden. The generalized traveling salesman problem: A new genetic algorithm approach. In E. Baker, A. Joseph, A. Mehrotra, and M. Trick, editors, *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, volume 37 of *Operations Research/Computer Science Interfaces Series*, pages 165–181. Springer US, 2007.

M.M. Silva, A. Subramanian, and L.S. Ochi. An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research*, 53(0):234 – 249, 2015.

S.S. Srivastava, S. Kumar, R.C. Garg, and P. Sen. Generalized traveling salesman problem through n sets of nodes. *CORS Journal*, 7:97–101, 1969.

D.O. Theis. A note on the relationship between the graphical traveling salesman polyhedron, the symmetric traveling salesman polytope, and the metric cone. *Discrete Applied Mathematics*, 158(10):1118 – 1120, 2010.

D.O. Theis. On the facial structure of symmetric and graphical traveling salesman polyhedra. *Discrete Optimization*, 12:10 – 25, 2014.

P. Toth and D. Vigo. *Vehicle Routing:Problems,Methods, and Applications*. MOS-SIAM Series on Optimization, 2014.

X. Wang, B. Golden, and D. Gulczynski. A worst-case analysis for the split delivery capacitated team orienteering problem with minimum delivery amounts. *Optimization Letters*, 8(8):2349–2356, 2014.

J. Wilck IV and T. Cavalier. A construction heuristic for the split delivery vehicle routing problem. *American Journal of Operations Research*, 2(2):153–162, 2012a.

J. Wilck IV and T. Cavalier. A genetic algorithm for the split delivery vehicle routing problem. *American Journal of Operations Research*, 2(2):207–216, 2012b.

Y. Xiong, D. Gulczynski, D. Kleitman, B. Golden, and E. Wasil. A worst-case analysis for the split delivery vehicle routing problem with minimum delivery amounts. *Optimization Letters*, 7(7):1597–1609, 2013.

F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers and Industrial Engineering*, 56 (4):1642–1648, 2009.