Bencomo Domínguez-Martín

# Design of optimal routes for Vehicles and Drivers

**Diseño de rutas óptimas para vehículos y conductores**

**Directores:**
**Juan José Salazar-González**
**Inmaculada Rodríguez-Martín**

2018

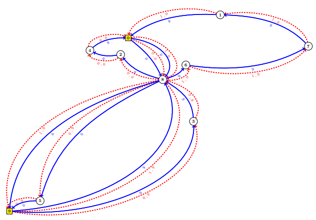
Universidad de La Laguna

Departamento de Matemáticas, Estadística
e Investigación Operativa

Bencomo Domínguez-Martín

# Design of optimal routes for Vehicles and Drivers

## Diseño de rutas óptimas para vehículos y conductores



**Under the guidance of:**
**Juan José Salazar-González**
**Inmaculada Rodríguez-Martín**

2018

D. Juan José Salazar González, Catedrático de Universidad del Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna y D.ª Inmaculada Rodríguez Martín, Profesora Titular de Universidad del Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna,

CERTIFICAN:

Que la presente memoria, titulada "Diseño de rutas óptimas para vehículos y conductores" (en inglés "Design of optimal routes for Vehicles and Drivers"), ha sido realizada bajo nuestra dirección por D. Bencomo Domínguez Martín y constituye su Tesis para optar al grado de Doctor en Matemáticas y Estadística 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 15 de junio de 2018

Juan José Salazar González | Inmaculada Rodríguez Martín

To my family

# Contents

# Acknowledgments

This memory has been done under the supervision of professors Juan José Salazar González and Inmaculada Rodríguez Martín, 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, have helped me in developing the PhD-research.

I appreciate the support by the Spanish Government through the Research Project MTM2013-36163-C06-01 for the grant 'Ayudas para contratos predoctorales para la formación de doctores 2013' (BES-2013-064640), which has made this PhD-research possible. This thesis has been also supported by the Research Project MTM2015-63680-R.

It was a great honour for me to work with the group of the Department of Computer Science in the Instituto de Matemática e Estatística of the Universidade de São Paulo of Brazil (São Paulo, Brazil, September 5-December 5, 2017) which for three months contributed to improve my knowledge in Operations Research. I could enjoy lectures related to my thesis, learn a new language, and meet interesting people. I would also like to thank the 'Agencia Estatal de Investigación', for the grant 'Movilidad predoctoral para la realización de estancias breves en Centros de I+D, convocatoria 2016', which made this work in Brazil possible.

I cannot forget to thank my parents, my sister, my girlfriend, and my friends for their support during these years.

# Preface

From the beginning of human life, the need to optimize has been present. The use of limited resources, the task scheduling or to look for the maximum benefit of the actions performed, have been some of the issues of concern for the different cultures and civilizations. Nowadays, these issues are the main goals for the majority of the companies, especially for the transportation companies. The use of a limited fleet of vehicles and drivers that must perform a known or unknown number of trips to meet the demand of the passengers, trying to minimize the cost of transportation network or to maximize the benefits, are usually the objectives of this type of companies. Thus, Combinatorial Optimization is used to develop efficient algorithms that provide solutions to these situations.

In this way, topics as the Vehicle Routing Problem (VRP) have arisen during the twentieth century as a generalization of the well-known Travelling Salesman Problem (TSP) and, in this particular case, has become one of the most studied problems in the field of Combinatorial Optimization. In the traditional VRP, some vehicles departing from a central depot must visit all the customers in a transportation network, in order to minimize the total solution cost. Determining the optimal solution is NP–hard, so, we could have troubles to find it depending on the size of the instance that we want to solve to optimality. Since the problem emerged, several variants of the VRP have been studied. Some of them, as the Capacitated Vehicle Routing Problem or the Multi-Depot Vehicle Routing Problem, are closely related to the content of this thesis.

Other problems related to the thesis are the Task Scheduling Problems. In these problems, there is a set of tasks, and each of them must be performed by one or more machines. There is a wide variety of applications for the problems of this type. For this reason, the objectives to achieve and the formulations proposed are more varied. Among these problems, the Vehicle and Crew Scheduling Problem (VCSP) is the most related to the subject of this thesis, and is applied to a real life situation that could be addressed by a transport company.

Although the problems mentioned above and their variants have been thoroughly studied, commonly the researches on these fields have focused on finding routes or sequences of tasks only for one operator. In the case of the routing problems these operators are often the vehicles. This thesis focuses on solving two new problems, one of each kind mentioned above, with the singularity that, in both cases, the routes and the sequences of tasks must be performed by two operators, vehicles and drivers, with different characteristics. Both problems are inspired by some particular characteristics of the local air traffic in the Canary Islands. The main regional airline company must operate a number of commercial flights every day between the airports of the islands and others nearby, using aircraft and crews. There are two main airports at the main islands, Tenerife and Gran Canaria, that are considered to be the depots, where the crews and the aircraft are located. Every two days the aircraft must be checked at the airport of Gran Canaria (LPA). For this reason an aircraft starts a route at one depot and must finish at the other one. In the case of the crews, to avoid overnight costs, they must end the day at the same depot where they started. In this thesis, from this situation, two general problems are proposed, considering these particular singularities that motivates this research.

The problems studied in this thesis are called the “Vehicle-and-Driver Scheduling Problem” (VDSP) and the “Driver-and-Vehicle Routing Problem” (DVRP). In addition to the mathematical models and the algorithms proposed to solve the problems, the importance of this thesis is also related to the practical applications. Besides the application to the air transportation, the algorithms proposed could be applied to design transportation networks for all types of means of transport. The benefits obtained using these tools could be very important in comparison with the traditional way of obtaining these solutions.

The research in this PhD thesis has been prepared at the Department of Mathematics, Statistic, and Operational Research of the Universidad de La Laguna, and has been supervised by professor Juan José Salazar González and professor Inmaculada Rodríguez Martín.

This work is organized in five chapters. In Chapter 1 some mathematical background and basic concepts, related to the Operational Research, are given. Chapter 2 describes problems that are related to the research proposed in this thesis as the TSP, VRP, and VCSP. Chapter 3 introduces the VDSP, a mathematical model, and a branch-and-cut algorithm to solve this problem, showing the results obtained in the computational experiments performed. Chapter 4 presents the DVRP, giving computational results after solving a mathematical model using a branch-and-cut algorithm. In Chapter 5, heuristic algorithms to solve the DVRP are proposed, showing the results obtained when we apply them. Finally, the conclusions of this thesis are remarked at the end of the manuscript.

During the course of this research the results obtained have been showed in the following national and international conferences:

- *Workshop on Combinatorial Optimization, Routing and Location (CORAL 2015).* Salamanca (Spain), September 30 – October 2, 2015.
- *XXIII EURO Working Group on Locational Analysis (EWGLA 2016).* Málaga (Spain), September 14–16, 2016.
- *II Encuentro de Jóvenes Investigadores del IUDR.* San Cristóbal de La Laguna (Spain), September 29, 2016.
- *VI Punto de Encuentro de Jóvenes Investigadores de Matemáticas (PEJIM 2016).* San Cristóbal de La Laguna (Spain), December 21–22, 2016.
- *I Encuentro del Grupo de Transporte de la SEIO.* Valencia (Spain), June 19–20, 2017.
- *Annual Workshop of the EURO Working Group on Vehicle Routing and Logistics optimization (VeRoLog 2017).* Amsterdam (Netherlands), July 10–12, 2017.

I have also attended to the following courses:

- *Winter School on Network Optimization (NetOpt 2015).* Estoril (Portugal), January 12 – 16, 2015.
- *EURO PhD School on Routing and Logistics (EPS-RaL 2015).* Brescia (Italy), June 24 – July 3, 2015.

In addition, I have made a research stay of three months with the research group of the Department of Computer Science at the University of São Paulo, Brazil.

Finally, the main results of this thesis have been published in Domínguez-Martín et al. (2017) and Domínguez-Martín et al. (2018). Also, Domínguez-Martín et al. (2018) is under submission in *Lecture Notes in Computer Science.*

# Chapter 1

# Basic concepts, models and approaches

The aim of this chapter is to introduce the basic concepts that constitute the basis of the problems and algorithms proposed in this research. Concepts related to graph theory, computational complexity, mathematical programming and polyhedral theory are essential to understand the theoretical background of this thesis. Moreover, the main method used to solve the problems proposed in this thesis, the so-called *branch-and-cut*, is also described here.

## 1.1 Graph Theory

An *undirected graph* is a pair $G = (V, E)$ formed by a finite and nonempty set of *vertices*, $V = \{1, \ldots, n\}$ and a set of non-ordered pairs of $V$, called *edges*, $E = \{e_1, \ldots, e_m\}$. Each element of $E$ is represented by $e_k = [i, j]$, for all $k \in \{1, \ldots, m\}$, with $i, j \in V$. Given $e_k = [i, j]$, it is said that $e_k$ is *incident* to $i$ and $j$. Note that, in this case, $e = [i, j] = [j, i]$. $G$ is said to be *simple* if it contains at most one edge linking each pair of vertices. A *complete graph* $G = (V, E)$ is a graph that contains edge $[i, j] \in E$ for all vertices $i, j \in V$. The complete graph of $n$ vertices, i.e. $|V| = n$, is denoted by $K_n$.

*Graph Theory* is a branch of Mathematics and Computer Science that studies the properties of the *graphs*, networks of points connected by lines. The origins of the Graph Theory are in the 1700s with the well-known problem of the *Seven Bridges of Königsberg*. The solution of this problem proposed by Leonhard Euler is considered to be the first research of this field. The graph theory arises as a branch of Topology, but, nowadays, it is an important tool in various fields as Physics, Chemistry, Genetics, Operations Research, etc. In this document, only a brief description of the basic notions is showed, but the reader is referred to read the books by Berge (1973), Bondy & Murty (2007), and Christofides (1975) for a in-depth study of this topic.

The set of all edges incident to vertex $i$ is denoted by $\delta(\{i\})$, but, for the sake of simplicity, it is written $\delta(i)$. The *degree* of a vertex $i$ is the cardinality of $\delta(i)$, i.e. $|\delta(i)|$. In general, given the graph $G = (V, E)$ and $S \subset V$, the set

$$\delta(S) := \{[i, j] \in E : i \in S, j \in V \setminus S\}$$

is called the *cut* induced by $S$, and the set

$$E(S) := \{[i, j] \in E : i, j \in S\}$$

denotes the set of all edges with both vertices in $S$. If $V' \subseteq V$ and $E' \subseteq E(V')$, then the pair $G' = (V', E')$ is a *subgraph* of $G = (V, E)$. A *walk* in $G$ is a sequence of vertices $\{v_1, \ldots, v_k\}$ such that $[v_i, v_{i+1}] \in E$ for $i \in \{1, \ldots, k-1\}$. The vertex $v_1$ is called the *origin*, the vertex $v_k$ is called the *destination*, and $v_2, \ldots, v_{k-1}$ are *intermediate vertices* along the walk. The *length* of the walk is the number of edges in it. A walk is called *odd* if its length is odd, and *even* if it has even length. A *path* is a walk with no node repetitions. A walk is closed if its origin and destination coincide, and, in this case, it is called *cycle*. A graph that does not contain cycles is said to be *acyclic* or *forest*. A graph is *connected* if there is a path for every pair of vertices of $V$. These paths induce equivalence relations on the vertices, and their classes are called *components*. A connected graph has only one component. A connected forest is called a *tree*.

Directions could be associated to the edges of the graphs. A *directed graph* or *digraph* is a pair $G = (V, A)$, where $V$ is a finite, nonempty set of elements $V = \{1, \ldots, n\}$ called vertices, and $A = \{(i, j) : i, j \in V, i \neq j\}$ is a set of ordered pairs of vertices called *arcs*. If $(i, j) \in A$, $i$ denotes the *tail* of the arc, and $j$ the *head*. Note that, in a digraph, $(i, j)$ and $(j, i)$ are different elements. When a digraph is represented, arrows are used to indicate the direction. The number of arcs leaving a vertex $i$ is called the *outdegree* of $i$ and the set of these arcs is denoted by $\delta^+(i)$. The number of arcs entering a vertex $i$ is called the *indegree* of $i$, and the set of these arcs is denoted by $\delta^-(i)$. Given the digraph $G = (V, A)$ and $S \subset V$, the set

$$\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$$

is used to refer the set of arcs with tail in $S$ and head in $V \setminus S$. Analogously,

$$\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$$

is used to refer the set of arcs with tail in $V \setminus S$ and head in $S$. And the set of arcs with tail and head in $S$ is denoted by

$$A(S) = \{(i, j) \in A : i \in S, j \in S\}.$$

Most of the definitions for undirected graphs can be reformulated in a straightforward way to directed graphs. The definitions of *diwalks*, *dipaths*, and *dicycles* are analogous to walks, paths and cycles, adding the condition that the arcs are directed in the same orientation. Replacing the arcs by edges, and removing edges duplication, we obtain the *underlying undirected graph* of the directed graph $G$. A digraph is *strongly connected* if there is a dipath between each pair of vertices. A *complete* digraph $G$ is a directed graph that, for all $i, j \in V$, contains the arcs $(i, j)$ and $(j, i)$.

Given a graph (digraph), $G$, with $n$ nodes, a *Hamiltonian tour* is a cycle (dicycle) of length $n$ in $G$. A graph (digraph) that contains a Hamiltonian tour is called *Hamiltonian*.

## 1.2 Computational Complexity

How difficult could be to solve a problem? How much computational resources are required to solve a given task? Answering these questions is the main purpose of the theory of Computational Complexity, classifying the problems according to time or space complexity. For a more detailed information about computational complexity concepts, the reader is referred to the books by Garey & Johnson (1979), Johnson & Papadimitriou (1985), and Papadimitriou & Steiglitz (1982).

A *problem* is an abstract description connected to a question that must be answered. The problems have variables and parameters whose values are not established. The problem is defined when a description of the parameters is made and the properties that the solution must satisfy are specified. An *instance* of the problem is proposed when the values of the parameters are set.

To formally describe the complexity of a problem we need symbols and string of symbols of a finite *alphabet* $\sum$. The elements of $\sum$ are called *letters* or *symbols*, and an ordered finite sequence of letters forms a *word* or a *string*. The set of the strings of symbols of $\sum$ is represented by $\sum^*$. The number of components of a string is called the *size*. With this notation, a problem can be formally described as a subset $\prod$ of $\sum^* \times \sum^*$, so that for a given string $x \in \sum^*$, we must find a string $y$ such that $(x, y) \in \prod$, or to establish that the string $y$ does not exist. The instance or *input* of the problem is the string $x$, and $y$ is called the *solution* or *output*.

An *algorithm* is a set of instructions to solve a problem. When an algorithm is applied to any instance of the problem $\prod$, it always finds a solution. Given a problem $\prod$, i.e., an infinite number of instances $(I_1, I_2, \dots)$, where the data of $I_i$ is a binary string of length $l_i = l(I_i)$, let $A$ be an algorithm that solves any instance of $\prod$ in finite time. The *running time* of $A$ is represented by a function $g_A : \prod \to \mathbb{R}^+$, that associates to each instance its computing time. The *time complexity function* of the algorithm $A$ can be defined as a function $f_A : \mathbb{N} \to \mathbb{N}$ such that $f_A(n) = max\{g_A(I_i) : l(I_i) = n\}$, i.e, the maximum computing time of all the instances of size $n$.

The $Big-O$ notation is used to talk about how long an algorithm takes to run. It is said that $f(n)$ is $O(g(n))$ if there exists a constant $c \geq 0$ and a positive integer $n'$ such that $f(n) \leq cg(n)$ for all $n \geq n'$. With this convention, a polynomial $\sum_{i=0}^{p} c_i n^i$ is $O(n^p)$. We consider the behavior of the function when $n \to \infty$. The algorithm $A$ is said to be a *polynomial time algorithm* for the problem $\prod$ if $f_A(n)$ is $O(n^p)$ for a fixed $p$. Otherwise, it is said to be *exponential*.

In Computational Complexity theory, a *decision problem* is problem whose solution is a 'yes' or "no' answer. Let $\mathcal{P}$ be the class of the decision problem that can be solved in polynomial time, i.e, a problem $\prod$ is in $\mathcal{P}$ if, and only if, there is a polynomial time algorithm that solves $\prod$. Another complexity class is $\mathcal{NP}$. In this case, a decision problem belongs to this class if it is possible to prove in polynomial time that a given instance will produce a yes-answer, for all instances that produce it. It is not required that the problem must be solved in polynomial time. Note that $\mathcal{P} \subseteq \mathcal{NP}$, but the equality is not proved.

There exists a subset of $\mathcal{NP}$ called $\mathcal{NPC}$, such that if there is a $\prod \in \mathcal{NPC} \cap \mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$. Problems in $\mathcal{NPC}$ are called $\mathcal{NP} - complete$. A problem $\prod_1$ is said to be *polynomially reduced* to another problem $\prod_2$ if each instance of $\prod_1$ can be transformed into an instance of $\prod_2$ using a polynomial time algorithm. With this definition we have the following results:

**Proposition 1.1.**
*If* $\prod_1$ *is polynomially reducible to* $\prod_2$ *and* $\prod_2 \in \mathcal{P}$, *then* $\prod_1 \in \mathcal{P}$.

**Proposition 1.2.**
*If* $\prod_1$ *is* $\mathcal{NP} - complete$ *and* $\prod_1$ *is polynomially reducible to* $\prod_2 \in \mathcal{NP}$, *then* $\prod_2 \in \mathcal{NP} - complete$.

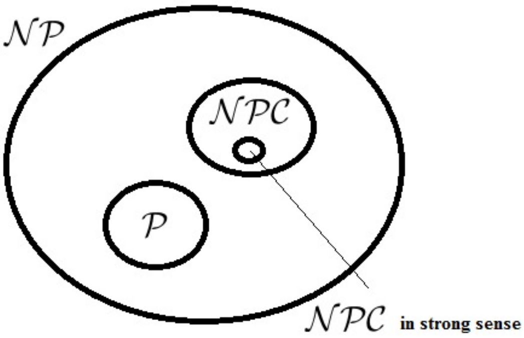



Figure 1.1: Complexity diagram

A problem is called $\mathcal{NP}$-hard if there is an $\mathcal{NP}$-complete problem that can be polynomially reduced to it. Then, an $\mathcal{NP}$-hard problem is at least as difficult as any $\mathcal{NP}$-complete problem. A *pseudo-polynomial* algorithm is an exponential algorithm that becomes polynomial if we impose a bound on the magnitude of the numbers defining the instances. The problems $\mathcal{NP}$-complete, for which the existence of a pseudopolynomial algorithm would imply $\mathcal{P} = \mathcal{NP}$, are called $\mathcal{NP}$-complete *in strong sense*.

Figure 1.1 illustrates the relationship between the different subsets of $\mathcal{NP}$.

For simplicity, the theory of $\mathcal{NP}$-completeness is formulated for decision problems, and not another optimization problems. This happens because all the optimization problems can be reformulated as a decision problem, imposing a bound on the value to be optimized.

## 1.3 Polyhedral Theory

In this section, some basic results and concepts of Polyhedral Theory and Linear Algebra are summarized. These elements are essential aspects for our dissertation, and we show here a brief outline of them. For a more detailed description the reader is encouraged to read Bachem & Gröstschel (1980), Nemhauser & Wosley (1988), Schrijver (1986) and Wolsey (1998).

**Definition 1.3.**
*Given $x_1, \dots, x_k \in \mathbb{R}^n$, a vector $x \in \mathbb{R}^n$ is called a* linear combination *of the*

*vectors* $x_1, \dots, x_k$*, if* $x = \sum_{i=1}^{k} \lambda_i x_i$*,* $\lambda_1, \dots, \lambda_k \in \mathbb{R}$*. When* $\sum_{i=1}^{k} \lambda_i = 1$*, then it is called* affine combination*. When also* $\lambda_i \geq 0$ *for all* $i \in \{1, \dots, k\}$*, then it is called a* convex combination*.*

**Definition 1.4.**
*Given a set* $S \subseteq \mathbb{R}^n$*,*

- *the set of all points that are linear combination of points in* $S$ *is called* linear hull *of* $S$ *and it is denoted by* $lin(S)$*.*
- *the set of all points that are convex combination of points in* $S$ *is called* convex hull *of* $S$ *and it is denoted by* $conv(S)$*.*
- *the set of all points that are affine combination of points in* $S$ *is called* affine hull *of* $S$ *and it is denoted by* $aff(S)$*.*

**Definition 1.5.**
*A set* $S \subseteq \mathbb{R}^n$*, with*

- $S = lin(S)$ *is called* linear subspace*.*
- $S = conv(S)$ *is called* convex set*.*
- $S = aff(S)$ *is called* affine subspace*.*

**Definition 1.6.**
*A set of points* $S = \{x_1, \dots, x_k\} \subseteq \mathbb{R}^n$ *is* linearly (affinely) independent *if the unique solution of* $\sum_{i=1}^{k} \lambda_i x_i = 0$ *(*$\sum_{i=1}^{k} \lambda_i x_i = 0$ *and* $\sum_{i=1}^{k} \lambda_i = 0$*) is* $\lambda_i = 0$ *for* $i \in \{1, \dots, k\}$*.*

Let $S$ be a set with at least two elements. Linear (affine) independence means that there is no $x \in S$ that can be represented as a linear (affine) combination of the vectors in $S \setminus \{x\}$. By convention, the empty set is linearly and affinely independent. Moreover, all sets with one element are affinely and linearly independent, and $\{0\}$ is linearly dependent but affinely independent. Linear independence implies affine independence, but the converse is not true.

**Definition 1.7.**
*A set* $H \subseteq \mathbb{R}^n$*,*

1. *is called a* hyperplane *if and only if there are* $a \in \mathbb{R}^n \setminus \{0\}$ *and* $a_0 \in \mathbb{R}$ *such that* $H = \{x \in \mathbb{R}^n : a^T x = a_0\}$*,*

2. *is called a* halfspace *if and only if there are* $a \in \mathbb{R}^n \setminus \{0\}$ *and* $a_0 \in \mathbb{R}$ *such that* $H = \{x \in \mathbb{R}^n : a^T x \leq a_0\}$.

If $a \neq 0$, it is said that the hyperplane $H = \{x \in \mathbb{R}^n : a^T x = a_0\}$ is the hyperplane defined by $a^T x = a_0$. It is also said that the halfspace $H = \{x \in \mathbb{R}^n : a^T x \leq a_0\}$ is the halfspace defined by $a^T x \leq a_0$.

**Definition 1.8.**
*A* polyhedron $P \subseteq \mathbb{R}^n$ *is the intersection of a finite number of halfspaces or, analogously, the set of points that satisfy a finite number of linear inequalities. That is,* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ *where* $(A, b)$ *is an* $m \times (n+1)$ *matrix.*

**Definition 1.9.**
*A polyhedron* $P \subseteq \mathbb{R}^n$ *is* bounded *if there exists* $\alpha \in \mathbb{R}^+$ *such that* $P \subseteq \{x \in \mathbb{R}^n : -\alpha \leq x_j \leq \alpha$ *for* $j = 1, \ldots, n\}$. *A bounded polyhedron is called* polytope.

The polytopes are the convex hull of a finite set of points, i.e., if $P$ is a polytope, it can be written as $P = conv(S)$, being $S \subseteq \mathbb{R}^n$ a finite set.

**Definition 1.10.**
*Given a set* $S \subseteq \mathbb{R}^n$, *the* rank *of* $S$, *denoted by* $rank(S)$, *is the cardinality of the largest linearly independent subset of* $S$. *The* dimension *of* $S$, *denoted by* $dim(S)$, *is the rank of* $S$ *minus one.*

The dimension of a polyhedron $P \subseteq \mathbb{R}^n$ is $k$, denoted by $dim(P) = k$, if the maximum number of affinely independent points in $P$ is $k+1$. If $dim(P) = n$, the polyhedron is *full-dimensional*, this is the equivalent to say that there is no hyperplane containing $S$.

**Definition 1.11.**
*The inequality* $a^T x \leq a_0$ *is called a* valid inequality *for a polyhedron* $P \subseteq \mathbb{R}^n$ *is it is satisfied by all points in* $P$, *i.e.,* $P \subseteq \{x \in \mathbb{R}^n : a^T x \leq a_0\}$

**Definition 1.12.**
*Given a valid inequality* $a^T x \leq a_0$ *for* $P$, *then the set* $F = P \cap \{x : a^T x = a_0\}$ *defines a* face *of* $P$. $P$ *itself and the empty set represent the* improper *faces of* $P$, *whereas, if* $\emptyset \neq F \neq P$, $F$ *is said to be a* proper *face of* $P$, *and also the face* induced *by* $a^T x \leq a_0$.

If $\{x \in P : a^T x \leq a_0\} = \{x \in P : b^T x \leq b_0\}$ the inequalities $a^T x \leq a_0$ and $b^T x \leq b_0$ induce the same face, and it is said that are *equivalent* with respect to $P$. Moreover, given two faces, $F$ and $F'$, of $P$ induced by $a^T x \leq a_0$ and $a'^T x \leq a'_0$, respectively, if $F \subset F' \subset P$ then it is said that $a'^T x \leq a'_0$ *dominates* $a^T x \leq a_0$.

The minimal proper face of a polyhedron $P$ is a face of dimension zero formed by one element called *vertex*, that cannot be represented by convex combinations of other points of the polyhedron. Other important faces of a polyhedron are the maximal proper faces, called *facets*. If $F = P \cap \{x \in \mathbb{R}^n : a^T x = a_0\}$ is a face, $a^T x \leq a_0$ is called a *facet defining inequality* for $P$.

A polyhedron is usually represented as an inequality system. The importance of the facet defining inequalities lies on the fact that when each inequality that defines a polyhedron is a facet, then each inequality is as strong as possible, and there are no better inequalities. In a *minimal linear system*, that determines a polyhedron $P$, an inequality cannot be removed without changing the polyhedron and an inequality cannot be replaced by an equality without reducing the polyhedron. The link between this type of systems and the facet defining inequalities is stated in the following result.

**Theorem 1.13.**
*Let $P = \{x \in \mathbb{R}^n : Ax \leq b, Cx = d\}$. Then $Ax \leq b$, $Cx = d$ is a minimal lineal system for $P$ if and only if*

1. *no equality in $Cx = d$ is a linear combination of the other equations of that system, and*
2. *each inequality in $Ax \leq b$ defines a facet of $P$ and conversely.*

In practice, to prove if a given inequality $a^T x \leq a_0$ defines a facet for a polyhedron $P$, the following theorem is used.

**Theorem 1.14.**
*Let $F$ be a proper face of $P = \{x \in \mathbb{R}^n : Ax \leq b, Cx = d\}$ and let $h$ be the number of equations in this system. Then the following statements are equivalent:*

1. *$F$ is a facet of $P$.*
2. *$dim(F) = dim(P) - 1$.*
3. *If $F = \{x \in P : a^T x = a_0\} = \{x \in P : \bar{a}^T x = \bar{a}_0\}$, then there exist $\alpha \geq \mathbb{R}^+$ and $\beta \in \mathbb{R}^h$ that $\bar{a} = \alpha a + \beta C$ and $\bar{a}_0 = \alpha a_0 + \beta d$.*

## 1.4 Mathematical Programming

The *Mathematical Programming* is a branch of the Applied Mathematics that addresses problems where the objective is to optimize (minimize or maximize) a function under some constraints. This type of problems can concern disciplines as varied as Economy, Biology, Chemistry, Engineering, etc. The origins of Mathematical Programming go back to 1947, when George B. Dantzig developed the *simplex algorithm* to solve linear programming problems.

A problem of Mathematical Programming can be formulated as follows:

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

i.e, to minimize a scalar function $f : X \to \mathbb{R}$, where

$$X = \{x : h_i(x) \geq 0, i = 1 \dots k; l_j(x) = 0, j = 1, \dots, n\} \subseteq \mathbb{R}^n,$$

being $h_i(x)$ and $l_j(x)$ also scalar functions. The *objective function* is represented by $f(x)$, and $X$ is considered to be the *feasible set*. A *feasible solution* $x^*$ of the problem is a point that verifies all the constraints in $X$. If a reformulation of the problem is made, it is possible to deal with maximization problems, because

$$max\{f(x) : x \in X\} = -min\{-f(x) : x \in X\}.$$

The search of a feasible solution $x^*$ that makes $f(x^*)$ as small as possible, can provide the following results. A problem is *not feasible*, when there is no feasible solution, i.e., $X = \emptyset$. In this case, it is written $min\{f(x) : x \in X\} = +\infty$. A problem is *not bounded* when there exist solutions that may decrease the objective function infinitely, i.e., for each real value $M$, a point $x \in X$ such that $f(x) < M$ can be found. If a problem is not bounded, it is said that $min\{f(x) : x \in X\} = -\infty$. A problem has *optimum* when there is a solution $x^* \in X$ such that $f(x^*) \leq f(x)$ for all $x \in X$. Then, $f(x^*) = min\{f(x) : x \in X\}$, and $x^*$ is called the *optimal solution* of the problem, but not necessarily the only optimal solution.

The field of Mathematical Programming can be split in different groups depending on the type of the variables, constraints, and objective function we consider. In *Linear programming*, the objective function and the constraints are linear. If the variables are integer, this type of problem is called *Integer Linear Programming* problem. The assumption that the variable must be integer makes the problems harder to solve. The *Quadratic programming* consider problems in which the objective function is quadratic and convex, but the restrictions are linear. In the case of the *Convex programming*, the constraints and the objective function are convex. In *Stochastic programming*, some parameters are undetermined. In this thesis, problems of Integer Linear Programming have been proposed as we will show in Chapters 3 and 4, introducing the Vehicle and Driver Scheduling Problem and the Driver and Vehicle Routing Problem, respectively.

## 1.5 Branch-and-Cut algorithm

In the field of Combinatorial Optimization, several exact and heuristic methods have been used to solve Integer Linear Programming (ILP) problems. The branch-and-cut method is one of the main exact algorithms used to solve this kind of problems. In this section a general description of the branch-and-cut method is presented. For a more thorough study of this type of algorithms see Jünger et al. (1995), Thienel (1995), Caprara & Fischetti (1997), and Padberg & Rinaldi (1991). In this last book, the term branch-and-cut was used for the first time.

This method is used to solve integer linear problems, using a branch-and-bound algorithm combined with a cutting plane algorithm to strengthen the Linear Programming relaxations of the mathematical model. The branch-and-bound method was proposed in Land & Doig (1960) to solve ILP problems. The time required to solve problems using this method increases exponentially when the number of variables is also increased. In the 50s and 60s, R. Gomory started to use cutting planes to solve the same type of problems, developing a new method. In this case, the method required less time to solve the problems. The combination of both methods showed greater efficiency and became very popular, giving rise to the branch-and-cut method, that appeared in Gröstschel et al. (1984) for the first time.

The first step of a branch-and-cut algorithm is the *Linear Programming (LP) relaxation* of an integer linear program (IP), that consists of dropping the restriction that all variables must be integer. If we want to minimize the objective function, the optimal value of the LP relaxation, denoted by $z_{LP}$, is a lower bound of the optimal value $z_{IP}$ of the integer linear program, i.e., $z_{LP} \leq z_{IP}$.

When the LP relaxation is solved, if the optimal solution $x^*$ is integral, we are done. Otherwise, two linear problems are proposed, choosing a variable $x_i^*$ that took fractional value. In the first program the constraint $x_i^* \geq \lceil x_i^* \rceil$ is added, and $x_i^*$, i.e, $x_i^* \leq \lfloor x_i^* \rfloor$ is added in the second one. Then, a branch-and-bound algorithm is applied. The algorithm set $z_{IP} = \infty$ and associates each linear program a node of a search tree. At each step a node is processed, and it is pruned if it is infeasible, if its optimal solution exceeds $z_{IP}$, or if its optimal solution is integer and smaller than $z_{IP}$. In the last case, $z_{IP}$ is updated and the found solution becomes the best known solution. If we are unable to prune the node, then we choose a non-integral solution to branch on, as we did in the root node, creating two new nodes that are added to the search tree. The algorithm finishes when the search tree is empty. When the number of variables or constraints of the IP is large, this process can be very slow, and a *cutting plane* algorithm must be used to tighten the LP relaxations.

The cutting plane algorithm proceeds as follows. We denote as $LP(\infty)$, the LP relaxation of an ILP problem. Given $h \geq 0$, the current iteration, let $LP(h)$ be a linear program formed by a subset of the constraints in $LP(\infty)$. Solving $LP(h)$, an optimal solution $x_h^*$ is obtained. If this solution is integer then it is optimal. Otherwise, an algorithm is applied, called *separation algorithm*. If there exists one constraint of $LP(\infty)$ violated by $x_h^*$, the separation algorithm will provide at least that constraint, and if none violated constraint is found, it means that all constraints are satisfied. If there exist some violated constraints, that constraints are added to $LP(h)$ to obtain $LP(h+1)$. When all the violated constraints are found, this process stops and an optimal solution is reached. Remark that $z_{LP(h)} \leq z_{LP(h+1)} \leq z_{LP(\infty)} \leq z_{IP}$, with $h \geq 0$, being $z_{LP(h)}$ the optimal solution of $LP(h)$.

The branch-and-cut method has several advantages and disadvantages. One of the advantages is that savings in terms of time and memory can be obtained using the constraints in the LP from previous lower bound generations. Furthermore, the fact that when the branch-and-cut method ends, the optimality of the solution is ensured, is an advantage over the use of heuristic methods. The main disadvantage of this method happens when large problems are considered, because the number of variables in the LP relaxations may become very large, and consequently, the computer spends too much time to obtain the solution of the LP.

# Chapter 2

# Related Problems

This chapter shows some ILP problems related to this thesis. The Travelling Salesman Problem, described in Section 2.1, and the Vehicle Routing Problem, described in Section 2.2 are related to the problems studied in this thesis. In addition, all the mathematical background under these classical problems have been adapted and generalized to propose the mathematical formulations of the problems in Chapters 3 and 4. The theme of the Vehicle Routing & Crew Scheduling Problem, described in Section 2.3, has inspired this research by the fact that the problems studied consider the design of schedulings for vehicles, but also for drivers.

## 2.1 The Travelling Salesman Problem

The first notions of the Travelling Saleman Problem (TSP) appeared in Voigt (1831), a book dedicated to other issues, but setting out the essence of the TSP without naming it in that way. The term 'travelling salesman problem' began to be used in 1931–1932, and at that time became very popular for various reason, as the difficulty of solving it. It is in Dantzig et. al. (1954), where the most popular formulation of the TSP was published. This fact is considered to be one of the most important events in the field of the combinatorial optimization. They were able to solve problems with 49 cities, and, although nowadays much larger instances have been solved optimally, the importance of this research is related to the future researches it inspired. Many exact and heuristic algorithms have been developed to solve the TSP and its several variants.

The TSP has numerous applications, some of them unexpected. Problems in vehicle routing, computer wiring, cutting wallpaper, job sequencing, clustering data array, etc. can be solved formulating them as TSP instances. But the main importance of the TSP is related to the different problems that have appeared as generalizations of the TSP, for example, the well-known *Vehicle Routing Problem*, that will be presented below. The problems studied in this thesis can be also seen as restricted variants of the TSP, and, for this reason, some of the valid inequalities that will be shown below can be adapted to formulate valid inequalities for the problems in this thesis.

Given a set of $n$ cities, the objective of the TSP is to find the minimum-cost-route that starts and ends at the same city, and visits every city exactly once. Hence, in Graph Theory, the aim of the TSP is to find a *Hamiltonian circuit* (or *tour*) of the graph. The cost of going from one city to another one is assumed to be known. The new problems proposed in this thesis have been modeled using a directed graph, that also can be used to model the *asymmetric TSP* (ATSP). In the ATSP the cost of going from a city to another one is different from the cost of traversing the same way in the opposite direction. Without loss of generality, we can assume that the graph is complete because, otherwise, we can add the missing arcs of the graph with arcs of large cost. If the considered graph $G$ is undirected, the problem is called Symmetric Travelling Salesman Problem (STSP).

There are several mathematical formulations proposed to solve the TSP, but, as stated above, the most popular formulation for this problem was proposed in Dantzig et. al. (1954). Some notation must be established to describe the mathematical model. Let $G = (V, A)$ be a complete directed graph with vertex set $V = \{0, 1, \ldots, n\}$ and arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. There is a known cost $c_{ij}$ associated with the arc $(i, j) \in A$. The ATSP consists of finding a Hamiltonian circuit $G^* = (V, A^*)$ of $G$ with minimum cost. The model uses a variable $x_{ij}$ that takes value 1 if arc $(i, j)$ is in the optimal tour, and 0, otherwise. In these conditions, the ATSP can be modeled as follows:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{2.3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subset V : S \neq \emptyset \tag{2.4}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V. \tag{2.5}$$

Constraints (2.2) and (2.3) indicate that the in-degree and out-degree of each vertex, respectively, is equal to one. Inequalities (2.4) are the well known *subtour elimination constraints* (SEC) that, as their name implies, avoid the occurrence of subtours, i.e., tours with length $n' < n$. This formulation has $n^2 - n$ variables and $O(2^n)$ constraints.

The number of SECs (2.4) can be halved by eliminating the sets $S$ including 0. Using the degree constraints (2.2) and (2.3). inequalities (2.4) can be reformulated as *connectivity constraints*:

$$\sum_{i\in S}\sum_{j\in V\setminus S} x_{ij} \geq 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset. \tag{2.6}$$

Solving the Linear Programming (LP) relaxation of the model (2.1)–(2.5), a valid lower bound on the optimal solution value of the ATSP can be obtained. This relaxation is obtained by replacing constraints (2.5) by

$$x_{ij} \geq 0 \qquad i,j \in \{1,\ldots,n\}.$$

Valid inequalities could be used to strengthen the LP relaxation of the model. Over the years, several families of inequalities have been proposed to improve the mathematical models of the TSP. The classical valid inequalities for the ATSP are the following ones:

### 2.1.1 $D_k^+$ and $D_k^-$ inequalities

Let $\{i_1, i_2, \ldots, \imath_k\} \subset V$, with $3 \leq k \leq n-1$. Then the $D_k^-$ inequalities have the form

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=3}^{k} x_{i_1 i_j} + \sum_{j=4}^{k}\sum_{h=3}^{j-1} x_{i_j i_h} \leq k-1,$$

and the $D_k^+$ inequalities

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=2}^{k-1} x_{i_j i_1} + \sum_{j=3}^{k-1}\sum_{h=2}^{j-1} x_{i_j i_h} \leq k-1.$$

These inequalities were proposed in Gröstschel & Padberg (1985) and are facets defining for $P$.

### 2.1.2 2-matching constraints

Given $H, T_1, \ldots, T_k \subset V$, $k \geq 3$, $k$ odd satisfying

$$|H \cap T_i| = 1, \ i \in \{1, \ldots k\},$$

$$|T_i \setminus H| = 1, \ i \in \{1, \ldots k\},$$

$$T_i \cap T_j = \emptyset, \ 1 \leq i < j \leq k,$$

the 2-matching constraint is given by

$$x(A(H)) + \sum_{i=1}^{k} x(A(T_i)) \leq |H| + \frac{k-1}{2}.$$

The 2-matching constraints were presented in Gröstschel & Padberg (1985).

### 2.1.3 SD inequalities

Given a *handle* $H \subset V$, and *teeth* $T_1, \ldots T_t$, t odd, such that $|T_i \cap H| = 1$, $|T_i \setminus H| = 1$, and (possibly empty) disjoint node sets $S$ and $D$, where $(S \cup D) \subset V \setminus (H \cup T_1 \cup \cdots \cup T_t)$ such that $|S| + |D| = t$. Then the SD-inequalities can be written as

$$x(S \cup H : D \cup H) + \sum_{i=1}^{t} x(A(T_i)) \leq |H| + \frac{|S| + |D| + t - 1}{2}. \qquad (2.7)$$

This family of inequalities was introduced in Balas & Fischetti (1993). The SD-inequalities can be seen as a generalization of the 2-matching inequalities.

### 2.1.4 Comb inequalities

A comb is defined by $H \subseteq V$ the handle and $T_1, \ldots, T_t \subseteq V$ the teeth, with $t \geq 3$ odd, that satisfy the following properties:

- $T_j \setminus H \neq \emptyset, \forall 1 \leq j \leq t,$
- $T_j \cap H \neq \emptyset, \forall 1 \leq j \leq t,$
- $T_i \cap T_j = \emptyset, \forall 1 \leq i < j \leq t.$

The comb inequality is formulated as

$$x(\delta(H)) + \sum_{i=1}^{t} x(\delta(T_i)) \geq 3t + 1.$$

The *comb inequalities* were proposed in Gröstschel & Padberg (1979), and are a generalization of (2.7) inequalities. Note that these inequalities are formulated for the STSP, but since STSP is a special case of the ATSP, any given inequality for the STSP polytope has an equivalent inequality for the ATSP. The ATSP can be reduced to a STSP on an undirected graph with twice as many nodes. For a more detailed explanation of this process the reader is encouraged to read Balas & Fischetti (2007).

## 2.2 The Vehicle Routing Problem

The problems studied in this thesis are closely related to the *Capacitated Vehicle Routing Problem* (CVRP), the simplest variant of the Vehicle Routing Problem (VRP), one of the most studied problems in the field of logistics. The VRP was proposed by Dantzig & Ramser (1959) to solve a problem of petrol deliveries, given the first algorithmic solution approach. After that, the models and algorithms proposed to solve this type of problems have been used for the solution of many real-world applications such as street cleaning, school bus routing, transportation of disabled people, solid waste collection, etc. As other combinatorial optimization problems, it represents a generalization of the TSP.

In general, in the VRP, the objective is to meet the demand of some customers, delivering goods that are located at a depot, using a fleet of vehicles and minimizing the total solution cost. The CVRP has the particular feature that the vehicles used have a limited capacity to carry goods to the different customers. All the vehicles are identical and start and end their routes at the single depot. In the literature, there are many surveys that show different approaches to present the CVRP; see Laporte (1992, 1997), and Toth & Vigo (2002). Several formulations have been proposed for this problem (see, e.g., Yaman (2006)), but the CVRP formulation used as starting point of this research is called *two index vehicle flow formulation*, and can be described in its asymmetric way, as follows.

Let $G = (V, A)$ be a directed graph where $V = \{0, \dots n\}$ represents the vertex set and $A$ the arc set. Vertex 0 stands for the depot and vertices $i = 1, \dots, n$ correspond to the customers. A nonnegative demand $d_i$ is assigned to each customer

$i$. Given a set of customers $S \subseteq V$, let $d(S) = \sum_{i \in S} d_i$ represent the total demand of $S$. The cost of travelling from a vertex $i$ to another vertex $j$ is represented by $c_{ij}$, for all $(i,j) \in A$. The capacity of each vehicle is denoted by $Q$. A fleet of $K$ vehicles is located at the depot. $\gamma(S)$ represents the minimum number of vehicles needed to serve the set $S$. The objective of the CVRP is to find an optimal set of $K$ routes to serve all the customers. The routes start and finish at the depot and each customer has to be served by a single vehicle. Then, the CVRP can be modeled as:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.8}$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{2.9}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \setminus \{0\} \tag{2.10}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.11}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.12}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \gamma(S) \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{2.13}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V. \tag{2.14}$$

Constraints (2.9) and (2.10) represent the *degree constraints* of each node, and ensure that each customer is visited by only one vehicle. In the case of the depot, the degree constraints, (2.11) and (2.12) impose that the number of vehicle that start and end the route at the depot is the same, $K$. The inequalities (2.13) are the *capacity inequalities* that avoid subtours, and are also used to avoid to exceed the capacity of the vehicle. In Section 2.2.1 the different versions of the capacity inequalities are explained. Finally, (2.14) establish that the variables $x$ take integer values, 0 or 1, i.e., $x_{ij}$ takes the value 1 if a vehicle traverses the arc $(i,j) \in A$ and value 0 otherwise.

For the problems studied in this thesis, two depots and a directed graph $G = (V, A)$ have been considered. For this reason, our problem is strongly related with the *Multi-Depot Asymmetric Capacitated Vehicle Routing Problems* (MDACVRP). In spite of the large number of papers that discuss about the CVRP, there are no many of them that address the MDACVRP and exact methods to solve it. In Laporte et al. (1984, 1988), there are examples of exact branch-and-bound methods that allow to solve relatively small instances.

The mathematical model proposed in (2.8)–(2.14) allows to solve instances with a small number of nodes. To solve larger instances it is very important to strengthen the linear programming relaxation of the model, adding new valid inequalities, to improve the effectiveness of the branch-and-cut algorithm, as we have explained in Section 1.5. Some of the classical families of valid inequalities used to solve the CVRP are the following:

### 2.2.1 Capacity inequalities

In the CVRP, the *capacity inequalities* (2.13) avoid to exceed the vehicle capacity, but these constraints are also used as subtour elimination constraints. There are different versions of capacity inequalities; all of them share the same left-hand side, but the right-hand side changes. A larger right-hand side produce a tighter inequality. The so-called *fractional capacity constraints* have the following expression:

$$x(\delta^+(S)) \geq \frac{d(S)}{Q} \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset. \tag{2.15}$$

The exact separation procedure of this family of inequalities is solvable in polynomial time. These inequalities can be strengthened if we replace the right-hand side by the nearest larger integer of $\frac{d(S)}{Q}$, i.e.,

$$x(\delta^+(S)) \geq \left\lceil \frac{d(S)}{Q} \right\rceil \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset. \tag{2.16}$$

These constraints are called *framed capacity constraints* and, in this case, it is not possible to obtain an exact separation procedure in polynomial time. A solution for this issue, often used, is to solve the exact separation procedure for (2.15), and to check if the inequalities (2.16) are violated.

### 2.2.2 Multistar inequalities

Although the Multistar inequalities were introduced in Araque et. al. (1994), it is in Gouveia (1995) where the so-called *generalized large multistar inequalities* were proposed for the CVRP, considering a directed graph. The formulation of these inequalities is the following:

$$x(\delta^+(S)) \geq \frac{1}{Q}\sum_{i\in S}(d_i + \sum_{j\notin S} d_j(x_{ij} + x_{ji})) \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset.$$

It is possible to propose an exact separation procedure for these inequalities, solvable in polynomial time.

## 2.3 The Vehicle and Crew Scheduling Problem

The development of the transport has contributed to the study of the *Vehicle and Crew Scheduling problems* (VCSP). It is very difficult to provide a definition of VCSP, because it depends of the real-life situation that must be addressed. Transport companies of airplanes, bus, taxis, etc. face each day this kind of problems. Commonly, a fleet of vehicles and crews must perform a set of *tasks* or *trips*, with the aim of finding a scheduling with minimum total cost.

In the literature, we found two approaches to obtain the desired scheduling. The *sequential approach* has been traditionally used by the transport companies to solve these situations, finding first a scheduling for the vehicles, and then, independently, for the crews. However, in the last years, some authors have proposed an *integrated approach* in which both schedulings, for vehicles and crews, can be obtained at the same time, and they have showed the benefits of using this strategy to solve the problem. A more exhaustive explanation of this approach to solve the VCSP can be found in Freling et al. (2003), Freling et al. (2003) or Haase et al. (2001). Furthermore, the multi-depot case has been also studied in Huisman et al. (2005).

Mesquita & Paias (2008) propose an integer linear programming formulation for the multi-depot case of the VCSP, and use a four-step solution approach to solve it. Kim et al. (2010) study a problem in which manpower teams must be transported by vehicles to perform multi-stage tasks at customer locations. There are precedences among the tasks and vehicles are not driven by teams, but only used as a way of transport between customer locations. They give a MIP formulation for the problem, although it has no practical use to solve it. To this end, they propose a particle swarm optimization based heuristics. In another context, the aircraft routing and crew scheduling is solved using Benders decomposition by Cordeau et. al (2001), and Papadakos (2009), and through a MIP-based heuristic method by Salazar-González (2014).

As we have mentioned above, we cannot make a general definition for the VCSP, conversely of the cases of the TSP and the VRP. Then, we cannot give a general formulation of the VCSP and, for this reason, the mathematical formulation of the problems studied in this thesis is not based in any formulation of the VCSP. The link between the VCSP and this research is related to the objective proposed. In both cases, a synchronization between vehicles and crews (or drivers) is needed to ensure a feasible solution. Also, terms as *changeover*, to refer to a change of vehicle performed by a driver, or *exchange locations* to call the places where a changeover can occur, have been imported from the VCSP literature.

# Chapter 3

# The Vehicle and Driver Scheduling Problem

The first problem addressed in this thesis, called The Vehicle and Driver Scheduling Problem (VDSP), can be stated in a general way as a task scheduling problem where each task must be assigned to one machine and one operator. Tasks assigned to the same machine or operator cannot be performed in parallel, and the execution of a task cannot be interrupted. Thus, the tasks assigned to a machine or operator must be ordered. There are two configurations on which machines and operators must be before and after processing their tasks. A machine changes from an initial to a final configuration during the processing of its tasks, while an operator must start and finish in the same configuration. Moreover, machines and operator can only process a limited number of tasks. There is a cost for changing from a task to the next one in a schedule, a cost for each operator used, and a cost associated to each pair of consecutive tasks performed by the same machine but by different operators (changeovers). The objective of the problem is to assign the tasks to machines and operators, and to sequence the tasks assigned to each one, in order to minimize the total solution cost.

The problem was inspired by some particular planning characteristics of an air-transportation company operating in the Canary Islands. A number of commercial flights (tasks) must be operated every day, using several aircraft (machines) and crews (operators). There are two hub airports, TFN and LPA, where the crews' homes are (the two configurations). To avoid overnight costs to the company, the flights assigned to a crew must allow that crew to start and end the day at the airport where its home is. The aircraft need to be checked every two days, and this operation is called *short-time maintenance*. The short-time maintenances take place during the night (between 23:00 and 07:00) and can only be performed at LPA airport, where the required equipments are located. The short-time maintenances have a major planning impact as they force that, in a particular night, half of the aircraft fleet stay at LPA while the other half stay at TFN. Thus, the flights assigned to each aircraft must be such that it starts the day in one hub airport and ends in the other one. This movement guarantees the short-time maintenance on each aircraft. In addition there are other constraints like the maximum number of flights that can be assigned to a crew.

Salazar-González (2014) studied the complex real-world problem posed by the air carrier in the same regional context. That problem also involves, among other aspects, the so-called *aircraft routing* and *crew routing* problems, where a *route* is a set of flights that can be sequentially operated. However, in the real-world problem the departure and arrival times for each flight are given in advance. This fact makes the graph of connecting flights acyclic, besides being sparse, and thus dynamic programming provides efficient approaches to find min-cost routes (see for example Freling et al. (2003)). Under this condition, Salazar-González (2014) described a heuristic approach and Cacchiani & Salazar-González (2017) gave an exact method to solve the airline problem. In this paper we study a more general problem, without assuming precedences among the tasks, or, using the air transportation example, with unfixed departure times for the flights.

The problem without given precedence constraints on the tasks can be modelled as a vehicle routing problem (described in Section 2.2). A route represents a sequence of tasks. The tasks can be considered as customers that need to be served by two types of vehicles. One type of vehicles (type 1) represents the operators. The other type of vehicles (type 2) represents the machines. The two machine configurations are called depots. There are costs associated to consecutive customers in a route. Such cost could represent, in the air transportation example, the waiting time in an airport for the crews, and the cleaning operations for aircraft between two consecutive flights. There are limits on the maximum number of tasks in a route (e.g. a crew cannot operate more than 6 flights each day). There is a cost associated to each operator (crew) used, and to each changeover (two consecutive flights done by the same plane but by different crews). The problem consists of designing min-cost routes so that each customer is served by one vehicle of each type. Vehicles of type 1 must start and end their routes at the same depot, while vehicles of type 2 must leave from one depot and arrive to the other.

The Section 3.1 formally describes the VDSP and details the mathematical formulation. It also presents several families of valid inequalities to strengthen the linear programming relaxation. Section 3.2 proposes a branch-and-cut algorithm to solve the problem and explains the separation procedures for each family of inequalities. Section 3.3 presents computational results obtained when the algorithm is implemented and used to solve different instances. We compare the results using the valid inequalities at different stages of the algorithm.

## 3.1 Formulation and valid inequalities

In this section we formally describe the VDSP, give a mathematical model and present the valid inequalities used to strengthen its linear programming relaxation. We start by setting up the notation.

We are given $n$ customers, two depots, and two types of vehicles. The set of customers is represented by $V_c = \{1, \ldots, n\}$ and the set of depots by $V_d = \{0, n+1\}$. Let $G = (V, A)$ be a complete directed graph with vertex set $V = V_c \cup V_d$ and arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. The cost to pay when a vehicle of type $k$, for $k = 1$ and $k = 2$, traverses an arc $(i, j)$ is denoted by $c^k_{ij}$, and it is assumed to be known. All vehicles of type $k$ have a capacity equal to $Q^k$, which represents the maximum number of customers that can be served by the vehicle. We assume that the number of vehicles of type $k$ available at depot $d$ is $K^k_d$.

The aim of the problem is to design feasible routes in $G$ in order to visit each customer with one vehicle of each type. A route for a vehicle of type 1 must end at the same depot where it starts (i.e., it must be a cycle). A route for a vehicle of type 2 must start and end at different depots (i.e., it must be an open path).

Ideally, if customers $i$ and $j$ are served by the same vehicle of type 1 then they should also be served by the same vehicle of type 2. In the air-transportation example, this would mean that each crew only flies one aircraft. However, this ideal situation is not always possible due to the capacity limits and the different types of routes, and in some cases a crew must change from one aircraft to another one. This case is called *changeover*, and is undesired by the transportation companies, not only because it forces extra work for the crew, but also because a delay in the first flight may affect other two flights. For that reason, a changeover is strongly penalized in the cost function with a big value $M$ in our problem definition. There is also another cost $N$ to be paid for using each vehicle of type 1 in a solution. This value $N$ is usually lower than $M$.

To illustrate the problem, Figure 3.1 shows the optimal solution of an instance with 2 depots (nodes 0 and 19) and 18 customers (nodes 1 to 18). There are two vehicles of type 2 that go from one depot to the other (solid lines). There are two

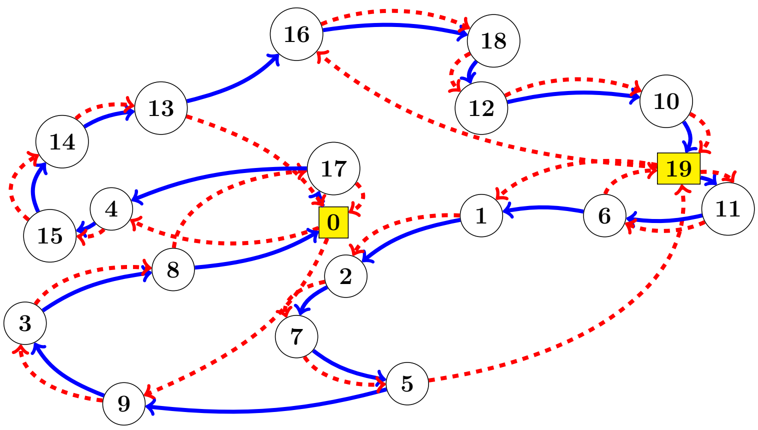



Figure 3.1: Solution example

vehicles of type 1 that make circular routes (dashed lines) from depot 0, and three vehicles of the same type that start and end their routes at depot 19. Vehicles of type 1 can serve at most 4 customers, and vehicles of type 2 can serve at most 9. Note that a line in this figure does not represent a space movement, like in the CVRP, but a precedence; for example, in the air-transportation context, a line from $i$ to $j$ means that flight $j$ will be operated after flight $i$ (being the arrival airport of $i$ equal to the departure airport of $j$). Changeovers occur when, between two consecutive non-depot nodes in a route, there is a dashed line and not a solid line. For example, in Figure 3.1, we can observe that there is a changeover between the nodes 8 and 17.

### 3.1.1 Mathematical model

We now model the VDSP, starting by defining the decision variables. Variable $x_{ij}^k$ takes value 1 if a vehicle of type $k$ traverses an arc $(i, j) \in A$, and value 0 otherwise. For brevity of notation, we will write $x^k(A')$ instead of $\sum_{(i,j)\in A'} x_{ij}^k$ for each vehicle of type $k$ and each subset of arcs $A' \subseteq A$. Variable $y_{ij}$ is used to indicate a changeover between nodes $i$ and $j$, i.e., $y_{ij}$ is equal to 1 when $x_{ij}^1 = 1$ and $x_{ij}^2 = 0$, and 0 otherwise. Variable $w_i \in \mathbb{R}$ represents the position in which customer $i$ is served, and variable $z_i^k \in \mathbb{R}$ determines the number of customers that a vehicle of type $k$ has served immediately after serving customer $i$.

Then a mathematical formulation for the VDSP is given by:

$$\min \sum_{(i,j)\in A} c^1_{ij}x^1_{ij} + \sum_{(i,j)\in A} c^2_{ij}x^2_{ij} + N \sum_{i\in V_d, j\in V_c} x^1_{ij} + M \sum_{(i,j)\in A} y_{ij} \tag{3.1}$$

subject to:

$$x^1(\delta^+(i)) = x^1(\delta^-(i)) = 1 \qquad i \in V_c \tag{3.2}$$

$$x^1(\delta^+(j)) = x^1(\delta^-(j)) \leq K^1_j \qquad j \in V_d \tag{3.3}$$

$$x^1(\delta^+(S)) \geq \sum_{i\in S}(x^1_{0,i} + x^1_{i,n+1}) \qquad S \subseteq V_c, S \neq \emptyset \tag{3.4}$$

$$x^1(\delta^+(S)) \geq \sum_{i\in S}(x^1_{n+1,i} + x^1_{i,0}) \qquad S \subseteq V_c, S \neq \emptyset \tag{3.5}$$

$$x^1_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{3.6}$$

$$x^2(\delta^+(i)) = x^2(\delta^-(i)) = 1 \qquad i \in V_c \tag{3.7}$$

$$x^2(\delta^+(0)) = x^2(\delta^-(n+1)) = K^2_0 \tag{3.8}$$

$$x^2(\delta^+(n+1)) = x^2(\delta^-(0)) = K^2_{n+1} \tag{3.9}$$

$$x^2(\delta^+(S)) \geq \sum_{i\in S}(x^2_{ji} + x^2_{ij}) \qquad S \subseteq V_c, S \neq \emptyset, j \in V_d \tag{3.10}$$

$$x^2_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{3.11}$$

$$z^k_j \geq z^k_i + x^k_{ij} - (Q^k - 1)(1 - x^k_{ij}) \qquad (i,j) \in A, i,j \in V_c, k = 1,2 \tag{3.12}$$

$$w_j \geq w_i + x^k_{ij} - (n-1)(1 - x^k_{ij}) \qquad (i,j) \in A, i,j \in V_c, k = 1,2 \tag{3.13}$$

$$x^1_{ij} \leq x^2_{ij} + y_{ij} \qquad (i,j) \in A \tag{3.14}$$

$$y_{ij} \geq 0 \qquad (i,j) \in A \tag{3.15}$$

$$w_i \geq 0 \qquad i \in V_c \tag{3.16}$$

$$z^k_i \geq 0 \qquad i \in V_c, k = 1,2. \tag{3.17}$$

The objective function (3.1) to minimize is the sum of the routing cost for the two types of vehicles, the cost of using crews (i.e., vehicles of type 1), and the cost of the changeovers. Equations (3.2) and (3.7) ensure that each customer is visited exactly by one vehicle of each type. Inequalities (3.3) limit the fleet size for vehicles of type 1, and equations (3.8)–(3.9) set the number of vehicles of type 2 that must go from one depot to the other. Constraints (3.4) and (3.5) impose that all the vehicles of type 1 have to start and finish their routes at the same depot. The vehicles of type 2 need to start and finish at different depots, which is guaranteed by (3.10). Inequalities (3.12) establish the maximum number of customers that a vehicle can handle in a single route. These inequalities also ensure that if a vehicle of type $k$ traverses an arc $(i,j) \in A$, the vehicle must serve the customer

$j$ after the customer $i$, thus avoiding subtours. Constraints (3.13) ensure that the customers are visited by two vehicles of type 1 and 2 at the same time. Constraints (3.12) and (3.13) are inspired by the Miller, Tucker and Zemlin inequalities subtour elimination constraints for the TSP (see Miller et al. (1960)) and their improved version by Desrochers & Laporte (1991). In our case these inequalities are not aimed to eliminate infeasible subtours (since we already do this with constraints (3.4), (3.5) and (3.10)), but to ensure that the capacity restriction in the vehicles is respected and to guarantee consistency in the positions at which each customer is visited within the routes of the two types of vehicles. Inequalities (3.14) relate vehicles and changeovers, i.e., if a vehicle of type 1 traverses an arc then, either a vehicle of type 2 also traverses the same arc, or a changeover occurs. Finally, constraints (3.6), (3.11) and (3.15)-(3.17) are variable restrictions.

Observe that the model could be extended to allow time windows constraints if required. To this end, let $t_i \in \mathbb{R}$ be the time needed to serve a customer $i$, and let $t_{ij}^k \in \mathbb{R}$ be the time that a vehicle of type $k$ needs to go from node $i$ to node $j$. Suppose that a customer $i$ has to be served within a time interval $[a_i, b_i]$. Then $w_i$ represents the starting time to serve customer $i$ and the time window constraint can easily be considered by adding the inequality $a_i \leq w_i \leq b_i$. These constraints are of particular relevance for an airline problem where an initial flight scheduling is given and limited retimings on some flights are allowed.

### 3.1.2 Valid inequalities

The mathematical model previously proposed can already provide optimal solutions for small instances. However, in order to find optimal solutions for larger instances it is fundamental to find additional inequalities to strengthen the linear programming relaxation of the model. In this section we present several families of valid inequalities for the VDSP adapted from others given in the literature on CVRP variants.

#### 3.1.2.1 Subtour elimination inequalities

Since subtours involving only customers are not allowed, a first family of valid inequalities that could strengthen the linear-programming relaxation of (3.1)-(3.17) are the so-called *subtour elimination constraints*:

$$x^k(\delta^+(S)) \geq 1 \qquad S \subseteq V_c, S \neq \emptyset, k = 1, 2. \tag{3.18}$$

These inequalities have been adapted from the inequalities (2.6).

#### 3.1.2.2 Capacity inequalities

The continuous variables $z_j^k$ and the inequalities (3.12) avoid partial routes with an infeasible number of customers. An alternative problem formulation can be obtained by replacing the $z_j^k$ variables with an exponential number of inequalities known in the VRP literature as *fractional capacity constraints*:

$$x^k(\delta^+(S)) \geq |S|/Q^k \qquad S \subseteq V_c, S \neq \emptyset, k = 1, 2, \tag{3.19}$$

which are dominated by the so-called *rounded capacity constraints*:

$$x^k(\delta^+(S)) \geq \lceil |S|/Q^k \rceil \qquad \forall S \subseteq V_c, S \neq \emptyset, k = 1, 2, \tag{3.20}$$

which include inequalities (3.18).

The inequalities (3.19) and (3.20) have been proposed as an adaptation of the inequalities, (2.15) and (2.16), respectively.

#### 3.1.2.3 Multistar inequalities

As mentioned in Section 2.2.2, Araque et. al. (1994) introduced the *Multistar Inequalities* for the CVRP with identical demands. These inequalities can be formulated for the VDSP as follows:

$$x^k(\delta^+(S)) \geq \frac{1}{Q^k}(|S| + \sum_{i \in S} \sum_{j \notin S} (x_{ij}^k + x_{ji}^k)) \qquad S \subseteq V_c, S \neq \emptyset, k = 1, 2. \tag{3.21}$$

#### 3.1.2.4 $D_p^+$ and $D_p^-$ inequalities

The strong cycle-breaking inequalities $D_p^+$ and $D_p^-$ *inequalities* (introduced in Section 2.1.1) can be adapted for the VDSP as follows. Let $S = \{i_1, \ldots, i_p\} \subset V_c$ and a vehicle type $k$. The $D_p^+$ inequality is:

$$\sum_{j=1}^{p-1} x_{i_j,i_{j+1}}^k + x_{i_p,i_1}^k + 2\sum_{j=2}^{p-1} x_{i_j,i_1}^k + \sum_{j=3}^{p-1}\sum_{l=2}^{j-1} x_{i_j,i_l}^k \leq p - 1, \tag{3.22}$$

and the $D_p^-$ inequality is:

$$\sum_{j=1}^{p-1} x_{i_j,i_{j+1}}^k + x_{i_p,i_1}^k + 2\sum_{j=3}^{p} x_{i_1,i_j}^k + \sum_{j=4}^{p}\sum_{l=3}^{j-1} x_{i_j,i_l}^k \leq p - 1. \tag{3.23}$$

#### 3.1.2.5 Comb inequalities

Although the *comb inequalities* were presented in Gröstschel & Padberg (1979) for the STSP, as mentioned in Section 2.1.4, Lysgaard et al. Lysgaard et al. (2004) introduced them for the CVRP. For a given vehicle type $k$, these inequalities are valid for the VDSP and are defined as follows. A comb consists of a handle $H \subset V_c$ and a set of teeth $T_1, \ldots, T_t \subset V_c$ $(t \geq 2)$ such that:

- $H \cap T_j \neq \emptyset$ and $T_j \setminus H \neq \emptyset$ for $j = 1, \ldots, t$;
- for each pair $\{i, j\} \subset \{1, \ldots, t\}$: $T_i \cap T_j \subset H$ or $T_i \cap T_j \cap H = \emptyset$.

For any subset $S \subset V$, let $\tilde{r}^k(S)$ be equal to $\lceil |S|/Q^k \rceil$ if $S \subset V_c$, and to $\lceil |V_c \setminus S|/Q^k \rceil$ otherwise. Let

$$S(H, T_1, \ldots, T_t) = \sum_{j=1}^{t} \tilde{r}^k(T_j \cap H) + \tilde{r}^k(T_j \setminus H) + \tilde{r}^k(T_j)$$

If this number is odd, the *comb inequality*

$$x^k(\delta(H)) + \sum_{j=1}^{t} x^k(\delta(T_j)) \geq S(H, T_1, \ldots, T_t) + 1 \tag{3.24}$$

is valid for the VDSP.

## 3.2 Solution approach

In this section we describe a branch-and-cut algorithm to solve the VDSP. As we have explained in Section 1.5, a branch-and-cut algorithm for integer programming problems combines a branch-and-bound method and a cutting-plane method. The branch-and-bound method explores a decision tree and the cutting-plane method computes lower bounds by solving linear programming relaxations improved by valid inequalities. We now describe how to apply this methodology to solve the VDSP with the elements introduced in the previous section.

### 3.2.1 Cutting-plane phase

The first linear-programming relaxation to be solved at the root node of the branch-and-cut tree is defined by (3.1)–(3.3), (3.7)–(3.9), (3.13) and the non-negativity of the variables. We now describe how the other inequalities are dynamically generated. Let $(x^*, y^*, w^*)$ be the solution of the current relaxation, then we apply the following algorithm in each cutting plane phase:

- **Step 1:** We first look for violated constraints (3.14) as described in Section 3.2.1.1.
- **Step 2:** We look for violated constraints (3.21) and (3.20). To separate constraints (3.21) we use the procedure described in Section 3.2.1.5. Finding a violated inequality (3.20) when any exists is a difficult problem. For that reason we have developed heuristic procedures that consists of applying exact separation procedures for similar inequalities. These inequalities are (3.18) and (3.19), and separation procedures for them are described in Sections 3.2.1.2, and 3.2.1.3, respectively. Each time a set $S$ is generated using these procedures or the separation routine for constraints (3.21), we check the violation of the associated inequality (3.20). All the found violated inequalities (3.21) and (3.20) are added to the current linear program.
- **Step 3:** We apply the approach given in Section 3.2.1.7 to look for violated inequalities (3.22) and (3.23).
- **Step 4:** The violation of inequalities (3.4), (3.5) and (3.10) is detected using the procedure described in Section 3.2.1.4.
- **Step 5:** Violated inequalities (3.24) are looked for as described in Section 3.2.1.6.

A step is executed only when the previous one fails to find any violated constraint. Based on preliminary computational experiments we decided to execute the separation of the inequalities (3.21), (3.22), (3.23), and (3.24) only at the root node. The reason is that the time consumed by the separation procedures of these inequalities did not compensate the minor improvement in the objective function when they are also separated in other nodes.

We next describe the procedures for performing the separation of each family of inequalities.

#### 3.2.1.1 Separation of inequalities (3.14)

Since the number of inequalities (3.14) is quadratic in $n$, their violation can be checked in polynomial time by simple enumeration. Note, anyway, that only the arcs $(i, j) \in A$ with $x_{ij}^{1*} > 0$ may induce a violated inequality.

#### 3.2.1.2 Separation of inequalities (3.18)

Although the separation procedure for inequalities (3.18) is well-known in the vehicle routing literature, we summarize it here because it motivates similar separation procedures for other inequalities in this article. The inequalities can be separated in an exact and efficient way by solving min-cut problems as follows. Given a vehicle type $k$, we construct a support graph $G' = (V', A')$ with $V' = V \cup \{t\}$, being $t$ a dummy node. The arc set $A'$ is defined as follows:

- all the arcs $(i, j) \in A$ such that $x_{ij}^{k*} > 0$, each one with capacity $x_{ij}^{k*}$, and
- the arcs connecting the nodes $i \in V_d$ with $t$ with infinite capacity.

We then solve a min-cut problem separating $i$ from $t$ for each node $i \in V_c$. Let $S \subseteq V$ be an optimal min-cut solution, $i \in S$. If the sum of the capacities of the arcs leaving $S$ is smaller than 1 then such $S$ defines a violated inequality (3.18).

#### 3.2.1.3 Separation of inequalities (3.19)

The previous separation procedure can be adapted to find the most violated inequalities (3.19), if any exist. Given a vehicle type $k$, the first step is to rewrite the inequality as

$$x^k(\delta^+(S)) + \frac{|V_c \setminus S|}{Q^k} \geq \frac{|V_c|}{Q^k}.$$

We define a support graph $G' = (V', A')$ with $V' = V \cup \{s, t\}$, being $s$ and $t$ dummy nodes. We build the arc set $A'$ as follows:

- all the arcs $(i, j) \in A$ such that $x_{ij}^{k*} > 0$, each one with capacity $x_{ij}^{k*}$,
- all arcs connecting $s$ with nodes $i \in V_c$, each one with capacity $1/Q^k$, and
- the arcs connecting the nodes $i \in V_d$ with $t$, each one with infinite capacity.

Let $S \subset V'$ be the solution of the min-cut problem separating $s$ from $t$, $s \in S$. If the capacity of the arcs leaving $S$ on $G'$ is smaller than $|V_c|/Q^k$ then $S$ defines a violated inequality (3.19).

#### 3.2.1.4 Separation of inequalities (3.4), (3.5) and (3.10)

Inequalities (3.4) can be rewritten as

$$x^1(\delta^+(S)) + \sum_{i \notin S}(x^1_{0,i} + x^1_{i,n+1}) \geq \sum_{i \in V_c}(x^1_{0,i} + x^1_{i,n+1}).$$

Let us consider a support graph $G' = (V', A')$ with $V' = V \cup \{s, t\}$, being $s$ and $t$ dummy nodes. The arc set $A'$ is defined as follows:

- all the arcs $(i,j) \in A$ such that $x^{1*}_{ij} > 0$, each one with capacity $x^{1*}_{ij}$,
- all arcs connecting $s$ with nodes $i \in V_c$, each one with capacity $x^{1*}_{0,i} + x^{1*}_{i,n+1}$, and
- the arcs connecting the nodes $i \in V_d$ with $t$, each one with infinite capacity.

Let $S \subset V'$ be the solution of the min-cut problem separating $s$ from $t$, $s \in S$. If the capacity of the arcs leaving $S$ on $G'$ is smaller than $\sum_{i \in V_c}(x^{1*}_{0,i} + x^{1*}_{i,n+1})$ then $S$ defines a violated inequality (3.4) that must be added to the current linear program.

The separation procedures for (3.5) and (3.10) are similar.

#### 3.2.1.5 Separation of inequalities (3.21)

The inequalities (3.21) can be rewritten as

$$(Q^k - 2)\sum_{i \in S}\sum_{j \notin S} x^{k*}_{ij} + \sum_{j \notin S}\sum_{i \in V_c} x^{k*}_{ij} \geq |V_c|.$$

We consider a support graph $G' = (V', A')$ with $V' = V \cup \{s, t\}$, being $s$ and $t$ dummy nodes. In this case, the arcs set $A'$ is defined as follows:

- all the arcs $(i,j) \in A$ such that $x^{k*}_{ij} > 0$, each one with capacity $(Q^k - 2)x^{k*}_{ij}$,
- the arcs connecting $s$ with the nodes $i \in V_d$ with infinite capacity, and
- the arcs connecting the nodes $j \in V_c$ with $t$ with capacity $\sum_{i \in V_c} x^{k*}_{ij}$.

Let $S \subset V'$ be the solution of the min-cut problem separating $s$ from $t$ with $t \in S$. If the capacity of the arcs leaving $S$ on $G'$ is smaller than $|V_c|$ then $S$ defines a violated inequality (3.21).

#### 3.2.1.6 Separation of inequalities (3.24)

To separate the inequalities (3.24), we apply a heuristic separation procedure adapting to the VDSP one of the routines of the CVRPSEP software by Lysgaard (2004). The adaptation consists of merging 0 and $n+1$ in a single depot.

#### 3.2.1.7 Separation of inequalities (3.22) and (3.23)

We use a heuristic separation procedure for the inequalities (3.22) and (3.23). Given a fractional solution and a vehicle type $k$, we apply a nearest-neighbor strategy to build a sequence of $p$ nodes by considering the arcs in decreasing order of the values $x_{ij}^{k*}$. When the arcs linking those $p$ nodes form a cycle in the support graph $G' = (V', A')$, with $V' = V$ and $A' = \{(i,j) \in A : x_{ij}^{k*} > 0\}$, then the inequalities (3.22) and (3.23) are checked for violation.

## 3.3 Computational results

In this section we show and analyze the computational results obtained when using the above described branch-and-cut approach to solve VDSP instances. The algorithm was coded in C++ and ran on a desktop computer with an Intel(R) Core(TM) i3-3240 CPU @ 4.40 GHz, 8GB RAM, Windows 7 Professional and CPLEX 12.6 as MIP solver. To solve the min-cut problems we used the routine included in the *Concorde TSP* software package by Applegate et al. (2003).

Since the VDSP has not been studied before, we have created new instances by adapting benchmark CVRP instances proposed by Christofides et al. (1979), and Augerat et al. (1995) (available at the CVRPLIB (2014)). We have generated instances with $n+2 \in \{16, 20, 32, 40, 50\}$. The routing cost $c_{ij}^k$ is defined as the Euclidean distance between $i$ and $j$. The VDSP instances are classified in three different classes according to the capacity of the vehicles. The vehicle capacity for the instances in Class $a$ is defined as:

$$Q^k = \left\lceil \frac{n}{K_0^k + K_{n+1}^k} \right\rceil$$

The vehicle capacity for the instances in Class $b$ is set to:

$$Q^k = \left\lceil \left( n + 2 + \left\lceil \frac{n}{K_0^k + K_{n+1}^k} \right\rceil \right) /3 \right\rceil$$

Instances of Class $c$ have $Q^k = n + 2$. Note that instances in Class $a$ have tighter capacities than the instances in Class $b$, and that Class $c$ instances are uncapacitated. For each problem size $n + 2$, we created different instances of each class by varying the values of the parameters $K_d^k$, producing a set of 69 instances. All instances are available from the authors upon request.

The first issue that we addressed was the comparison between the basic model given in Section 3.1.1 (Model 1) and the alternative model mentioned in Section 3.1.2.2 (Model 2), that uses constraints (3.19) instead of the constraints involving variables $z_i^k$. We implemented a branch-and-cut for solving each model, and display the obtained results in Table 3.1. Each line in the table corresponds to a VDSP instance. We report results for only a subset of representative instances, with 16 and 20 nodes, since the algorithms reach the imposed time limit on larger instances. The first column shows the instance name, and the next 7 columns show its main features. For each model, column *route* display the routing cost value, $x$ is the number of vehicles of type 1 used in an optimal solution, and $y$ is the number of changeovers. Columns *gap* show the percentage ratio between the optimal objective value minus the value of the LP-relaxation at the end of the root node, and the optimal objective value. Columns *r-time*, *time*, *nodes* and *cuts* show the computing time in seconds to solve the LP relaxation, the total computing time in seconds, the number of explored nodes in the branch-and-cut tree, and the overall number of cuts added, respectively. We imposed a time limit of 2 hours on each instance, and if reached we write T.L. in the column *time*, and use the cost of the best solution found instead of the optimal solution value to compute the ratio in column *gap*. All instances in the table were solved using $M = 100$ an $N = 10$.

From Table 3.1 we can infer the difficulty of solving to optimality the instances of Class $a$ with more than 16 nodes using only Model 1 or Model 2, without additional valid inequalities. Though not reported in the table, the branch-and-cut algorithms rarely find the optimal solution for instances in Class $b$ with more than 20 nodes, and it happens so even with the uncapacitated instances in Class $c$ with 40 or 50 nodes.

In general, we can conclude that we obtain better computational results in terms of *time* and *gap* using Model 2. The improvement obtained is larger in the instances of Class $a$, where the lower bound at the root node is better for all the instances and the times are considerably better in some cases. In the Class $b$, still most of the times Model 2 compares favorably with Model 1, finding only a few cases where the lower bound is better with the latter. Finally, the small Class $c$ instances in the table are easily solved by both algorithms in a short time, and their performance is rather similar. In view of these results, we conclude that it is better to use Model 2, that is, to eliminate the variables $z_i^k$ and use instead inequalities (3.19).

As might be expected, the performance of the branch-and-cut algorithm is better when the additional valid inequalities described in Section 3.1.2 are used. Table 3.2 reports the performance of the enhanced algorithm on VDSP instances with two different settings of the parameters $M$ and $N$. The first setting is the one used in the previous computational experiment, i.e. $M = 100$ and $N = 10$. The second setting reduces the penalties on the number of changeovers and the number of vehicles of type 1 in the objective function, by defining $M = 10$ and $N = 1$.

A first observation from Table 3.1 (columns under Model 2) and Table 3.2 (columns under $M = 100$, $N = 10$) is that the valid inequalities described in Section 3.1.2 are useful. Indeed, in all the common instances (those with 16 and 20 nodes) they noticeably help to reduce the gap at the root node, besides reducing also, in most cases, the computing times. Using the valid inequalities we manage to solve to optimality 3 instances more than without them.

Table 3.2 shows that the reduction of the penalties $M$ and $N$ makes the problem easier to solve. In fact, the algorithm fails to solve to optimality 17 out of the 69 instances within the given time limit when $M = 100$ and $N = 10$, while this figure goes down to only 6 when $M = 10$ and $N = 1$. We can say that a general rule is that the larger the instance, the harder to solve. Also, for a given size, the instances with the tightest capacities (Class $a$) are the more difficult. However, there are clear differences on these statements depending on the penalties. With the reduced penalties all instances but one with 32 nodes or less are solved, and the 6 unsolved instances belong to Class $a$. With the larger penalties, we are unable to solve to optimality even 2 of the smallest instances with 16 nodes. Most of the instances of Class $a$ are unsolved (13 out of 23), and there are 4 unsolved instances of Class $b$. Summarizing, the computing times are generally smaller on the instances with $M = 10$ and $N = 1$, and this reduction is especially remarkable in the instances of Class $a$.

Regarding the LP gaps at the root node, we can not conclude from Table 3.2 that they are affected by the values of the penalties. Observe also, that when $M$ and $N$ change, the optimal routes can change, and consequently, their costs. Anyway, the gaps are below 4% in the great majority of instances solved to optimality, with both penalty settings.

| | | | | | | | | Model 1 | | | | | | | | Model 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *name* | $n+2$ | $Q^2$ | $Q^1$ | $K_0^2$ | $K_{n+1}^2$ | $K_0^1$ | $K_{n+1}^1$ | route | $x$ | $y$ | gap | r-time | time | nodes | cuts | route | $x$ | $y$ | gap | r-time | time | nodes | cuts |
| n16-1-1-2-2a | 16 | 7 | 4 | 1 | 1 | 2 | 2 | 486 | 4 | 0 | 19.58 | 0.16 | 2279.52 | 64806 | 2084 | 486 | 4 | 0 | 10.09 | 0.09 | 64.23 | 3633 | 1053 |
| n16-1-1-2-2b | 16 | 8 | 7 | 1 | 1 | 2 | 2 | 443 | 2 | 0 | 8.64 | 0.14 | 10.53 | 914 | 302 | 443 | 2 | 0 | 7.78 | 0.06 | 2.89 | 275 | 343 |
| n16-1-1-2-2c | 16 | 16 | 16 | 1 | 1 | 2 | 2 | 411 | 2 | 0 | 1.86 | 0.12 | 0.19 | 4 | 121 | 411 | 2 | 0 | 2.47 | 0.08 | 0.12 | 4 | 126 |
| n16-1-1-1-1a | 16 | 7 | 7 | 1 | 1 | 1 | 1 | 447 | 2 | 0 | 9.21 | 0.14 | 13.92 | 1250 | 330 | 447 | 2 | 0 | 8.35 | 0.12 | 6.26 | 512 | 539 |
| n16-1-1-1-1b | 16 | 8 | 8 | 1 | 1 | 1 | 1 | 445 | 2 | 0 | 8.82 | 0.19 | 10.83 | 1040 | 325 | 445 | 2 | 0 | 8.44 | 0.14 | 7 | 530 | 688 |
| n16-1-1-1-1c | 16 | 16 | 16 | 1 | 1 | 1 | 1 | 411 | 2 | 0 | 1.62 | 0.20 | 0.25 | 4 | 126 | 411 | 2 | 0 | 2.23 | 0.08 | 0.14 | 8 | 131 |
| n16-1-1-2-3a | 16 | 7 | 3 | 1 | 1 | 2 | 3 | 521 | 5 | 1 | 36.96 | 0.12 | T.L. | 57705 | 5358 | 537 | 5 | 1 | 24.81 | 0.11 | T.L. | 38372 | 8193 |
| n16-1-1-2-3b | 16 | 8 | 7 | 1 | 1 | 2 | 3 | 443 | 2 | 0 | 8.64 | 0.16 | 9.47 | 837 | 304 | 443 | 2 | 0 | 7.78 | 0.06 | 2.93 | 275 | 343 |
| n16-1-1-2-3c | 16 | 16 | 16 | 1 | 1 | 2 | 3 | 411 | 2 | 0 | 3.05 | 0.16 | 0.2 | 7 | 115 | 411 | 2 | 0 | 2.47 | 0.11 | 0.14 | 4 | 126 |
| n16-2-2-1-1a | 16 | 4 | 7 | 2 | 2 | 1 | 1 | 507 | 2 | 2 | 8.73 | 0.14 | 250.65 | 16933 | 752 | 507 | 2 | 2 | 7.42 | 0.14 | 179.96 | 7923 | 1559 |
| n16-2-2-1-1b | 16 | 7 | 8 | 2 | 2 | 1 | 1 | 481 | 2 | 2 | 6.32 | 0.11 | 18.94 | 2223 | 382 | 481 | 2 | 2 | 5.94 | 0.12 | 11.9 | 1423 | 618 |
| n16-2-2-1-1c | 16 | 16 | 16 | 2 | 2 | 1 | 1 | 454 | 2 | 2 | 2.63 | 0.11 | 0.27 | 32 | 216 | 454 | 2 | 2 | 2.63 | 0.12 | 0.39 | 70 | 233 |
| n16-2-2-4-4a | 16 | 4 | 2 | 2 | 2 | 4 | 4 | 626 | 7 | 0 | 25.72 | 0.19 | 1266.24 | 32829 | 1762 | 626 | 7 | 0 | 7.69 | 0.25 | 131.34 | 6084 | 1276 |
| n16-2-2-4-4b | 16 | 7 | 6 | 2 | 2 | 4 | 4 | 497 | 4 | 0 | 3.72 | 0.19 | 3 | 269 | 261 | 497 | 4 | 0 | 5.06 | 0.25 | 2.39 | 228 | 349 |
| n16-2-2-4-4c | 16 | 16 | 16 | 2 | 2 | 4 | 4 | 484 | 4 | 0 | 3.63 | 0.31 | 0.5 | 20 | 192 | 484 | 4 | 0 | 3.62 | 0.20 | 0.34 | 25 | 207 |
| n16-2-3-1-1a | 16 | 3 | 7 | 2 | 3 | 1 | 1 | 537 | 2 | 4 | 18.03 | 0.11 | T.L. | 77063 | 4328 | 551 | 2 | 4 | 16.94 | 0.16 | T.L. | 40141 | 8192 |
| n16-2-3-1-1b | 16 | 7 | 8 | 2 | 3 | 1 | 1 | 504 | 2 | 3 | 5.12 | 0.11 | 15.88 | 2039 | 366 | 504 | 2 | 3 | 5.38 | 0.11 | 35.21 | 3953 | 815 |
| n16-2-3-1-1c | 16 | 16 | 16 | 2 | 3 | 1 | 1 | 480 | 2 | 3 | 3.12 | 0.12 | 0.42 | 77 | 218 | 480 | 2 | 3 | 3.12 | 0.08 | 0.56 | 150 | 265 |
| n20-1-1-2-3a | 20 | 9 | 4 | 1 | 1 | 2 | 3 | 675 | 5 | 1 | 41.09 | 0.11 | T.L. | 32379 | 7530 | 678 | 5 | 1 | 30.25 | 0.30 | T.L. | 20808 | 11061 |
| n20-1-1-2-3b | 20 | 10 | 8 | 1 | 1 | 2 | 3 | 531 | 3 | 0 | 13.50 | 0.14 | 5020.6 | 67709 | 3010 | 531 | 3 | 0 | 12.12 | 0.19 | 1238.29 | 16878 | 3301 |
| n20-1-1-2-3c | 20 | 20 | 20 | 1 | 1 | 2 | 3 | 503 | 2 | 0 | 7.07 | 0.11 | 4.27 | 204 | 343 | 503 | 2 | 0 | 8.14 | 0.12 | 4.99 | 377 | 471 |
| n20-1-1-3-3a | 20 | 9 | 3 | 1 | 1 | 3 | 3 | 666 | 6 | 0 | 33.06 | 0.11 | T.L. | 37032 | 5482 | 652 | 6 | 0 | 11.42 | 0.30 | T.L. | 38704 | 6033 |
| n20-1-1-3-3b | 20 | 10 | 8 | 1 | 1 | 3 | 3 | 531 | 3 | 0 | 13.50 | 0.17 | 4056.32 | 59987 | 2776 | 531 | 3 | 0 | 12.12 | 0.19 | 2124.2 | 21108 | 4192 |
| n20-1-1-3-3c | 20 | 20 | 20 | 1 | 1 | 3 | 3 | 503 | 2 | 0 | 7.07 | 0.11 | 3.99 | 193 | 343 | 503 | 2 | 0 | 8.14 | 0.12 | 6.55 | 506 | 537 |
| n20-2-2-5-5a | 20 | 5 | 2 | 2 | 2 | 5 | 5 | 860 | 10 | 0 | 40.10 | 0.20 | T.L. | 25832 | 6582 | 861 | 10 | 0 | 16.19 | 0.47 | T.L. | 25228 | 8905 |
| n20-2-2-5-5b | 20 | 9 | 8 | 2 | 2 | 5 | 5 | 589 | 4 | 0 | 8.59 | 0.17 | 191.91 | 7088 | 780 | 589 | 4 | 0 | 9.10 | 0.30 | 202.27 | 6168 | 1333 |
| n20-2-2-5-5c | 20 | 20 | 20 | 2 | 2 | 5 | 5 | 554 | 4 | 0 | 5.74 | 0.19 | 2.65 | 150 | 307 | 554 | 4 | 0 | 5.60 | 0.16 | 2.68 | 214 | 370 |
| n20-2-3-1-1a | 20 | 4 | 9 | 2 | 3 | 1 | 1 | 641 | 2 | 4 | 18.14 | 0.28 | T.L. | 44232 | 4576 | 685 | 2 | 4 | 18.12 | 0.44 | T.L. | 21641 | 10268 |
| n20-2-3-1-1b | 20 | 8 | 10 | 2 | 3 | 1 | 1 | 587 | 2 | 3 | 4.44 | 0.22 | 52.21 | 2406 | 548 | 587 | 2 | 3 | 3.33 | 0.30 | 27.67 | 1535 | 829 |
| n20-2-3-1-1c | 20 | 20 | 20 | 2 | 3 | 1 | 1 | 552 | 2 | 3 | 0.00 | 0.19 | 0.27 | 4 | 322 | 552 | 2 | 3 | 0.44 | 0.30 | 0.37 | 9 | 349 |
| n20-5-5-2-2a | 20 | 2 | 5 | 5 | 5 | 2 | 2 | 939 | 4 | 6 | 11.22 | 0.27 | T.L. | 71429 | 2346 | 951 | 4 | 6 | 7.17 | 1.01 | T.L. | 23951 | 7107 |
| n20-5-5-2-2b | 20 | 8 | 9 | 5 | 5 | 2 | 2 | 787 | 4 | 6 | 3.05 | 0.22 | 21.98 | 1735 | 466 | 787 | 4 | 6 | 1.62 | 0.25 | 8.99 | 917 | 576 |
| n20-5-5-2-2c | 20 | 20 | 20 | 5 | 5 | 2 | 2 | 771 | 4 | 6 | 1.74 | 0.20 | 1.34 | 94 | 374 | 771 | 4 | 6 | 0.92 | 0.30 | 0.62 | 42 | 377 |

Table 3.1: Comparing the two problem models.

| | | | | | | | | $M = 100,\ N = 10$ | | | | | | | | $M = 10,\ N = 1$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | $n+2$ | $Q^2$ | $Q^1$ | $K_0^2$ | $K_{n+1}^2$ | $K_0^1$ | $K_{n+1}^1$ | route | $x$ | $y$ | gap | r-time | time | nodes | cuts | route | $x$ | $y$ | gap | r-time | time | nodes | cuts |
| n16-1-1-2-2a | 16 | 7 | 4 | 1 | 1 | 2 | 2 | 486 | 4 | 0 | 3.92 | 0.16 | 0.87 | 69 | 443 | 486 | 4 | 0 | 2.03 | 0.19 | 0.95 | 50 | 235 |
| n16-1-1-2-2b | 16 | 8 | 7 | 1 | 1 | 2 | 2 | 443 | 2 | 0 | 3.56 | 0.09 | 0.73 | 51 | 358 | 443 | 2 | 0 | 3.71 | 0.11 | 1.06 | 80 | 463 |
| n16-1-1-2-2c | 16 | 16 | 16 | 1 | 1 | 2 | 2 | 411 | 2 | 0 | 0.46 | 0.17 | 0.19 | 4 | 136 | 397 | 1 | 1 | 0.25 | 0.09 | 0.09 | 3 | 93 |
| n16-1-1-1-1a | 16 | 7 | 7 | 1 | 1 | 1 | 1 | 447 | 2 | 0 | 3.21 | 0.19 | 2.95 | 136 | 1016 | 447 | 2 | 0 | 3.34 | 0.16 | 2.43 | 140 | 598 |
| n16-1-1-1-1b | 16 | 8 | 8 | 1 | 1 | 1 | 1 | 445 | 2 | 0 | 3.87 | 0.06 | 1.19 | 98 | 396 | 445 | 2 | 0 | 6.04 | 0.06 | 2.43 | 169 | 652 |
| n16-1-1-1-1c | 16 | 16 | 16 | 1 | 1 | 1 | 1 | 411 | 2 | 0 | 0.23 | 0.20 | 0.22 | 3 | 137 | 397 | 1 | 1 | 0.25 | 0.11 | 0.12 | 3 | 89 |
| n16-1-1-2-3a | 16 | 7 | 3 | 1 | 1 | 2 | 3 | 519 | 5 | 1 | 15.31 | 0.36 | T.L. | 37634 | 5819 | 519 | 5 | 1 | 2.39 | 0.25 | 0.84 | 57 | 243 |
| n16-1-1-2-3b | 16 | 8 | 7 | 1 | 1 | 2 | 3 | 443 | 2 | 0 | 3.56 | 0.09 | 0.75 | 51 | 358 | 443 | 2 | 0 | 3.71 | 0.11 | 1.08 | 80 | 463 |
| n16-1-1-2-3c | 16 | 16 | 16 | 1 | 1 | 2 | 3 | 411 | 2 | 0 | 0.46 | 0.20 | 0.22 | 4 | 136 | 397 | 1 | 1 | 0.25 | 0.09 | 0.11 | 3 | 93 |
| n16-2-2-1-1a | 16 | 4 | 7 | 2 | 2 | 1 | 1 | 507 | 2 | 2 | 2.96 | 0.16 | 2.98 | 174 | 870 | 507 | 2 | 2 | 3.69 | 0.39 | 2.87 | 143 | 418 |
| n16-2-2-1-1b | 16 | 7 | 8 | 2 | 2 | 1 | 1 | 481 | 2 | 2 | 2.50 | 0.08 | 0.66 | 78 | 383 | 481 | 2 | 2 | 5.17 | 0.06 | 2.04 | 168 | 668 |
| n16-2-2-1-1c | 16 | 16 | 16 | 2 | 2 | 1 | 1 | 454 | 2 | 2 | 0.67 | 0.09 | 0.11 | 4 | 224 | 440 | 1 | 3 | 0.21 | 0.08 | 0.11 | 4 | 103 |
| n16-2-2-4-4a | 16 | 4 | 2 | 2 | 2 | 4 | 4 | 626 | 7 | 0 | 0.44 | 0.41 | 0.58 | 12 | 273 | 626 | 7 | 0 | 0.57 | 0.20 | 0.73 | 33 | 190 |
| n16-2-2-4-4b | 16 | 7 | 6 | 2 | 2 | 4 | 4 | 497 | 4 | 0 | 3.72 | 0.16 | 0.62 | 47 | 449 | 484 | 3 | 1 | 2.11 | 0.11 | 1.03 | 49 | 323 |
| n16-2-2-4-4c | 16 | 16 | 16 | 2 | 2 | 4 | 4 | 484 | 4 | 0 | 1.34 | 0.20 | 0.31 | 18 | 232 | 440 | 1 | 3 | 0.21 | 0.12 | 0.12 | 3 | 97 |
| n16-2-3-1-1a | 16 | 3 | 7 | 2 | 3 | 1 | 1 | 536 | 2 | 4 | 11.45 | 0.39 | T.L. | 37052 | 16837 | 535 | 2 | 4 | 4.36 | 0.34 | 5.52 | 242 | 629 |
| n16-2-3-1-1b | 16 | 7 | 8 | 2 | 3 | 1 | 1 | 504 | 2 | 3 | 2.79 | 0.08 | 0.97 | 132 | 568 | 504 | 2 | 3 | 3.92 | 0.11 | 1.92 | 158 | 493 |
| n16-2-3-1-1c | 16 | 16 | 16 | 2 | 3 | 1 | 1 | 480 | 2 | 3 | 0.56 | 0.09 | 0.12 | 7 | 232 | 466 | 1 | 4 | 0.20 | 0.06 | 0.06 | 3 | 83 |
| n20-1-1-2-3a | 20 | 9 | 4 | 1 | 1 | 2 | 3 | 624 | 5 | 1 | 16.37 | 0.69 | T.L. | 20324 | 10269 | 613 | 5 | 1 | 5.39 | 0.78 | 37.39 | 982 | 917 |
| n20-1-1-2-3b | 20 | 10 | 8 | 1 | 1 | 2 | 3 | 531 | 3 | 0 | 4.46 | 0.33 | 9.81 | 392 | 792 | 529 | 4 | 0 | 4.80 | 0.42 | 7.24 | 317 | 544 |
| n20-1-1-2-3c | 20 | 20 | 20 | 1 | 1 | 2 | 3 | 503 | 2 | 0 | 7.07 | 0.11 | 3.88 | 225 | 555 | 466 | 1 | 1 | 1.89 | 0.09 | 0.16 | 6 | 61 |
| n20-1-1-3-3a | 20 | 9 | 3 | 1 | 1 | 3 | 3 | 651 | 6 | 0 | 1.84 | 0.75 | 4.56 | 124 | 561 | 651 | 6 | 0 | 2.44 | 0.45 | 1.84 | 61 | 328 |
| n20-1-1-3-3b | 20 | 10 | 8 | 1 | 1 | 3 | 3 | 531 | 3 | 0 | 4.46 | 0.33 | 6.22 | 252 | 883 | 529 | 4 | 0 | 5.16 | 0.27 | 15.02 | 590 | 998 |
| n20-1-1-3-3c | 20 | 20 | 20 | 1 | 1 | 3 | 3 | 503 | 2 | 0 | 7.07 | 0.14 | 3.90 | 225 | 555 | 466 | 1 | 1 | 1.89 | 0.08 | 0.14 | 6 | 61 |
| n20-2-2-5-5a | 20 | 5 | 2 | 2 | 2 | 5 | 5 | 821 | 10 | 0 | 4.23 | 0.78 | 140.24 | 3679 | 1292 | 803 | 10 | 1 | 2.83 | 0.69 | 17.10 | 595 | 544 |
| n20-2-2-5-5b | 20 | 9 | 8 | 2 | 2 | 5 | 5 | 589 | 4 | 0 | 3.97 | 0.41 | 7.35 | 291 | 1084 | 572 | 3 | 1 | 2.91 | 0.31 | 2.22 | 113 | 418 |
| n20-2-2-5-5c | 20 | 20 | 20 | 2 | 2 | 5 | 5 | 554 | 4 | 0 | 3.20 | 0.20 | 0.87 | 47 | 304 | 529 | 3 | 1 | 0.55 | 0.17 | 0.20 | 4 | 207 |
| n20-2-3-1-1a | 20 | 4 | 9 | 2 | 3 | 1 | 1 | 654 | 2 | 4 | 13.09 | 0.66 | T.L. | 19823 | 31614 | 638 | 2 | 4 | 4.90 | 0.64 | 121.09 | 2883 | 1789 |
| n20-2-3-1-1b | 20 | 8 | 10 | 2 | 3 | 1 | 1 | 587 | 2 | 3 | 1.05 | 0.33 | 0.64 | 21 | 467 | 587 | 2 | 3 | 0.62 | 0.53 | 1.50 | 36 | 369 |
| n20-2-3-1-1c | 20 | 20 | 20 | 2 | 3 | 1 | 1 | 552 | 2 | 3 | 0.00 | 0.17 | 0.19 | 2 | 406 | 552 | 2 | 3 | 0.00 | 0.17 | 0.17 | 1 | 186 |
| n20-5-5-2-2a | 20 | 2 | 5 | 5 | 5 | 2 | 2 | 927 | 4 | 6 | 1.24 | 1.61 | 36.16 | 781 | 1181 | 907 | 4 | 7 | 1.16 | 1.83 | 11.37 | 264 | 810 |
| n20-5-5-2-2b | 20 | 8 | 9 | 5 | 5 | 2 | 2 | 787 | 4 | 6 | 0.88 | 0.33 | 1.42 | 89 | 531 | 777 | 3 | 7 | 1.53 | 0.28 | 2.37 | 120 | 563 |
| n20-5-5-2-2c | 20 | 20 | 20 | 5 | 5 | 2 | 2 | 771 | 4 | 6 | 0.35 | 0.39 | 0.51 | 16 | 427 | 761 | 3 | 7 | 0.48 | 0.20 | 0.28 | 11 | 213 |
| n32-1-1-4-4a | 32 | 15 | 4 | 1 | 1 | 4 | 4 | 1566 | 8 | 0 | 4.27 | 5.54 | T.L. | 25290 | 4489 | 1534 | 8 | 1 | 3.39 | 4.96 | 491.48 | 3958 | 2111 |
| n32-1-1-4-4b | 32 | 16 | 12 | 1 | 1 | 4 | 4 | 1155 | 4 | 0 | 8.10 | 6.86 | T.L. | 18266 | 13555 | 1098 | 3 | 2 | 3.28 | 5.29 | 416.10 | 4002 | 2389 |
| n32-1-1-4-4c | 32 | 32 | 32 | 1 | 1 | 4 | 4 | 979 | 2 | 0 | 4.70 | 2.06 | 73.68 | 1081 | 1616 | 913 | 1 | 1 | 0.00 | 1.73 | 1.73 | 1 | 535 |
| n32-2-2-1-1a | 32 | 8 | 15 | 2 | 2 | 1 | 1 | 1269 | 2 | 2 | 3.80 | 10.51 | 1540.79 | 4907 | 9962 | 1216 | 2 | 6 | 4.29 | 4.27 | 1183.05 | 10214 | 2510 |
| n32-2-2-1-1b | 32 | 14 | 16 | 2 | 2 | 1 | 1 | 1170 | 2 | 2 | 1.97 | 15.68 | 197.86 | 1374 | 2612 | 1149 | 2 | 3 | 1.86 | 6.10 | 144.68 | 1342 | 3556 |
| n32-2-2-1-1c | 32 | 32 | 32 | 2 | 2 | 1 | 1 | 1053 | 2 | 2 | 0.00 | 6.40 | 6.40 | 1 | 1065 | 1038 | 1 | 3 | 0.00 | 1.87 | 1.89 | 1 | 521 |
| n32-2-2-5-5a | 32 | 8 | 3 | 2 | 2 | 5 | 5 | 1970 | 10 | 2 | 11.06 | 10.75 | T.L. | 16075 | 5803 | 1920 | 10 | 4 | 2.45 | 10.02 | 974.30 | 6729 | 1955 |
| n32-2-2-5-5b | 32 | 14 | 12 | 2 | 2 | 5 | 5 | 1281 | 4 | 0 | 6.60 | 8.85 | 3592.36 | 16702 | 7943 | 1203 | 3 | 4 | 3.47 | 4.02 | 700.27 | 6506 | 3271 |
| n32-2-2-5-5c | 32 | 32 | 32 | 2 | 2 | 5 | 5 | 1139 | 4 | 0 | 2.63 | 2.01 | 15.71 | 283 | 1017 | 1038 | 1 | 3 | 0.16 | 2.42 | 3.06 | 11 | 585 |
| n32-3-3-3-3a | 32 | 5 | 5 | 3 | 3 | 3 | 3 | 1860 | 6 | 0 | 4.27 | 14.60 | T.L. | 15609 | 6014 | 1829 | 6 | 2 | 2.73 | 14.12 | T.L. | 20366 | 5003 |
| n32-3-3-3-3b | 32 | 13 | 13 | 3 | 3 | 3 | 3 | 1454 | 6 | 0 | 4.60 | 11.40 | 784.61 | 5859 | 3771 | 1334 | 3 | 4 | 2.78 | 4.13 | 212.22 | 2756 | 2104 |
| n32-3-3-3-3c | 32 | 32 | 32 | 3 | 3 | 3 | 3 | 1331 | 6 | 0 | 1.33 | 4.13 | 5.97 | 46 | 747 | 1174 | 1 | 5 | 0.33 | 1.51 | 2.37 | 16 | 505 |
| n32-4-4-1-1a | 32 | 4 | 15 | 4 | 4 | 1 | 1 | 1732 | 2 | 6 | 2.95 | 14.09 | T.L. | 13646 | 14437 | 1661 | 2 | 10 | 2.10 | 12.42 | 984.73 | 6036 | 4814 |
| n32-4-4-1-1b | 32 | 12 | 16 | 4 | 4 | 1 | 1 | 1463 | 2 | 6 | 1.10 | 12.64 | 63.87 | 447 | 2609 | 1463 | 2 | 6 | 2.47 | 4.80 | 148.78 | 1782 | 3222 |
| n32-4-4-1-1c | 32 | 32 | 32 | 4 | 4 | 1 | 1 | 1330 | 2 | 6 | 0.00 | 7.14 | 7.16 | 1 | 1037 | 1321 | 1 | 7 | 0.40 | 2.06 | 3.04 | 21 | 456 |
| n40-1-1-4-4a | 40 | 19 | 5 | 1 | 1 | 4 | 4 | 974 | 8 | 0 | 8.34 | 7.83 | T.L. | 8543 | 10478 | 951 | 8 | 1 | 7.49 | 9.86 | T.L. | 11444 | 9077 |
| n40-1-1-4-4b | 40 | 20 | 15 | 1 | 1 | 4 | 4 | 796 | 4 | 0 | 8.13 | 4.70 | T.L. | 9737 | 32716 | 752 | 3 | 1 | 3.07 | 3.73 | 1300.99 | 7676 | 6731 |
| n40-1-1-4-4c | 40 | 40 | 40 | 1 | 1 | 4 | 4 | 736 | 2 | 0 | 2.38 | 6.04 | 177.48 | 1526 | 3064 | 716 | 1 | 1 | 0.96 | 3.63 | 11.87 | 99 | 745 |
| n40-2-2-10-10a | 40 | 10 | 2 | 2 | 2 | 10 | 10 | 1374 | 19 | 0 | 1.19 | 14.38 | 1512.37 | 6498 | 3457 | 1374 | 19 | 0 | 1.48 | 15.62 | 1343.12 | 6508 | 2789 |
| n40-2-2-10-10b | 40 | 17 | 14 | 2 | 2 | 10 | 10 | 824 | 4 | 0 | 6.57 | 7.97 | T.L. | 11663 | 12594 | 769 | 3 | 3 | 3.67 | 7.96 | 4915.97 | 21148 | 8594 |
| n40-2-2-10-10c | 40 | 40 | 40 | 2 | 2 | 10 | 10 | 791 | 4 | 0 | 3.43 | 8.08 | 2754.65 | 12835 | 4813 | 748 | 2 | 2 | 0.65 | 4.17 | 17.05 | 188 | 880 |
| n40-4-4-4-4a | 40 | 5 | 5 | 4 | 4 | 4 | 4 | 1203 | 8 | 0 | 9.57 | 29.80 | T.L. | 9192 | 7810 | 1252 | 8 | 4 | 15.73 | 41.93 | T.L. | 10533 | 7921 |
| n40-4-4-4-4b | 40 | 15 | 15 | 4 | 4 | 4 | 4 | 946 | 8 | 0 | 3.75 | 22.89 | T.L. | 8847 | 14005 | 902 | 6 | 2 | 2.37 | 7.16 | 1522.34 | 6769 | 7199 |
| n40-4-4-4-4c | 40 | 40 | 40 | 4 | 4 | 4 | 4 | 938 | 8 | 0 | 3.00 | 19.69 | 5840.10 | 16343 | 4519 | 849 | 3 | 5 | 0.17 | 6.35 | 7.47 | 8 | 752 |
| n40-10-10-2-2a | 40 | 2 | 10 | 10 | 10 | 2 | 2 | 1560 | 4 | 16 | 0.34 | 81.26 | 1242.30 | 2581 | 3471 | 1560 | 4 | 16 | 0.59 | 63.99 | 1067.11 | 2822 | 3503 |
| n40-10-10-2-2b | 40 | 14 | 17 | 10 | 10 | 2 | 2 | 1326 | 4 | 16 | 0.37 | 19.02 | 590.00 | 2172 | 4282 | 1326 | 4 | 16 | 0.83 | 13.03 | 457.77 | 3158 | 3389 |
| n40-10-10-2-2c | 40 | 40 | 40 | 10 | 10 | 2 | 2 | 1310 | 4 | 16 | 0.10 | 11.72 | 12.53 | 6 | 1828 | 1299 | 3 | 17 | 0.07 | 6.32 | 7.22 | 10 | 708 |
| n50-6-6-2-2a | 50 | 4 | 12 | 6 | 6 | 2 | 2 | 1464 | 4 | 8 | 8.93 | 204.42 | T.L. | 3766 | 7773 | 1343 | 4 | 8 | 6.03 | 116.22 | T.L. | 6328 | 9525 |
| n50-6-6-2-2b | 50 | 18 | 21 | 6 | 6 | 2 | 2 | 1077 | 4 | 8 | 0.49 | 36.99 | 918.38 | 1521 | 5318 | 1077 | 4 | 8 | 0.81 | 41.11 | 429.24 | 1575 | 4648 |
| n50-6-6-2-2c | 50 | 50 | 50 | 6 | 6 | 2 | 2 | 1065 | 4 | 8 | 0.03 | 18.63 | 19.44 | 11 | 2699 | 1065 | 4 | 8 | 0.09 | 15.32 | 16.89 | 9 | 941 |
| n50-8-8-2-2a | 50 | 3 | 12 | 8 | 8 | 2 | 2 | 1643 | 4 | 12 | 6.18 | 311.30 | T.L. | 3286 | 10281 | 1578 | 4 | 18 | 10.08 | 210.60 | T.L. | 6423 | 10364 |
| n50-8-8-2-2b | 50 | 18 | 21 | 8 | 8 | 2 | 2 | 1214 | 4 | 12 | 0.41 | 29.33 | 715.12 | 1333 | 5389 | 1214 | 4 | 12 | 0.75 | 29.05 | 219.31 | 1050 | 2429 |
| n50-8-8-2-2c | 50 | 50 | 50 | 8 | 8 | 2 | 2 | 1202 | 4 | 12 | 0.04 | 28.20 | 38.58 | 39 | 3330 | 1202 | 4 | 12 | 0.08 | 23.40 | 27.02 | 22 | 1360 |
| n50-8-8-8-8a | 50 | 3 | 3 | 8 | 8 | 8 | 8 | 1958 | 16 | 0 | 2.85 | 308.82 | T.L. | 4315 | 6957 | 1930 | 16 | 0 | 3.83 | 149.85 | T.L. | 8803 | 4218 |
| n50-8-8-8-8b | 50 | 18 | 18 | 8 | 8 | 8 | 8 | 1420 | 16 | 0 | 2.03 | 46.24 | 6275.34 | 10041 | 5821 | 1238 | 7 | 10 | 1.34 | 21.04 | 1515.88 | 4889 | 7362 |
| n50-8-8-8-8c | 50 | 50 | 50 | 8 | 8 | 8 | 8 | 1415 | 16 | 0 | 1.71 | 29.76 | 1451.36 | 3909 | 3556 | 1219 | 6 | 10 | 0.08 | 22.59 | 25.80 | 14 | 1203 |

Table 3.2: Branch-and-cut results

It is also important to remark how the number of changeovers increases in some instances when we consider lower penalties. This might happen because the small cost of the changeover makes their use more advantageous.

Table 3.3 displays the average number of cuts generated in the cutting-plane phase when running the branch-and-cut algorithm with the two proposed settings for the penalty parameters. For each number of nodes $n + 2$ and each problem class, we show the average number of found violated inequalities of types (3.14), (3.20), (3.21),(3.4)-(3.5)-(3.10), (3.22)-(3.23), and (3.24), in the corresponding columns. Columns *Total* report the average total number of cuts generated for each group of

| | | $M = 100$, $N = 10$ | | | | | | | $M = 10$, $N = 1$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n+2$ | *Class* | (3.14) | (3.20) | (3.21) | (3.4)-(3.5)-(3.10) | (3.22)-(3.23) | (3.24) | Total | (3.14) | (3.20) | (3.21) | (3.4)-(3.5)-(3.10) | (3.22)-(3.23) | (3.24) | Total |
| 16 | a | 167.50 | 3347.33 | 26.50 | 663.00 | 1.33 | 4.00 | 4209.67 | 121.33 | 196.00 | 27.17 | 36.67 | 1.50 | 2.83 | 385.50 |
| | b | 156.50 | 211.17 | 8.67 | 42.33 | 0.00 | 0.00 | 418.67 | 132.50 | 306.33 | 9.00 | 62.50 | 0.00 | 0.00 | 510.33 |
| | c | 137.00 | 31.00 | 2.67 | 12.00 | 0.17 | 0.00 | 182.83 | 51.50 | 32.33 | 3.50 | 5.67 | 0.00 | 0.00 | 93.00 |
| 20 | a | 285.80 | 7584.80 | 51.40 | 1049.80 | 3.20 | 8.40 | 8983.40 | 226.20 | 453.40 | 48.80 | 140.40 | 3.00 | 5.80 | 877.60 |
| | b | 277.20 | 345.40 | 10.20 | 113.20 | 1.20 | 4.20 | 751.40 | 175.20 | 277.20 | 16.20 | 102.80 | 2.60 | 4.40 | 578.40 |
| | c | 252.80 | 97.20 | 4.20 | 92.40 | 2.80 | 0.00 | 449.40 | 76.20 | 50.60 | 3.20 | 14.40 | 1.20 | 0.00 | 145.60 |
| 32 | a | 869.60 | 5903.20 | 115.00 | 1209.60 | 16.00 | 27.60 | 8141.00 | 713.60 | 1878.80 | 134.60 | 515.60 | 16.80 | 19.20 | 3278.60 |
| | b | 853.20 | 4095.80 | 26.60 | 1058.00 | 11.80 | 52.60 | 6098.00 | 616.80 | 1846.80 | 28.80 | 355.60 | 16.20 | 44.20 | 2908.40 |
| | c | 665.60 | 249.20 | 12.20 | 167.60 | 1.60 | 0.20 | 1096.40 | 199.60 | 270.00 | 14.00 | 22.40 | 12.20 | 2.20 | 520.40 |
| 40 | a | 1199.50 | 4138.25 | 131.25 | 791.00 | 12.00 | 32.00 | 6304.00 | 1024.00 | 3931.50 | 153.75 | 663.00 | 11.25 | 38.75 | 5822.25 |
| | b | 1322.25 | 11743.00 | 18.50 | 2761.50 | 11.50 | 42.50 | 15899.25 | 711.25 | 4292.75 | 20.75 | 1404.00 | 7.25 | 42.25 | 6478.25 |
| | c | 1157.50 | 962.25 | 13.75 | 1402.00 | 9.50 | 13.75 | 3558.75 | 297.00 | 353.50 | 20.75 | 79.50 | 9.00 | 20.75 | 780.50 |
| 50 | a | 2256.00 | 5127.67 | 469.67 | 407.33 | 11.33 | 65.00 | 8337.00 | 1716.33 | 5364.00 | 402.00 | 488.67 | 9.67 | 56.33 | 8037.00 |
| | b | 2231.67 | 2171.67 | 21.33 | 1047.33 | 18.00 | 19.33 | 5509.33 | 917.33 | 2979.67 | 30.00 | 854.00 | 13.67 | 18.33 | 4813.00 |
| | c | 2212.00 | 533.33 | 12.00 | 422.00 | 13.00 | 2.67 | 3195.00 | 439.00 | 609.33 | 14.67 | 88.67 | 13.67 | 2.67 | 1168.00 |

Table 3.3: Average number of cuts generated during the branch-and-cut algorithm

instances. We can see that, in general, the average number of cuts is considerably smaller when we use $M = 10$, $N = 1$, i.e., when we consider low penalizations. For both parameter settings, the most numerous cuts usually are the rounded capacity constraints (3.20), followed by (3.14), the constraints (3.4)-(3.5)-(3.10) on the route patterns for the two types of vehicles, and the multistar inequalities (3.21). On the other hand, only few violated $D_p$ (3.22)-(3.23) and comb (3.24) inequalities are found. This may be explained because, in the cutting plane phase, we only look for violated cuts of a family if we have not found any violated cut of the previously examined family. Moreover, we only have heuristic separation procedures for this two types of cuts, and they are separated only at the root node of the branch-and-cut algorithm. Anyway, we opt by separating them because they contribute to reduce the gaps in some particular cases.

Finally, we show in Figure 3.2 the optimal solutions for the three variants of the n16-1-1-2-2 instance with $M = 10$, $N = 1$. In that instance there are, at each depot, two vehicles of type 1, that must perform circular routes, and one vehicle of type 2, that must perform a route from one depot to the other one. As in the Figure 3.1, the dashed lines represent the routes for the vehicles of type 1 and the solid lines are the routes for the vehicles of type 2. In the instance of Class $a$ we can observe that all the available vehicles of type 1 are used and there are not changeovers. In the instance of Class $b$, due to the less restrictive capacity, only two vehicles of type 1 are needed, one departing from each depot, and there are not changeovers. In the instance of Class $b$, which is uncapacitated, a single vehicle of type 1 visits all the nodes, doing a changeover between nodes 5 and 12. In larger instances it is be possible to verify how the number of changeovers varies when the penalization setting is modified, as we mentioned above.

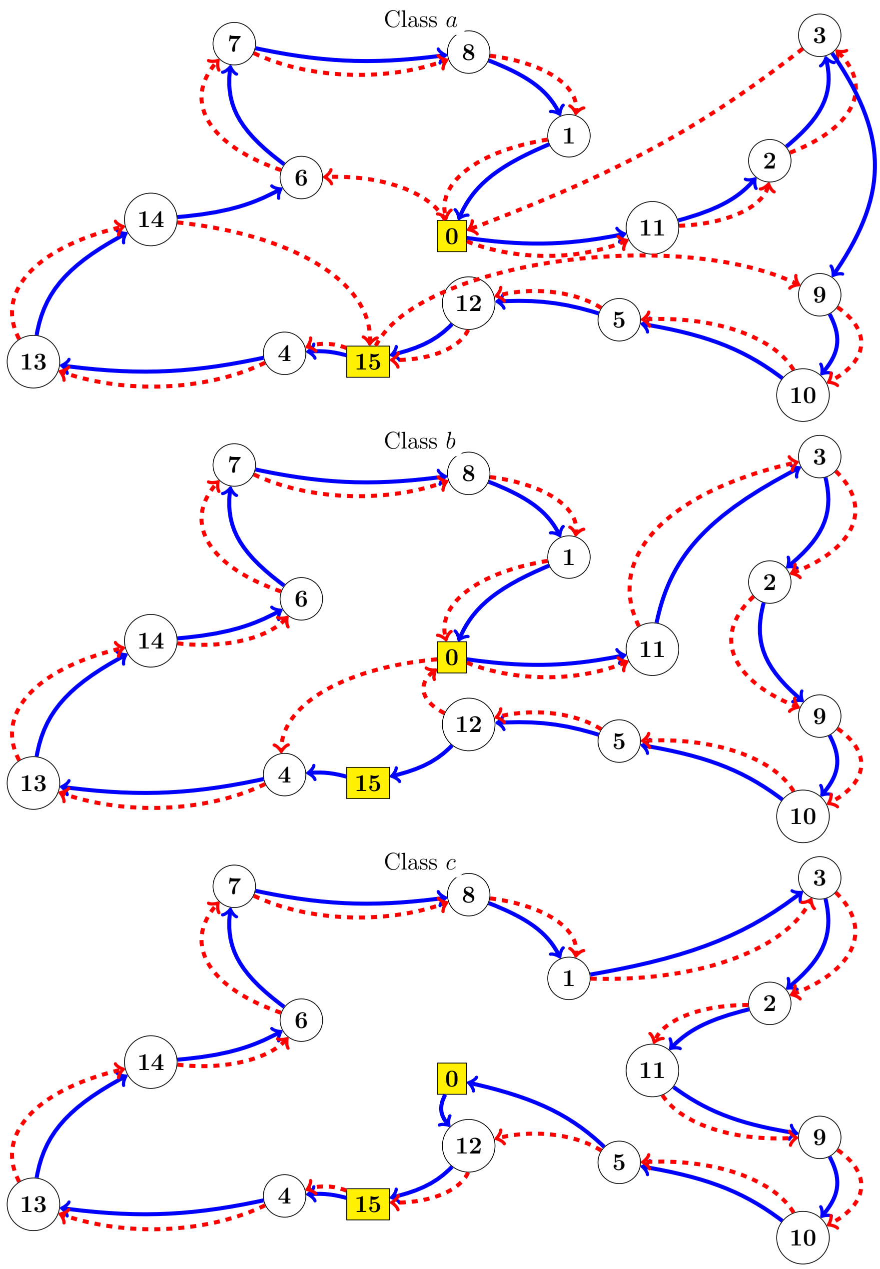



Figure 3.2: Solutions of the instance n16-1-1-2-2 with $M = 10$, $N = 1$

# Chapter 4

# The Driver and Vehicle Routing Problem

The research in the field of the routing problems focusses on finding routes for vehicles assuming that each one will be driven by a unique driver. As we have showed in Section 2.2, this is the case of the Vehicle Routing Problem (VRP), even when including also several side constraints like time windows or pickup and deliveries (see Toth & Vigo (2014) for problem variants). In the *Driver and Vehicle Routing Problem* this assumption is relaxed and a new problem is addressed where, moreover, vehicles and drivers have different types of routes. The requirement of starting and ending each route at the same location is quite natural for drivers, willing to return home, but it is unnecessary or unwanted for vehicles in some circumstances. This may be case, for example, when vehicles travel long distances while drivers or crews have limitations on their operating times and return to their home bases after finishing their work shifts. This situation may arise in long distance ground transportation (e.g., long distance cargo trains), in the maritime transport of freights, or in other different contexts as in humanitarian or military logistics (see Lam et al. (2015)).

The problem was inspired by the same situation than the VDSP, the planning characteristics of the local air traffic in the Canary Islands. In this case, instead of sequencing the tasks assigned to two different kinds of operators, as we made in the VDSP, we aim to study exclusively the underlying routing problem. A major difference between the VDSP and the DVRP is the number of visits to a node. In the scheduling problem each task must be performed by a unique operator and a unique machine. In the DVRP, some locations may be visited by several vehicles and drivers. Thus, the DVRP is defined as follows. We are given two depots, where a given number of vehicles and drivers are based, and a set of customers locations. Each customer must be served by a vehicle and a driver. Vehicles start their routes at their base depot and end at the other depot, while drivers must start and end their routes at their base depot. The vehicles have to be always lead by a driver, and drivers need a vehicle to move from one location to another, either driving themselves or as passengers. When there are more than one driver in a vehicle, any of them can lead the vehicle. The duration of a driver route is the time between the departure from and the arrival to the depot, and it includes the time driving and travelling as passenger. Moreover, drivers' routes cannot exceed a given time duration. Drivers can switch vehicles only at some given locations known as *exchange locations*, which are the only customer locations that can be visited by more than one vehicle. In other to make feasible the interaction between drivers and vehicles, their routes must be synchronized. The objective is to decide the routes of the vehicles and the drivers in order to minimize the total drivers' travel cost. Vehicles' costs are not considered because usually both, the drivers' and vehicles' costs, are dependent on the distance traveled, and therefore the objective function is just a function of that distance. Moreover, there are real-world situations in which human resources are considerably more costly than vehicles. However, the problem objective can be easily changed to also include vehicles' costs if necessary.

To illustrate the problem, Figure 4.1 depicts the optimal solution of an instance with 5 customers (nodes 1 to 5). Nodes 0 and 6 are the two depots, and vehicles' exchanges can take place only at node 5. There are four drivers and four vehicles available at each depot, the duration of all the travels between nodes is set to 1, and the maximum duration of the drivers' routes is set to 4. The figure shows that only two vehicles and two drivers from each depot are used in the optimal solution. The drivers make circular routes (dashed lines) starting and ending at the same depot, and vehicles' routes start at a depot and end at the other one (solid lines). We see that drivers exchange vehicles at node 5. For example, one of the drivers leaving from depot 0 makes the route 0-5-4-0. At node 5 he moves from one vehicle going form 0 to 6 to another going from 6 to 0 in order to be able to return to his base. The second driver leaving from depot 0 makes the route 0-5-2-0. The two vehicles leaving from 0 make the routes 0-5-1-6 and 0-5-3-6, respectively. As for depot 6, the drivers leaving it make the routes 6-5-1-6 and 6-5-3-6, while its vehicles' routes are 6-5-2-0 and 6-5-4-0. There is always a unique driver in each vehicle.

As we have mentioned, an important aspect of the DVRP is the synchronization between vehicles and drivers. We propose an *integrated* approach to solve the

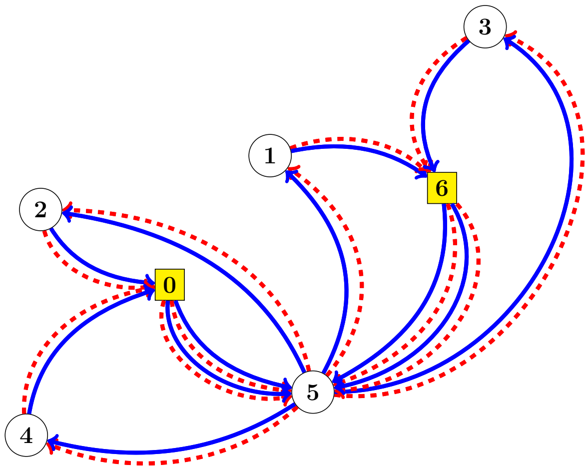



Figure 4.1: Solution example

DVRP, computing the routes for vehicles and drivers through a unique model. At first glance we could alternatively think about solving the problem in a *sequential* way, computing the drivers' and vehicles' routes in different steps. However, fixing the drivers' routes first, could easily create infeasible problems when determining vehicles' routes in a second stage. Take the instance whose optimal solution is shown in Figure 4.1 as example. Figure 4.2 shows the optimal driver's routes for that instance. Note that three drivers leave from depot 0 and one from depot 6. All drivers leave from their bases in different directions, meaning that they all drive a vehicle (i.e., none travels as passenger). Then there is only one vehicle going from depot 6 to 0, and the three drivers with base at 0 must take it to go back home together. However, we see in Figure 4.2 that two of those drivers return to 0 together, through the link $(5, 0)$, but the other uses the link $(4, 0)$. Therefore, it is impossible to find feasible vehicles' routes for these drivers.

As far as we know, the DVRP, has not been studied before. As we have remarked in Section 2.3, the problems that involve drivers and vehicles usually fall within the field of the *Vehicle-and-Crew Scheduling*. Drexl (2012) presents a survey and a classification scheme for vehicle routing problems with multiple synchronization constraints. The DVRP fits into the category he calls *movement synchronization en route*. This refers to non-autonomous vehicles that require autonomous vehicles to move in space, and that may join and separate at locations, different from the depot, that they visit during their route. Another problem that would fall into the same class is the *Joint Vehicle and Crew Routing and Scheduling Problem* studied

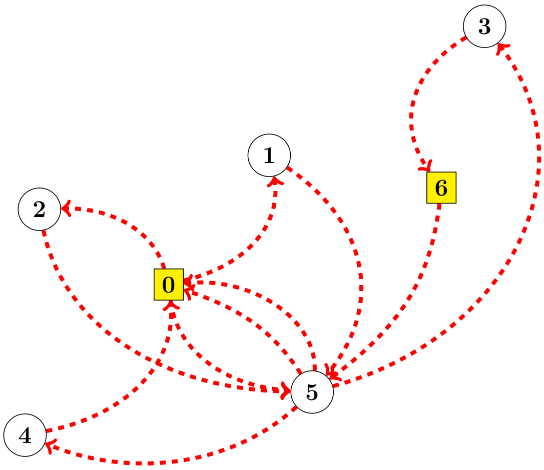



Figure 4.2: Optimal solution for drivers

by Lam et al. Lam et al. (2015). In that problem, motivated by applications in military and humanitarian logistics, vehicles lead by crews must service pickup and delivery requests before returning to the depot. Crews have limitations on their operating time, can switch vehicles at any given location, and can travel as passengers before and after their work shifts. The authors give a MIP formulation for the problem, with more than twenty types of decision variables, and propose a constraint programming approach to solve it.

Section 4.1 presents the notation, a MIP formulation, and several families of valid inequalities. Section 4.2 describes a branch-and-cut algorithm to solve the DVRP, and the separation procedures for the valid inequalities. And the Section 4.3 shows the results of our computational experiments for the VDRP.

## 4.1 Formulation and valid inequalities

In this section we formally describe the DVRP, give a mathematical model, and present several families of valid inequalities to strengthen its linear-programming relaxation. First, we set up the notation needed to describe the mathematical model.

We are given a set of customer locations $V_c = \{1, ...n\}$, two depots, and a set of drivers and vehicles. The $n$ customer locations are partitioned in two subsets, so that $V_c = V_r \cup V_e$. $V_r$ is the set of *regular* customer locations, those that must be visited exactly once by a vehicle and therefore, by at least a driver leading that vehicle. $V_e$ is the set of *exchange* locations, that is, those where drivers can exchange vehicles, and they must be visited at least once by a vehicle. $D = \{0, n+1\}$ is the set of depots. Let $G = (V, A)$ be a complete directed graph with vertex set $V = D \cup V_r \cup V_e$ and arc set $A = \{(i,j) : i, j \in V, i \neq j\}$. To refer to the set of arcs with tail in a set $S \subseteq V$ and head in $V \setminus S$, we use $\delta^+(S)$ instead of $\{(i,j) \in A : i \in S, j \notin S\}$, and we use $\delta^-(S)$ instead of $\delta^+(V \setminus S)$. Given two subsets of vertices $S$ and $S'$, $(S : S')$ represents the arc set $\{(i,j) \in A : i \in S, j \in S'\}$. The set of drivers and vehicles available at each depot $d \in D$ is denoted by $K_d$ and $L_d$ respectively. We define $K = \cup_{d \in D} K_d$ and $L = \cup_{d \in D} L_d$. The time needed to traverse an arc $(i,j)$ is represented by $t_{ij}$. Drivers' routes cannot exceed a given time limit $T$. No time limit is imposed to the vehicles' routes. There is a known cost $c_{ij}$ to pay when a driver traverses the arc $(i,j) \in A$. This cost may represent the salary of the driver and be related to the distance or the time needed to go from $i$ to $j$. Each driver must start and end its route at the same depot and, conversely, each vehicle must start its route at a depot and end it at the other one. The objective is to find feasible routes for vehicles and drivers in $G$ in order to serve each customer as required, and to minimize the total cost.

### 4.1.1 Mathematical model

To model the DVRP we use the following five decision variables. Variable $x_{ij}^k$ takes value 1 if the driver $k \in K$ traverses the arc $(i,j) \in A$, and 0 otherwise. Variable $y_{ij}^d$ represents the number of vehicles that start their routes from depot $d$ and traverse an arc $(i,j) \in A$. Variable $u_i^k$ takes value 1 if the driver $k \in K$ visits the customer $i$, and 0 otherwise. Variable $v_i^d$ represents the number of vehicles originating from depot $d \in D$ and visiting customer $i$. Finally, variable $w_i$ represents the (ordered) position in which customer $i$ is served. For simplicity of notation, we write $x^k(A')$ and $y^d(A')$ instead of $\sum_{(i,j)\in A'} x_{ij}^k$ and $\sum_{(i,j)\in A'} y_{ij}^d$, respectively, for each $A' \subset A$, $k \in K$ and $d \in D$.

Then a mathematical formulation for the DVRP can be described as follows. The objective function minimizes the cost of the drivers' routes:

$$\min \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{4.1}$$

The constraints can be divided in three blocks. A first block corresponds to the drivers' routes:

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) = u_i^k \qquad i \in V_c, k \in K \tag{4.2}$$

$$\sum_{k\in K} u_i^k \geq 1 \qquad i \in V_c \tag{4.3}$$

$$x^k(\delta^+(n+1)) = x^k(\delta^-(n+1)) = 0 \qquad k \in K_0 \tag{4.4}$$

$$x^k(\delta^+(0)) = x^k(\delta^-(0)) = 0 \qquad k \in K_{n+1} \tag{4.5}$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij}^k \leq T \qquad k \in K \tag{4.6}$$

$$w_j \geq w_i + x_{ij}^k - (n-1)(1 - x_{ij}^k) \qquad k \in K, (i,j) \in A,\ i,j \in V_c \tag{4.7}$$

$$w_i \geq 0 \qquad i \in V_c \tag{4.8}$$

$$x_{ij}^k \in \{0,1\} \qquad k \in K, (i,j) \in A \tag{4.9}$$

$$u_i^k \in \{0,1\} \qquad i \in V, k \in K. \tag{4.10}$$

Equations (4.2) determine if a driver $k$ visits the customer location $i$. Constraints (4.3) establish that at least one driver must visit each node. Equalities (4.4) and (4.5) guarantee that a driver from one depot will not end up at the other depot. Inequalities (4.6) limit to $T$ the maximum time that a driver can spend on a route. Constraints (4.7) avoid subtours involving only customers, and are known in the literature as Miller-Tucker-Zemlin inequalities (see Miller et al. (1960); Desrochers & Laporte (1991)). Constraints (4.8), (4.9), and (4.10) are the restrictions on the variables.

A second block of constraints corresponds to the vehicles' routes:

$$y^d(\delta^+(i)) = y^d(\delta^-(i)) = v_i^d \qquad i \in V_c, d \in D \tag{4.11}$$

$$\sum_{d\in D} v_i^d = 1 \qquad i \in V_r \tag{4.12}$$

$$\sum_{d\in D} v_i^d \geq 1 \qquad i \in V_e \tag{4.13}$$

$$y^0(\delta^+(n+1)) = y^0(\delta^-(0)) = 0 \tag{4.14}$$

$$y^{n+1}(\delta^+(0)) = y^{n+1}(\delta^-(n+1)) = 0 \tag{4.15}$$

$$1 \leq y^0(\delta^+(0)) = y^0(\delta^-(n+1)) \leq |L_0| \tag{4.16}$$

$$1 \leq y^{n+1}(\delta^+(n+1)) = y^{n+1}(\delta^-(0)) \leq |L_{n+1}| \tag{4.17}$$

$$w_j \geq w_i + y_{ij}^d - (n-1)(1 - y_{ij}^d) \qquad d \in D, (i,j) \in A,\ i,j \in V_c \tag{4.18}$$

$$y_{ij}^d \geq 0 \qquad d \in D, (i,j) \in A \tag{4.19}$$

$$v_i^d \geq 0 \qquad i \in V_c, d \in D. \tag{4.20}$$

Equations (4.11) define the number of vehicles leaving depot $d$ to visit a customer $i$. Constraints (4.12) and (4.13) impose the requirement on the number of visits on regular and exchange customers, respectively. Equalities (4.14) and (4.15) guarantee that the vehicles departing from one depot will arrive at the other one, while (4.16) and (4.17) limit the number of vehicles moving from one depot to the other. Subtours among customers are not allowed to vehicles, and this is ensured by the Miller-Tucker-Zemlin inequalities (4.18). Finally, since vehicle routes have no resource limitations, the variables represent a flow in a network, and integrability is unnecessary in (4.19) and (4.20).

A third block of constraints make compatible the two types of routes:

$$\sum_{k\in K} x_{ij}^k \geq \sum_{d\in D} y_{ij}^d \qquad (i,j)\in A \tag{4.21}$$

$$x_{ij}^k \leq \sum_{d\in D} y_{ij}^d \qquad k\in K, (i,j)\in A. \tag{4.22}$$

Inequalities (4.21) establish that each vehicle in the route has a driver, and inequalities (4.22) guarantee that each driver in the route is in a vehicle.

### 4.1.2 Valid inequalities

The mathematical model proposed in the previous section provides optimal (integer) solutions for small instances. In order to strengthen the linear programming relaxation of the model and to find optimal solutions for larger instances, we propose some additional inequalities.

#### 4.1.2.1 One-driver constraints

The following simple inequalities help to avoid fractional solutions with less than one driver leaving each depot:

$$\sum_{k\in K_d} x^k(\delta^+(d)) \geq 1 \qquad d\in D. \tag{4.23}$$

They ensure that at least one driver leaves each depot.

#### 4.1.2.2 Symmetry breaking constraints

Different solutions of the formulation in Section 4.1.1 describe the same routes because each driver is represented by a different set of variables while all drivers based in a depot have the same time limitation. Therefore, a permutation of drivers may change the values of the variables but not the solution value. This behavior is called *symmetricity* and it creates troubles to any Mathematical Programming solver. However, we did not find the way to model the driver routes with aggregated variables, as done for the vehicle routes. The reason is the time-limitation constraint.

A way to partially reduce the drawback of using disaggregated variables for the driver routes is through the following inequalities. Let us consider a depot $d \in D$, and let us label the drivers in $K_d$ with consecutive numbers, say $K_d = \{1, \ldots, m\}$. Then the inequalities

$$x^k(\delta^+(d)) \geq x^{k+1}(\delta^+(d)) \qquad k = 1, \ldots, m-1 \tag{4.24}$$

force to use drivers with smaller labels before than others, thus eliminating some of the symmetries.

#### 4.1.2.3 No-change constraints

Since a vehicle cannot return to the starting depot after it has initiated its route, each driver must visit an exchange customer and change vehicle. Although the model always ensures this behavior on integer solutions, we have found fractional solutions violating the following inequalities:

$$x^k(S : V_c \setminus S) \geq x^k(d : S) \qquad d \in D, k \in K_d, S \subseteq V_r. \tag{4.25}$$

Since these inequalities forbid a driver route to return to its depot after having visited regular customers only, we refer them as *no-change constraints*.

#### 4.1.2.4 Subtour elimination constraints

The described formulation uses the Miller-Tucker-Zemlin inequalities (4.7) and (4.18) to avoid subtours between customers. They have the advantage of providing a *compact* formulation, i.e., a model with a polynomial number of variables and constraints. However, it is known that the linear programming formulation is stronger when using the following exponential family of subtour elimination inequalities:

$$x^k(\delta^-(S)) \geq u_i^k \qquad k \in K, S \subseteq V_c, i \in S. \tag{4.26}$$

## 4.2 Solution approach

This section describes a branch-and-cut algorithm to solve the DVRP. As we have showed in Section 1.5, it combines a branch-and-bound method, exploring a decision tree, and a cutting-plane method that computes lower bounds by solving linear programming relaxations improved by valid inequalities. We now describe the main features of our algorithm.

### 4.2.1 Initial LP

The initial program to solve is the linear-programming relaxation of (4.1)–(4.20), plus the valid inequalities (4.23) and (4.24). Although (4.21) and (4.22) are a polynomial number of inequalities, we found more efficient in practice to add them dynamically to the relaxation when needed.

### 4.2.2 Cutting-plane phase

The inequalities that are not included in the initial LP are dynamically generated in the following order: (4.26), (4.25), (4.21), (4.22). At most 100 violated cuts of each type are added within each cutting plane iteration. After performing some preliminary computational experiments, we decided to separate inequalities (4.26) and (4.25) only at the root node. The reason is that their separation procedures are quite time consuming.

We describe next the separation procedure for each family of inequalities, given a fractional solution $(x^*, y^*, u^*, v^*, w^*)$.

#### 4.2.2.1 Separation of inequalities (4.25)

For each $d \in D$ and $k \in K_d$, the inequality (4.25) can be rewritten as

$$x^k(S : V_c \setminus S) + x^k(d : V_r \setminus S) \geq x^k(d : V_r)$$

This inequality can be separated in an exact and efficient way by solving a min-cut problem as follows. Let us consider two dummy nodes $s$ and $t$, and define a graph $G' = (V', A')$ where the node set is $V' = V_c \cup \{s, t\}$ and the arc set $A'$ includes all the arcs $(i, j) \in A$ with $i, j \in V_c$, each one with capacity $x^{*k}_{ij}$, a new arc $(s, i)$ for each $i \in V_r$, with capacity $x^{*k}_{di}$, and a new arc $(i, t)$ for each $i \in V_e$, with infinite capacity. Let $S' \subset V'$ be the optimal min-cut solution separating $s$ from $t$ in $G'$, with $d \in S'$. Note that the set $S'$ contains only the depot $d$ and some customers in $V_r$. If the capacity of the arcs leaving $S'$ is smaller than $x^{*k}(d : V_r)$, then $S = S' \setminus \{d\}$ defines a violated inequality (4.25) that must be added to the current linear program.

#### 4.2.2.2 Separation of inequalities (4.26)

For each $i \in V_c$ and each $k \in K$, the separation of (4.26) is identical to that of the classical subtour elimination constraint for the Travelling Salesman Problem. It consists of solving a min-cut problem between the base depot of driver $k$ and the customer $i$ on the network $G = (V, A)$, being $x^{k*}_{ij}$ the capacity of each arc $(i, j) \in A$ . Let $S \subset V$ be the optimal min-cut solution, with $i \in S$. If the capacity of the arcs entering $S$ is smaller than $u^{*k}_i$ then $S$ defines a violated inequality (4.26) that must be added to the current linear program.

#### 4.2.2.3 Separation of inequalities (4.21) and (4.22)

The violation of the inequalities (4.21) and (4.22) can be checked in polynomial time by simple enumeration.

## 4.3 Computational results

This section analyzes the results obtained in our computational experiments, using the branch-and-cut algorithm described above to solve the DVRP. This algorithm was coded in C++ and ran on a desktop computer with an Intel(R) Core(TM) i3-3240 CPU @ 4.40 GHz, 8GB RAM, Windows 7 Professional and CPLEX 12.7 as MILP solver. Default settings for CPLEX were used, except for the variable selection strategy that was set to "strong branching", and for the automatic generation of MIP cuts that was disabled. To solve the min-cut problems we used the routine included in the *Concorde TSP* software package by Applegate et al. (2003).

To the best of our knowledge, the DVRP has not been studied before, and we do not have real world or benchmark instances. Therefore we have created our own set of random instances, aiming to simulate ground transportation cases. In particular, we have generated instances with sizes $n+2 \in \{10, 15, 20, 25, 30\}$ and node coordinates in the square $[0, 100] \times [0, 100]$. The first and the last nodes generated are the depots 0 and $n+1$. We created five instances for each problem size, resulting in a set of 25 instances. The whole set is available from the authors upon request. The costs $c_{ij}$ are defined as the Euclidean distances between $i$ and $j$, and could be considered as the distances in kilometers between two cities. Let us consider a parameter $v$ representing the average velocity in kilometers per hour of a vehicle. In our computational experiments we set $v = 60$. Then, we defined $t_{ij} = c_{ij}/v + t_i$, being $t_i$ the time needed to serve customer $i$. We considered $t_i = 0.5$. As mentioned before, the parameter $T$ represents the time limitation on the drivers' routes. In these computational experiments, we used four different settings for $T$, called $T_A, T_B, T_C, T_D$, such that $T_A < T_B < T_C < T_D$. The value in $T_A$ for each instance is generally the tightest value of $T$ that allows to find a feasible solution. We have considered that the number of vehicles and drivers available at both depots is three, i.e. $|K_d| = |L_d| = 3$ for $d \in \{0, n+1\}$. We run each instance with a time limit of 2 hours.

To prove the contribution of the cuts used in our branch-and-cut scheme, we performed an experiment consisting of comparing three versions of the branch-and-cut algorithm that differ by the sets of valid inequalities used. The three branch-and-cut algorithms compared are: B&C0, solving (4.1)–(4.24), B&C1 solving (4.1)–(4.25), B&C2 solving (4.1)–(4.24) plus (4.26), and B&C3, solving (4.1)–(4.26). Table 4.1 shows the results obtained when applying these algorithm on the 25 instances with $|V_e| = 1$ and $T = T_C$. We chose these instances in order to carry out the experiment in a set of a priori tractable or easy cases, with just one exchange node and with loose time limitations on the drivers' routes. For each algorithm we report the total computing time in seconds ("T.L." means that the time limit was reached), and the percentage gap between the linear programming relaxation at the end of the root node and the optimal solution.

Note that computing times and gaps are large for the B&C0, and we can not solve instance with more than 15 nodes to proven optimality. The use of inequalities (4.25) reduces the computing times and the gaps in all the cases, and allows us to find the optimal solution for some instances with up to 25 nodes, and even one instance with 30 nodes. Incorporating constraints (4.26) to B&C0 has also positive effects (see B&C2). Yet, the best results are obtained by BC&3. Adding both (4.25) and (4.26) to BC&0 considerably reduces the computing times and the gaps, which are generally under 5% and are even zero in some cases. The algorithm with all the valid inequalities solves all instances to proven optimality in quite satisfactory computing times.

| | | | B&C0 | | B&C1 | | B&C2 | | B&C3 | |
|---|---|---|---|---|---|---|---|---|---|---|
| name | $n+2$ | $T_C$ | time | GAP | time | GAP | time | GAP | time | GAP |
| n10-1 | 10 | 8 | 34.24 | 33.66 | 2.28 | 16.59 | 7.89 | 23.78 | 0.73 | 5.08 |
| n10-2 | 10 | 7 | 2.00 | 9.25 | 0.23 | 4.11 | 0.31 | 3.60 | 0.08 | 0.00 |
| n10-3 | 10 | 7 | 20.11 | 33.59 | 0.87 | 1.68 | 4.59 | 25.26 | 0.14 | 1.08 |
| n10-4 | 10 | 7 | 10.53 | 18.75 | 0.41 | 11.46 | 0.53 | 8.94 | 0.27 | 0.18 |
| n10-5 | 10 | 7 | 18.52 | 34.83 | 1.00 | 2.53 | 5.52 | 24.82 | 0.12 | 1.02 |
| n15-1 | 15 | 9 | 532.42 | 30.37 | 129.03 | 27.79 | 17.55 | 13.75 | 0.97 | 6.94 |
| n15-2 | 15 | 9 | 87.30 | 24.14 | 0.36 | 0.00 | 21.86 | 20.94 | 0.23 | 0.00 |
| n15-3 | 15 | 9 | 138.09 | 17.27 | 5.49 | 0.96 | 6.80 | 13.90 | 0.19 | 0.00 |
| n15-4 | 15 | 9 | 1390.84 | 21.93 | 125.07 | 13.01 | 47.14 | 16.28 | 0.69 | 1.15 |
| n15-5 | 15 | 9 | T.L. | 37.37 | 17.67 | 10.10 | 130.56 | 20.34 | 0.59 | 0.18 |
| n20-1 | 20 | 10 | T.L. | 67.08 | 6877.34 | 13.64 | T.L. | 26.48 | 10.41 | 6.00 |
| n20-2 | 20 | 10 | T.L. | 24.66 | 1433.32 | 11.89 | 188.96 | 17.86 | 9.63 | 4.61 |
| n20-3 | 20 | 10 | T.L. | 56.83 | T.L. | 16.11 | 1886.21 | 19.34 | 32.84 | 4.56 |
| n20-4 | 20 | 10 | T.L. | 14.99 | 295.95 | 9.17 | 53.73 | 11.62 | 9.66 | 3.96 |
| n20-5 | 20 | 10 | T.L. | 23.79 | T.L. | 18.59 | 485.27 | 17.36 | 7.78 | 5.72 |
| n25-1 | 25 | 11 | T.L. | 47.54 | T.L. | 8.42 | 591.27 | 8.52 | 110.18 | 1.53 |
| n25-2 | 25 | 11 | T.L. | 61.30 | T.L. | 53.79 | 4722.38 | 17.51 | 167.98 | 6.53 |
| n25-3 | 25 | 11 | T.L. | 20.08 | 4033.76 | 8.88 | 60.92 | 10.12 | 9.87 | 2.93 |
| n25-4 | 25 | 11 | T.L. | 47.21 | 1479.67 | 13.65 | 581.18 | 13.39 | 22.28 | 1.70 |
| n25-5 | 25 | 11 | T.L. | 28.68 | 3852.10 | 14.87 | T.L. | 52.45 | 58.72 | 5.54 |
| n30-1 | 30 | 13 | T.L. | 61.22 | T.L. | 47.76 | T.L. | 47.14 | 135.27 | 6.11 |
| n30-2 | 30 | 13 | T.L. | 67.56 | T.L. | 50.41 | T.L. | 15.89 | 62.06 | 1.25 |
| n30-3 | 30 | 13 | T.L. | 43.40 | T.L. | 14.86 | T.L. | 22.47 | 38.78 | 3.46 |
| n30-4 | 30 | 13 | T.L. | 57.35 | T.L. | 48.39 | 778.87 | 12.41 | 146.89 | 7.75 |
| n30-5 | 30 | 13 | T.L. | 33.85 | 448.96 | 3.33 | T.L. | 14.57 | 18.35 | 1.03 |

Table 4.1: Different branch-and-cut algorithms on instances with $T = T_C$ and $|V_e| = 1$

To measure the impact of $T$ in the performance of the complete branch-and-cut algorithm (B&C3), Table 4.2 shows the results obtained when solving the 25 instances with $|V_e| = 1$ and the different settings for $T$. Again, each line in the table corresponds to a DVRP instance, but each instance is now solved with the four values for $T$. The first column shows the name of the instance, and the second column $n + 2$ shows the number of nodes including the depots. Columns $T$ show the values of this parameter. Columns $x^0$ and $x^{n+1}$ show the number of drivers from the two depots used in the optimal solution. Columns $GAP$ show the percentage ratio between the optimal objective value minus the value of the continuous relaxation at the end of the root node, and the optimal objective value. Columns *sol, time, r-time, nodes* and *cuts* display the optimal solution value of the problem, the computing time in seconds to solve the continuous relaxation, the total computing time in seconds, the number of explored nodes in the branch-and-cut tree, and the overall number of cuts added, respectively. As done before, if the time limit is reached we report "T.L." in the column *time*. For the instance named `n10-1`, the value of $T$ in the setting $T_A$ is different from the values of the other instances with 10 nodes because a smaller value would result in infeasibility.

In the subsequent $T$ configurations we increase that value in the same proportion used for the other instances.

When we use the tightest values $T = T_A$, the branch-and-cut algorithm does not find the optimal solution for the instances with 30 nodes. The difficulty of the problem is showed in the column GAP, with values close to 30% in general. Due to the restricted value of $T$, more than one driver from each depot is needed to serve all the customers. For the second set of values of the parameter, $T = T_B$, the gaps are smaller in almost all the instances, but unfortunately this is not reflected on the computing times and the algorithm has difficulties to solve instances with more than 20 nodes to optimality within the time limit. The last two groups of columns display the results for the loosest values $T = T_C$ and $T = T_D$. In these cases the improvement obtained is more remarkable, and for all the instances only one driver from each depot is needed in the optimal solution. When $T = T_C$, the gaps are all under 8%, and we can solve to optimality all the instances in competitive computing times. The situation is even better when $T = T_D$ where, in a high proportion of the instances, we obtain a gap of 0% and the total computing times are drastically reduced in many cases.

From Table 4.2, we can infer that, as expected, the parameter $T$ has a strong effect on the computational results. The problem gets harder to solve as $T$ gets tighter. Indeed, the need of using more than one driver from each depot makes the problem more difficult. This fact is easily observable if we compare the columns of gaps and computing times for the tightest and the loosest values of $T$.

In order not to overload Table 4.2 we do not show the number of vehicles leaving each depot in the optimal solution. That number is obviously one when the number of drivers leaving the depot is also one. It also coincides with the number of drivers in most other cases in Table 4.2, but there are some exceptions. For example, Figure 4.3 shows the optimal solution for the instance n10-3 with $T = T_A$ (i.e., $T = 5$). There are 10 nodes, being nodes 0 and 9 the two depots and node 8 the only exchange location. As seen in the figure, three drivers leave from each depot (dashed lines), but the number of vehicles that leave from depots 0 and 9 is one and three respectively (solid lines). This means that the three drivers from depot 0 leave in the unique vehicle departing from that depot. In the same way, the three drivers from depot 9 have to share one vehicle to be able to go back home. In those situations one driver leads the vehicle and the other ones travel as passengers. The issue of knowing which one is driving the vehicle is not relevant for the problem.

| | | $T_A$ | | | | | | | | | $T_B$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | $n+2$ | $T$ | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts | $T$ | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts |
| n10-1 | 10 | 6 | 2 | 2 | 34.83 | 652 | 7.22 | 0.12 | 296 | 541 | 7 | 1 | 1 | 8.53 | 442 | 0.41 | 0.12 | 14 | 266 |
| n10-2 | 10 | 5 | 2 | 2 | 38.32 | 486 | 17.43 | 0.12 | 771 | 515 | 6 | 1 | 1 | 0.00 | 292 | 0.06 | 0.06 | 1 | 170 |
| n10-3 | 10 | 5 | 3 | 3 | 56.69 | 987 | 272.77 | 0.14 | 15206 | 650 | 6 | 2 | 2 | 39.95 | 646 | 327.38 | 0.11 | 23669 | 586 |
| n10-4 | 10 | 5 | 2 | 2 | 33.21 | 610 | 4.52 | 0.09 | 211 | 414 | 6 | 2 | 2 | 28.15 | 534 | 8.00 | 0.08 | 674 | 375 |
| n10-5 | 10 | 5 | 2 | 2 | 35.39 | 595 | 6.91 | 0.14 | 309 | 532 | 6 | 1 | 1 | 2.88 | 365 | 0.59 | 0.09 | 29 | 289 |
| n15-1 | 15 | 6 | 2 | 2 | 22.35 | 454 | 34.43 | 0.64 | 353 | 1129 | 8 | 1 | 1 | 4.58 | 349 | 0.83 | 0.58 | 4 | 872 |
| n15-2 | 15 | 6 | 2 | 2 | 34.19 | 746 | 60.59 | 0.48 | 616 | 1193 | 8 | 1 | 1 | 1.67 | 414 | 1.06 | 0.30 | 20 | 509 |
| n15-3 | 15 | 6 | 2 | 2 | 33.95 | 660 | 50.44 | 0.47 | 663 | 1090 | 8 | 1 | 1 | 12.14 | 442 | 9.72 | 0.28 | 165 | 912 |
| n15-4 | 15 | 6 | 3 | 3 | 46.94 | 1094 | T.L. | 0.70 | 91794 | 1742 | 8 | 2 | 2 | 28.71 | 715 | 473.82 | 1.34 | 5871 | 1562 |
| n15-5 | 15 | 6 | 2 | 2 | 30.61 | 787 | 83.09 | 0.73 | 995 | 1347 | 8 | 2 | 2 | 37.31 | 751 | T.L. | 0.28 | 139200 | 1472 |
| n20-1 | 20 | 7 | 3 | 3 | 54.47 | 1285 | T.L. | 2.22 | 20292 | 3320 | 9 | 2 | 2 | 36.32 | 846 | T.L. | 2.54 | 21172 | 3244 |
| n20-2 | 20 | 7 | 2 | 2 | 29.72 | 666 | 4071.75 | 2.32 | 19308 | 2623 | 9 | 1 | 1 | 4.69 | 450 | 4.84 | 1.33 | 12 | 1155 |
| n20-3 | 20 | 7 | 2 | 2 | 32.15 | 845 | 6576.33 | 3.03 | 22774 | 2985 | 9 | 2 | 2 | 30.13 | 757 | T.L. | 3.67 | 25669 | 2909 |
| n20-4 | 20 | 7 | 2 | 2 | 23.40 | 600 | 997.44 | 1.75 | 5561 | 2103 | 9 | 2 | 2 | 25.14 | 580 | 6305.20 | 1.08 | 43657 | 2286 |
| n20-5 | 20 | 7 | 2 | 2 | 30.14 | 647 | 3623.64 | 3.29 | 15121 | 2577 | 9 | 2 | 2 | 22.32 | 538 | 1574.83 | 1.54 | 8714 | 2619 |
| n25-1 | 25 | 8 | 2 | 2 | 25.41 | 718 | 2002.73 | 8.36 | 3007 | 3342 | 10 | 2 | 2 | 29.36 | 711 | T.L. | 10.78 | 10592 | 4065 |
| n25-2 | 25 | 8 | 2 | 2 | 39.53 | 825 | T.L. | 19.72 | 6716 | 4963 | 10 | 2 | 2 | 33.71 | 716 | T.L. | 11.90 | 8570 | 4674 |
| n25-3 | 25 | 8 | 2 | 2 | 24.19 | 603 | 3463.28 | 6.54 | 5692 | 3517 | 10 | 2 | 2 | 22.07 | 555 | 4473.84 | 13.17 | 9606 | 3741 |
| n25-4 | 25 | 8 | 2 | 2 | 26.39 | 683 | 3530.49 | 13.32 | 4505 | 4050 | 10 | 2 | 2 | 30.85 | 680 | T.L. | 15.10 | 8099 | 4649 |
| n25-5 | 25 | 8 | 2 | 2 | 28.77 | 721 | 3720.39 | 8.58 | 6438 | 3953 | 10 | 2 | 2 | 32.62 | 703 | T.L. | 10.36 | 8920 | 4438 |
| n30-1 | 30 | 9 | 2 | 2 | 26.98 | 877 | T.L. | 34.63 | 3380 | 4925 | 12 | 2 | 2 | 31.76 | 864 | T.L. | 42.15 | 3098 | 7004 |
| n30-2 | 30 | 9 | 2 | 2 | 33.64 | 927 | T.L. | 106.47 | 878 | 7211 | 12 | 2 | 2 | 35.18 | 863 | T.L. | 149.62 | 1726 | 7970 |
| n30-3 | 30 | 9 | 2 | 2 | 31.68 | 812 | T.L. | 85.61 | 2209 | 6275 | 12 | 2 | 2 | 40.06 | 818 | T.L. | 10.59 | 7737 | 5418 |
| n30-4 | 30 | 9 | 2 | 2 | 29.10 | 756 | T.L. | 43.06 | 2534 | 6084 | 12 | 2 | 2 | 25.94 | 676 | T.L. | 15.49 | 4612 | 6687 |
| n30-5 | 30 | 9 | 2 | 2 | 29.67 | 756 | T.L. | 16.02 | 4293 | 4427 | 12 | 2 | 2 | 33.90 | 739 | T.L. | 6.93 | 12091 | 4769 |

| | $T_C$ | | | | | | | | | $T_D$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | $T$ | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts | $T$ | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts |
| n10-1 | 8 | 1 | 1 | 5.08 | 410 | 0.73 | 0.09 | 28 | 304 | 10 | 1 | 1 | 0.00 | 369 | 0.05 | 0.05 | 2 | 163 |
| n10-2 | 7 | 1 | 1 | 0.00 | 292 | 0.08 | 0.08 | 1 | 172 | 10 | 1 | 1 | 0.00 | 292 | 0.05 | 0.05 | 3 | 165 |
| n10-3 | 7 | 1 | 1 | 1.08 | 390 | 0.14 | 0.14 | 4 | 195 | 10 | 1 | 1 | 0.00 | 383 | 0.06 | 0.05 | 2 | 162 |
| n10-4 | 7 | 1 | 1 | 0.18 | 384 | 0.27 | 0.12 | 7 | 215 | 10 | 1 | 1 | 0.00 | 383 | 0.06 | 0.06 | 3 | 201 |
| n10-5 | 7 | 1 | 1 | 1.02 | 356 | 0.12 | 0.08 | 5 | 193 | 10 | 1 | 1 | 0.00 | 350 | 0.06 | 0.06 | 1 | 169 |
| n15-1 | 9 | 1 | 1 | 6.94 | 349 | 0.97 | 0.58 | 7 | 851 | 12 | 1 | 1 | 0.00 | 302 | 0.23 | 0.22 | 2 | 537 |
| n15-2 | 9 | 1 | 1 | 0.00 | 406 | 0.23 | 0.23 | 1 | 427 | 12 | 1 | 1 | 0.00 | 406 | 0.25 | 0.25 | 1 | 427 |
| n15-3 | 9 | 1 | 1 | 0.00 | 388 | 0.19 | 0.17 | 3 | 348 | 12 | 1 | 1 | 0.00 | 388 | 0.20 | 0.20 | 1 | 314 |
| n15-4 | 9 | 1 | 1 | 1.15 | 497 | 0.69 | 0.41 | 5 | 543 | 12 | 1 | 1 | 1.29 | 460 | 1.37 | 0.62 | 11 | 511 |
| n15-5 | 9 | 1 | 1 | 0.18 | 471 | 0.59 | 0.33 | 6 | 512 | 12 | 1 | 1 | 0.00 | 469 | 0.31 | 0.30 | 2 | 465 |
| n20-1 | 10 | 1 | 1 | 6.00 | 557 | 10.41 | 1.42 | 41 | 999 | 14 | 1 | 1 | 0.74 | 520 | 2.48 | 1.79 | 6 | 911 |
| n20-2 | 10 | 1 | 1 | 4.61 | 438 | 9.63 | 1.84 | 31 | 1244 | 14 | 1 | 1 | 2.85 | 399 | 2.06 | 0.70 | 10 | 860 |
| n20-3 | 10 | 1 | 1 | 4.56 | 540 | 32.84 | 1.50 | 148 | 1446 | 14 | 1 | 1 | 0.57 | 507 | 2.82 | 1.72 | 7 | 972 |
| n20-4 | 10 | 1 | 1 | 3.96 | 447 | 9.66 | 0.94 | 50 | 910 | 14 | 1 | 1 | 0.00 | 415 | 0.44 | 0.39 | 3 | 523 |
| n20-5 | 10 | 1 | 1 | 5.72 | 432 | 7.78 | 2.42 | 21 | 1201 | 14 | 1 | 1 | 8.31 | 409 | 13.71 | 2.26 | 58 | 1235 |
| n25-1 | 11 | 1 | 1 | 1.53 | 499 | 110.18 | 3.48 | 205 | 2918 | 16 | 1 | 1 | 0.00 | 483 | 4.57 | 3.49 | 4 | 1259 |
| n25-2 | 11 | 1 | 1 | 6.53 | 501 | 167.98 | 5.51 | 369 | 2822 | 16 | 1 | 1 | 5.36 | 483 | 52.12 | 5.88 | 95 | 1820 |
| n25-3 | 11 | 1 | 1 | 2.93 | 439 | 9.87 | 4.06 | 10 | 1602 | 16 | 1 | 1 | 0.00 | 405 | 1.31 | 1.14 | 3 | 676 |
| n25-4 | 11 | 1 | 1 | 1.70 | 469 | 22.28 | 8.80 | 20 | 1846 | 16 | 1 | 1 | 0.22 | 454 | 9.50 | 4.01 | 11 | 1416 |
| n25-5 | 11 | 1 | 1 | 5.54 | 491 | 58.72 | 3.98 | 90 | 2240 | 16 | 1 | 1 | 0.00 | 451 | 2.87 | 2.34 | 4 | 1038 |
| n30-1 | 13 | 1 | 1 | 6.11 | 615 | 135.27 | 8.27 | 224 | 2361 | 18 | 1 | 1 | 2.58 | 581 | 57.35 | 20.90 | 41 | 2403 |
| n30-2 | 13 | 1 | 1 | 1.25 | 560 | 62.06 | 13.71 | 72 | 2553 | 18 | 1 | 1 | 0.00 | 552 | 22.12 | 15.99 | 10 | 2601 |
| n30-3 | 13 | 1 | 1 | 3.46 | 506 | 38.78 | 11.65 | 28 | 2071 | 18 | 1 | 1 | 0.00 | 485 | 10.41 | 9.66 | 3 | 1960 |
| n30-4 | 13 | 1 | 1 | 7.75 | 536 | 146.89 | 25.82 | 70 | 3810 | 18 | 1 | 1 | 2.60 | 495 | 121.43 | 26.13 | 66 | 3198 |
| n30-5 | 13 | 1 | 1 | 1.03 | 493 | 18.35 | 5.38 | 12 | 1678 | 18 | 1 | 1 | 0.97 | 490 | 35.32 | 6.83 | 32 | 1751 |

Table 4.2: B&C3 on instances with different values of $T$ and $|V_e| = 1$

| | | | $\|V_e\|=2$ | | | | | | | | $\|V_e\|=4$ | | | | | | | | $\|V_e\|=\lfloor n/2\rfloor$ | | | | | | | | $\|V_e\|=$ n | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | $n+2$ | $T$ | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts | $x^0$ | $x^{n+1}$ | GAP | sol | time | r-time | nodes | cuts |
| n10-1 | 10 | 8 | 1 | 1 | 14.30 | 381 | 1.58 | 0.08 | 101 | 288 | 1 | 1 | 7.54 | 338 | 0.70 | 0.12 | 26 | 402 | 1 | 1 | 7.54 | 338 | 0.70 | 0.12 | 26 | 402 | 1 | 1 | 7.54 | 338 | 0.45 | 0.11 | 16 | 332 |
| n10-2 | 10 | 7 | 1 | 1 | 0.00 | 292 | 0.08 | 0.06 | 5 | 240 | 1 | 1 | 3.22 | 292 | 0.28 | 0.11 | 11 | 263 | 1 | 1 | 3.22 | 292 | 0.28 | 0.11 | 11 | 263 | 1 | 1 | 3.83 | 292 | 0.27 | 0.08 | 11 | 253 |
| n10-3 | 10 | 7 | 1 | 1 | 13.85 | 390 | 2.61 | 0.11 | 178 | 264 | 1 | 1 | 8.33 | 354 | 0.87 | 0.11 | 37 | 256 | 1 | 1 | 8.33 | 354 | 0.87 | 0.11 | 37 | 256 | 1 | 1 | 14.01 | 339 | 1.59 | 0.09 | 62 | 312 |
| n10-4 | 10 | 7 | 1 | 1 | 3.72 | 384 | 0.34 | 0.16 | 9 | 264 | 1 | 1 | 6.93 | 384 | 0.34 | 0.09 | 10 | 303 | 1 | 1 | 6.93 | 384 | 0.34 | 0.09 | 10 | 303 | 1 | 1 | 8.94 | 384 | 0.48 | 0.11 | 15 | 338 |
| n10-5 | 10 | 7 | 1 | 1 | 7.01 | 295 | 0.48 | 0.12 | 16 | 291 | 1 | 1 | 4.75 | 281 | 0.33 | 0.14 | 9 | 323 | 1 | 1 | 4.75 | 281 | 0.33 | 0.14 | 9 | 323 | 1 | 1 | 4.75 | 281 | 0.41 | 0.17 | 12 | 311 |
| n15-1 | 15 | 9 | 1 | 1 | 10.77 | 349 | 6.90 | 0.53 | 60 | 940 | 1 | 1 | 9.32 | 333 | 15.12 | 0.70 | 149 | 1157 | 1 | 1 | 9.60 | 333 | 7.82 | 0.47 | 87 | 913 | 1 | 1 | 9.60 | 333 | 3.68 | 0.53 | 42 | 1005 |
| n15-2 | 15 | 9 | 1 | 1 | 20.08 | 406 | 49.39 | 0.31 | 1223 | 843 | 1 | 1 | 14.39 | 379 | 6.58 | 0.36 | 123 | 653 | 1 | 1 | 14.29 | 375 | 14.38 | 0.31 | 242 | 723 | 1 | 1 | 14.40 | 375 | 5.73 | 0.28 | 133 | 580 |
| n15-3 | 15 | 9 | 1 | 1 | 0.72 | 359 | 0.39 | 0.23 | 5 | 480 | 1 | 1 | 2.16 | 359 | 2.25 | 0.47 | 42 | 522 | 1 | 1 | 6.82 | 359 | 2.03 | 0.39 | 34 | 550 | 1 | 1 | 6.95 | 359 | 2.03 | 0.37 | 37 | 581 |
| n15-4 | 15 | 9 | 1 | 1 | 1.86 | 458 | 0.76 | 0.41 | 7 | 599 | 1 | 1 | 4.20 | 451 | 3.23 | 0.45 | 45 | 703 | 1 | 1 | 5.49 | 446 | 1.90 | 0.41 | 21 | 720 | 1 | 1 | 6.71 | 446 | 1.93 | 0.33 | 26 | 691 |
| n15-5 | 15 | 9 | 1 | 1 | 13.74 | 454 | 23.52 | 0.64 | 287 | 944 | 1 | 1 | 11.83 | 431 | 11.01 | 0.59 | 137 | 1108 | 1 | 1 | 12.30 | 431 | 17.92 | 0.76 | 187 | 1115 | 1 | 1 | 12.94 | 431 | 13.65 | 0.55 | 155 | 1064 |
| n20-1 | 20 | 10 | 1 | 1 | 3.71 | 479 | 5.34 | 1.51 | 27 | 995 | 1 | 1 | 14.52 | 479 | 131.37 | 3.82 | 353 | 2538 | 1 | 1 | 14.34 | 478 | 57.94 | 3.40 | 244 | 2103 | 1 | 1 | 14.51 | 478 | 82.84 | 1.98 | 505 | 2215 |
| n20-2 | 20 | 10 | 1 | 1 | 8.26 | 438 | 20.14 | 1.62 | 88 | 1371 | 1 | 1 | 13.42 | 438 | 53.49 | 1.58 | 261 | 1654 | 1 | 1 | 16.10 | 438 | 112.43 | 1.15 | 687 | 1643 | 1 | 1 | 17.05 | 432 | 65.01 | 0.95 | 497 | 1436 |
| n20-3 | 20 | 10 | 1 | 1 | 11.02 | 499 | 69.53 | 0.95 | 468 | 1609 | 1 | 1 | 8.74 | 481 | 72.81 | 2.31 | 215 | 2440 | 1 | 1 | 6.35 | 468 | 24.21 | 1.87 | 106 | 1733 | 1 | 1 | 4.90 | 458 | 10.51 | 1.28 | 38 | 1849 |
| n20-4 | 20 | 10 | 1 | 1 | 8.09 | 447 | 80.56 | 1.31 | 351 | 1901 | 1 | 1 | 10.01 | 447 | 53.37 | 1.95 | 339 | 1445 | 1 | 1 | 8.66 | 438 | 44.49 | 1.34 | 229 | 1680 | 1 | 1 | 9.80 | 438 | 49.30 | 0.90 | 390 | 1244 |
| n20-5 | 20 | 10 | 1 | 1 | 5.56 | 410 | 8.11 | 1.70 | 30 | 1222 | 1 | 1 | 3.47 | 381 | 16.85 | 1.73 | 69 | 1352 | 1 | 1 | 6.30 | 381 | 10.39 | 1.26 | 52 | 1149 | 1 | 1 | 3.78 | 371 | 4.29 | 1.37 | 17 | 1063 |
| n25-1 | 25 | 11 | 1 | 1 | 3.17 | 499 | 27.63 | 6.97 | 58 | 1795 | 1 | 1 | 4.61 | 499 | 146.13 | 6.13 | 201 | 3203 | 1 | 1 | 8.34 | 498 | 209.85 | 6.41 | 474 | 2984 | 1 | 1 | 8.34 | 498 | 139.70 | 5.27 | 580 | 2516 |
| n25-2 | 25 | 11 | 1 | 1 | 6.56 | 474 | 276.12 | 7.75 | 532 | 3511 | 1 | 1 | 7.14 | 474 | 169.99 | 5.85 | 359 | 2684 | 1 | 1 | 10.20 | 467 | 333.47 | 9.42 | 547 | 3081 | 1 | 1 | 11.51 | 467 | 124.99 | 4.09 | 279 | 2491 |
| n25-3 | 25 | 11 | 1 | 1 | 7.23 | 439 | 330.29 | 4.23 | 632 | 3335 | 1 | 1 | 8.20 | 439 | 120.73 | 4.40 | 238 | 2312 | 1 | 1 | 7.09 | 427 | 40.79 | 3.81 | 76 | 1984 | 1 | 1 | 7.60 | 427 | 53.60 | 3.90 | 152 | 2071 |
| n25-4 | 25 | 11 | 1 | 1 | 7.31 | 458 | 140.37 | 7.44 | 220 | 2203 | 1 | 1 | 8.06 | 458 | 212.05 | 6.04 | 441 | 2742 | 1 | 1 | 6.71 | 445 | 31.36 | 4.65 | 66 | 2488 | 1 | 1 | 8.74 | 445 | 173.33 | 6.61 | 279 | 3219 |
| n25-5 | 25 | 11 | 1 | 1 | 10.85 | 481 | 904.15 | 7.35 | 1217 | 4194 | 1 | 1 | 14.93 | 481 | 365.60 | 6.49 | 757 | 3743 | 1 | 1 | 13.54 | 456 | 759.29 | 5.80 | 2215 | 3090 | 1 | 1 | 13.71 | 456 | 893.20 | 3.56 | 2513 | 3126 |
| n30-1 | 30 | 13 | 1 | 1 | 6.17 | 554 | 323.64 | 31.68 | 239 | 4131 | 1 | 1 | 6.10 | 541 | 281.16 | 28.22 | 113 | 4043 | 1 | 1 | 7.14 | 539 | 961.48 | 23.60 | 654 | 5401 | 1 | 1 | 7.33 | 539 | 321.28 | 11.78 | 313 | 4814 |
| n30-2 | 30 | 13 | 1 | 1 | 5.36 | 560 | 1008.19 | 19.03 | 1066 | 4946 | 1 | 1 | 5.68 | 560 | 996.92 | 36.30 | 722 | 4402 | 1 | 1 | 14.23 | 557 | 3161.44 | 63.73 | 2215 | 5570 | 1 | 1 | 15.14 | 555 | 5050.03 | 85.15 | 2454 | 7043 |
| n30-3 | 30 | 13 | 1 | 1 | 11.83 | 498 | 6787.17 | 13.46 | 14005 | 3308 | 1 | 1 | 2.43 | 450 | 174.99 | 20.17 | 167 | 2768 | 1 | 1 | 9.75 | 441 | 1039.59 | 17.33 | 806 | 5359 | 1 | 1 | 9.54 | 440 | 302.19 | 18.70 | 225 | 4500 |
| n30-4 | 30 | 13 | 1 | 1 | 9.35 | 536 | 974.27 | 25.71 | 487 | 5412 | 1 | 1 | 11.85 | 536 | 637.90 | 21.54 | 484 | 4490 | 1 | 1 | 10.17 | 523 | 1175.62 | 27.47 | 634 | 5337 | 1 | 1 | 10.24 | 523 | 358.41 | 24.73 | 362 | 4343 |
| n30-5 | 30 | 13 | 1 | 1 | 10.49 | 493 | 1767.62 | 12.21 | 3585 | 3557 | 1 | 1 | 14.00 | 493 | 4498.48 | 16.46 | 6014 | 4103 | 1 | 1 | 7.22 | 457 | 247.26 | 16.88 | 198 | 3441 | 1 | 1 | 7.28 | 457 | 418.58 | 18.58 | 610 | 3430 |

Table 4.3: B&C3 on instances with $T = T_C$ and different values of $|V_e|$

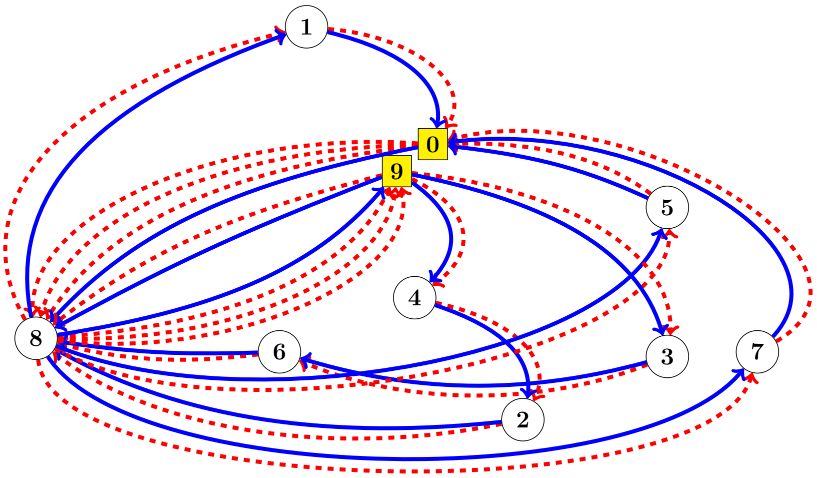



Figure 4.3: Optimal solution for n10-3 with $|V_e| = 1$ and $T = 5$

Finally, we also analyzed the effect of the number of *exchange* nodes allowed in each instance. Table 4.3 shows the results of the experiments when we consider $|V_e| \in \{2, 4, \lfloor n/2 \rfloor, n\}$. In order to not overload the table, we just show the results for $T = T_C$.

A first observation of Table 4.3 is that in many cases (12 out of 25) we obtain the best gaps when we consider a smaller number of exchange nodes, i.e., $|V_e| = 2$. However, in terms of computing times, there is not a clear tendency. The differences showed in the four groups of columns are not so relevant as if we compare these results with the information showed in the columns under $T_C$ of Table 4.2, where the gaps and computing times are smaller in almost all the cases. As a conclusion we can say that the number of exchange nodes has a minor impact on the performance of the branch-and-cut algorithm when compared to the impact of $T$.

# Chapter 5

# A heuristic approach to the DVRP

Section 4.3 showed that it is hard to solve the DVRP when we apply an exact algorithm over instances with particular characteristics and more than 25 nodes, using a restrictive time to limit the routes of the drivers. In order to obtain feasible solutions for larger instances of the DVRP, we propose heuristic strategies to address the problem. Moreover, the objective is to obtain these solutions spending the shortest possible time.

We cannot solve the DVRP only for the drivers and then to build the vehicle routes, without imposing any condition. This sequential approach could produce infeasible solutions, as we have explained at the beginning of Chapter 4, with the Figures 4.1 and 4.2. We propose two sequential heuristic algorithms that obtain suitable routes for the drivers in a first step, and then add to these routes the vehicles' routes.

For the first step of the heuristic we propose a mathematical model only with the variables related to the drivers, and with some conditions inferred from the computational results showed in Section 4.3. This model is solved using a branch-and-cut algorithm. Since in the DVRP we only want to minimize the cost of the drivers' routes, when these are computed, we can apply a second step to obtain the vehicles' routes, imposing that they must be related to the drivers' routes in a correct way.

The second heuristic strategy proposed is to add to the first heuristic algorithm a previous assignment of the *regular customers* to the depots.

This chapter is organized as follows. In Section 5.1, a mathematical formulation is proposed to obtain the routes of the drivers, with valid inequalities that strengthen the formulation. Section 5.2 proposes two heuristic algorithms used to obtain feasible solutions for the DVRP. The results of the computational experiments are showed in Section 5.3.

## 5.1 Formulation and valid inequalities for the drivers' routes problem

In this section, we introduce the mathematical model used to obtain a feasible solution for the drivers' routes of the DVRP. In the mathematical model some inequalities have been included to impose the conditions that make possible to solve the problem in a sequential way. We also propose some valid inequalities, used to strengthen the mathematical formulation.

To mathematically model the drivers' routes, we use the same notation than in Section 4.1, that is, $V_c = \{1, \ldots, n\}$ is the set of customer locations, and there are two depots, $D = \{0, n+1\}$. The $n$ customer locations are partitioned in two subsets, so that $V_c = V_r \cup V_e$. $V_r$ is the set of *regular* customer locations, and $V_e$ is the set of *exchange* locations. In this case, both types of locations must be visited by at least one driver. Let $G = (V, A)$ be a complete directed graph with vertex set $V = D \cup V_r \cup V_e$ and arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. The set of drivers available at each depot $d \in D$ is denoted by $K_d$. We define $K = \cup_{d \in D} K_d$. Each arc $(i, j) \in A$ has an associated traversing time $t_{ij}$ and a cost $c_{ij}$. Drivers' routes cannot exceed a given time limit $T$.

### 5.1.1 Mathematical model

In the mathematical model proposed to obtain the routes of the drivers, we use the following variables. Variable $x_{ij}^k$ takes value 1 if the driver $k \in K$ traverses the arc $(i, j) \in A$, and 0 otherwise. Variable $u_i^k$ takes value 1 if the driver $k \in K$ visits the customer $i$, and 0 otherwise. As we have mentioned before, we write $x^k(A')$ instead of $\sum_{(i,j) \in A'} x_{ij}^k$, for each $A' \subset A$, $k \in K$ and $d \in D$.

Then a mathematical formulation to obtain feasible driver routes can be described as follows. The objective function minimizes the cost of the drivers' routes:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \tag{5.1}$$

s.t.:

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) = u_i^k \qquad i \in V_c, k \in K \tag{5.2}$$

$$\sum_{k \in K} u_i^k \geq 1 \qquad i \in V_c \tag{5.3}$$

$$x^k(\delta^+(n+1)) = x^k(\delta^-(n+1)) = 0 \qquad k \in K_0 \tag{5.4}$$

$$x^k(\delta^+(0)) = x^k(\delta^-(0)) = 0 \qquad k \in K_{n+1} \tag{5.5}$$

$$\sum_{(i,j) \in A} t_{ij} x_{ij}^k \leq T \qquad k \in K \tag{5.6}$$

$$\sum_{k \in K_0} x^k(\delta^+(0)) = \sum_{k \in K_{n+1}} x^k(\delta^+(n+1)) \tag{5.7}$$

$$x^k(\delta^-(S)) \geq u_i^k \qquad k \in K, S \subseteq V_c, i \in S \tag{5.8}$$

$$x_{ij}^k + x_{lj}^{k'} \leq 1 \qquad j \in V_r, i, l \in V, d \in D, \; k, k' \in K_d, k \neq k' \tag{5.9}$$

$$u_i^k + u_i^{k'} \leq 1 \qquad i \in V_r, k \in K_d, k' \in K_{d'}, \; d \neq d' \tag{5.10}$$

$$\sum_{i \in V_e} u_i^k \geq x^k(\delta^+(d)) \qquad k \in K, d \in D \tag{5.11}$$

$$x_{ij}^k \in \{0,1\} \qquad k \in K, (i,j) \in A \tag{5.12}$$

$$u_i^k \in \{0,1\} \qquad i \in V, k \in K. \tag{5.13}$$

Equations (5.2) determine if a driver $k$ visits the customer location $i$. Constraints (5.3) establish that at least one driver must visit each node. Equalities (5.4) and (5.5) guarantee that a driver from one depot will not end up at the other depot. Inequalities (5.6) limit to $T$ the maximum time that a driver can spend on a route. Constraints (5.7) impose that the number of drivers that starts the route at the depot 0 is the same that starts the route at the depot $n+1$. Constraints (5.8) avoid subtours between customers. Inequalities (5.9) impose that if two drivers from the same depot arrive to a regular customer, then they cannot come from different customers. Constraints (5.10) avoid that two drivers from different depots visit a regular customer. Together, (5.9) and (5.10) make impossible that a regular customer is visited by more than one vehicle. Inequalities (5.11) ensure that all the drivers visit at least an exchange location. Constraints (5.12) and (5.13) are the restrictions on the variables.

The inequalities (5.7), (5.9), (5.10) and (5.11) have been added to the block of constraints for the drivers proposed in Section 4.1.1. This constraints are used to ensure that the routes of the drivers obtained solving this mathematical model are feasible drivers' routes for the DVRP. The inequalities (5.7) have been added because observing the results obtained in Section 4.3, the optimal solutions for all the instances have the particular feature that the number of drivers that starts the route at a depot matches the number of drivers that starts the route at the other one. This assumption avoids the situation that we mentioned at the beginning of Chapter 4, described with Figures 4.1 and 4.2. Moreover, the inequalities (5.8) become essential to define the problem, because we have removed the inequalities (4.7) that we used to avoid subtours in the mathematical model proposed in Section 4.1.1.

### 5.1.2 Valid inequalities

As in the model proposed in Section 4.1.1, in order to strengthen the linear programming relaxation of the model and to find solutions for larger instances, we propose some valid inequalities. These inequalities have been previously introduced in Section 4.1.2 and are the following:

- inequalities (4.23), to ensure that at least one driver starts a route at each depot,
- inequalities (4.24), to avoid some symmetric solutions,
- inequalities (4.25), to impose that a driver cannot return to the depot, after starting the route, without visiting an exchange customer and changing vehicle.

## 5.2 Solution approach

This section describes two heuristic algorithms, called Heuristic 1 and 2, to obtain feasible solutions for the DVRP. Both algorithms are based on the same branch-and-cut algorithm to obtain a feasible drivers' solution for the DVRP. In another step, an iterative process is used to build the vehicle routes based on the routes obtained for the drivers. As we have mentioned before, the difference between both algorithms consists of a previous assignment of the customers that we make in Heuristic 2.

### 5.2.1 Heuristic 1

The first heuristic algorithm has the following scheme:

- **Step 1:** To solve the mathematical model (5.1)-(5.13) with the valid inequalities (4.23), (4.24), and (4.25).
- **Step 2:** To build the vehicles' routes using the process described in Section 5.2.1.3.

The branch-and-cut algorithm described in the Step 1 has the following characteristics.

#### 5.2.1.1 Initial LP

The initial program to solve is the linear-programming relaxation of (5.1)–(5.7), (5.11)–(5.13) and the valid inequalities (4.23) and (4.24). We found more efficient in practice to add the inequalities (5.9) and (5.10) dynamically to the relaxation when needed.

#### 5.2.1.2 Cutting-plane phase

The inequalities that are not included in the initial LP are dynamically generated in the following order: (5.8), (4.25), (5.9), and then (5.10). At most 100 violated cuts of each type are added within each cutting plane iteration. The separation procedure of the inequalities (4.25) is only executed at the root node, based on preliminary computational experiments, because the separation procedure is quite time consuming.

The violation of the inequalities (5.9) and (5.10) can be checked in polynomial time by simple enumeration. The separation procedures of (4.25) and (5.8) were described in Sections 4.2.2.1 and 4.2.2.2, respectively.

#### 5.2.1.3 Building vehicles' routes

The building procedure of the vehicles' routes is made considering the drivers' routes obtained in a first step. We must establish that each driver is in a vehicle, driving it or as passenger, and also that each vehicle is driven by a driver, i.e., at least a driver is in the vehicle. Moreover, the route of a vehicle that starts at a depot ends at the other one. The set of vehicles available at each depot $d \in D$ is denoted by $L_d$, and we define $L = \cup_{d \in D} L_d$. While each driver is individually differentiated, in the case of the vehicles, we only differentiate the depot where they belong.

We design an iterative process that works as follows. If an arc $(d, i) \in \delta^+(d)$ is traversed by one or more drivers $k \in K_d$, then a vehicle departs from depot $d$ using the same arc. Then, the vehicle follows the route of the driver (or drivers, if some of them go into the vehicle as passenger), visiting the same customers until a exchange location is reached. From that point, the vehicle leaves the exchange location with a driver (or drivers) $k' \in K_{d'}$ and follows the route of the driver, until the depot is reached.

When all arcs $(d, i) \in \delta^+(d)$ have been processed this way, the routes for drivers and vehicles that we have obtained could not represent a feasible solution for the DVRP. In some cases, it is needed that more than a vehicle leave the depot by the same arc. This situation can only happen when the arc links a depot and an exchange location, as established in the mathematical formulation of the DVRP in Section 4.1.1.

To detect this particular cases, when we have finished the iterative process described above, we check if the number of vehicles departing from depot $d$ coincides with the number of drivers arriving to depot $d'$ through different arcs, i.e, using different vehicles. If yes, we already have a feasible DVRP solution. If not, it means that there is some driver $k' \in K_{d'}$, going from an exchange location to $d'$ without a vehicle. To correct this situation, an extra vehicle is taken from $d$ to that exchange location, and then from there to $d'$ following the route of driver $k'$.

With this strategy we are able to ensure that the routes that have been built for drivers and vehicles form a feasible solution for the DVRP.

### 5.2.2 Heuristic 2

In the second heuristic used, called Heuristic 2, we solve the same mathematical model used in 5.2.1, but the regular customers are previously assigned to a depot. Thus, the scheme of the Heuristic 2 is the following:

- **Step 0:** To assign the nodes to the depots. To this end, we establish that $u_i^k = 0$, with $i \in V_r, k \in K_d$, if $c_{id} > c_{id'}$, $d, d' \in D$, i.e., we assign each regular customer to the nearest depot.
- **Step 1:** To solve the mathematical model (5.1)-(5.13) with the valid inequalities (4.23), (4.24), and (4.25).
- **Step 2:** To build the vehicles' routes using the process described in Section 5.2.1.3.

## 5.3 Computational results

In this section, the results obtained in our computational experiments are showed and examined. We show the performance of the heuristic algorithms described above to solve DVRP instances. This algorithms were coded in C++ and ran on a desktop computer with an Intel(R) Core(TM) i3-3240 CPU @ 4.40 GHz, 8GB RAM, Windows 7 Professional and CPLEX 12.7 as MIP solver. To solve the min-cut problems we used the routine included in the *Concorde TSP* software package by Applegate et al. (2003).

In Section 4.3, we explained how the DVRP instances are created. For these heuristic approaches to solve the DVRP, we create larger DVRP instances with different characteristics. In this case, the coordinates of the regular customers $i \in V_r$ are generated randomly in the square $[0, 100] \times [0, 100]$, the coordinates of the exchange locations, $i \in V_e$, in $[40, 60] \times [0, 100]$, and the first and the second depot are located in $[0, 20] \times [0, 100]$ and $[80, 100] \times [0, 100]$, respectively. This strategy to generate the nodes is followed thinking about a ground transportation application where two cities (the depots) are placed at a long distance. The drivers and the vehicles start their workday at these cities. The places where the drivers can make a change of vehicle, the exchange locations, are located in a place at a similar distance between the depots. We create 10 instances with $n+2 \in \{50, 100\}$ nodes, five of each size.

The different parameters keep the definitions and the values proposed in Section 4.3, for the costs, $c_{ij}$, and the times, $t_{ij}$, of each arc. The number of vehicles and drivers available at both depots is three, i.e., $|K_d| = |L_d| = 3$ for $d \in \{0, n+1\}$. Our objective is to find solutions for the instances as quick as possible, thus the time limit used in this case is 10 minutes. For all the experiments performed, we have considered that there is only one exchange location, i.e., $|V_e| = 1$.

The first element studied is the performance of the Heuristic 1, described in Section 5.2.1, when it is applied to the instances of the DVRP in Domínguez-Martín et al. (2018). The objective is to obtain solutions quicker and to measure their quality. The results in Table 5.1 should be compared with the results presented in Table 4.2, for this reason, the results showed in this table are organized in the following columns: *name* shows the name of the instance, column *n+2* represents the number of nodes including the depots, columns $T_A, T_B, T_C$ and $T_D$ are the same values used in Section 4.3 for $T$, the upper bound for the total time of a driver route. Columns *SolE* and *TimeE* show the solution and the computing time in seconds, respectively, given in Table 4.2. *SolH1* and *TimeH1* represent the solution and the computing times in seconds, respectively, obtained using the Heuristic 1, proposed in Section 5.2.1. Finally, the column *GAP* shows the difference between the solution *SolE* and *SolH1* in percentage. Note that when "T.L." is reported in the column *TimeE*, the time limit reached is 2 hours, while in the column *TimeH1*, the time limit corresponds to 10 minutes.

From Table 5.1, we can infer that Heuristic 1 shows a good performance over the instances proposed in Domínguez-Martín et al. (2018). When the optimal solution is obtained using the branch-and-cut algorithm B&C3 presented in Section 4.3, we also obtain the same solution using Heuristic 1. Moreover, in almost all the cases that "T.L." is reported for B&C3, we obtain a better solution using Heuristic 1. Another important feature showed in the Table 5.1 is the computing time reduction using Heuristic 1 in comparison with the times obtained using B&C3. The reduction is more remarkable on the most restrictive instances, using the values $T_A$ and $T_B$ of the parameter $T$. For example, in the instances of 25 nodes with $T_A$, in almost all the cases, the computing time is reduced from 1 hour, using the exact method, to less than 3 minutes with the Heuristic 1.

As the Heuristic 1 shows a good performance when we solve the instances of the benchmark proposed in Domínguez-Martín et al. (2018), new instances of 50 and 100 nodes have been created to check if we are able to obtain solutions using this heuristic algorithm. The values of the parameter $T$ have been established as 20 for the instances of 50 nodes, and 40 for the instances of 100 nodes. In Table 5.2 we show the results obtained when the Heuristic 1 is applied over the instances of 50 nodes.

The first conclusion that we can infer from Table 5.2 is that the method followed to create the larger instances makes easier to solve the problem using the exact algorithm B&C3, proposed in Section 4.3. In this case, we achieve to solve with optimality instances of 50 nodes, while with the method used in Section 4.3 to create the instances, we only solve with optimality instances with up to 25 nodes when $T$ is tight, and with up to 30 when $T$ is less restrictive. Moreover, using Heuristic 1, we obtain the optimal solution for all the instances of 50 nodes, and with a significant reduction of the computing time spent to solve the problem. In general, we can say that the time reduction obtained is approximately about 90%.

The instances of 100 nodes are not included in Table 5.2, because we do not find feasible solutions for the DVRP, using Heuristic 1 or B&C3. To find feasible solutions for larger instances, we propose the Heuristic 2, described in Section 5.2.2. This heuristic makes a previous assignment of the customers to the depots to make easier to solve the problem. In Table 5.3, we compare the results obtained

| name | $n+2$ | $T_A$ | SolE | TimeE | SolH1 | TimeH1 | GAP | $T_B$ | SolE | TimeE | SolH1 | TimeH1 | GAP | $T_C$ | SolE | TimeE | SolH1 | TimeH1 | GAP | $T_D$ | SolE | TimeE | SolH1 | TimeH1 | GAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n10-1 | 10 | 6 | 652 | 7.22 | 652 | 1.15 | 0.00 | 7 | 442 | 0.41 | 442 | 0.51 | 0.00 | 8 | 410 | 0.73 | 410 | 0.42 | 0.00 | 10 | 369 | 0.05 | 369 | 0.08 | 0.00 |
| n10-2 | 10 | 5 | 486 | 17.43 | 486 | 0.37 | 0.00 | 6 | 292 | 0.06 | 292 | 0.12 | 0.00 | 7 | 292 | 0.08 | 292 | 0.09 | 0.00 | 10 | 292 | 0.05 | 292 | 0.09 | 0.00 |
| n10-3 | 10 | 5 | 987 | 272.77 | 987 | 30.72 | 0.00 | 6 | 646 | 327.38 | 646 | 0.87 | 0.00 | 7 | 390 | 0.14 | 390 | 0.67 | 0.00 | 10 | 383 | 0.06 | 383 | 0.03 | 0.00 |
| n10-4 | 10 | 5 | 610 | 4.52 | 610 | 1.95 | 0.00 | 6 | 534 | 8.00 | 534 | 0.90 | 0.00 | 7 | 384 | 0.27 | 384 | 0.14 | 0.00 | 10 | 383 | 0.06 | 383 | 0.09 | 0.00 |
| n10-5 | 10 | 5 | 595 | 6.91 | 595 | 1.28 | 0.00 | 6 | 365 | 0.59 | 365 | 0.64 | 0.00 | 7 | 356 | 0.12 | 356 | 0.27 | 0.00 | 10 | 350 | 0.06 | 350 | 0.06 | 0.00 |
| n15-1 | 15 | 6 | 454 | 34.43 | 454 | 8.38 | 0.00 | 8 | 349 | 0.83 | 349 | 1.68 | 0.00 | 9 | 349 | 0.97 | 349 | 1.68 | 0.00 | 12 | 302 | 0.23 | 302 | 0.36 | 0.00 |
| n15-2 | 15 | 6 | 746 | 60.59 | 746 | 3.06 | 0.00 | 8 | 414 | 1.06 | 414 | 1.58 | 0.00 | 9 | 406 | 0.23 | 406 | 0.22 | 0.00 | 12 | 406 | 0.25 | 406 | 0.23 | 0.00 |
| n15-3 | 15 | 6 | 660 | 50.44 | 660 | 4.24 | 0.00 | 8 | 442 | 9.72 | 442 | 3.49 | 0.00 | 9 | 388 | 0.19 | 388 | 0.41 | 0.00 | 12 | 388 | 0.20 | 388 | 0.28 | 0.00 |
| n15-4 | 15 | 6 | 1094 | T.L. | 1094 | 46.33 | 0.00 | 8 | 715 | 473.82 | 715 | 7.97 | 0.00 | 9 | 497 | 0.69 | 497 | 1.23 | 0.00 | 12 | 460 | 1.37 | 460 | 1.50 | 0.00 |
| n15-5 | 15 | 6 | 787 | 83.09 | 787 | 14.71 | 0.00 | 8 | 751 | T.L. | 751 | 35.40 | 0.00 | 9 | 471 | 0.59 | 471 | 0.55 | 0.00 | 12 | 469 | 0.31 | 469 | 0.53 | 0.00 |
| n20-1 | 20 | 7 | 1285 | T.L. | 1230 | T.L. | -4.28 | 9 | 846 | T.L. | 843 | 368.12 | -0.35 | 10 | 557 | 10.41 | 557 | 4.79 | 0.00 | 14 | 520 | 2.48 | 520 | 2.82 | 0.00 |
| n20-2 | 20 | 7 | 666 | 4071.75 | 666 | 13.37 | 0.00 | 9 | 450 | 4.84 | 450 | 5.57 | 0.00 | 10 | 438 | 9.63 | 438 | 2.06 | 0.00 | 14 | 399 | 2.06 | 399 | 3.43 | 0.00 |
| n20-3 | 20 | 7 | 845 | 6576.33 | 845 | 73.84 | 0.00 | 9 | 757 | T.L. | 757 | 194.61 | 0.00 | 10 | 540 | 32.84 | 540 | 3.39 | 0.00 | 14 | 507 | 2.82 | 507 | 3.10 | 0.00 |
| n20-4 | 20 | 7 | 600 | 997.44 | 600 | 20.37 | 0.00 | 9 | 580 | 6305.20 | 580 | 104.47 | 0.00 | 10 | 447 | 9.66 | 447 | 4.74 | 0.00 | 14 | 415 | 0.44 | 415 | 0.67 | 0.00 |
| n20-5 | 20 | 7 | 647 | 3623.64 | 647 | 140.71 | 0.00 | 9 | 538 | 1574.83 | 538 | 85.96 | 0.00 | 10 | 432 | 7.78 | 432 | 5.85 | 0.00 | 14 | 409 | 13.71 | 409 | 6.02 | 0.00 |
| n25-1 | 25 | 8 | 718 | 2002.73 | 718 | 155.10 | 0.00 | 10 | 711 | T.L. | 707 | T.L. | -0.56 | 11 | 499 | 110.18 | 499 | 6.96 | 0.00 | 16 | 483 | 4.57 | 483 | 2.61 | 0.00 |
| n25-2 | 25 | 8 | 825 | T.L. | 797 | T.L. | -3.39 | 10 | 716 | T.L. | 720 | T.L. | 0.56 | 11 | 501 | 167.98 | 501 | 12.28 | 0.00 | 16 | 483 | 52.12 | 483 | 12.89 | 0.00 |
| n25-3 | 25 | 8 | 603 | 3463.28 | 603 | 151.01 | 0.00 | 10 | 555 | 4473.84 | 555 | 207.22 | 0.00 | 11 | 439 | 9.87 | 439 | 5.82 | 0.00 | 16 | 405 | 1.31 | 405 | 1.98 | 0.00 |
| n25-4 | 25 | 8 | 683 | 3530.49 | 683 | 168.98 | 0.00 | 10 | 680 | T.L. | 680 | T.L. | 0.00 | 11 | 469 | 22.28 | 469 | 5.38 | 0.00 | 16 | 454 | 9.50 | 454 | 3.51 | 0.00 |
| n25-5 | 25 | 8 | 721 | 3720.39 | 721 | 41.04 | 0.00 | 10 | 703 | T.L. | 689 | T.L. | -1.99 | 11 | 491 | 58.72 | 491 | 4.74 | 0.00 | 16 | 451 | 2.87 | 451 | 2.37 | 0.00 |
| n30-1 | 30 | 9 | 877 | T.L. | 890 | T.L. | 1.48 | 12 | 864 | T.L. | 824 | T.L. | -4.63 | 13 | 615 | 135.27 | 615 | 38.06 | 0.00 | 18 | 581 | 57.35 | 581 | 17.14 | 0.00 |
| n30-2 | 30 | 9 | 927 | T.L. | 856 | T.L. | -7.66 | 12 | 863 | T.L. | 826 | T.L. | -4.29 | 13 | 560 | 62.06 | 560 | 25.26 | 0.00 | 18 | 552 | 22.12 | 552 | 7.69 | 0.00 |
| n30-3 | 30 | 9 | 812 | T.L. | 812 | T.L. | 0.00 | 12 | 818 | T.L. | 795 | T.L. | -2.81 | 13 | 506 | 38.78 | 506 | 12.45 | 0.00 | 18 | 485 | 10.41 | 485 | 5.34 | 0.00 |
| n30-4 | 30 | 9 | 756 | T.L. | 719 | 222.50 | -4.89 | 12 | 676 | T.L. | 662 | T.L. | -2.07 | 13 | 536 | 146.89 | 536 | 21.82 | 0.00 | 18 | 495 | 121.43 | 495 | 17.96 | 0.00 |
| n30-5 | 30 | 9 | 756 | T.L. | 767 | T.L. | 1.46 | 12 | 739 | T.L. | 729 | T.L. | -1.35 | 13 | 493 | 18.35 | 493 | 7.11 | 0.00 | 18 | 490 | 35.32 | 490 | 8.38 | 0.00 |

Table 5.1: Heuristic 1 and B&C3 results on instances with up to 30 nodes

| name | $n+2$ | $T$ | SolE | TimeE | SolH1 | TimeH1 | GAP |
|---|---|---|---|---|---|---|---|
| n50-1 | 50 | 20 | 604 | 908.99 | 604 | 94.13 | 0.00 |
| n50-2 | 50 | 20 | 592 | 2051.05 | 592 | 206.90 | 0.00 |
| n50-3 | 50 | 20 | 575 | 1233.33 | 575 | 332.58 | 0.00 |
| n50-4 | 50 | 20 | 621 | 3631.03 | 621 | 231.40 | 0.00 |
| n50-5 | 50 | 20 | 612 | 1331.20 | 612 | 209.06 | 0.00 |

Table 5.2: Heuristic 1 and B&C3 results for DVRP instances of 50 nodes

using this strategy, and the results obtained using the algorithm B&C3 presented in Section 4.3. The columns showed are the same than in Table 5.1, except *SolH2* and *TimeH2* that show the solution and the computing times, respectively, obtained using the Heuristic 2. In Table 5.2 also when we report "T.L." under the columns *TimeE*, the time limit reached by the exact algorithm B&C3 is 2 hours, while "T.L." in the columns *TimeH2* means that Heuristic 2 reached its time limit of 10 minutes.

From Table 5.3, we can conclude that using the Heuristic 2, the time spent to solve the problem is considerably reduced in comparison with the time spent using B&C3. This time reduction is more visible for the instances solved using the tighter value of $T$, $T_A$. When "Inf." is reported in column *SolH2*, it means that we have used such a tight value of $T$ that the problem is infeasible, using the heuristic strategy mentioned to assign the customers to the depots. In several cases, the value showed in the columns *GAP* is not so competitive, although for some instances the optimal solution is reached. The advantage of using the Heuristic 2 is showed in Table 5.4.

In Table 5.4, for the instances with 50 and 100 nodes, we use Heuristic 2, obtaining feasible solutions for the DVRP, and spending less than 10 minutes of computing time. The average time reduction obtained, comparing Heuristic 2 with B&C3, is about 95%. Moreover, the GAPs between the solutions obtained using B&C3 and Heuristic 2, are under 5%, for all the instances of 50 nodes. For the instances of 100 nodes, we have not showed results for the exact algorithm B&C3 because when the time limit of 2 hours is reached, no feasible solution has been obtained. With Heuristic 2, we are able to obtain feasible solutions for instances of 100 nodes, using less than 10 minutes of computing time.

| name | $n+2$ | $T_A$ | SolE | TimeE | SolH2 | TimeH2 | GAP | $T_B$ | SolE | TimeE | SolH2 | TimeH2 | GAP | $T_C$ | SolE | TimeE | SolH2 | TimeH2 | GAP | $T_D$ | SolE | TimeE | SolH2 | TimeH2 | GAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n10-1 | 10 | 6 | 652 | 7.22 | 888 | 0.47 | 36.20 | 7 | 442 | 0.41 | 648 | 0.11 | 46.61 | 8 | 410 | 0.73 | 648 | 0.11 | 58.05 | 10 | 369 | 0.05 | 416 | 0.03 | 12.74 |
| n10-2 | 10 | 5 | 486 | 17.43 | 509 | 0.09 | 4.73 | 6 | 292 | 0.06 | 509 | 0.08 | 74.32 | 7 | 292 | 0.08 | 319 | 0.03 | 9.25 | 10 | 292 | 0.05 | 319 | 0.06 | 9.25 |
| n10-3 | 10 | 5 | 987 | 272.77 | 994 | 1.19 | 0.71 | 6 | 646 | 327.38 | 662 | 0.50 | 2.48 | 7 | 390 | 0.14 | 660 | 0.28 | 69.23 | 10 | 383 | 0.06 | 422 | 0.02 | 10.18 |
| n10-4 | 10 | 5 | 610 | 4.52 | 728 | 0.53 | 19.34 | 6 | 534 | 8.00 | 534 | 0.84 | 0.00 | 7 | 384 | 0.27 | 384 | 0.03 | 0.00 | 10 | 383 | 0.06 | 384 | 0.03 | 0.26 |
| n10-5 | 10 | 5 | 595 | 6.91 | 601 | 0.23 | 1.01 | 6 | 365 | 0.59 | 576 | 0.28 | 57.81 | 7 | 356 | 0.12 | 356 | 0.03 | 0.00 | 10 | 350 | 0.06 | 356 | 0.02 | 1.71 |
| n15-1 | 15 | 6 | 454 | 34.43 | 506 | 0.70 | 11.45 | 8 | 349 | 0.83 | 467 | 0.90 | 33.81 | 9 | 349 | 0.97 | 352 | 0.14 | 0.86 | 12 | 302 | 0.23 | 352 | 0.14 | 16.56 |
| n15-2 | 15 | 6 | 746 | 60.59 | 1090 | 2.92 | 46.11 | 8 | 414 | 1.06 | 748 | 0.81 | 80.68 | 9 | 406 | 0.23 | 748 | 1.11 | 84.24 | 12 | 406 | 0.25 | 431 | 0.11 | 6.16 |
| n15-3 | 15 | 6 | 660 | 50.44 | 887 | 1.40 | 34.39 | 8 | 442 | 9.72 | 640 | 1.81 | 44.80 | 9 | 388 | 0.19 | 391 | 0.08 | 0.77 | 12 | 388 | 0.20 | 391 | 0.09 | 0.77 |
| n15-4 | 15 | 6 | 1094 | T.L. | Inf. | | | 8 | 715 | 473.82 | 759 | 0.83 | 6.15 | 9 | 497 | 0.69 | 759 | 0.73 | 52.72 | 12 | 460 | 1.37 | 510 | 0.11 | 10.87 |
| n15-5 | 15 | 6 | 787 | 83.09 | 1072 | 5.37 | 36.21 | 8 | 751 | T.L. | 761 | 0.62 | 1.33 | 9 | 471 | 0.59 | 475 | 0.17 | 0.85 | 12 | 469 | 0.31 | 475 | 0.19 | 1.28 |
| n20-1 | 20 | 7 | 1285 | T.L. | Inf. | | | 9 | 846 | T.L. | 864 | 3.67 | 2.13 | 10 | 557 | 10.41 | 833 | 1.86 | 49.55 | 14 | 520 | 2.48 | 549 | 0.53 | 5.58 |
| n20-2 | 20 | 7 | 666 | 4071.75 | 959 | 15.38 | 43.99 | 9 | 450 | 4.84 | 695 | 1.25 | 54.44 | 10 | 438 | 9.63 | 695 | 2.03 | 58.68 | 14 | 399 | 2.06 | 491 | 1.08 | 23.06 |
| n20-3 | 20 | 7 | 845 | 6576.33 | 1110 | 17.91 | 31.36 | 9 | 757 | T.L. | 812 | 3.00 | 7.27 | 10 | 540 | 32.84 | 552 | 1.12 | 2.22 | 14 | 507 | 2.82 | 552 | 1.23 | 8.88 |
| n20-4 | 20 | 7 | 600 | 997.44 | 792 | 4.80 | 32.00 | 9 | 580 | 6305.20 | 618 | 2.46 | 6.55 | 10 | 447 | 9.66 | 617 | 1.28 | 38.03 | 14 | 415 | 0.44 | 458 | 0.17 | 10.36 |
| n20-5 | 20 | 7 | 647 | 3623.64 | 794 | 4.85 | 22.72 | 9 | 538 | 1574.83 | 596 | 1.90 | 10.78 | 10 | 432 | 7.78 | 596 | 2.14 | 37.96 | 14 | 409 | 13.71 | 425 | 0.25 | 3.91 |
| n25-1 | 25 | 8 | 718 | 2002.73 | 762 | 3.07 | 6.13 | 10 | 711 | T.L. | 762 | 3.84 | 7.17 | 11 | 499 | 110.18 | 762 | 4.23 | 52.71 | 16 | 483 | 4.57 | 553 | 1.42 | 14.49 |
| n25-2 | 25 | 8 | 825 | T.L. | 983 | 24.87 | 19.15 | 10 | 716 | T.L. | 769 | 9.19 | 7.40 | 11 | 501 | 167.98 | 752 | 7.91 | 50.10 | 16 | 483 | 52.12 | 511 | 1.17 | 5.80 |
| n25-3 | 25 | 8 | 603 | 3463.28 | 603 | 4.56 | 0.00 | 10 | 555 | 4473.84 | 603 | 5.97 | 8.65 | 11 | 439 | 9.87 | 603 | 6.74 | 37.36 | 16 | 405 | 1.31 | 463 | 1.25 | 14.32 |
| n25-4 | 25 | 8 | 683 | 3530.49 | 683 | 2.75 | 0.00 | 10 | 680 | T.L. | 682 | 4.10 | 0.29 | 11 | 469 | 22.28 | 469 | 0.73 | 0.00 | 16 | 454 | 9.50 | 469 | 0.69 | 3.30 |
| n25-5 | 25 | 8 | 721 | 3720.39 | 755 | 2.78 | 4.72 | 10 | 703 | T.L. | 755 | 4.51 | 7.40 | 11 | 491 | 58.72 | 755 | 6.15 | 53.77 | 16 | 451 | 2.87 | 513 | 0.80 | 13.75 |
| n30-1 | 30 | 9 | 877 | T.L. | 1204 | 26.24 | 37.29 | 12 | 864 | T.L. | 911 | 8.63 | 5.44 | 13 | 615 | 135.27 | 911 | 9.52 | 48.13 | 18 | 581 | 57.35 | 641 | 1.84 | 10.33 |
| n30-2 | 30 | 9 | 927 | T.L. | Inf. | | | 12 | 863 | T.L. | 1125 | 23.45 | 30.36 | 13 | 560 | 62.06 | 849 | 30.69 | 51.61 | 18 | 552 | 22.12 | 844 | 19.31 | 52.90 |
| n30-3 | 30 | 9 | 812 | T.L. | 1125 | 23.63 | 38.55 | 12 | 818 | T.L. | 800 | 13.62 | -2.20 | 13 | 506 | 38.78 | 800 | 13.81 | 58.10 | 18 | 485 | 10.41 | 504 | 8.55 | 3.92 |
| n30-4 | 30 | 9 | 756 | T.L. | 973 | 13.43 | 28.70 | 12 | 676 | T.L. | 758 | 9.41 | 12.13 | 13 | 536 | 146.89 | 758 | 13.40 | 41.42 | 18 | 495 | 121.43 | 582 | 4.09 | 17.58 |
| n30-5 | 30 | 9 | 756 | T.L. | 1035 | 36.25 | 36.90 | 12 | 739 | T.L. | 770 | 15.79 | 4.19 | 13 | 493 | 18.35 | 763 | 12.14 | 54.77 | 18 | 490 | 35.32 | 543 | 18.35 | 10.82 |

Table 5.3: Heuristic 2 and B&C3 results on instances with up to 30 nodes

| name | $n+2$ | $T$ | SolE | TimeE | SolH2 | TimeH2 | GAP |
|---|---|---|---|---|---|---|---|
| n50-1 | 50 | 20 | 604 | 908.99 | 606 | 15.23 | 0.33 |
| n50-2 | 50 | 20 | 592 | 2051.05 | 614 | 33.12 | 3.72 |
| n50-3 | 50 | 20 | 575 | 1233.33 | 584 | 21.08 | 1.57 |
| n50-4 | 50 | 20 | 621 | 3631.03 | 643 | 13.60 | 3.54 |
| n50-5 | 50 | 20 | 612 | 1331.20 | 638 | 17.75 | 4.25 |
| n100-1 | 100 | 40 | - | - | 760 | 429.55 | - |
| n100-2 | 100 | 40 | - | - | 808 | 432.59 | - |
| n100-3 | 100 | 40 | - | - | 810 | 560.76 | - |
| n100-4 | 100 | 40 | - | - | 780 | 388.29 | - |
| n100-5 | 100 | 40 | - | - | 800 | 517.74 | - |

Table 5.4: Heuristic 2 and B&C3 results for DVRP instances with up to 100 nodes

# Conclusions

As we have mentioned in this thesis, commonly the literature on the fields of Vehicle Routing or Scheduling study problems where only a type of vehicle, driver, machine or operator is considered and, in the majority of cases, they work with a single depot. In this thesis two new problems are introduced: The Vehicle-and-Driver Scheduling Problem and The Driver and Vehicle Routing Problem. The tasks or the routes must be performed by a driver and a vehicle at the same time, and there are two depots where the vehicles and the drivers start and end their routes. The routes or sequences of tasks for drivers and vehicles have different characteristics.

In the case of the Vehicle-and-Driver Scheduling Problem (VDSP), a set of tasks must be conveniently sequenced so that they can be done by synchronized machines and operators. There are two possible configurations that mark the start/end of the process for operators and machines, and while the former must start and end the process at the same configuration, the latter must alternate between them. This problem can be modelled as a routing problem with two depots and with two types of vehicles, one type performing open routes between the two depots, and the other type performing circular routes from a predefined depot. The problem was inspired by the study of an air-transportation situation where a set of flights must be assigned to both crews and aircraft, under some conditions.

We present a mathematical formulation for the VDSP and several families of valid inequalities that serve to strengthen its LP relaxation. We have designed separation procedures for these inequalities, and we have used them to develop an exact branch-and-cut algorithm that is able to solve efficiently instances with up to 50 nodes and different features.

The second problem introduced in this thesis is the Driver and Vehicle Routing Problem (DVRP). We consider two depots and a set of customers that must be visited by a non-autonomous vehicle (that is, a vehicle with a driver). The particularity of this new problem is that vehicles' routes start at a depot and end at the other one, while drivers perform circular routes that start and end at the same depot. These differences make it mandatory that drivers exchange vehicles at some point, and thus vehicles are not always lead by the same driver along their entire routes. The synchronization of drivers and vehicles is crucial. Additionally, drivers are allowed to travel also as passengers, and they can exchange vehicles only at some given locations.

The DVRP might have applications in transportation situations where vehicles travel long distances, they are not always lead by the same driver or crew along the entire route, and drivers or crews return to their home bases at the end of their work shifts. This may happen, for example, in long distance ground transportation.

The problem is modelled as a vehicle routing problem with two depots. We present a mathematical formulation for the DVRP and some families of valid inequalities to strengthen the continuous relaxation of the mathematical model. Separation procedures for these valid inequalities have been developed, and we have used them to implement an exact branch-and-cut algorithm that is able to solve instances with up to 30 nodes and different features.

In addition, for the DVRP, two heuristic strategies have been used to obtain solutions for larger instances, spending less time. We have generated larger instances to test these heuristics. Both algorithms achieve the objective of decreasing the total computing time, but the quality of the solutions obtained with Heuristic 1 is better than the solutions obtained with Heuristic 2. The solutions obtained with Heuristic 1 are the optimal solutions for almost all the instances with up to 50 nodes. Furthermore, the Heuristic 2 manages to solve instances with up to 100 nodes in competitive computing times.

This thesis also poses interesting questions. As a future research, it would be interesting to improve the quality of the solutions obtained using Heuristic 2, applying local search strategies as two or three optimality. Also, the solution of the DVRP could be addressed using other heuristic techniques, not implying to solve any mathematical model. Finally, for both problems, VDSP and DVRP, the cases in which we consider more than two depots could be studied .

# Bibliography

D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde TSP solver. http://www.math.uwaterloo.ca/tsp/concorde/index.html , 2003.

P. Augerat, J.M. Belenger, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report RR 949-M, University Joseph Fourier, Grenoble, France, 1995.

J.R. Araque, G. Kudva, T.L. Morin, and J.F. Pekny. A branch-and-cut algorithm for vehicle routing problems. Annals of Operations Research 50, 37–59, 1994.

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

E. Balas and M. Fischetti. A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. Mathematical Programming 58, 325–352, 1993.

E. Balas and M. Fischetti. Polyhedral Theory for the asymmetric traveling salesman problem. Gutin G., Punnen A.P. (eds) The Traveling Salesman Problem and Its Variations. Combinatorial Optimization vol 12. Springer, Boston, MA, 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.

V. Cacchiani and J.J. Salazar-González. Optimal solutions to a real-world integrated airline scheduling problem. Transportation Science 51(1), 250–268, 2017.

A. Caprara and M. Fischetti. Branch-and-cut algorithms. Dell' Amico, F. Maffioli, and S. Martello, editors, Annotated Bibliographies in Combinatorial Optimization, Wiley, 45–64, 1997.

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

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth and C.Sandi (eds), Combinatorial optimization, John Wiley, New York, 1979.

J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transportation Science 35, 375–388, 2001.

CVRPLIB: capacitated vehicle routing problem library. http://vrp.atd-lab.inf.puc-rio.br/index.php/en/ , 2014

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

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

M. Desrochers and G. Laporte Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. Operations Research Letters 10, 27–36, 1991.

B. Domínguez-Martín, I. Rodríguez-Martín, and J-J. Salazar–González. An exact algorithm for a vehicle-and-driver scheduling problem. Computers and Operations Research 81, 247–256, 2017.

B. Domínguez-Martín, I. Rodríguez-Martín, and J-J. Salazar–González. The driver and vehicle routing problem. Computers and Operations Research 92, 56–64, 2018.

B. Domínguez-Martín, I. Rodríguez-Martín, and J-J. Salazar–González. A heuristic approach to the driver and vehicle routing problem. Submitted to Springer's Lecture Notes in Computer Science, 2018.

M. Drexl. Synchronization in vehicle routing - A Survey of VRPs with multiple synchronization constraints. Transportation Science 46, 297–316, 2012.

R. Freling, D. Huisman, and A.P.M. Wagelmans. Applying an integrated approach to vehicle and crew scheduling in practice. Vo S., Daduna J.R. (eds) Computer-Aided Scheduling of Public Transport. Lecture notes in Economics and Mathematical Systems vol 505. Springer, Berlin, Heidelberg, 2001.

R. Freling, D. Huisman, and A.P.M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling 6, 63, 2003.

M. R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco, 1979.

L. Gouveia. A result on projection for the Vehicle Routing Problem. European Journal of Operational Research 85, 610–624, 1995.

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

M. Gröstschel and MW. Padberg. On the symmetric traveling salesman problem I: inequalities. Math. Program. 16, 265–280, 1979.

M. Gröstschel and MW. Padberg. Polyhedral Theory. The traveling salesman problem. E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys, eds., John Wiley, 251–306, 1985.

D. Huisman, R. Freling, and A.P.M. Wagelmans. Multiple-depot integrated vehicle and crew scheduling. Transp Sci 39, 491–502, 2005.

D. Johnson and C. Papadimitriou. In the traveling salesman problem. A guided tour of Combinatorial Optimization. E. Lawler, J. Lenstra, A. H. G. Rinnooy Kan and D.B. Shmoys (eds), Wiley, Chichester, 1985.

M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, Network Models, Handbooks in Operations Research and Management Science 7, 225–330, 1995.

K. Haase, G. Desaulniers, and J. Desrosiers. Transportation Science 35(3), 286–303, 2001.

B. Kim, J. Koo, and J. Park. The combined manpower-vehicle routing problem for multi-staged services. Expert Systems with Applications 37, 8424–8431, 2010.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. Econometrica 28(3), 497–520, 960.

G. Laporte. The vehicle routing problem: an overview of exact and approximate algorithms. Annals of Oper. Res. vol. 61, 345-358, 1992.

G. Laporte. Vehicle routing. In Dell'Amico, Maffioli & Martelo (Eds.) Annotated bibliographies in Combinatorial Optimization. New York, Wiley, 1997.

G. Laporte, Y. Nobert, and D. Arpin. Optimal solutions to capacitated multidepot vehicle routing problems. Congressus Numerantium 44, 283–292, 1984.

G. Laporte, Y. Nobert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. Transportation Science 22, 161–172, 1988.

E. Lam, P. Van Hentenryck, and P. Kilby Joint vehicle and crew routing and scheduling. Lecture Notes in Computer Science 9255, 654–670, 2015.

J. Lysgaard, A.N. Letchford, and R.W. Eglese A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100, 423–445, 2004.

Lysgaard, J. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. http://www.hha.dk/lys/CVRPSEP.htm, 2004.

M. Mesquita and A. Paias. Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. Computers and Operations Research 35, 1562–1575, 2008.

C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and traveling salesman problems. Journal of the ACM 7, 326–329, 1960.

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

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

N. Papadakos. Integrated airline scheduling. Computers and Operations Research 36, 176–195, 2009.

C. Papadimitriou and K. Steiglitz. Combinatorial Optimization: algorithms and complexity. Pretince-Hall, New Jersey, 1982.

J.J. Salazar-González. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. Omega 43, 71–82, 2014.

A. Schrijver. Theory of Linear and Integer Programming. John Wiley & Sons, 1986.

S. Thienel. ABACUS: A branch and cut system, Ph.D. thesis, Universität zu Köln, Germany, 1995.

P. Toth and D. Vigo (Editors). The vehicle routing problem. SIAM Monographs on Discr. Appl. Math. vol. 9, 2002.

P. Toth and D. Vigo (Editors). The vehicle routing problem: problems, methods, and applications. MOSS-SIAM Series on Optimization, Philadelphia, Pennsylvania, 2014.

B. F. Voigt. Der Handlungsreisende wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glcklichen Erfolgs in seinen Geschften gewiss zu sein, Von einem alten Commis-Voyageur, Ilmenau, 1831. (Republished (1981) Verlag Bernd Schramm, Kiel.)

L. A. Wolsey. Integer Programming. John Wiley & Sons, Inc., 1998.

H. Yaman. Formulations and valid inequalities for the heterogeneous vehicle routing problem. Mathematical Programming 106(2), 365–390, 2006.