**POLYTECHNIC SCHOOL OF THE UNIVERSITY OF SÃO PAULO
DEPARTMENT OF NAVAL ARCHITECTURE AND OCEAN ENGINEERING**

MICHEL POVLOVITSCH SEIXAS

**HEURISTIC AND EXACT METHODS APPLIED TO A RICH VEHICLE
ROUTING AND SCHEDULING PROBLEM**

São Paulo
2013

MICHEL POVLOVITSCH SEIXAS

**HEURISTIC AND EXACT METHODS APPLIED TO A RICH VEHICLE
ROUTING AND SCHEDULING PROBLEM**

**Thesis submitted to the Department of
Naval Architecture and Ocean
Engineering of the University of São
Paulo to attain the Degree of the
Doctorate Program in Naval Architecture
and Ocean Engineering.**

**Knowledge Area: Naval and Ocean
Engineering.**

Advisor: Prof. Dr. André Bergsten Mendes

São Paulo
2013

FICHA CATALOGRÁFICA

Seixas, Michel Povlovitsch

**Heuristic and exact methods applied to a rich vehicle
Routing and scheduling problem / M.P. Seixas. -- versão corr. --
São Paulo, 2013.
126 p.**

**Tese (Doutorado) - Escola Politécnica da Universidade de
São Paulo. Departamento de Engenharia Naval e Oceânica.**

**1.Roteirização 2.Metaheurística 3.Programação inteira e
fluxos em rede 4.Programação dinâmica I.Universidade de São
Paulo. Escola Politécnica. Departamento de Engenharia Naval e
Oceânica II.t.**

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA NAVAL E OCEÂNICA

MICHEL POVLOVITSCH SEIXAS

**MÉTODOS HEURÍSTICOS E EXATOS APLICADOS A UM PROBLEMA
RICO DE ROTEIRIZAÇÃO E PROGRAMAÇÃO DE VEÍCULOS**

São Paulo
2013

MICHEL POVLOVITSCH SEIXAS

**MÉTODOS HEURÍSTICOS E EXATOS APLICADOS A UM PROBLEMA
RICO DE ROTEIRIZAÇÃO E PROGRAMAÇÃO DE VEÍCULOS**

**Tese apresentada ao Departamento de
Engenharia Naval e Oceânica da Escola
Politécnica da Universidade de São Paulo
para Obtenção do Título de Doutor em
Engenharia Naval e Oceânica.**

**Área de Concentração: Engenharia Naval
e Oceânica.**

**Orientador: Prof. Dr. André Bergsten
Mendes**

São Paulo
2013

FICHA CATALOGRÁFICA

Seixas, Michel Povlovitsch

**Heuristic and exact methods applied to a rich vehicle
Routing and scheduling problem / M.P. Seixas. -- versão corr. --
São Paulo, 2013.
126 p.**

**Tese (Doutorado) - Escola Politécnica da Universidade de
São Paulo. Departamento de Engenharia Naval e Oceânica.**

**1.Roteirização 2.Metaheurística 3.Programação inteira e
fluxos em rede 4.Programação dinâmica I.Universidade de São
Paulo. Escola Politécnica. Departamento de Engenharia Naval e
Oceânica II.t.**

Acknowledgments

I am very thankful to my mother and my father for respecting my decisions and being so supportive. I am grateful to all my closest friends. Special thanks to Rogério Aversa for encouraging me to start this research.

Many thanks to Prof. André Mendes for guiding this research, providing milestones, important references and insights and for being so supportive during all these years. I am also very thankful to Prof. Marco Brinati for the rich discussions and relevant insights since the beginning of this research.

Thanks to the lecturers and professors of both Polytechnic School of the University of São Paulo and Mathematics and Statistics Institute of the University of São Paulo, who shared their valuable knowledge and made possible the achievements of this research.

Abstract

This study considers a vehicle routing problem with time windows, accessibility restrictions on customers and a fleet that is heterogeneous with regard to capacity, average speed and cost. A vehicle can perform multiple routes per day, all starting and ending at a single depot, and it is assigned to a single driver, whose total work hours are limited. The available fleet is divided into an owned fleet, for which a variable cost is incurred, and a chartered fleet, for which only a fixed cost is incurred for each vehicle used. A column generation algorithm embedded in a branch-and-bound framework is proposed. The column generation pricing subproblem required a specific elementary shortest path problem with resource constraints algorithm to address the possibility for each vehicle performing multiple routes per day and to address the need to determine the workday's start time within the planning horizon. To make the algorithm efficient, a constructive heuristic and a learning metaheuristic algorithm based on tabu search were also developed. Both were used on branch-and-bound tree nodes to generate a good initial solution to the linear restricted master problem; particularly, to find a good initial primal bound to the branch-and-bound tree.

Keywords: Routing, Tabu Search, Column Generation, Dynamic Programming, Branch-and-price.

Resumo

Este estudo aborda um problema de roteirização de veículos com janelas de tempo, restrições de acessibilidade nos clientes e uma frota que é heterogênea em relação à capacidade de carga, velocidade média de deslocamento e custo. Um veículo pode percorrer múltiplas rotas por dia, todas começando e terminando em um mesmo depósito, e está designado a um único motorista, cujo total de horas trabalhadas no dia está limitado a um valor máximo.

A frota disponível é dividida em uma frota própria, para a qual um custo variável é incorrido, e uma frota de freteiros, para a qual apenas um custo fixo é incorrido para cada veículo utilizado. Um algoritmo baseado em geração de colunas, integrado a um procedimento de branch-and-bound, é proposto neste estudo. O subproblema de precificação da geração de colunas requereu um algoritmo específico para o problema do caminho mínimo elementar com restrições sobre recursos capaz de lidar com a possibilidade de cada veículo percorrer múltiplas rotas por dia e capaz de lidar com a necessidade de determinar o instante de início do dia de trabalho do motorista dentro do horizonte de planejamento. Para tornar o algoritmo eficiente, uma heurística construtiva e uma heurística de melhoria baseada em busca tabu também foram desenvolvidos. Ambos são utilizados nos nós da árvore de branch-and-bound para gerar boas soluções iniciais para o problema mestre restrito da geração de colunas; particularmente, para encontrar um bom limitante primal inicial para a árvore de branch-and-bound.

Palavras-chave: Roteirização, Busca Tabu, Geração de Colunas, Programação Dinâmica, Branch-and-Price.

Contents

1. INTRODUCTION.....	1
1.1. MOTIVATION	2
1.2. RESEARCH OBJECTIVES.....	2
1.3. METHODOLOGY	3
1.4. CONTRIBUTIONS.....	6
2. PROBLEM DESCRIPTION.....	8
3. LITERATURE REVIEW	11
4. MATHEMATICAL FORMULATION	17
4.1. ARC FLOW FORMULATION	19
4.2. MULTIPLE ROUTE CUTS	20
5. SOLUTION ALGORITHM.....	24
5.1. CONSTRUCTIVE HEURISTIC.....	24
5.1.1. <i>Algorithm for Building the Initial Solution</i>	24
5.1.2. <i>Procedure for Verifying the Insertion Feasibility of a Customer into a Workday</i>	27
5.1.3. <i>Procedure for Verifying the Removal Feasibility of a Customer from a Workday</i>	31
5.2. TABU SEARCH	31
5.2.1. <i>Neighbourhood Structure</i>	31
5.2.2. <i>Overview of the Tabu Search Algorithm</i>	32
5.2.3. <i>Tabu Activation Rule</i>	34
5.2.4. <i>Dynamic Tabu Tenure</i>	35
5.2.5. <i>Long-term Memory and Diversification</i>	35
5.2.6. <i>Chartered Vehicles Treatment</i>	36
5.3. EXTENSIVE FORMULATION AND COLUMN GENERATION	44
5.3.1. <i>Initialization</i>	47
5.3.2. <i>Column Generation Lower Bounding and Anticipated Finalization</i>	48
5.3.3. <i>Stabilization</i>	51
5.4. PRICING SUBPROBLEM.....	58
5.4.1. <i>Bounded Bidirectional Dynamic Programming Algorithm</i>	60
5.4.2. <i>Cost Resource</i>	62
5.4.3. <i>Load Capacity Resource</i>	63
5.4.4. <i>Time Resource</i>	63
5.4.5. <i>Workload Resource</i>	64
5.4.6. <i>Idle Time Reducing Capacity Resource</i>	67
5.4.7. <i>Elementary Path Constraints</i>	68
5.4.8. <i>Multiple Routes</i>	68
5.4.9. <i>Lexicographic Ordering of Labels</i>	69
5.4.10. <i>Bounding on Resource</i>	70
5.4.11. <i>Treatment of Cuts</i>	70
5.4.11.1. <i>Arc Flow Variable Cuts</i>	70
5.4.11.2. <i>Multiple Route Cuts</i>	71
5.4.12. <i>Simplified Algorithm for Problems without Arc Flow Variable Cuts</i>	71
5.4.13. <i>Dominance Rule</i>	72
5.4.14. <i>Heuristic Labeling Algorithm</i>	76

5.4.15.	<i>Forward-backward Joint Feasibility Tests and Search for Optimal Solution</i>	77
5.4.16.	<i>State-Space Relaxation and Decremental State-Space Relaxation</i>	80
5.5.	BRANCH-AND-PRICE	81
5.5.1.	<i>Search Strategies</i>	81
5.5.2.	<i>Initial Upper Bound</i>	82
5.5.3.	<i>Branching Strategy</i>	82
5.6.	PROBLEM LOWER BOUNDING	82
5.6.1.	<i>Alternative Lower Bounding</i>	82
5.6.2.	<i>Pseudo Lower Bound</i>	84
6.	COMPUTATIONAL RESULTS	85
6.1.	FIRST SET OF SCENARIOS: SMALL INSTANCES	86
6.2.	SECOND SET OF SCENARIOS: LARGE INSTANCES	92
6.2.1.	<i>Scenario L1: No Chartered Vehicle Available</i>	93
6.2.2.	<i>Scenario L2: Expensive Chartered Vehicles Available</i>	107
6.2.3.	<i>Scenario L3: Expensive and Inexpensive Chartered Vehicles Available</i>	111
7.	CONCLUSIONS AND FUTURE WORK	116
8.	REFERENCES	119

List of Figures

Figure 1: Research methodology by Mitroff et al. (1974).....	4
Figure 2: Methodology framework.	5
Figure 3: Axiomatic quantitative research steps.	6
Figure 4: Example of a digraph G regarding one vehicle and its workday.	17
Figure 5: Equivalent workdays.	21
Figure 6: Customer insertion into a vehicle workday.	28
Figure 7: Customer removal from a vehicle workday.....	31
Figure 8: Tabu search algorithm flowchart	33
Figure 9: IP primal block diagonal structure.....	44
Figure 10: Example of column generation iterations data.....	50
Figure 11: Example of column generation last iterations data.	51
Figure 12: Dual box and dual linear penalty for a customer dual variable	52
Figure 13: Dual box and dual linear penalty for a vehicle dual variable	53
Figure 14: Example where idle time reducing capacity resource is required.....	61
Figure 15: Time basis and arrival/departure time window.....	64
Figure 16: equivalent backward labels in different routes	72
Figure 17: Example where component (t-w) is required in the dominance rule.	73
Figure 18: Path DC-2-1-4-3-5-DC labels in the example given.....	75
Figure 19: Path DC-1-2-3-5-DC labels in the example given.....	76
Figure 20: R201 solution example.	104
Figure 21: R207 solution example	106

List of Tables

Table 1: Column generation iterations data and branch-and-price incumbent solution objective value in an example.....	48
Table 2: Four different paths from DC to vertex 3.....	74
Table 3: Results obtained for R1 instances with the first 20 customers and only owned vehicles available.	87
Table 4: Results obtained for R2 instances with the first 20 customers and only owned vehicles available.	88
Table 5: Results obtained for C1 instances with the first 20 customers and only owned vehicles available.	89
Table 6: Results obtained for C2 instances with the first 20 customers and only owned vehicles available.	90
Table 7: Results obtained for RC1 instances with the first 20 customers and only owned vehicles available.	91
Table 8: Results obtained for RC2 instances with the first 20 customers and only owned vehicles available.	92
Table 9: Results obtained for R1 instances with the first 50 customers and only owned vehicles available.	94
Table 10: Average workday cost and average number of vehicles used for R1 instances.....	95
Table 11: Results obtained for C1 instances with the first 50 customers and only owned vehicles available.	95
Table 12: Average workday cost and average number of vehicles used for C1 instances.....	96
Table 13: Results obtained for RC1 instances with the first 50 customers and only owned vehicles available.	96
Table 14: Average workday cost and average number of vehicles used for RC1 instances.	97
Table 15: Results obtained for R2 instances with the first 50 customers and only owned vehicles available.	97
Table 16: Average workday cost and average number of vehicles used for R2 instances.....	98
Table 17: Average number of routes performed by each type of vehicle in R2 instances with the first 50 customers and only owned vehicles available.	99
Table 18: Results obtained for C2 instances with the first 50 customers and only owned vehicles available.	100
Table 19: Average workday cost and average number of vehicles used for C2 instances.....	100
Table 20: Average number of routes performed by each type of vehicle in C2 instances with the first 50 customers and only owned vehicles available.	101
Table 21: Results obtained for RC2 instances with the first 50 customers and only owned vehicles available.	102
Table 22: Average workday cost and average number of vehicles used for RC2 instances.	102
Table 23: Average number of routes performed by each type of vehicle in RC2 instances with the first 50 customers and only owned vehicles available.	103
Table 24: Results obtained for R1 instances with the first 50 customers and expensive chartered vehicles available.	109

Table 25: Results obtained for C1 instances with the first 50 customers and expensive chartered vehicles available.	110
Table 26: Results obtained for RC1 instances with the first 50 customers and expensive chartered vehicles available.	111
Table 27: Results obtained for R1 instances with the first 50 customers and varied charter costs.	113
Table 28: Results obtained for C1 instances with the first 50 customers and varied charter costs.	114
Table 29: Results obtained for RC1 instances with the first 50 customers and varied charter costs.	115

1. Introduction

The transportation activity is a very important area in the logistic context and absorbs, on average, a higher percentage of the total costs than the other logistic activities (Ballou, 2004).

The vehicle routing problem (VRP) belongs to a broad category of operational research problems known as network optimization problems and it is concerned with the optimal design of routes to be used by a fleet of vehicles to serve a set of customers.

Theoretical research and practical applications in the field of vehicle routing started in 1959 with the truck dispatching problem introduced by Dantzig and Ramser (1959). In this problem, the objective is to find the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal. Dantzig and Ramser (1959) used a method based on a linear programming formulation to produce by hand calculations a near-optimal solution with four routes to a problem with twelve service stations. The authors proclaimed: “No practical applications of the method have been made as yet”.

Since it was first proposed by Dantzig and Ramser (1959), thousands of papers have been devoted to the exact and approximate solution of the many variants of this problem, such as the Capacitated VRP (CVRP), in which a homogeneous fleet of vehicles is available and the only constraint is the vehicle capacity, or the VRP with Time Windows (VRPTW), where customers may be served within a specified time interval and the schedule of the vehicle trips needs to be determined.

The June 2006 issue of OR/MS Today provided a survey with 17 vendors of commercial routing software whose packages are currently capable of finding solutions for average-sized problems with 1000 stops, 50 routes, and two-hour hard-time windows in two to ten minutes (Hall, 2006). In practice, vehicle routing is one of the success stories in operations research. For example, each day 103500 drivers at UPS follow computer-generated routes. The drivers visit 7.9 million customers and handle an average of 15.6 million packages (UPS url).

More recently, greater attention has been devoted to more complex variants of the VRP, sometimes named “rich” VRPs, which are closer to the practical distribution problems. In particular, these variants are characterized by multiple depots, multiple trips to be performed by the vehicles, multiple vehicle types or other operational issues such as loading constraints.

A rich variant of the vehicle routing problem with time windows (VRPTW) is considered herein, highlighting that each vehicle is assigned to a single driver who can travel several routes during a workday and be within the limit of his/her work hours. This feature requires the optimal point in time to start each route to be identified. Some customers have accessibility restrictions, thereby requiring specific vehicles. The fleet is heterogeneous with regard to capacity, speed and cost structure. The available fleet is divided into an owned fleet, for which only a variable cost is incurred, and a chartered fleet, for which only a fixed cost is incurred for each vehicle used. Concerning the objective function, the purpose is to minimize the total routing cost, which depends both on the distance travelled by owned vehicles and on the use of chartered vehicles.

1.1. Motivation

In recent decades, the vehicle routing problem (VRP) has been a subject of great interest due to the combination of two main factors, namely the increasing importance of routing decisions within competitive distribution environments and technological innovations such as: global positioning systems, radio frequency identification, parallel computing, which enables management of the complexity of actual instances of this combinatorial problem when searching for near-optimal solutions, among others.

The development of new solution methods based on more efficient, effective and robust algorithms to solve large-scale problems is highly desired and poses a challenge, especially to new rich variants of the vehicle routing and scheduling problem.

1.2. Research Objectives

In the real-life context analyzed, an instance of the problem usually contains more than a hundred customers and tens of vehicles. These real instances have to be solved in less than one hour by regular computers in order to not jeopardize overall company warehouse and distribution operations.

The purpose of this research is to develop an efficient and robust exact algorithm that is able to find optimal solutions for the problem studied. As the vehicle routing problem considered in this research generalizes the VRPTW and is therefore NP-hard (Lenstra and Rinnooy, 1981), for large instances, the purpose of this research is to find near-optimal solutions with proven quality (gap) by dual bounds.

In summary, the main objective of this research is to develop a complete solution procedure that is able to solve a scarcely approached problem in the literature and that is effective, efficient and robust for a daily distribution operation.

1.3. Methodology

Quantitative model-based research has been the basis of most of the initial research in operations, labeled as operations research in Europe, and was also the basis of initial management consulting and operations research in USA.

Quantitative models are based on a set of variables that vary over a specific domain, while quantitative and casual relationships have been defined between these variables. Therefore, model-based quantitative research can be defined as research where models of casual relationships between control variables and performance variables are developed, analyzed and tested. This quantitative research may be distinguished between axiomatic and empirical, and furthermore between descriptive and normative research.

In an axiomatic quantitative research, the primary concern is to obtain solutions from a model and to ensure that the solutions obtained provide information about the problem structure and behavior. This sort of research generates knowledge about the behavior of certain model variables, based on assumptions about the behavior of other variables in the model. It may also produce knowledge about how to manipulate certain variables in the model in order to obtain a desired behavior.

An axiomatic quantitative research may be normative or descriptive, although it is typically normative.

Normative research is primarily interested in developing policies, strategies and actions to improve over the results available in the existing literature, to find an optimal solution for a newly defined problem, or to compare various strategies for addressing a specific problem.

Descriptive research is primarily interested in analyzing a model, which leads to understanding and explanation of the characteristics of the model, i.e., it aims to analyze a model and fully know and explain its features and behavior.

Empirical research, the second class of model-based research, is primarily driven by empirical findings and measurements. In this case, the primary concern is to ensure that

there is a model fit between observations and actions in reality and the model made of that reality.

Similar to the axiomatic research, the empirical quantitative research may be normative or descriptive.

Normative empirical research aims to develop policies, strategies and actions to improve the current situation.

Descriptive empirical research is primarily interested in creating a model that adequately describes the casual relationships that may exist in reality, which leads to understanding of the processes going on.

The research methodology based on quantitative modeling, which is the methodology employed in this research, is defined by Mitroff et al. (1974) in four phases: conceptualization, modeling, model solving and implementation.

The four phases are executed in the order described in figure 1 and according to the framework shown in figure 2:

Figure 1: Research methodology by Mitroff et al. (1974).

Each phase of the methodology is described below:

- *Conceptualization*: in this phase, the scope of the problem and of the model to be investigated is defined. Also in this phase, variables to be used, constraints and objective function are defined.

- *Modeling*: in this phase, the researcher actually builds the quantitative model to be addressed, thus defining casual relationships between the variables.
- *Model Solving*: search for an exact solution or for an optimal solution of the problem.
- *Implementation*: implementation of the solution found in the previous phase.

Figure 2: Methodology framework.

The model-based quantitative research considered in this research is classified as axiomatic and normative according to the framework previously described.

The main body of a theoretical axiomatic quantitative operations research article or thesis generally contains chapters that cover the subjects outlined below:

- Conceptual model of the process or the problem.
- Scientific model of the process or the problem.
- Solution.
- Proof of the solution.
- Insights relating the solution to the conceptual model.

Axiomatic quantitative research starts with a condensed description of the characteristics of the operational process or the operational decision problem that is going to be studied. This corresponds to the conceptual model in figure 2.

The second phase is specification of the scientific model of the process or problem. The scientific model must be presented in formal mathematical terms, such that either mathematical or numerical analysis is possible, or computer simulation can be carried out.

In the last steps, a solution is found and a proof to the solution is presented. Proofs generally can only be delivered with mathematical analysis and solution identification criteria. In summary, high quality solutions result from a combination of a mathematical proof of the quality of the solution and of insight into what might be a solution.

In figure 3 it is possible to visualize the steps of axiomatic quantitative research.

Figure 3: Axiomatic quantitative research steps.

1.4. Contributions

The major contribution of this research is the development of an exact solution procedure for a rich variant of the vehicle routing problem which contemplates: a mathematical formulation, valid cuts to strengthen the mathematical formulation developed, a constructive heuristic procedure, a learning tabu search algorithm, a state-of-the-art column generation procedure embedded in a branch-and-price scheme and a dynamic programming algorithm to solve a rich elementary shortest path problem with resource constraints; the latter one being the major contribution of this work.

It is worth highlighting that the rich problem considered in this study has been scarcely approached by heuristic or exact methods in the literature to the best of our knowledge and it can be found in many warehouse and distribution operations throughout the world.

Regarding the tabu search algorithm, the learning algorithm development is able to address, besides routing and scheduling decisions, chartering decisions with regard to vehicles that do not have associated variable costs in a limited and heterogeneous fleet

context. Also with regard to the tabu search developed, a new intensification strategy based on a short-term memory deactivation is introduced to obtain a high intensification during the local search of the tabu search framework.

This research is organized as follows: Chapter 2 provides a detailed description of the characteristics of the problem studied and of the objective function to be optimized; Chapter 3 presents a literature review regarding the problem studied; Chapter 4 presents a mathematical formulation and valid cuts; Chapter 5 describes procedures and algorithms; Chapter 6 presents parameter settings, problem instances and the results obtained. Conclusions and suggestions of future work are given in Chapter 7.

2. Problem Description

The vehicle routing problem considered in this work is based on the distribution operation of a beverage company in São Paulo, Brazil. The division responsible for routing the vehicles divides the city land area in several sectors, with each sector served by vehicles of a distribution center located inside the metropolitan area. Therefore, each customer located in São Paulo metropolitan area is designated to a particular distribution center.

In summary, the vehicle routing problem studied in this work consists of determining the workday of each vehicle of the fleet to minimize the total cost of the distribution operation from a single depot.

A vehicle workday is defined as a sequence of customers to be visited; for this purpose, the vehicle may return to the distribution center to be reloaded and then start a new delivery route. That is, a workday route is defined as a sequence of customers that must be visited by the vehicle in a specified order after the vehicle is loaded at the distribution center. Thus, a workday consists of a feasible sequence of routes throughout the day. For convenience purposes, the maximum number of routes in a workday is predetermined. Vehicles that are not used run a fictitious route in which a vehicle leaves the distribution center and returns to it without visiting any customer at no cost. When a vehicle arrives at the distribution center to be reloaded, the products to be delivered were previously picked and packed in the warehouse and they are ready to be loaded inside the vehicle cargo compartment.

It is worth stressing that the possibility for a vehicle to perform multiple routes in a workday is highly desired as it maximizes its use, particularly, in case of small and agile vehicles (e.g. vans) that can serve few customers in a route due to a limited cargo compartment capacity.

Demand of each customer is known before the day starts, and it is defined as a demand for units of four different product families: Pet bottle, can, cellaret and barrel. Every customer must be served only once by a vehicle in the planning horizon; that is, a delivery is not fragmented and is compulsory.

Each customer has a specific service time, which depends on the vehicle used to perform the service, and a strict time window that must be respected. If the vehicle reaches a customer before the beginning of its time window, it must wait until the time

window starts to begin the service. If the vehicle arrives after the end of the time window, the service is not provided.

Some customers can only be visited by specific vehicles due to accessibility restrictions at the delivery location.

The fleet is heterogeneous in terms of both capacity and average displacement speed and it is divided into van-type vehicles, LCVs (light cargo vehicles) and trucks. The differences with regard to average displacement speed among different types of vehicles are, primarily, based on parking time, vehicle manoeuvring time in congested streets along the trip and traffic movement restrictions for trucks.

In addition to owned vehicles, a fleet of chartered vehicles is available. The cost of an owned vehicle is variable, depending only on the total distance traveled and on the vehicle type. On the grounds that a common practice in some distribution operations is to charter vehicles based only on a daily fixed cost agreement, in the present study, the cost of a chartered vehicle is fixed and incurred only if the vehicle is used to serve any customer. These chartered vehicles usually have high associated fixed costs and they are primarily employed to absorb seasonal peaks of delivery orders (peaks in demand), that could not be fulfilled by employing only owned vehicles of the available fleet.

No considerations are made about how products are arranged inside a vehicle cargo compartment and its maximum capacity is defined in units of each product family, which allows one to define the fraction of the vehicle total cargo compartment capacity occupied by a customer demand. A more precise approach, that defines how products are packed inside the vehicle cargo compartment, may also be addressed. This implies in additional packing decisions to be made, besides routing, fleet size and mix and scheduling decisions, and it is responsible for a significant increase in complexity in terms of both mathematical formulation and solution algorithm. It is worth highlighting that the container loading problem (Moura and Oliveira, 2005) is notoriously NP hard (Scheithauer, 1992), meaning that the solution is too complex to be solved exactly in polynomial time. Consequently, only the simplified approach is considered and no packing decisions are addressed in this study.

The total number of hours worked by the driver in a workday cannot exceed a specified maximum limit. The vehicle loading time at the distribution center before the route begins and possible waiting times at customers' locations until the beginning of their time windows are considered in the calculation of the worked hours, along with the

travelling time between customers and the service time at each customer's location. The loading time of the vehicle at the distribution center depends on the vehicle type.

The traveling times between customers and between the distribution center and customers depend on both the distance traveled and the average speed of the vehicle used.

3. Literature Review

Great advancements have been obtained for different VRPTW variants, such as those by Solomon and Desrosiers (1988) and Desrochers et al. (1988). Approaches with column generation for set partitioning formulations of the VRPTW and other variants have been presented as well. Desrochers et al. (1992) proposed the use of the column generation technique to solve the linear relaxation of the VRPTW set partitioning formulation. Columns are added, when necessary, by means of the resolution of the shortest path problem with time windows and capacity constraints using a dynamic programming algorithm. The solutions obtained generally yielded excellent dual bounds which are used in a branch-and-bound algorithm to solve the integer formulation. The same approach employed by Desrochers et al. (1992) is used in the present study.

The design and analysis of algorithms for vehicle routing and scheduling problems with time window constraints were addressed by Solomon (1987). All the algorithms presented were extensions of known VRP heuristics and his approach consisted in incorporating not only the distance but also the time dimension in the heuristic process. As a consequence, Solomon (1987) examined the necessary and sufficient conditions for time feasibility when inserting a customer on a partially constructed feasible route and defined a *push forward* created by the customer insertion on subsequent customers service time along the route. When computing the *push forward* created by the insertion in each customer, Solomon's procedure minimizes idle time generated, which is the same approach used in the present study by the dynamic programming algorithm of the column generation framework proposed.

Tabu search, which is the heuristic method considered in this study, is a local search metaheuristic introduced by Glover (1986) that has been successfully used to solve vehicle routing problems with time windows. Garcia et al. (1994) were the first to apply tabu search for VRPTW, and since then, many authors have presented numerous tabu search implementations involving sophisticated techniques and strategies (Bräysy and Gendreau, 2005). Further information regarding the tabu search algorithm is given by Glover (1989, 1990), Hertz et al. (1997), Glover and Laguna (1997) and Gendreau (2003).

Obtaining solutions to the vehicle routing problem with multiple routes, but without time windows, has been approached by means of heuristics by Taillard et al. (1996), Brandão and Mercer (1998), Olivera and Vieira (2007), Petch and Salhi (2003, 2007),

among others. In the study by Taillard et al. (1996), different routes generated by tabu search are combined to produce workdays through the resolution of a bin-packing problem. In the present work, the tabu search developed acts on a space with multiple routes without the need to combine them to form workdays. This approach, which is more costly for seeking neighbour solutions within a larger space, eliminates the computational cost of analyzing the feasibility of combining different routes and finding a good combination.

Brandão et al. (1997) solved a multi-trip vehicle routing and scheduling problem that is similar to the problem studied. The main differences are that the driver's legal time breaks are not considered herein, therefore idle time must be minimized, and that there are no variable cost associated with hired vehicles, consequently, new procedures to evaluate solutions must be considered. The tabu search algorithm developed by Brandão et al. (1997) consists of three phases, and it is based on both tabu search theory and on an algorithm developed by Gendreau et al. (1992) for the travelling salesman problem called GENI, which is not considered herein.

Azi et al. (2010) proposed a method, which is either exact or heuristic, to solve the VRPTW with multiple routes. It is a generalization of a previous method (Azi et al., 2007) for the same problem, but with only one vehicle available. In their work, a preprocessing is conducted to generate the set of all feasible routes or a subset of these routes so that a branch-and-price algorithm is employed to find the optimal solution over the set of routes generated during the preprocessing stage. In the branch-and-price algorithm of Azi et al. (2010), the pricing subproblem, which can be cast as an elementary shortest path problem with resource constraints, consists of finding a sequence of feasible routes on a network where each vertex corresponds to such a route. Alternatively, in the present study, the subproblem consists of finding a sequence of customers on a network where a vertex corresponds to a customer or the depot, and no preprocessing is required to find the optimal solution over the set of all feasible routes.

It is important to emphasize that this study deals with a heterogeneous fleet vehicle routing problem (HFVRP) which is considered to be much harder than corresponding problems with a homogeneous fleet. The HFVRP was proposed by Taillard (1999) and it is a very important problem, since fleets are likely to be heterogeneous in most practical situations. The author developed an algorithm based on tabu search, adaptive memory and column generation which was also applied to solve the fleet size and mix vehicle routing problem (FSMVRP). Prins (2002) developed an algorithm for the

HFVRP with multiple routes that extends a number of VRP classical heuristics followed by a local search procedure based on the steepest descent local search and tabu search. Li et al. (2010) proposed a multi-start adaptive memory procedure combined with path relinking and a modified tabu search to solve the HFVRP with fixed and variable costs. More recently, Brandão (2011) proposed a tabu search algorithm for the HFVRP which includes additional features such as strategic oscillation, shaking and frequency-based memory.

Concerning the treatment typically given to a workday, it is considered that a number of working hours cannot be exceeded for any route, forcing the last customer to be served within a maximum time interval from when the route began (Azi et al., 2010, and Cordeau et al., 2001). Cordeau and Laporte (2003) proposed a tabu search heuristic for the static dial-a-ride problem with route duration constraints that evaluates the capacity to delay the departure time from the depot to help improve solution feasibility during the neighbourhood search. In the present problem, a similar control is necessary, but it is applied to multiple routes forming a workday in an exact, rather than heuristic, context.

Ceselli et al. (2009) solved a real-world, multi-trip vehicle routing problem with driver constraints that are similar to those in this study. In their work, each vehicle is described by a maximum daily duty length and, according to work rules, the waiting time and unloading time count as rest time or driving time. In the present problem, the waiting time must always be counted as driving time, which requires the ability to handle the freedom of a variable workday start time.

It is also worth stressing that classical dynamic programming algorithms developed to solve the shortest path problem with resource constraints, such as in Dejax et al. (2004), do not include considerations about the determination of the beginning of each vehicle path in relation to the planning horizon. As to the driver allocation, if the legal and labor requirements are abided by, it is not reasonable that the different drivers' workdays start at the same time, when dealing with a distribution operation with time windows. In the present study, the instant when the workday starts is a decision to be made together with the determination of the sequence of customers to be visited.

This characteristic, together with the possibility of the vehicle being able to carry out multiple routes per day, is treated by a specific algorithm to solve the subproblem in the ambit of column generation. This is the major contribution of this work, in addition to

providing a framework that also contains a constructive heuristic and a very effective tabu search.

The existence of a fixed cost, which is incurred only when a vehicle is used, in addition to a variable cost, is an inherent characteristic of fleet size and mix routing problems (FSMVRP) in which the number of vehicles available in the fleet is considered to be unrestricted. In FSMVRP, the fixed cost associated with vehicle use in the objective function, whose value is minimized and which also contains vehicle variable costs, is responsible for pricing the use of a vehicle so as to minimize the number of vehicles required, among other objectives. Other variants of the FSMVRP are also approached in the literature, such as the FSMVRP with fixed costs but without variable costs and the FSMVRP with variable costs only.

This latter approach has been successfully used by Desrosiers et al. (1988) in a school bus scheduling algorithm, which finds the minimum number of vehicles required to visit a set of nodes, each subject to time window constraints, considering the case of an homogeneous fleet of vehicles located at a common depot. Desrosiers et al. (1988) formulated the problem such that the fixed cost is incurred in the objective function when a vehicle travels the arc that leaves the depot to a trip node in a graph where each node specifies a known and previously defined bus trip. A similar approach was used in this study, in which a fixed cost is incurred when a chartered vehicle travels at least one arc leaving the depot node to a customer node.

Many metaheuristic based approaches were proposed for the FSMVRP over the years. Ochi et al. (1998a) proposed a hybrid evolutionary procedure that combines scatter search with genetic algorithm to solve the FSMVRP with fixed costs but without variable costs. A parallel implementation of the same algorithm was presented by Ochi et al. (1998b). Gendreau et al. (1999) developed a heuristic algorithm that combines a GENIUS approach, tabu search and adaptive memory. A hybrid genetic algorithm that employs local search as a mutation approach was developed by Liu et al. (2009) to solve the FSMVRP with fixed costs only and the FSMVRP with variable costs only. Two memetic algorithms were developed by Prins (2009) to solve all FSMVRP variants and the HFVRP with variable costs but without fixed costs.

The possibility of outsourcing vehicles has also been approached in the literature. The vehicle routing problem with private fleet and common carrier (VRPPC) describes a situation where, by hypothesis, the total demand exceeds the capacity of the internal fleet, so an external transporter is necessary (common carrier). In this situation the

problem is twofold: select customers that should be served by the external carrier and define routes of the internal fleet to serve remaining customers. The cost of serving a customer by the common carrier is fixed for each customer served, without referring to any routing. The single vehicle case was formulated by Volgenant and Jonker (1987) and solved exactly by Diaby and Ramesh (1995). The VRPPC was formally introduced by Chu (2005) who solved it heuristically: he firstly applied a modified version of the classical saving procedure (Clarke and Wright, 1964) followed by some local exchanges between routes. Bolduc et al. (2007) have proposed a heuristic called SRI that is composed of three steps: the selection of customer served by the external carrier, the construction of the solution (routing) and the improvement, through the application of sophisticated exchanges. Subsequently, Bolduc et al. (2008) have presented a perturbation metaheuristic, called RIP (Randomized construction, Improvement, Perturbation), which essentially combines a descent method with diversification strategies. Recently, Côté and Potvin (2009) have obtained the best-known results on a set of benchmarks using a Tabu Search approach.

It is worth emphasizing that the vehicle routing problem considered in the present study is slightly different from the VRPPC previously introduced, as the routes of the chartered (outsourced) vehicles must be designed and the fixed cost of outsourcing a vehicle is independent of the number of customers it serves.

Motivated by a real-life case, the study of Zäpfel and Bögl (2008) considered short-range weekly planning on the part of postal companies that must decide about pickup tours and delivery tours for fluctuating volume (number of shipments), with time windows for the demand points, in consideration of variable vehicle capacities and personnel planning, and including outsourcing decisions for tours and drivers. The authors propose a hybrid metaheuristic combined with a construction heuristic to solve the problem, which can be cast as a multi-period vehicle routing and crew scheduling problem with outsourcing options.

Tan et. Al. (2006) considered a transportation problem for moving empty or laden containers for a logistic company that, owing to the limited resource of its vehicles (trucks and trailers), often needs to sub-contract certain job orders to outsourced companies. The authors propose a multi-objective truck and trailer vehicle routing problem model and a hybrid multi-objective evolutionary algorithm, which is featured with specialized genetic operators, variable-length representation and local search heuristic, to find the Pareto optimal routing solutions.

It should be stressed that the vehicle routing problem considered in this study generalizes the VRPTW and is therefore NP-hard (Lenstra and Rinnooy, 1981).

4. Mathematical Formulation

Let digraph $G^* = (N \cup \{O_1\} \cup \{O_2\}, A^*)$ be given where N is the set of customers; O_1 and O_2 are fictitious origin and destination vertices, respectively, both corresponding to the depot; A^* is the set of arcs joining two distinct vertices of G^* ; and any feasible route corresponds to a path from O_1 to O_2 on G^* .

Let R represent the maximum number of routes driven during a vehicle workday. To represent a workday with R routes, digraph G^* is sequentially replicated R times, forming digraph G , in which a path from O_1 (origin of route 1) to O_{R+1} (destination of route R) represents a vehicle workday with the R routes (see figure 4).

Figure 4: Example of a digraph G regarding one vehicle and its workday.

Computationally, only a reduced digraph of a single route G^* is stored and used so that memory consumption is not significantly increased.

An instance of the problem is defined by the data listed below:

- Directed graph $G = (\{N_1, N_2, \dots, N_R\} \cup \{O_1, O_2, \dots, O_R, O_{R+1}\}, A)$ in which the set of vertices is composed of the set of customers N replicated R times by adding the R route Origins and the vertex of the Final Destination (O_{R+1}); set A contains the arcs between any two vertices of the same route. There are no arcs reaching origin O_1 and there are no arcs leaving destination O_{R+1} . Vehicles not used in route r during the workday travel the fictitious arc (O_r, O_{r+1}) at zero cost.
- Set of heterogeneous vehicles $K = \{K^o \cup K^c\}$, where K^o is the set of owned vehicles and K^c is the set of chartered vehicles available.
- N^K – set of customers to be served by specific authorized vehicles: $N^K \subseteq N$.

- Each customer i in N^K has a group of authorized vehicles K_i : $K_i \subseteq K$.
- Q_k^{pet} – maximum cargo capacity of vehicle k in units of Pet bottle, i.e., number of bottles that fill the entire cargo compartment of vehicle k .
- Q_k^{can} – maximum cargo capacity of vehicle k in units of can, i.e., number of cans that fill the entire cargo compartment of vehicle k .
- Q_k^{cel} – maximum cargo capacity of vehicle k in units of cellaret, i.e., number of cellarrets that fill the entire cargo compartment of vehicle k .
- Q_k^{bar} – maximum cargo capacity of vehicle k , in units of barrel, i.e., number of barrels that fill the entire cargo compartment of vehicle k .
- D_i^{pet} – customer i demand for units of Pet bottle.
- D_i^{can} – customer i demand for units of can.
- D_i^{cel} – customer i demand for units of cellaret.
- D_i^{bar} – customer i demand for units of barrel.
- $[a_i, b_i]$ – time window of customer i . For the vertices $\{O_1, \dots, O_{R+1}\}$ representing the distribution center (depot), the time windows correspond to the planning horizon, i.e., $[a_{O_1}, b_{O_1}] = \dots = [a_{O_{R+1}}, b_{O_{R+1}}] = [a_{DC}, b_{DC}] = [0, b_{DC}]$.
- s_i^k – service time at customer i when the service is provided by vehicle $k \in K$. At the distribution center, this corresponds to the average loading time of vehicle $k \in K$, s_{DC}^k .
- c_{ijr}^k – cost of traveling the arc (i, j) in route r with vehicle $k \in K^o$. This value depends on the arc (i, j) distance, d_{ij} , and on the variable cost of the vehicle k , VC_k , $k \in K^o$, given in \$/distance.
- FC^k – daily charter cost of vehicle $k \in K^c$.
- $\tilde{t}_{ij}^k$ – travel time of arc (i, j) with vehicle k .
- $W_{max,k}$ – maximum driver working hours for vehicle k , i.e., maximum duration of a vehicle k workday.
- R – maximum number of routes allowed per day to the vehicles. This value is an upper bound that is sufficiently large to hold all possible trips according to past daily routing operation experience.
- M_{ij}^k is a sufficiently large number defined as: $M_{ij}^k = \max\{b_i + s_i + \tilde{t}_{ij}^k - a_j, 0\}$ $k \in K, i, j \in A$.

4.1. Arc Flow Formulation

The mathematical formulation developed has three decision variables x_{ijr}^k , a binary variable that is equal to 1 if arc (i, j) is travelled by vehicle k along route r and that is equal to 0 otherwise; t_{ir}^k , a continuous variable that defines the instant in time at which vehicle k starts serving customer i along route r , and v^k , a binary variable that is equal to 1 if chartered vehicle k is used to serve any customer and that is equal to 0 otherwise. Below is the mathematical formulation of the initial problem, denoted IP.

(IP)

$$\text{Min} \sum_{r=1}^R \sum_{k \in K^o} \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijr}^k + \sum_{k \in K^c} FC^k \cdot v^k$$

s. t.

$$\sum_{r=1}^R \sum_{k \in K} \sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k = 1 \quad i \in N \quad (1)$$

$$\sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} x_{ijr}^k = 0 \quad j \in N^K, k \notin K_j \quad (2)$$

$$\sum_{j \in N \cup \{O_{r+1}\}} x_{O_r j r}^k = 1 \quad k \in K, r = 1, \dots, R \quad (3)$$

$$\sum_{i \in N \cup \{O_r\}} x_{i O_{r+1} r}^k = 1 \quad k \in K, r = 1, \dots, R \quad (4)$$

$$\sum_{i \in N \cup \{O_r\}} x_{ihr}^k - \sum_{j \in N \cup \{O_{r+1}\}} x_{hjr}^k = 0 \quad k \in K, h \in N, r = 1, \dots, R \quad (5)$$

$$a_i \leq t_{ir}^k \leq b_i \quad k \in K, r = 1, \dots, R, i \in N \cup \{O_r, O_{r+1}\} \quad (6)$$

$$t_{ir}^k + s_i^k + \tilde{t}_{ij}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad k \in K, r = 1, \dots, R, (i, j) \in A \setminus \{(O_r, O_{r+1})\} \quad (7)$$

$$t_{ir}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad k \in K, r = 1, \dots, R, (i, j) = (O_r, O_{r+1}) \quad (8)$$

$$\sum_{i \in N} \left[\left(\frac{D_i^{pet}}{Q_k^{pet}} + \frac{D_i^{can}}{Q_k^{can}} + \frac{D_i^{cel}}{Q_k^{cel}} + \frac{D_i^{bar}}{Q_k^{bar}} \right) \left(\sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k \right) \right] \leq 1 \quad k \in K, r = 1, \dots, R \quad (9)$$

$$t_{O_{R+1}R}^k - t_{O_{1,1}}^k \leq W_{max,k} \quad k \in K \quad (10)$$

$$v^k \geq \sum_{j \in N} x_{O_r j r}^k \quad k \in K^c, r = 1, \dots, R \quad (11)$$

$$x_{ijr}^k \in \{0,1\} \quad k \in K, (i,j) \in A, r = 1, \dots, R \quad (12)$$

$$t_{ir}^k \geq 0 \quad k \in K, r = 1, \dots, R, i \in N \cup \{O_1, \dots, O_{R+1}\} \quad (13)$$

$$v^k \in \{0,1\} \quad k \in K^c \quad (14)$$

Constraint (1) imposes that each customer i must be visited only once by a vehicle k in a route r . According to constraint (2), unauthorized vehicles are kept from visiting customers that have accessibility restrictions. In constraint (3), it is imposed that every vehicle k must leave from each route origin. Analogously, constraint (4) imposes that every vehicle k must reach each route destination. Constraint (5) ensures the flow conservation for a customer, i.e., if a vehicle reaches a vertex then this vehicle must leave from the same vertex. Constraint (6) imposes that time windows $[a_i, b_i]$ must be respected. The group of constraints (7) and (8) establishes the relationship between the vehicle departure time from a customer and that customer's immediate successor service time. The particular constraint (8) for arcs (O_r, O_{r+1}) $r = 1, \dots, R$ is required on the grounds that average loading time at the distribution center and travel time must not be considered in these cases. Constraint (9) states that a vehicle can only be loaded up to its capacity. Constraint (10) guarantees that the maximum daily working hours of a vehicle do not exceed its limit $W_{max,k}$. Constraint (11), together with the objective function, ensures that the binary variable associated with a chartered vehicle is equal to 1 if it is used and 0 otherwise. The group of constraints (12), (13) and (14) defines the solution space of the decision variables.

4.2. Multiple Route Cuts

It is important to notice that the fictitious route arc (O_r, O_{r+1}) may be responsible for the existence of equivalent solutions from a physical point of view, but distinct mathematically, when the maximum number of routes allowed per day to the vehicles is greater than or equal to 2, i.e., when $R \geq 2$.

Equivalent solutions of a current solution exist when $R \geq 2$ and there is, at least, one workday of any vehicle in which at least one fictitious arc is traveled and there is, at least, one route serving any customer. In these cases, there is freedom in rearranging fictitious routes inside the workday without changing the relative sequence of routes

serving customers, so as to generate equivalent workdays and, therefore, equivalent solutions.

Figure 5 shows all equivalent workdays in an example with four routes on total ($R = 4$) and only two routes serving customers. In this case, there are six equivalent workdays.

Figure 5: Equivalent workdays.

Let:

- X – a feasible solution of IP formulation;
- WD_k – workday of vehicle k of a feasible solution X . Therefore, X is the set of workdays of the vehicles of the fleet, i.e., $X = \{WD_k | k \in K\}$;
- n_{wd}^k – total number of equivalent workdays, based on the sequence of routes serving customers in workday WD_k of solution X , including WD_k ;
- n_r^k – number of routes of workday WD_k serving customers, $n_r \leq R$;
- r_i^k – i -th route serving at least one customer in the sequence of routes of workday WD_k serving customers. The index i , where $1 \leq i \leq n_r$, refers to the route position in the sequence of routes serving customers.

For a workday WD_k , when $1 \leq n_r^k < R$, the total number of equivalent workdays n_{wd}^k , i.e., workdays with the same sequence of routes serving customers, is calculated as:

$$n_{wd}^k = \sum_{r_1^k=1}^{R-n_r^k+1} \left(\sum_{r_2^k=r_1^k+1}^{R-n_r^k+2} \left(\sum_{r_3^k=r_2^k+1}^{R-n_r^k+3} \left(\dots \left(\sum_{r_{n-1}^k=r_{n-2}^k+1}^{R-n_r^k+(n_r^k-1)} \left(\sum_{r_n^k=r_{n-1}^k+1}^{R-n_r^k+n_r^k} 1 \right) \right) \right) \right) \right)$$

If $n_r^k = 1$:

$$n_{wd}^k = \sum_{r_1^k=1}^R 1 = R$$

If $n_r^k = 2$:

$$n_{wd}^k = \sum_{r_1^k=1}^{R-1} \left(\sum_{r_2^k=r_1^k+1}^R 1 \right) = \frac{R^2 - R}{2}$$

If $n_r^k = 3$:

$$n_{wd}^k = \sum_{r_1^k=1}^{R-2} \left(\sum_{r_2^k=r_1^k+1}^{R-1} \left(\sum_{r_3^k=r_2^k+1}^R 1 \right) \right) = \frac{R^3 - 3R^2 + 2R}{6}$$

If $n_r^k = 4$:

$$n_{wd}^k = \sum_{r_1^k=1}^{R-3} \left(\sum_{r_2^k=r_1^k+1}^{R-2} \left(\sum_{r_3^k=r_2^k+1}^{R-1} \left(\sum_{r_4^k=r_3^k+1}^R 1 \right) \right) \right) = \frac{R^4 - 6R^3 + 11R^2 - 6R}{24}$$

...

In the example of Figure 5 where $n_r^k = 2$ and $R = 4$, $n_{wd}^k = \frac{4^2 - 4}{2} = 6$.

Based on the above equations, it is possible to conclude that for a workday with n_r^k routes serving customers when $1 \leq n_r^k < R$, an increase in the total number of routes, R , with n_r^k constant is followed by a polynomial increase in the number of equivalent workdays, i.e., $n_{wd}^k = O(R^{n_r^k})$ in big-O notation.

It is important to stress that a feasible solution may have several workdays that can be rewritten in an equivalent form. Hence, the total number of equivalent solutions of X , including X , is given by: $\prod_{\{WD_k \subseteq X | 1 \leq n_r^k < R\}} n_{wd}^k$.

The significant increase, of combinatorial nature, in the number of equivalent solutions due to an increase of the maximum number of routes allowed per day to the vehicles is highly undesired, on the grounds that it is responsible for an increase of the size of the solution space and, therefore, of the computational cost of finding optimal and near-optimal solutions.

In order to reduce the expansion of the solution space when the value of R increases, valid inequalities where (15) derived.

$$x_{o_{r+1}o_{r+2}r+1}^k \geq x_{o_r o_{r+1}r}^k \quad k \in K, r = 1, \dots, (R - 1) \quad (15)$$

According to constraints (15), applicable only when $R \geq 2$, if a vehicle k travels the fictitious arc in route r , i.e., $x_{o_r o_{r+1}r}^k = 1$, then the fictitious route arc must be traveled along the following routes until the final route R . Consequently, the routes serving customers are always placed in the beginning of the workday and the fictitious routes are always traveled in the end of the workday so as to fulfill R routes.

5. Solution Algorithm

To solve the problem described in Chapter 2 and formulated in Chapter 4 to optimality, a branch-and-price procedure was proposed. The linear relaxation of each branch-and-bound tree node was solved using column generation technique, which requires an efficient algorithm for solving the pricing subproblem. The pricing subproblem, which is responsible for generating feasible workdays for each vehicle of the available fleet, was solved by a dynamic programming algorithm.

A constructive heuristic and a metaheuristic based on tabu search were also developed to find a good initial upper bound at the root node of the branch-and-price tree, as well as to generate a good initial solution for the linear restricted master problem of the other tree nodes.

It is described in the following chapters: Constructive Heuristic, Tabu Search, Dantzig Wolfe Decomposition and Column Generation, Pricing Subproblem, Branch-and-Price and Problem Lower Bounding.

5.1. Constructive Heuristic

5.1.1. Algorithm for Building the Initial Solution

In summary, the developed procedure inserts customers into vehicle workdays and builds an initial solution after inserting all of them according to the pseudocode *Constructive_Heuristic()*.

Constructive Heuristic Algorithm – procedure to find a feasible solution.

```
Constructive_Heuristic()
1  find customers' geometric center
2  divide customers into 4 quadrants with their geometric center as a reference
3  sort each quadrant list of customers in increasing order of time window
   beginning
4  //Travel quadrants in anti-clockwise sequence starting from the upper right
   quadrant //
5  for each quadrant customer list do
6      for every customer with a list of authorized vehicles do
7          find the smallest insertion cost position among authorized vehicle
           workdays
8          if a position was found then
9              insert customer and remove it from the quadrant list
10  sort the list of vehicles in decreasing order of cargo capacity
11  sort the list of vehicles in increasing order of cost (variable or fixed) with a
   stable sorting algorithm
12  for each vehicle in the sorted list of vehicles do
13      //Travel quadrants in anti-clockwise sequence starting from the upper
           right quadrant //
14      for each quadrant customer list do
15          for each customer do
16              find smallest insertion cost position in current vehicle
               workday
17              if a position was found then
18                  insert customer and remove it from quadrant list
19  if there are unserved customers then
20      if tabu search was never called then
21          call tabu search to improve current solution
22          go to line 5 to insert unserved customers
23      else
24          call a commercial software to find a feasible solution
```

Initially, customers are divided into four groups, each one corresponding to a quadrant, with customers' geometric center serving as the reference. For each quadrant, a list of all of its customers is generated and sorted in increasing order of time window beginning. It is worth highlighting that some customers have accessibility restrictions and, thus, have a list of authorized vehicles. The other customers will not have such list and they can be served by any vehicle.

Next, the four quadrants are traveled in an anti-clockwise sequence, starting from the upper right quadrant, to first allocate the customers with accessibility restrictions. The choice of the upper right quadrant as the first quadrant to be analyzed, as well as the anti-clockwise sequence instead of clockwise, were both arbitrary. In each quadrant, the list of customers not yet served and previously organized is traveled to identify and

allocate those with accessibility restrictions. After a customer with this characteristic is identified, the customer is inserted into a position within a vehicle workday from its list of authorized vehicles that results in the smallest insertion cost.

When all customers with accessibility restriction are allocated, the remaining customers are then treated. First, a vehicle is chosen according to the criterion which gives higher priority for owned vehicles and which, among vehicles with equal cost structures, selects the vehicle with the largest capacity. After the vehicle selection is completed, the four quadrants are traveled in an anti-clockwise sequence and in each quadrant, the list of unserved customers is traveled. For each unserved customer, all possible insertion positions within the chosen vehicle workday are verified and the feasible position that results in the smallest insertion cost is chosen. Once inserted, the customer is removed from the list of the corresponding quadrant.

This process is repeated until all customers of each quadrant are analyzed. For further details on the verification of the insertion feasibility of a customer into a vehicle workday, see Section 5.1.2.

After travelling all four quadrants with the vehicle previously chosen, a new vehicle that has not yet been analyzed is chosen, according to the criterion presented and all four quadrants are traveled again to allocate unserved customers to the chosen vehicle workday. It is important to note that a vehicle may be chosen only once by the algorithm.

In most cases, this procedure is sufficient to build an initial solution. However, an initial solution may not be generated in cases such as those in which there are few vehicles available, those in which many customers must be served by specific vehicles or those in which service times are large. When this occurs, the strategy adopted to build an initial solution shortly consists in:

- 1) Relaxing the problem so as to allow that not all customers are served;
- 2) Conducting a tabu search with few iterations to improve the workdays (i.e., the routes) of the relaxed problem, aiming at making the insertion of all customers not yet served feasible;
- 3) Executing the procedure for generating an initial solution again to insert only the customers not allocated in the first execution.

Let S^1 be the solution space defined by the constraints of the original problem, IP. By relaxing the constraint (1) such that each customer can be visited or not, space S^2 is obtained.

When the customer insertion procedure generates a solution X^1 within space $S^2 \setminus S^1$, the tabu search is conducted, as described in Section 5.2, with few iterations (i.e., with low computational cost), to find a new solution X^2 of smaller cost and that visits the same customers served in X^1 . Cost reduction implies a reduction in the total distance traveled by the vehicles, which “makes room” in the route, and allows for the insertion of customers not yet served by the insertion procedure described before.

In order to find a feasible solution, CPLEX™ is called when the procedure described above fails. The problem is considered to be infeasible only if CPLEX™ is unable to find a feasible solution.

It should be stressed that the constructive heuristic will be called in each node of the branch-and-price tree, where the existence of arc flow variable cuts of the type $x_{ijr}^k = 0$ or $x_{ijr}^k = 1$ requires a preprocessing before executing the initial solution building procedure previously described, which does not consider these cuts. This preprocessing has to be able to generate workdays that consider all the cuts, even if some customers are not served, i.e., even if the solution obtained belongs to space $S^2 \setminus S^1$.

For this, a solution with as few customers as possible is found by solving the elementary minimum cost path problem of each vehicle with the exact dynamic programming algorithm described in Section 5.4, where the cost of traveling an arc is equal to its distance for all vehicles. Each vehicle is treated sequentially and the customers served in the elementary minimum cost path of a vehicle are prevented from being visited by the other vehicles that have not yet been analyzed. In case it is not possible to find a solution to this problem for a vehicle, it can be concluded that no feasible solution exists for the tree node problem.

Only after obtaining workdays which do not violate any cut, the procedure previously described will be executed to insert customers that have not been allocated yet.

5.1.2. Procedure for Verifying the Insertion Feasibility of a Customer into a Workday

It is important for the constructive heuristic to verify at a low computational cost whether the insertion of a customer into a position of a vehicle workday is feasible.

Let WD_k be the workday of vehicle k , pos the insertion position and j the customer to be inserted. The insertion of j in the pos position where customer v of WD_k is positioned is represented in figure 6:

Figure 6: Customer insertion into a vehicle workday.

Let w_i be the accumulated amount of worked time (which is limited to, at most, $W_{max,k}$) from the service start time of the origin (O_1) to the service start time of i . Let $C(r)$ be the set of customers served in route r of the workday. In figure 6, before j insertion, $C(1) = \{h, i, v\}$.

The insertion of customer j in workday WD_k of vehicle k in position pos of customer v is only feasible if the eight statements (verifications) below are true. These verifications are conducted in increasing order of computational cost because if one fails to be true, the verification can be interrupted with an infeasible result.

- 1) Vehicle k of WD_k is authorized to serve customer j ;
- 2) In the best case, it is still possible to serve v after serving j at the beginning of its time window: $a_j + s_j^k + \tilde{t}_{jv}^k \leq b_v$. This is a quick initial verification of violation of the time window of customer v . The accurate calculation of each of the new service start times after the insertion of j is performed during verification 8, for being the verification of the highest computational cost;
- 3) There are arcs (i, j) and (j, v) . In case the digraph is complete, this verification is not necessary;
- 4) Cut $x_{ivr}^k = 1$ does not exist in route r ;
- 5) Cuts $x_{ijr}^k = 0$ and $x_{jvr}^k = 0$ do not exist in route r ;
- 6) If $r \geq 2$, then in the previous route, $r - 1$, the vehicle must not travel the fictitious arc (O_{r-1}, O_r) , i.e., $x_{O_{r-1}O_r}^k = 0$ must be true.
- 7) The maximum load capacity of vehicle k , which visits the set of customers $C(r)$

in route r where the insertion of j occurs, is not violated: $\sum_{i \in \{C(r) \cup j\}} \left(\frac{D_i^{pet}}{Q_k^{pet}} + \right.$

$$\left. \frac{D_i^{can}}{Q_k^{can}} + \frac{D_i^{cel}}{Q_k^{cel}} + \frac{D_i^{bar}}{Q_k^{bar}} \right) \leq 1;$$

- 8) In the resulting workday, each customer is served within its time window and the total worked time does not exceed the maximum time allowed for the vehicle driver.

The computation of item 8 is performed according to the algorithm described in the pseudocode *Feasible_Insertion*(WD_k, pos, j) which returns true if the insertion of customer j in position pos is feasible in relation to time windows and worked time, and false otherwise. It is worth highlighting that the calculation of each service start time and the hours worked by *Feasible_Insertion*(WD_k, pos, j) requires the calculation of the maximum possible displacement value of the O_1 service start time, Δ . Whenever a vehicle reaches a customer before its time window beginning, it has to wait until it starts. This creates a waiting time (i.e., an idle time) which is incorporated to the worked hours and that may be avoided if it is possible to serve the origin later, by displacing the previous service start times and reducing the generated idle time.

Insertion Feasibility Algorithm – verification of the feasibility of inserting a customer into a vehicle workday.

```

1   Boolean Feasible_Insertion( $WD_k, pos, j$ )
2   //Initialization//
3    $WD_k \leftarrow \{j\}$     // $WD_k$  receives  $j$  in position  $pos$ 
4    $p \leftarrow 0$          //initial position of workday  $WD_k$ 
5    $t_p \leftarrow 0$        //service start time in position  $p$  of  $WD_k$ 
6    $w_p \leftarrow 0$        //worked hours in position  $p$  of  $WD_k$ 
7    $\Delta \leftarrow b_{DC}$     // $\Delta = \max.$  service start time displacement at the origin ( $O_1$ )
8   //Calculation and verification//
9   for  $p \leftarrow 1$  to length of  $WD_k$  do
10      if  $WD_k[p-1] = DC$  and  $WD_k[p] = DC$  then //where  $p = 0 = O_1$ 
11          $t_p \leftarrow t_{p-1}$ 
12          $w_p \leftarrow w_{p-1}$ 
13      else
14          $t_p \leftarrow t_{p-1} + s_{p-1}^k + \tilde{t}_{p-1,p}^k$ 
15          $w_p \leftarrow w_{p-1} + s_{p-1}^k + \tilde{t}_{p-1,p}^k$ 
16      end
17      if  $t_p < a_p$  then
18          $t_p \leftarrow a_p$ 
19          $t_{idle} \leftarrow t_p - (t_{p-1} + s_{p-1}^k + \tilde{t}_{p-1,p}^k)$ 
20         if  $\Delta = 0$  then
21             $w_p \leftarrow w_{p-1} + (t_p - t_{p-1})$ 
22         else if  $\Delta > 0$  and  $\Delta \geq t_{idle}$  then
23             $\Delta \leftarrow \Delta - t_{idle}$ 
24            if  $b_p - t_p < \Delta$  then
25                $\Delta \leftarrow b_p - t_p$ 
26            end
27         else if  $\Delta > 0$  and  $\Delta < t_{idle}$  then
28             $w_p \leftarrow w_{p-1} + s_{p-1}^k + \tilde{t}_{p-1,p}^k + (t_{idle} - \Delta)$ 
29             $\Delta \leftarrow 0$ 
30         end
31      else if  $t_p \geq a_p$  and  $t_p \leq b_p$  then
32         if  $b_p - t_p < \Delta$  then
33             $\Delta \leftarrow b_p - t_p$ 
34         end
35      else if  $t_p > b_p$  then
36         return false
37      end
38      if  $w_p > W_{max,k}$  then
39         return false
40      end
41       $t_{p-1} \leftarrow t_p$ 
42       $w_{p-1} \leftarrow w_p$ 
43  end
44  return true

```

5.1.3. Procedure for Verifying the Removal Feasibility of a Customer from a Workday

It is also important for the constructive heuristic to verify whether the removal of a customer from a position of a vehicle workday is feasible with regard to arc flow variable cuts (cuts of the type $x_{ijr}^k = 0$ or $x_{ijr}^k = 1$) and multiple route cuts (see Section 4.2 Multiple Route Cuts for further details). These cuts are additional constraints for the problem that must be respected in every feasible solution and that may be violated by a customer removal from a vehicle workday.

Denote by WD_k the workday of vehicle k , j the customer to be removed and r the route in which customer j is served. The removal of j from WD_k is represented in figure 7:

Figure 7: Customer removal from a vehicle workday.

The removal operation is only feasible if the verifications below are true:

- 1) Cut $x_{ivr}^k = 0$ does not exist in route r ;
- 2) Cuts $x_{ijr}^k = 1$ or $x_{jvr}^k = 1$ do not exist in route r ;
- 3) If $R \geq 2$ and $r < R$ and the removal of customer j from route r transforms the route r into a fictitious route (i.e., j is the only customer served in route r), then no customer can be served in route $r + 1$. If any customer in route $r + 1$ is served by vehicle k , under these conditions, the multiple route cuts are violated after the removal of j from route r .

5.2. Tabu Search

In the following sections, the tabu search algorithm developed is described in detail.

5.2.1. Neighbourhood Structure

The tabu search algorithm is based on moves that define the neighbourhood of a solution. In the problem studied, a move consists of the removal of a customer from a vehicle workday position and the insertion of this customer into a workday position of

another vehicle, i.e., a single insertion move. The internal customer repositioning move, i.e., a single insertion move in which the destination vehicle and the origin vehicle are the same, is also considered to define the neighbourhood of a solution.

5.2.2. Overview of the Tabu Search Algorithm

The search starts with an initial solution given by the constructive heuristic (described in Section 5.1) or by the random solution generator (described in Section 5.2.5); in the latter case, this occurs after a diversification has been performed. Then, the formal local search according to the tabu search framework is performed.

In each local search iteration of the tabu search algorithm, all possible single insertion movements and internal customer repositioning movements are verified; therefore, the entire current solution neighbourhood is examined. For each feasible movement, according to the criteria described in Section 5.1.2 and in Section 5.1.3, whether the movement is tabu (see Section 5.2.3 Tabu Activation Rule) and the total cost of the solution that would be obtained if the movement were performed are verified. After verifying the entire neighbourhood, the best tabu and non-tabu movements that were found are determined.

In case there are chartered vehicles in the available fleet, a use flag is assigned to each chartered vehicle and the cost structure of these vehicles is modified in order to obtain solutions in compliance with their use flags during the local search (Section 5.2.6 describes in details the role of vehicle use flags).

The aspiration criterion adopted is the acceptance of a tabu solution when it globally improves the objective function and its cost is lower than the best non-tabu solution cost.

Whenever a move fails to improve the local minimum solution, it is verified whether the maximum number of iterations without local minimum improvement was reached. In this case, a diversification will be executed with the aim of generating a solution distant from the local minimum solution, which will serve as the basis for initiating a new local search. If there are chartered vehicles in the available fleet, their use flags are updated according to a predetermined strategy before each diversification.

Additionally, whenever a customer reallocation move is conducted during an iteration, the move is inserted into the short-term memory of the tabu search, which consists of a FIFO line of moves whose maximum size is dynamically determined by the algorithm.

After a diversification procedure, the local minimum solution is updated with the random solution generated, and a new local search process is initiated. The tabu search ends when the maximum number of diversifications is reached.

The major parameters of the proposed tabu search are:

- 1) Size of the short-term memory: θ , which is dynamically updated by the algorithm;
- 2) Number of iterations without local minimum solution improvement that triggers diversification, I_{div} ;
- 3) Number of random moves required to diversify a solution;
- 4) Maximum number of attempts made to find a random feasible move during the diversification process;
- 5) Maximum number of diversifications (stopping criterion), Q_{div} .

In figure 8 it is possible to visualize the tabu search framework of the proposed algorithm.

Figure 8: Tabu search algorithm flowchart

A brief description of each flowchart process is given as follows:

- *Procedure 1*: initialization of variables, flags and counters;
- *Procedure 2*: generation of an initial solution from which the local search is initiated;
- *Procedure 3*: verification of current solution neighbourhood and selection of a tabu move if the aspiration criterion is met or selection of a non-tabu move otherwise;
- *Procedure 4*: updating of iteration counters, the current solution and, when necessary, the local minimum and global solution; the tabu short-term memory is updated with the move executed in this procedure;
- *Procedure 5*: updating of chartered vehicles use flags according to the local minimum solution found;
- *Procedure 6*: updating of *Tabu Tenure* according to the number of consecutive iterations that do not yield local minimum solution improvement;
- *Procedure 7*: resets *Tabu Tenure* and iteration counters; clears short-term memory; updates long-term memory with the minimum local solution.

5.2.3. Tabu Activation Rule

The tabu activation rule, which is used to define whether or not a move is tabu, is based on the following move attributes: customer and destination vehicle. If there is a move that has removed the customer from the destination vehicle workday in the short-term memory, then the move under analysis is tabu; otherwise, the move is not tabu.

It is worth emphasizing that the tabu activation rule is active only during the hill climbing process, regardless of the fact that the short-term memory is always updated with the last move performed during the entire local search by the algorithm proposed. This deactivation is important to avoid missing the minimum local solution by the short-term memory role when it is active during the entire search. Consequently, the entire neighbourhood of a solution is explored by the descent heuristic; in particular, the entire neighbourhood of the local minimum solution found.

5.2.4. Dynamic Tabu Tenure

The short-term memory size θ , or *Tabu Tenure*, is one of the most influential parameters of the tabu search algorithm, and it varies during the hill-climbing process in the proposed algorithm. The implemented θ value update approach divides the local search into three stages, and it is similar to the approach used by Brandão (2009).

In the first iteration, the value of the *Tabu Tenure* is defined as: $\theta = \max\left\{\frac{|N|}{2}, 10\right\}$.

After $\left\lfloor \frac{1}{2} \cdot I_{div} \right\rfloor$ consecutive iterations have been performed without local minimum solution improvement, and under the hypothesis that the current solution is far from the local minimum solution, the short-term memory size is reduced to $\theta = \max\left\{\frac{|\theta|}{2}, 5\right\}$ to provide added flexibility to the neighbourhood search.

After $\left\lfloor \frac{2}{3} \cdot I_{div} \right\rfloor$ consecutive iterations have been performed without local minimum solution improvement, the local search is constrained by triplicating the *Tabu Tenure* value in a last attempt to escape from the local minimum solution region in the solution space.

During the local search, if the aspiration criterion is met or if a new local minimum solution is found, the short-term memory size is reset to $\theta = \max\left\{\frac{|N|}{2}, 10\right\}$. It is important to also note that an increase in the value of θ does not change the FIFO line of previously executed moves. However, a reduction in the value of θ corresponds to an elimination of the oldest moves stored until the desired value of θ is obtained.

5.2.5. Long-term Memory and Diversification

A frequency-based long-term memory to perform a biased and more effective diversification that guides the search to regions that markedly contrast with those examined so far, as suggested by Glover (1989,1990), was implemented. This long-term memory, represented by a two-dimensional matrix F , stores for each customer and vehicle pair (i, k) the number of local minimum solutions already found in which customer i is served by vehicle k .

In general, diversification is performed by randomly choosing two distinct vehicles, k_o and k_d . Next, a position pos_o of the k_o workday containing a customer is randomly chosen, as well as a position pos_d of the k_d workday. If the customer insertion move

from position pos_o of k_o to position pos_d of k_d is feasible, the random reallocation is conducted.

This random reallocation procedure of customers is repeated until the specified number of random moves is reached. If no feasible random move has been found after the specified maximum number of attempts have been made, the tabu search is ended.

Given the value of Q_{div} , defined as the maximum number of diversifications, the long-term memory in the first $\left\lfloor \frac{Q_{div}}{2} \right\rfloor$ diversifications is not used, and vehicles and customers (or positions) are randomly selected, according to a uniform probability distribution, from a pool of vehicles and a pool of customers without replacement to define a diversification move.

After $\left\lfloor \frac{Q_{div}}{2} \right\rfloor$ diversifications have been performed and with the long-term memory filled with a considerable amount of past data, the diversification becomes biased. Hence, among the four steps required to define a move (in order), namely the selection of k_o , pos_o , k_d and pos_d , the selection of position pos_o occupied by customer i during one workday by vehicle k_o is based on a probability distribution in which the probability to select customer i or position pos_o , $P(i, k_o)$, is directly proportional to the frequency in which it was served by vehicle k_o according to the local minimum solutions already found. In contrast, in the other three steps used to define a move, the selection remains based on a uniform probability distribution.

Let $f_{ik_o} = F(i, k_o)$ be the number of times that vehicle k_o serves customer i in local minimum solutions already found, and let $N' \subseteq N$ be the set of customers served in workday of vehicle k_o that may be served by other vehicles. The probability $P(i, k_o)$ of selecting customer i served by vehicle k_o is defined as:

$$P(i, k_o) = \begin{cases} 0, & \text{if } |K_i| = 1 \\ \frac{f_{ik_o}}{\sum_{j \in N'} f_{jk_o}}, & \text{otherwise} \end{cases} .$$

5.2.6. Chartered Vehicles Treatment

The tabu search framework previously described performs well when all vehicles are owned vehicles and, therefore, only a variable cost, depending on the total distance travelled and on the vehicle type, is incurred.

However, the availability of chartered vehicles with only fixed costs induces a behaviour in the local search scheme that prevents good solutions from being found.

This occurs because the evaluation criterion of a move is the total cost of the solution obtained after the move execution is computed. According to this search direction criterion that is responsible for guiding local search, the removal of a customer from an owned vehicle workday and the insertion of this customer into a workday of a chartered vehicle in use is advantageous, moves that exchange customers between chartered vehicles are generally irrelevant and moves that remove a customer from a chartered vehicle workday and insert this customer into an owned vehicle workday are generally disadvantageous.

If no procedure concerning the use of chartered vehicles is conducted, the local minimum solution found will have chartered vehicles with “chaotic” routes and underutilized owned vehicles. Thus, the tabu search algorithm has to decide whether to use or not a chartered vehicle (this decision is modelled in IP as variables $v^k \in \{0,1\}$ for every $k \in K^c$) and, once a chartered vehicle is selected to be used, the tabu search algorithm has to effectively define its workday.

In general, charter costs are high enough to make them non-competitive regarding the costs incurred by owned vehicles (i.e., an optimal solution is supposed to have as few chartered vehicles as possible serving customers, and once one is used, the chartered vehicle workday must serve as many customers as possible). However, there may be chartered vehicles with competitive fixed costs such that they are used in an optimal solution.

Concerning the decision to be made of whether using or not a chartered vehicle, a vector of flags, which is denoted by $vhcFlag[k]$ and has one flag for each vehicle $k \in K^c$, is considered where $vhcFlag[k] = true$ if the chartered vehicle k should be used, and $vhcFlag[k] = false$ otherwise. It is worth emphasizing that $vhcFlag[k] = true$ implies that the use of the chartered vehicle k is desired but not mandatory.

Concerning the additional requirement of evaluating distances when performing moves involving chartered vehicles to avoid “chaotic” routes, it is considered that every vehicle $k \in K^c$ has only a variable cost, VC_k , instead of having only a fixed cost, FC_k . Furthermore, it is considered that the value of the variable cost VC_k of vehicle $k \in K^c$ depends on the value of the vehicle use flag as follows: if $vhcFlag[k] = false$ then $VC_k = FC_k$, and $VC_k = 0.5 \min\{VC_v, v \in K^o\} \cdot \frac{FC_k}{\max\{FC_v, v \in K^c\}}$ otherwise.

In summary, when $vhcFlag[k] = false$, the chartered vehicle k has a high variable cost that favours the removal of customers from its workday during local search. When

$hcFlag[k] = true$, the number of customers served by vehicle k is maximized during local search by assuming that it has a variable cost that is substantially lower than the lowest variable cost among owned vehicles.

The proposed cost structure for chartered vehicles when evaluating neighbour solutions has the effect, among others, of guiding local search to regions of the solution space where the values of the binary variables v^k match the corresponding $vhcFlag[]$ vector values.

The general heuristic approach proposed when there are chartered vehicles in the available fleet is to consider the solution space of IP as a set of $2^{|K^c|}$ subspaces (i.e., one subspace for each combination of v^k binary variables values) and test $vhcFlag[]$ vectors before diversifications, such that the tabu search learns with the corresponding subspaces minimum local solutions how to move to promising regions of the solution space.

The procedure to set the values of $vhcFlag[k] \forall k \in K^c$ before each diversification, which corresponds to Process 5 of the flowchart depicted in figure 8, is described in details in pseudocode *Update_Flags*. The pseudocode *Initialize_Vars* is executed immediately after the tabu search starts and describes how variables, flags and counters required by the *Update_Flags* procedure are initialized. The pseudocode *Generate_Random_Vector* contains the procedure to generate a random vector of vehicle use flags. Also, the following procedures and variables need to be defined:

- div : number of the current diversification ($1 \leq div \leq Q_{div}$).
- X : local minimum solution found by the tabu search algorithm after diversification div . The cost of the solution X according to the objective function of IP is denoted by $X.cost$.
- $\hat{X}$: current best solution found by the tabu search algorithm.
- $divergence(X, VhcFlag, k)$: returns true if $VhcFlag[k] = false$ and the vehicle k serves any customer in solution X , and returns false in the other cases.
- $divergenceCount$: number of consecutive diversifications performed with vehicle use and flag divergence in local minimum solutions.
- $divergenceCount[k]$: vector with the number of divergences found in local minimum solutions regarding the use of vehicle k and its flag.
- $Reset(divergenceCount[])$: sets every position $k \in K^c$ of the vector $divergenceCount[]$ to 0.

- *GetMaxDivVhc()*: returns the vehicle k corresponding to the position in vector *divergenceCount*[] of maximum value, i.e., returns the vehicle with the maximum number of divergences found.
- *Reset(vhcFlag[])*: sets every position $k \in K^c$ of the vector *vhcFlag*[] to false.
- *Copy(origin[], destination[])* : copies vector *origin*[] to vector *destination*[].
- X^1 : best solution found during the first phase of the algorithm.
- *vhcBaseFlag*[]: vector of flags obtained after the end of the first phase of the algorithm.
- *LB()*: returns the sum of the fixed costs of the activated chartered vehicles, i.e., $LB = \text{sum}\{FC_k | VhcFlag[k] = \text{true}, k \in K^c\}$.
- X^2 : best solution found during the second phase of the algorithm with a particular vehicle flag activated.
- $k_{activated}$: vehicle whose flag was activated and whose use is under analysis by the algorithm during the second phase.
- *accordance*($X, VhcFlag, k$) : returns true if $VhcFlag[k] = \text{true}$ and the vehicle k serves any customer in solution X , and returns false in the other cases.
- *accordanceCount*: number of diversifications during the second phase with vehicle use and flag accordance in local minimum solutions.
- *GetNextVehicle()*: returns the next vehicle k of the list of chartered vehicles where $vhcUseFlag[k] = \text{false}$ and $LB + FC_k < \hat{X}.cost$. Returns *null* if no vehicle is found with these two conditions satisfied.
- *benefit*[]: vector of flags for chartered vehicles which identifies whether the activation of a chartered vehicle generated a solution with a cost lower than the cost of the best solution found during the first phase or not.
- *Reset(benefit[])*: sets every position $k \in K^c$ of the vector *benefit*[] to false.

Below are presented the pseudocodes *Initialize_Vars*, *Update_Flags* and *Generate_Random_Vector* followed by a brief description of them.

Variable Initialization: initialization of counters, vectors and flags.

```
Initialize_Vars()  
1 //Initialization  
2 Reset(vhcFlag[ ])   
3 //1st phase  
4 divergenceCount  $\leftarrow$  0  
5 Reset(divergenceCount[ ])   
6 //2nd phase  
7 accordanceCount  $\leftarrow$  0  
8  $k_{activated} \leftarrow null$   
9 Reset(benefit[ ])   
10 //3rd phase  
11 lbCheckSuccess  $\leftarrow false$ 
```

Flag Update Algorithm: pseudocode of Procedure 5 of the tabu search algorithm presented in figure 8.

```
Update_Flags()  
1 //Phase 1: activates chartered vehicles that should be used  
2 if  $div < \lfloor 0.4Q_{div} \rfloor$  then  
3   //Updates divergence counters  
4   divergenceFound  $\leftarrow false$   
5   for each  $k \in K^c$  do  
6     if  $divergence(X, VhcFlag, k) = true$  then  
7       divergenceCount[k]  $\leftarrow$  divergenceCount[k] + 1  
8       divergenceFound  $\leftarrow true$   
9     end  
10  end  
11  if divergenceFound = true then  
12    divergenceCount  $\leftarrow$  divergenceCount + 1  
13  else  
14    divergenceCount  $\leftarrow$  0  
15  end  
16  //Check the number of consecutive diversifications with divergence  
17  if  $divergenceCount > \left\lfloor \frac{0.4Q_{div}}{|K^c|+1} \right\rfloor$  then  
18    VhcFlag[GetMaxDivVhc()]  $\leftarrow true$   
19    divergenceCount  $\leftarrow$  0  
20    Reset(divergenceCount[ ])   
21  end  
22  
23 //Preparation for phase 2  
24 else if  $div = \lfloor 0.4Q_{div} \rfloor$  then  
25   //Stores phase 1 data for future analyzes  
26   Copy(VhcFlag, VhcBaseFlag)  
27    $X^1 \leftarrow \hat{X}$   
28   //Select an inactivated vehicle to be analysed in phase 2  
29   for each  $k \in K^c$  do  
30     if  $vhcFlag[k] = false$  then  
31       //Check if the activation is promising
```



```

32         if  $LB() + FC_k < \hat{X}.cost$  then
33              $vhcFlag[k] \leftarrow true$ 
34              $k_{activated} \leftarrow k$ 
35              $X^2 \leftarrow X^1$ 
36             break
37         end
38     end
39 end
40
41 //Phase 2: tests each inactivated vehicle
42 else if  $div > \lfloor 0.4Q_{div} \rfloor$  and  $div < \lfloor 0.7Q_{div} \rfloor$  then
43     if  $vhcActivated \neq null$  then
44         //Updates 2nd phase solution when improved
45         if  $X.cost < X^2.cost$  then
46              $X^2 \leftarrow X$ ;
47         end
48         //Updates accordance counter
49         if  $accordance(X, VhcFlag, k_{activated}) = true$  then
50              $accordanceCount \leftarrow accordanceCount + 1$ 
51         else
52              $accordanceCount \leftarrow 0$ 
53         end
54         //Check the number diversifications with accordance
55         if  $accordanceCount > \lfloor \frac{0.3Q_{div}}{|K^c|} \rfloor$  then
56             //Deactivate vehicle selected to test another vehicle
57              $vhcFlag[k_{activated}] \leftarrow false$ 
58             //Test if the activation improved the 1st phase best solution
59             if  $X^2.cost < X^1.cost$  then
60                  $benefit[k_{activated}] \leftarrow true$ 
61             else
62                 //Resets data for the next vehicle
63                  $accordanceCount \leftarrow 0$ ;
64                  $k_{activated} \leftarrow GetNextVhc()$ 
65                 if  $k_{activated} \neq null$  then
66                      $X^2 \leftarrow X^1$ 
67                      $vhcFlag[k_{activated}] \leftarrow true$ 
68                 end
69             end
70         end
71
72 //Phase 3: activates every vehicle with 1st phase cost reduction during 2nd phase
73 else if  $div = \lfloor 0.7Q_{div} \rfloor$  then
74     //Check if the activation of all beneficial vehicles is promising
75     for each  $k \in K^c$  do
76         if  $vhcBaseFlag[k] = true$  then
77              $vhcFlag[k] \leftarrow true$ 
78         else
79             if  $benefit[k] = true$  then
80                  $vhcFlag[k] \leftarrow true$ 
81             else
82                  $vhcFlag[k] \leftarrow false$ 
83             end

```

```

84         end
85     end
86     if  $LB() < \hat{X}.cost$  then
87          $lbCheckSuccess \leftarrow true$ 
88     else
89          $lbCheckSuccess \leftarrow false$ 
90     end
91     //In case the activation is not promising, randomly activates vehicles
92     if  $lbCheckSuccess = false$  then
93          $GenerateRandomVector()$ 
94     end
95
96 else if  $div > \lfloor 0.7Q_{div} \rfloor$  and  $div < \lfloor 0.75Q_{div} \rfloor$  then
97     if  $lbCheckSuccess = false$  then
98          $GenerateRandomVector()$ 
99     end
100
101 //Phase 4: randomly activates vehicles
102 else
103      $GenerateRandomVector()$ 
104 end

```

Random Vector Generation: procedure to generate a random vector of vehicle use flags.

```

    Generate_Random_Vector()
1   $counter \leftarrow 0$ 
2  while true do
3      for each  $k \in K^c$  do
4          if  $vhcBaseFlag[k] = true$  then
5               $vhcFlag[k] \leftarrow true$ 
6          else
7              if  $benefit[k] = true$  then
8                   $vhcFlag[k] \leftarrow randomBooleanGenerator()$ 
9              else
10                  $vhcFlag[k] \leftarrow false$ 
11             end
12         end
13     end
14     if  $LB() < \hat{X}.cost$  then
15         return
16     else
17          $counter \leftarrow counter + 1$ 
18         //Use flags of the 1st phase after 50 unsuccessful attempts
19         if  $counter = 50$  then
20              $copy(vhcBaseFlag, vhcFlag)$ 
21             return
22         end
23     end
24 end

```

The tabu search algorithm with regard to the update of $vhcFlag[]$ values is divided in four phases, having as reference the number of diversifications executed. The number of diversifications executed in the first, second, third and fourth phases corresponds to 40%, 30%, 5% and 25% of Q_{div} , respectively.

In order to improve the effectiveness of the search during phases 2, 3 and 4, the subspace lower bound, $LB = \sum\{FC_k | VhcFlag[k] = true, k \in K^c\}$, is always computed when defining a new $vhcFlag[]$ to avoid guiding the search to subspaces that certainly do not have solutions better than the incumbent solution (when $LB \geq \hat{X}.cost$).

Briefly, according to pseudocode *Update_Flags*, the first phase is responsible for identifying chartered vehicles that are supposed to be used in an optimal solution. These vehicles are activated (use flag is set as true) when they are used in local minimum solutions despite of their high associated variable cost. The vehicles activated during the first phase remain activated during the entire search.

The second phase individually tests every chartered vehicle that was not activated during the first phase by activating it and then verifying if a local minimum solution better than the best solution found during the first phase was generated. In case this is true, the vehicle use is identified as promising by setting $benefit[k] \leftarrow true$, and false otherwise.

The third phase guides the search to the subspace where every promising vehicle, identified during the second phase, is activated. In case the use of all activated vehicles certainly do not generate a solution better than the incumbent solution (by computing the corresponding subspace lower bound), then a random activation of vehicles is performed according to the procedure of the fourth phase.

At last, during the fourth phase, random $vhcFlag[]$ vectors are generated in an attempt to test different combinations of use of chartered vehicles. It is worth emphasizing that the vehicles activated during the first phase are certainly activated in the fourth phase and the vehicles randomly activated are only those which are promising, i.e. vehicles that were not identified as promising during the second phase are never activated during the fourth phase.

5.3. Extensive Formulation and Column Generation

It is possible to provide an alternative formulation to the routing problem after observing that IP has primal block diagonal structure, as shown in figure 9. This structure is composed by linking constraint (1) and independent constraint blocks from (2) to (13) for each vehicle k , corresponding to workdays.

Figure 9: IP primal block diagonal structure.

Let P^k be the set of all feasible workdays p of vehicle k . The alternative mathematical formulation of the problem, denoted Master Problem (MP), is as follows:

(MP)

$$\text{Min} \sum_{k \in K} \sum_{p \in P^k} c_p^k \cdot \lambda_p^k$$

s. t.

$$\sum_{k \in K} \sum_{p \in P^k} a_{ip}^k \cdot \lambda_p^k = 1 \quad i \in N \quad (16)$$

$$\sum_{p \in P^k} \lambda_p^k = 1 \quad k \in K \quad (17)$$

$$\lambda_p^k \in \{0,1\} \quad k \in K, p \in P^k \quad (18)$$

where:

- λ_p^k is a binary decision variable that is equal to 1 if and only if workday p is chosen for vehicle k and 0 otherwise;
- $c_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k$ for $k \in K^o, p \in P^k$ will be the cost of workday p of owned vehicle k ;
- $c_p^k = \begin{cases} FC^k & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r \\ 0, & \text{otherwise} \end{cases}$ for $k \in K^c, p \in P^k$ will be the cost of workday p of chartered vehicle k ;
- $a_{ip}^k = \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ijrp}^k$ for $k \in K, i \in N, p \in P^k$ is the number of times customer i is visited by vehicle k on workday p . It is important to notice that $a_{ip}^k \in \{0,1\}$ because a customer can be visited only once in a workday. However, in case constraints (2) to (15) are relaxed, in order to compute a valid lower bound to IP by allowing cycles in a workday and by allowing a customer to be visited more than once (when the workday is not an elementary path from O_1 to O_{R+1}), then $a_{ip}^k \in \mathbb{N}$;
- $x_{ijrp}^k \in \{0,1\}$ for $k \in K, r = 1, \dots, R, (i,j) \in A, p \in P^k$ is equal to 1 if vehicle k along route r travels the arc (i,j) on workday p ; $x_{ijrp}^k = 0$ otherwise.

Constraints (16) impose that each customer must be served only once, constraints (17) impose that exactly one duty is selected for each vehicle and constraints (18) ensure that the solution is integer.

It is important to notice that MP has an exponential number of variables. Therefore, we resort to column generation (Lübbecke and Desrosiers, 2005) to solve the linear relaxation of the master problem (LMP). It is also important to remark that LMP is equivalent to the Dantzig Wolfe reformulation operated on constraints (2)...(15) of IP, where the resulting constraints (16) are the so-called linking constraints and constraints (17) are the convexity constraints.

In LMP, the set partitioning constraints (16) are replaced with set covering constraints: $\sum_{k \in K} \sum_{p \in P^k} a_{ip}^k \cdot \lambda_p^k \geq 1$ for $i \in N$. Indeed, when the triangular inequality holds on the cost matrix and load splitting is not allowed, no optimal solution visits any customer more than once. Consequently, the two formulations are equivalent. Set covering constraints are preferable to set partitioning constraints because the associated dual variables are restricted to assume non-negative values.

To solve LMP, it is applied the column generation technique. Therefore, it is considered a subset of workdays for each vehicle obtaining a restricted master problem (LRMP) so that a feasible solution to LMP exists. LMP is solved and the dual variables associated with constraints (16) and (17) are denoted π and σ , respectively. A pricing subproblem for every vehicle k is solved and the algorithm iterates until no column with negative reduced cost exists for every vehicle.

The pricing subproblem consists of finding a workday that potentially contributes to reducing the objective function value of the LRMP, that is, a workday whose reduced cost of the new variable λ_p^k is negative.

From the linear programming theory, the reduced cost $\bar{c}_p^k$ of variable λ_p^k (column) is:

$\bar{c}_p^k = c_p^k - \sum_{l=1}^n a_{lp}^k \cdot \pi_l - 1 \cdot \sigma_k$, where $l = (1, \dots, n)$ is the customer index (MP row) regarding constraints (16).

It is possible to rewrite the above equation in function of IP tableaux coefficients and variables. For an owned vehicle $k \in K^o$, given that:

$$c_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k \text{ for } k \in K^o, p \in P^k \text{ and}$$

$$a_{lp}^k = \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \text{ for } k \in K, l = (1, \dots, n), p \in P^k$$

then,

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k - \sum_{l=1}^n \pi_l \cdot \left(\sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \right) - \sigma_k ;$$

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k - \sum_{l=1}^n \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} \pi_l \cdot x_{ljrp}^k - \sigma_k ;$$

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k - \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} \sum_{l=1}^n \pi_l \cdot x_{ljrp}^k - \sigma_k ;$$

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k - \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} \sum_{i \in N} \pi_i \cdot x_{ijrp}^k - \sigma_k ;$$

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} c_{ijr}^k \cdot x_{ijrp}^k - \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} \left(\sum_{i \in N} \pi_i \cdot x_{ijrp}^k + \pi_{O_r} \cdot x_{O_r j r p}^k \right) - \sigma_k \text{ where } \pi_{O_r} = 0;$$

$$\bar{c}_p^k = \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (c_{ijr}^k - \pi_i) \cdot x_{ijrp}^k - \sigma_k.$$

Therefore, the pricing subproblem of an owned vehicle, denoted SP^o , has the objective function $\text{Min} \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (c_{ijr}^k - \pi_i) \cdot x_{ijr}^k - \sigma_k$, where $\pi_{O_r} = 0$ for $r = 1, \dots, R$, subject to constraints (2) to (15) regarding vehicle k .

It is also possible to rewrite the reduced cost expression of a chartered vehicle variable (column) in function of IP tableaux coefficients and variables. Given that:

$$c_p^k = \begin{cases} FC^k & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r \\ 0, & \text{otherwise} \end{cases} \quad \text{for } k \in K^c, p \in P^k \text{ and}$$

$$a_{lp}^k = \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \quad \text{for } k \in K, l = (1, \dots, n), p \in P^k$$

then,

$$\bar{c}_p^k = \begin{cases} FC^k - \sum_{l=1}^n \pi_l \cdot \left(\sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \right) - \sigma_k, & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r; \\ 0 - \sum_{l=1}^n \pi_l \cdot \left(\sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \right) - \sigma_k, & \text{otherwise} \end{cases};$$

$$\bar{c}_p^k = \begin{cases} FC^k - \sum_{l=1}^n \pi_l \cdot \left(\sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} x_{ljrp}^k \right) - \sigma_k, & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r; \\ 0 - 0 - \sigma_k, & \text{otherwise} \end{cases};$$

$$\bar{c}_p^k = \begin{cases} FC^k - \sum_{l=1}^n \sum_{r=1}^R \sum_{j \in N \cup \{O_{r+1}\}} \pi_l \cdot x_{ljrp}^k - \sigma_k, & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r; \\ -\sigma_k, & \text{otherwise} \end{cases};$$

$$\bar{c}_p^k = \begin{cases} FC^k + \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (-\pi_i) \cdot x_{ijrp}^k - \sigma_k, & \text{if } \exists x_{O_r j r p}^k = 1, \forall j \in N, \forall r \\ -\sigma_k, & \text{otherwise} \end{cases}$$

where $\pi_{O_r} = 0$.

The pricing subproblem of a chartered vehicle, denoted SP^c , has the objective function $\text{Min} \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (-\pi_i) \cdot x_{ijr}^k + FC^k \cdot v^k - \sigma_k$, where $\pi_{O_r} = 0$ for $r = 1, \dots, R$, subject to constraints (2) to (15) regarding vehicle k .

5.3.1. Initialization

The LRMP of each tree node is initialized with a feasible solution by calling the constructive heuristic, followed by a tabu search of few iterations to obtain a good initial solution at low computational cost. In most cases, the number of columns generated for LRMP optimality is significantly reduced with a good initial solution (good initial set of columns).

5.3.2. Column Generation Lower Bounding and Anticipated Finalization

In some cases, it is possible to interrupt the column generation scheme before obtaining the optimal solution with the calculation of a lower bound, LB_{CG} , according to expression: $LB_{CG} = Z_{LRMP} + \sum_{k=1}^{|K|} Z_{SP,k}$ where $Z_{SP,k}$ is the optimal solution or a lower bound of the subproblem of vehicle k . A detailed deduction of the expression of the lower bound in the column generation technique can be obtained in Desrosiers and Lübbecke (2005).

If the LB_{CG} value in an iteration is higher than the objective value of the best integer solution known in the tree (incumbent solution), the generation of new columns is stopped and the objective function value of the LRMP is returned, since it is a case of pruning by bound.

The following table shows column generation data of an example where the node is pruned by bound before iterating until its optimal solution. In iteration 48, the column generation dual bound (112.6228195) is greater than the objective value of the incumbent solution (112.4775785), therefore the node may be pruned by bound, saving the computational cost of the remaining 14 iterations. As expected, after solving linear relaxation to optimality, the node is indeed pruned by bound since $114.1355953 > 112.4775785$.

Table 1: Column generation iterations data and branch-and-price incumbent solution objective value in an example.

Iteration	C. G. Primal Bound	C. G. Dual Bound	Gap %	B&P Primal Bound
1	206.753384	-805.1013661	489.4%	112.4775785
2	206.753384	-882.3240218	526.8%	112.4775785
3	206.753384	-768.4942949	471.7%	112.4775785
4	206.753384	-667.6338205	422.9%	112.4775785
5	206.753384	-777.5674923	476.1%	112.4775785
6	206.753384	-734.3676056	455.2%	112.4775785
7	206.753384	-643.9664741	411.5%	112.4775785
8	206.753384	-610.0550647	395.1%	112.4775785
9	206.753384	-728.6851036	452.4%	112.4775785
10	206.753384	-210.6172595	201.9%	112.4775785
11	202.9459891	-153.9985891	175.9%	112.4775785
12	187.3846854	-213.2927196	213.8%	112.4775785
13	159.8958666	-95.24707021	159.6%	112.4775785
14	155.7520304	-35.74969744	123.0%	112.4775785
15	155.746342	-3.494835044	102.2%	112.4775785
16	153.1785893	10.22910263	93.3%	112.4775785
17	146.9321209	30.95533279	78.9%	112.4775785

18	144.3691309	50.83810973	64.8%	112.4775785
19	136.9486174	71.67284913	47.7%	112.4775785
20	135.7900311	83.06989489	38.8%	112.4775785
21	135.3541523	40.12413383	70.4%	112.4775785
22	133.6805759	79.635277	40.4%	112.4775785
23	129.9548344	85.15943897	34.5%	112.4775785
24	129.1348896	94.2210199	27.0%	112.4775785
25	123.5087978	74.92043286	39.3%	112.4775785
26	122.2791023	81.06579877	33.7%	112.4775785
27	119.8012507	89.62332147	25.2%	112.4775785
28	119.5850446	94.83860648	20.7%	112.4775785
29	119.3139441	102.3657768	14.2%	112.4775785
30	119.2552997	104.0618706	12.7%	112.4775785
31	118.688924	108.3776403	8.7%	112.4775785
32	118.4502473	110.5559996	6.7%	112.4775785
33	116.4338693	106.0584488	8.9%	112.4775785
34	114.326598	92.79177152	18.8%	112.4775785
35	114.326598	104.9881711	8.2%	112.4775785
36	114.326598	94.42326069	17.4%	112.4775785
37	114.326598	105.3547351	7.8%	112.4775785
38	114.326598	109.6248009	4.1%	112.4775785
39	114.326598	107.3593726	6.1%	112.4775785
40	114.326598	108.2464452	5.3%	112.4775785
41	114.326598	110.3464607	3.5%	112.4775785
42	114.326598	107.9129988	5.6%	112.4775785
43	114.326598	105.2502426	7.9%	112.4775785
44	114.326598	111.8752729	2.1%	112.4775785
45	114.326598	111.8629614	2.2%	112.4775785
46	114.326598	112.0895227	2.0%	112.4775785
47	114.326598	108.7687739	4.9%	112.4775785
48	114.326598	112.6228195	1.5%	112.4775785
49	114.326598	112.631691	1.5%	112.4775785
50	114.1964473	113.7133117	0.4%	112.4775785
51	114.1964473	111.7723072	2.1%	112.4775785
52	114.1964473	112.7374857	1.3%	112.4775785
53	114.1964473	113.6637336	0.5%	112.4775785
54	114.1355953	110.1622955	3.5%	112.4775785
55	114.1355953	113.3298805	0.7%	112.4775785
56	114.1355953	112.2067576	1.7%	112.4775785
57	114.1355953	113.8172575	0.3%	112.4775785
58	114.1355953	113.3533733	0.7%	112.4775785
59	114.1355953	113.9793087	0.1%	112.4775785
60	114.1355953	114.0061923	0.1%	112.4775785
61	114.1355953	114.0673088	0.1%	112.4775785
62	114.1355953	114.1355953	0.0%	112.4775785

In figures 10 and 11 it is possible to visualize iterations data presented in table 1.

Figure 10: Example of column generation iterations data.

Figure 11: Example of column generation last iterations data.

5.3.3. Stabilization

A well-known column generation property is that the values of the dual variables do not smoothly converge to their respective optima but significantly oscillate with seemingly no regular pattern. This behaviour is undesirable, as it represents a major efficiency issue.

Several stabilization methods that attempt to accelerate convergence have been proposed in the column generation literature.

Two different and widely used stabilization techniques, namely the interior point technique described by Rousseau et al. (2007) and the method commonly referred to as BoxPen stabilization, proposed by Merle et al. (1999), were implemented and compared with regard to the present problem.

The interior point technique described by Rousseau et al. (2007) is a method that addresses degeneracy and convergence difficulties by selecting a dual solution inside the dual optimal space rather than retrieving an extreme point. To achieve the centralization of dual values, several extreme points of the optimal dual polyhedron are generated, and the interior point is computed as a convex combination of these extreme points.

With a different approach, the BoxPen method proposed by Merle et al. (1999), combines two other stabilization techniques. The first technique, described by Marsten et al. (1975), consists of defining a box around the previous dual solution and modifying the master problem so that the feasible dual space is limited to the area defined by these boxes. The second technique, proposed by Kim et al. (1995), is to adapt the master problem so that the distance separating a dual solution from the previous dual solution is linearly penalized. In summary, the BoxPen method imposes soft limits on the dual variables and a penalty in the objective when dual variables take values outside of these limits (box).

Computational results have shown that the BoxPen method typically requires fewer iterations and a lower overall computing time to solve the LMP when compared to the interior point stabilization technique, and therefore, it was the stabilization method chosen for the present problem.

Regarding the BoxPen method implementation for the present study, in figures 12 and 13 it is possible to see a representation of the dual boxes around π_i and σ_k dual variables as well as the linear penalty in the objective function when they lie outside of the intervals $[d_i^-, d_i^+]$ and $[d_k^-, d_k^+]$, respectively. This linear penalty is defined by an angular coefficient ε and by the value of a ω variable, which is the measure of how far the dual solution is from the closest border of the dual box.

Figure 12: Dual box and dual linear penalty for a customer dual variable

Figure 13: Dual box and dual linear penalty for a vehicle dual variable

The primal linear restricted master problem, obtained according to the procedure described in Section 5.3 and denoted by LRMP, is presented below. The dual variables associated with constraints (19) and (20) are denoted π and σ , respectively.

LRMP – Primal

$$\text{Min} \sum_{k \in K} \sum_{p \in P^k} c_p^k \cdot \lambda_p^k$$

s. t.

$$\sum_{k \in K} \sum_{p \in P^k} a_{ip}^k \cdot \lambda_p^k \geq 1 \quad i \in N \quad (19) \quad \pi_i$$

$$\sum_{p \in P^k} \lambda_p^k = 1 \quad k \in K \quad (20) \quad \sigma_k$$

$$\lambda_p^k \geq 0 \quad k \in K, p \in P^k \quad (21)$$

Constraints (19) impose that each customer must be served at least once, constraints (20), as constraints (17) of the MP, impose that exactly one duty is selected for each vehicle and constraints (21) are the convexity constraints.

Based on the linear restricted master problem primal formulation, the dual of the LRMP is defined as:

LRMP – Dual

$$\text{Max} \sum_{i \in N} \pi_i + \sum_{k \in K} \sigma_k$$

s. t.

$$\sum_{i \in N} a_{ip}^k \pi_i + \sum_{k \in K} b_p^k \sigma_k \leq c_p^k \quad k \in K, p \in P^k \quad (22) \quad \lambda_p^k$$

$$\pi_i \geq 0 \quad i \in N \quad (23)$$

$$\sigma_k \text{ unrestricted} \quad k \in K \quad (24)$$

According to the BoxPen method, the modified dual of the LRMP is obtained after incorporating the linear penalties for leaving the dual boxes in the objective function which are based on the new variables ω , that compute the distances separating dual variables values from the corresponding box borders, as well as on new constraints to ensure that these distances are computed correctly.

Below is presented the modified dual of the LRMP, obtained after adding the new variables and constraints and after modifying the objective function.

LRMP – Modified Dual

$$\text{Max} \sum_{i \in N} \pi_i + \sum_{k \in K} \sigma_k + \sum_{i \in N} (-\varepsilon_i^- \omega_i^- - \varepsilon_i^+ \omega_i^+) + \sum_{k \in K} (-\varepsilon_k^- \omega_k^- - \varepsilon_k^+ \omega_k^+)$$

s. t.

$$\sum_{i \in N} a_{ip}^k \pi_i + \sum_{k \in K} b_p^k \sigma_k \leq c_p^k \quad k \in K, p \in P^k \quad (25) \quad \lambda_p^k$$

$$\pi_i \geq 0 \quad i \in N \quad (26)$$

$$\sigma_k \text{ unrestricted} \quad k \in K \quad (27)$$

$$\pi_i + \omega_i^- \geq d_i^- \quad i \in N \quad (28) \quad y_i^-$$

$$\pi_i - \omega_i^+ \leq d_i^+ \quad i \in N \quad (29) \quad y_i^+$$

$$\sigma_k + \omega_k^- \geq d_k^- \quad k \in K \quad (30) \quad y_k^-$$

$$\sigma_k - \omega_k^+ \leq d_k^+ \quad k \in K \quad (31) \quad y_k^+$$

$$\omega_i^- \geq 0 \quad i \in N \quad (32)$$

$$\omega_i^+ \geq 0 \quad i \in N \quad (33)$$

$$\omega_k^- \geq 0 \quad k \in K \quad (34)$$

$$\omega_k^+ \geq 0 \quad k \in K \quad (35)$$

Constraints (28), (29), (30) and (31) ensure that the distance ω separating the dual variable value from the box border when the dual variable lies outside the box is computed correctly. If the dual variable value lies inside the box, then ω^+ and ω^- are equal to zero according to the objective function, whose value is maximized.

Constraints (32), (33), (34) and (35) define the solution space of the new dual variables. After obtaining the modified dual of the linear restricted master problem, the modified primal of the linear restricted master problem is defined as follows:

LRMP – Modified Primal

$$\text{Min} \sum_{k \in K} \sum_{p \in P^k} c_p^k \cdot \lambda_p^k + \sum_{i \in N} (-d_i^- y_i^- + d_i^+ y_i^+) + \sum_{k \in K} (-d_k^- y_k^- + d_k^+ y_k^+)$$

s. t.

$$\sum_{k \in K} \sum_{p \in P^k} a_{ip}^k \cdot \lambda_p^k - y_i^- + y_i^+ \geq 1 \quad i \in N \quad (36)$$

$$\sum_{p \in P^k} \lambda_p^k - y_k^- + y_k^+ = 1 \quad k \in K \quad (37)$$

$$y_i^- \leq \varepsilon_i^- \quad i \in N \quad (38)$$

$$y_i^+ \leq \varepsilon_i^+ \quad i \in N \quad (39)$$

$$y_k^- \leq \varepsilon_k^- \quad k \in K \quad (40)$$

$$y_k^+ \leq \varepsilon_k^+ \quad k \in K \quad (41)$$

$$\lambda_p^k \geq 0 \quad k \in K, p \in P^k \quad (42)$$

Where y_i^- , y_i^+ , y_k^- and y_k^+ are the dual variables associated with constraints (28), (29), (30) and (31) of the modified dual LRMP, respectively.

Denote by P the Primal LRMP, by D the Dual LRMP, by $\tilde{D}$ the Modified Dual LRMP and by $\tilde{P}$ the Modified Primal LRMP. Also, denote by (λ_p^*) , (π_i^*, σ_k^*) , $(\tilde{\lambda}_p^*, y_i^{-*}, y_i^{+*}, y_k^{-*}, y_k^{+*})$ and $(\tilde{\pi}_i^*, \tilde{\sigma}_k^*, \omega_i^{-*}, \omega_i^{+*}, \omega_k^{-*}, \omega_k^{+*})$ optimal solutions of P , D , $\tilde{P}$ and $\tilde{D}$, respectively. Then, $P \equiv \tilde{P}$ (i.e., $y_i^- = y_i^+ = 0 \ \forall i \in N$ and $y_k^- = y_k^+ = 0 \ \forall k \in K$) if one of the following two conditions is met:

- (i) $\varepsilon_i^- = \varepsilon_i^+ = 0 \ \forall i \in N$ and $\varepsilon_k^- = \varepsilon_k^+ = 0 \ \forall k \in K$
- (ii) $d_i^- \leq \tilde{\pi}_i^* \leq d_i^+ \ \forall i \in N$ and $d_k^- \leq \tilde{\sigma}_k^* \leq d_k^+ \ \forall k \in K$

Conditions (i) and (ii) provide stopping criteria for the stabilized column generation algorithm. The algorithm implemented for the present study is described in details in pseudocode Stabilized Column Generation Algorithm. At an iteration it , a restricted linear program is given by $\tilde{P}^{it}$ and its dual by $\tilde{D}^{it}$. Also, the following procedures need to be defined:

- $(\tilde{\lambda}_p^{it}, y_i^-, y_i^+, y_k^-, y_k^+; \tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it}, \omega_i^-, \omega_i^+, \omega_k^-, \omega_k^+) \leftarrow \text{optimizer}(\tilde{P}^{it})$: returns optimal solutions $(\tilde{\lambda}^{it}, y_i^-, y_i^+, y_k^-, y_k^+)$ and $(\tilde{\pi}^{it}, \tilde{\sigma}^{it}, \omega_i^-, \omega_i^+, \omega_k^-, \omega_k^+)$ of $\tilde{P}^{it}$ and $\tilde{D}^{it}$, respectively.
- $(\hat{a}, \hat{c}) \leftarrow \text{oracle}(\tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it}, k)$: returns a column $p = (\hat{a}, \hat{c}) \in P^k$, such that the reduced cost $\bar{c}_p^k$ of the associated $\tilde{\lambda}_p$ variable is minimum over all columns of vehicle $k \in K$, where $\hat{a}$ is the vector of the column p constraint coefficients and $\hat{c}$ is the associated $\tilde{\lambda}_p$ variable coefficient in the objective function of $\tilde{P}$.
- $d^{k+1} \leftarrow \text{recenter}(\tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it})$: updates every interval $[d_i^-, d_i^+]$ and $[d_k^-, d_k^+]$ according to a predefined box size after recentering every dual box around the corresponding dual value, $\tilde{\pi}_i^{it}$ and $\tilde{\sigma}_k^{it}$, respectively.

Stabilized Column Generation Algorithm

```

1 //Assumptions:  $\tilde{P}^0$  feasible and bounded
2 Initialization of  $\tilde{P}$  with a feasible solution:  $A^0, c^0, d^0, \varepsilon^0$ 
3  $it \leftarrow 0$ 
4 while true
5    $(\tilde{\lambda}_p^{it}, y_i^-, y_i^+, y_k^-, y_k^+; \tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it}, \omega_i^-, \omega_i^+, \omega_k^-, \omega_k^+) \leftarrow \text{optimizer}(\tilde{P}^{it})$ 
6    $colAdded \leftarrow false$ 
7    $end \leftarrow true$ 
8   for each vehicle  $k \in K$  do
9      $(\hat{a}, \hat{c}) \leftarrow \text{oracle}(\tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it}, k)$ 
10    if  $\tilde{c}_p^k < 0$  then
11       $(A^{it+1}; c^{it+1}) \leftarrow (A^{it}, \hat{a}; c^{it}, \hat{c})$ 
12       $colAdded \leftarrow true$ 
13       $end \leftarrow false$ 
14    else
15      if  $!(y_i^- = y_i^+ = 0 \ \forall i \in N \text{ and } y_k^- = y_k^+ = 0 \ \forall k \in K)$  then
16         $end \leftarrow false$ 
17      end
18    end
19  end
20  if  $end = true$  then
21     $\lambda^* \leftarrow \tilde{\lambda}^k$ 
22    break
23  end
24  if  $colAdded = true$  then
25     $\varepsilon_i^{-it+1} \leftarrow 1.1\varepsilon_i^{-it} \ \forall i \in N$ 
26     $\varepsilon_i^{+it+1} \leftarrow 1.1\varepsilon_i^{+it} \ \forall i \in N$ 
27     $\varepsilon_k^{-it+1} \leftarrow 1.1\varepsilon_k^{-it} \ \forall k \in K$ 
28     $\varepsilon_k^{+it+1} \leftarrow 1.1\varepsilon_k^{+it} \ \forall k \in K$ 
29  else
30     $\varepsilon_i^{-it+1} \leftarrow 0.001\varepsilon_i^{-it} \ \forall i \in N$ 
31     $\varepsilon_i^{+it+1} \leftarrow 0.001\varepsilon_i^{+it} \ \forall i \in N$ 
32     $\varepsilon_k^{-it+1} \leftarrow 0.001\varepsilon_k^{-it} \ \forall k \in K$ 
33     $\varepsilon_k^{+it+1} \leftarrow 0.001\varepsilon_k^{+it} \ \forall k \in K$ 
34     $d^{k+1} \leftarrow \text{recenter}(\tilde{\pi}_i^{it}, \tilde{\sigma}_k^{it})$ 
35  end
36   $it \leftarrow it + 1$ 
37 end

```

The updating strategies for ε_i^- , ε_i^+ , ε_k^- , ε_k^+ values and for intervals $[d_i^-, d_i^+]$ and $[d_k^-, d_k^+]$ must together ensure finite convergence of the algorithm. As depicted in the pseudocode Stabilized Column Generation Algorithm, the strategy adopted has this property because after an iteration in which the oracle is not able to find a negative reduced cost column, $[d_i^-, d_i^+]$ and $[d_k^-, d_k^+]$ are updated as in the BoxStep method (condition (ii))

holds after a finite number of iterations) and ε values are decreased (condition (i) holds after a finite number of iterations).

Finally, even though $[d_i^-, d_i^+]$ and $[d_k^-, d_k^+]$ may take any values when $\tilde{P}$ is initialized (values subject to $d_i^- \leq d_i^+$ and $d_k^- \leq d_k^+$), a good estimate of the optimal dual variables $(\tilde{\pi}_i^*, \tilde{\sigma}_k^*)$ should be preferred over arbitrary dual values when centering the boxes to define these intervals.

5.4. Pricing Subproblem

The pricing subproblem, SP^0 or SP^c , can be cast as the computation of an elementary minimum cost path considering resources constraints (ESPPRC) in digraph G , in which the costs in the arcs are modified according to the shadow prices of constraints (16) of the LRMP.

The mathematical formulation of the pricing subproblem of an owned vehicle $k \in K^0$, denoted by SP^0 , is presented below:

(SP^0)

$$\text{Min} \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (c_{ijr}^k - \pi_i) \cdot x_{ijr}^k - \sigma_k$$

s. t.

$$\sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} x_{ijr}^k = 0 \quad j \in N^K, k \notin K_j \quad (2)$$

$$\sum_{j \in N \cup \{O_{r+1}\}} x_{O_r jr}^k = 1 \quad r = 1, \dots, R \quad (3)$$

$$\sum_{i \in N \cup \{O_r\}} x_{iO_{r+1}r}^k = 1 \quad r = 1, \dots, R \quad (4)$$

$$\sum_{i \in N \cup \{O_r\}} x_{ihr}^k - \sum_{j \in N \cup \{O_{r+1}\}} x_{hjr}^k = 0 \quad h \in N, r = 1, \dots, R \quad (5)$$

$$a_i \leq t_{ir}^k \leq b_i \quad r = 1, \dots, R, i \in N \cup \{O_r, O_{r+1}\} \quad (6)$$

$$t_{ir}^k + s_i^k + \tilde{t}_{ij}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad r = 1, \dots, R, (i, j) \in A \setminus \{(O_r, O_{r+1})\} \quad (7)$$

$$t_{ir}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad r = 1, \dots, R, (i, j) = (O_r, O_{r+1}) \quad (8)$$

$$\sum_{i \in N} \left[\left(\frac{D_i^{pet}}{Q_k^{pet}} + \frac{D_i^{can}}{Q_k^{can}} + \frac{D_i^{cel}}{Q_k^{cel}} + \frac{D_i^{bar}}{Q_k^{bar}} \right) \left(\sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k \right) \right] \leq 1 \quad r = 1, \dots, R \quad (9)$$

$$t_{O_{R+1}R}^k - t_{O_1,1}^k \leq W_{max,k} \quad (10)$$

$$x_{ijr}^k \in \{0,1\} \quad (i,j) \in A, r = 1, \dots, R \quad (12)$$

$$t_{ir}^k \geq 0 \quad r = 1, \dots, R, i \in N \cup \{O_1, \dots, O_{R+1}\} \quad (13)$$

$$x_{O_{r+1}O_{r+2}r+1}^k \geq x_{O_rO_{r+1}r}^k \quad r = 1, \dots, (R-1) \quad (15)$$

For further information about each constraint of SP^0 formulation, see Chapter 4: Mathematical Formulation.

The mathematical formulation of the pricing subproblem of a chartered vehicle $k \in K^c$, denoted by SP^c , is presented below:

(SP^c)

$$\text{Min} \sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} \sum_{j \in N \cup \{O_{r+1}\}} (-\pi_i) \cdot x_{ijr}^k + FC^k \cdot v^k - \sigma_k$$

s. t.

$$\sum_{r=1}^R \sum_{i \in N \cup \{O_r\}} x_{ijr}^k = 0 \quad j \in N^K, k \notin K_j \quad (2)$$

$$\sum_{j \in N \cup \{O_{r+1}\}} x_{O_rjr}^k = 1 \quad r = 1, \dots, R \quad (3)$$

$$\sum_{i \in N \cup \{O_r\}} x_{iO_{r+1}r}^k = 1 \quad r = 1, \dots, R \quad (4)$$

$$\sum_{i \in N \cup \{O_r\}} x_{ihr}^k - \sum_{j \in N \cup \{O_{r+1}\}} x_{hjr}^k = 0 \quad h \in N, r = 1, \dots, R \quad (5)$$

$$a_i \leq t_{ir}^k \leq b_i \quad r = 1, \dots, R, i \in N \cup \{O_r, O_{r+1}\} \quad (6)$$

$$t_{ir}^k + s_i^k + \tilde{t}_{ij}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad r = 1, \dots, R, (i,j) \in A \setminus \{(O_r, O_{r+1})\} \quad (7)$$

$$t_{ir}^k - M_{ij}^k(1 - x_{ijr}^k) \leq t_{jr}^k \quad r = 1, \dots, R, (i,j) = (O_r, O_{r+1}) \quad (8)$$

$$\sum_{i \in N} \left[\left(\frac{D_i^{pet}}{Q_k^{pet}} + \frac{D_i^{can}}{Q_k^{can}} + \frac{D_i^{cel}}{Q_k^{cel}} + \frac{D_i^{bar}}{Q_k^{bar}} \right) \left(\sum_{j \in N \cup \{O_{r+1}\}} x_{ijr}^k \right) \right] \leq 1 \quad r = 1, \dots, R \quad (9)$$

$$t_{O_{R+1}R}^k - t_{O_1,1}^k \leq W_{max,k} \quad (10)$$

$$v^k \geq \sum_{j \in N} x_{O_rjr}^k \quad r = 1, \dots, R \quad (11)$$

$$x_{ijr}^k \in \{0,1\} \quad (i,j) \in A, r = 1, \dots, R \quad (12)$$

$$t_{ir}^k \geq 0 \quad r = 1, \dots, R, i \in N \cup \{O_1, \dots, O_{R+1}\} \quad (13)$$

$$v^k \in \{0,1\} \quad (14)$$

$$x_{O_{r+1}O_{r+2}r+1}^k \geq x_{O_rO_{r+1}r}^k \quad r = 1, \dots, (R-1) \quad (15)$$

For further information about each constraint of SP^c formulation, see Chapter 4: Mathematical Formulation.

A common and widely employed technique for solving the elementary minimum cost path problem considering resources constraints is dynamic programming, as successfully used by Desrosiers et al. (1995) and by Dejax et al. (2004).

It should be stressed that the real-life routing problem considered here has some particular constraints, such as the possibility of a vehicle travelling multiple routes per day and the limitation on vehicle driver working hours that require a particular treatment, that, to the best of our knowledge, have not been addressed to date in the literature by exact methods.

A shortest path problem where waiting time is undesired along the path, namely the shortest path problem with waiting costs (SPWC), was introduced by Desaulniers et al. (1998). In the SPWC, the elapsed time between the arrival and the departure time at a node is referred to as waiting time and there is linear cost penalty that is imposed for each unit of time spent waiting along the path. Desaulniers and Villeneuve (2000) propose two alternative formulations of the SPWC for which algorithms already exist, namely the shortest path problem with linear node costs and a two-resource generalized shortest path problem with resource constraints.

Unlike the SPWC, in the present problem, there is only the cost of travelling arcs in the objective function, and idle time must be minimized only enough to ensure path feasibility with regard to the maximum workday duration along the minimum cost path.

5.4.1. Bounded Bidirectional Dynamic Programming Algorithm

The algorithm developed herein (Seixas and Mendes, 2013) was based on the bounded bidirectional dynamic programming algorithm proposed by Righini and Salani (2006, 2008) that, in turn, is based on the algorithm developed by Desrochers and Soumis (1988) to solve the resource constrained shortest path problem (RCSPP).

In the present problem, a particular resource called the workload resource was introduced. The workload resource is limited to $W_{max,k}$. This resource consists of the

time elapsed since the service start time of O_1 and its consumption depends on the arcs travelled, the sequence of vertices visited and the service start time of O_1 .

Idle time is generated whenever a vehicle arrives at j before the beginning of its time window. In this case, the vehicle is forced to wait until a_j to start the service, and this waiting time is incorporated into the workload resource. In traditional dynamic programming algorithms for solving the ESPPRC (Righini and Salani, 2006), the O_1 service start time is always fixed at 0 with no implication for the final result, even though the mathematical formulation allows this variable to assume any positive value. In the present problem, the workload resource depends on the value of this variable. In short, whenever idle time is created, the dynamic programming algorithm has to recursively recalculate the service start times along the path to minimize the workload resource consumption.

However, the simple existence of the workload resource in a label is not sufficient for finding the optimal solution because there are situations in which a dominated label may be part of the optimal path. This behaviour occurs because of the capacity to serve the origin later and thus eliminate idle time along the route is smaller in the first case. Because of this feature, it is necessary to create an additional resource, the value of which represents the potential to eliminate idle time such that the idle time generated during the extension of the labels is minimized or even fully eliminated.

Figure 14: Example where idle time reducing capacity resource is required.

In figure 14, it is possible to visualize an example with four customers in which $W_{max,k} = 5$, $s_{DC}^k = 0$ and $[a_{DC}, b_{DC}] = [0, 10]$. Values t and c represent the time and cost of travelling an arc, respectively. Note that path DC-1-2-3 is apparently better

than path DC-2-1-3 for reaching 3 at the same instant in time, with the same number of hours worked and with a lower cost by 1 unit. However, path DC-1-2-3 serves 4 with 5 units of workload consumed and is incapable of travelling arc 4-DC with cost $c = -10$. In contrast, path DC-2-1-3 is capable of leaving the origin at $t = 1$ and travelling path 3-4-DC, defining the optimal solution: DC-2-1-3-4-DC. The new resource proposed is initially (at the origin) equal to the planning horizon, and along the path, it is consumed progressively until it is exhausted or not at the final destination.

The vector of resources $\mathbf{R}$ is composed of the following components: q (consumed fraction of vehicle capacity), t (service start time in a forward label or departure time in a backward label), w (time consumed out of the working hours), Δ (idle time reducing capacity), $cutQty$ (number of cuts $x_{ijr}^k = 1$ travelled), S (visit vector) and $vstQty$ (number of vertices served along the path). The label also has component r , which indicates the route to which the label belongs, and the cost C .

When a label $(\mathbf{R}, C, i)$ associated with vertex i is extended to vertex j generating the label $(\mathbf{R}', C', j)$, the consumption of each resource is updated according to the rules presented in the next sections. It is worth highlighting that the extension to j is only feasible if the vehicle is authorized to serve customer j .

5.4.2. Cost Resource

Cost is an unlimited resource and, at the origin vertex of the first route, its value is initialized with 0. For being an unlimited resource, there is no feasibility criterion to be applied on the label generated during extension. It is worth observing that the cost function is not strictly increasing in the case studied, as there are arcs in the digraph with a negative cost. Cost updating in the bidirectional algorithm is performed according to the expression as follows, both for the forward and backward extension: $C' = C + c_{ij}, \forall (i, j) \in A$.

It is worth stressing that a fixed cost must be incurred in the objective function if the vehicle is chartered and used to serve at least one customer. In this case, the fixed cost must not be computed in a label cost resource during its extension but, only when forward labels are joined with backward labels to define workdays as described in 8.4.13. Feasibility Tests and Search for Optimal Solution.

5.4.3. Load Capacity Resource

A demand for four product families: D_i^{pet} , D_i^{can} , D_i^{cel} and D_i^{bar} is associated to each vertex i and each vehicle k has a maximum load capacity in units of each product family: Q_k^{pet} , Q_k^{can} , Q_k^{cel} and Q_k^{bar} .

The total demand of each vertex is converted into the fraction of the vehicle cargo capacity it occupies and added to the resource capacity, which represents the occupied fraction of the vehicle cargo capacity after serving the demand of the vertices visited.

Let q be the occupied fraction of the vehicle cargo capacity. When the vehicle leaves the origin at the beginning of any route, $q = 0$. Whenever a vertex j is visited, the total capacity fraction occupied by the demand of vertex j is added to q . The updating of the vehicle consumed capacity, both for forward and backward labels, is performed

according to expression: $q' = q + \left(\frac{D_j^{pet}}{Q_k^{pet}} + \frac{D_j^{can}}{Q_k^{can}} + \frac{D_j^{cel}}{Q_k^{cel}} + \frac{D_j^{bar}}{Q_k^{bar}} \right)$. For being a limited resource, the label is only feasible if: $q' \leq 1$.

5.4.4. Time Resource

Time resource, in forward labels, consists of the time passed since the beginning of the planning horizon to the beginning of the service of the path vertex.

To each vertex i , a service time s_i^k and a time window $[a_i, b_i]$ are associated, within which the service must be started. If the vehicle reaches i before a_i , it has to wait until a_i for the service to be started. The travelling time $\tilde{t}_{ij}^k$ for traveling the arc leaving from i and arriving in j is known. The time consumption along the forward path is strictly increasing and its value is initialized in 0 at the origin ($a_{DC} = 0$). The rule for updating time in the forward direction obeys the expression: $t' = \max\{t + s_i^k + \tilde{t}_{ij}^k, a_j\}$. The forward label created by the extension is only feasible if: $t' \leq b_j$. Otherwise, the time window of vertex j is violated.

It should be observed that, in the situation in which the vehicle reaches j before the beginning of its time window, an idle time is generated until a_j , which is initially incorporated to the working hours of the workday.

In backward labels, the time consists of the time passed since the end of the planning horizon (“departure time from the final destination”) until the departure time of each path vertex. To each vertex i , a departure time window $[(a_i + s_i^k), (b_i + s_i^k)]$ can be

associated, within which the vehicle must leave the vertex. If, in the backward calculation, the vehicle leaves after the end of the departure window $(b_i + s_i^k)$, leaving earlier at $(b_i + s_i^k)$ is necessary and idle time is generated.

The consumption of time along the backward path is strictly increasing and its value is initialized in 0 at the final destination, O_{R+1} . The rule for updating time in the backward direction, assuming $a_{DC} = 0$, follows expression: $t' = \max\{t + s_i^k + \tilde{t}_{ji}^k, b_{DC} - (b_j + s_j^k)\}$. The backward label created with the extension is only feasible if: $t' \leq b_{DC} - (a_j + s_j^k)$. Otherwise, the departure time window of vertex j is violated.

It should be stressed that the time in forward direction is written on a different base from that of the backward direction as represented in figure 15.

Figure 15: Time basis and arrival/departure time window.

It can be observed from figure 15 that: $t_{backward} = (b_{DC} - a_{DC}) - t_{forward} = b_{DC} - t_{forward}$ because $a_{DC} = 0$.

5.4.5. Workload Resource

The calculation of w' , the workload resource, for forward labels is initially given by $w' = \max\{w + (s_i^k + \tilde{t}_{ij}^k), w + (a_j - t)\}$. For backward labels, the value of w' is $w' = \max\{w + s_i^k + \tilde{t}_{ji}^k, w + \{[b_{DC} - (b_j + s_j)] - t\}\}$.

However, idle time is generated when the vehicle reaches vertex j before the beginning of its time window, $t + s_i^k + \tilde{t}_{ij}^k < a_j$, in a forward extension or, when the vehicle

departs from vertex j after the end of its departure time window, $t + s_i^k + \tilde{t}_{ji}^k \leq b_{DC} - (b_j + s_j^k)$, in a backward extension. Based on the expressions above, the idle time is incorporated into the workday duration. To ensure future viability in relation to the hours worked on a path after adding new arcs, it is necessary to eliminate the generated idle time as much as possible. To do so, idle time, t_{idle} , must first be calculated either when reaching j before a_j in a forward extension, $t_{idle} = a_j - (t + s_i^k + \tilde{t}_{ij}^k)$, or when leaving from j after $(b_j + s_j^k)$ in a backward extension, $t_{idle} = [b_{DC} - (b_j + s_j^k)] - (t + s_i^k + \tilde{t}_{ji}^k)$.

Eliminating idle time in the forward direction is only possible when there is a margin to serve the origin later, displacing the service start time of all vertices previous to j along the path. For the backward direction, the elimination of idle time is only possible when there is a margin to “leave earlier from the final destination”, displacing the departure time of all vertices previous to j along the path.

This is verified in the forward direction by calculating the maximum displacement capacity of the origin service start time without making the path infeasible according to expression $\Delta' = \min\{\Delta_v | \Delta_v = (b_v - t_v), v \in path(\mathbf{R}, C, i)\}$. For the backward direction, $\Delta' = \min\{\Delta_v | \Delta_v = [b_{DC} - (a_v + s_v)] - t_v, v \in path(\mathbf{R}, C, i)\}$.

For forward direction, after calculating t_{idle} and Δ' , the service start time of each path vertex and the working hours in j , w' , are updated according to the procedure `Update_Forward_Path`.

For backward direction, the departure time and the working hours of each vertex of the path are updated according to the procedure `Update_Backward_Path`.

The forward or backward label created is only feasible if: $w' \leq W_{max,k}$.

Update_Forward_Path Algorithm – updates the hours worked in j and the service start times in the forward labels of the path in function of the idle time in j and of resource Δ .

```

1  void Update_Forward_Path( $t_{idle}, \Delta'$ )
2  if  $\Delta' = 0$  then
3       $w' \leftarrow w + (a_j - t)$ 
4  else
5      if  $\Delta' \geq t_{idle}$  then
6           $w' \leftarrow w + s_i^k + \tilde{t}_{ij}^k$ 
7          for each  $v \in path(\mathbf{R}, C, r, i)$  do
8               $t_v \leftarrow t_v + t_{idle}$ 
9          end
10          $\Delta' \leftarrow \Delta' - t_{idle}$ 
11         if  $(b_j - t' < \Delta')$  then
12              $\Delta' \leftarrow b_j - t'$ 
13         end
14     else
15          $w' \leftarrow w + s_i^k + \tilde{t}_{ij}^k + (t_{idle} - \Delta')$ 
16         for each  $v \in path(\mathbf{R}, C, r, i)$  do
17              $t_v \leftarrow t_v + \Delta'$ 
18         end
19          $\Delta' \leftarrow 0$ 
20 end

```

In summary, for the forward direction, after calculating t_{idle} and Δ' , when $\Delta' > 0$ and $\Delta' \geq t_{idle}$, then $w' = w + s_i^k + \tilde{t}_{ij}^k$ and $t_v = t_v + t_{idle}, \forall v \in path(\mathbf{R}, C, i)$, i.e., t_{idle} units of Δ' are consumed. When $\Delta' > 0$ and $\Delta' < t_{idle}$, then $w' = w + s_i^k + \tilde{t}_{ij}^k + (t_{idle} - \Delta')$ and $t_v = t_v + \Delta', \forall v \in path(\mathbf{R}, C, i)$, i.e., all Δ' is consumed.

Update_Backward_Path Algorithm – updates the hours worked in j and the departure times in the backward labels of the path in function of the idle time in j and of resource Δ .

```

1  void Update_Backward_Path( $t_{idle}, \Delta'$ )
2  if  $\Delta' = 0$  then
3       $w' \leftarrow w + \{[b_{DC} - (b_j + s_j^k)] - t\}$ 
4  else
5      if  $\Delta' \geq t_{idle}$  then
6           $w' \leftarrow w + s_i^k + \tilde{t}_{ji}^k$ 
7          for each  $v \in path(\mathbf{R}, C, r, i)$  do
8               $t_v \leftarrow t_v + t_{idle}$ 
9          end
10          $\Delta' \leftarrow \Delta' - t_{idle}$ 
11         if  $([b_{DC} - (a_j + s_j^k)] - t') < \Delta'$  then
12              $\Delta' \leftarrow [b_{DC} - (a_j + s_j^k)] - t'$ 
13         end
14     else
15          $w' \leftarrow w + s_i^k + \tilde{t}_{ji}^k + (t_{idle} - \Delta')$ 
16         For each  $v \in path(\mathbf{R}, C, r, i)$  do
17              $t_v \leftarrow t_v + \Delta'$ 
18         end
19          $\Delta' \leftarrow 0$ 
20     end

```

For the backward direction, after calculating t_{idle} and Δ' , in the case $\Delta' > 0$ and $\Delta' \geq t_{idle}$, then $w' = w + s_i^k + \tilde{t}_{ji}^k$ and $t_v = t_v + t_{idle}, \forall v \in path(\mathbf{R}, C, i)$, i.e., t_{idle} units of Δ' are consumed. In the other case, when $\Delta' > 0$ and $\Delta' < t_{idle}$, then $w' = w + s_i^k + \tilde{t}_{ji}^k + (t_{idle} - \Delta')$ and $t_v = t_v + \Delta', \forall v \in path(\mathbf{R}, C, i)$, i.e., all Δ' is consumed.

5.4.6. Idle Time Reducing Capacity Resource

The workload resource is not sufficient to fully identify a label in relation to the working hours and requires the use of a new resource that is capable of identifying the capacity of the path to reduce idle time, Δ . This resource is crucial for calculating working hours during the extension of a label so that the path feasibility is not lost after the incorporation of new arcs.

As this resource is progressively consumed along the path, Δ can be considered a new resource and updated according to the algorithms `Update_Forward_Path` and `Update_Backward_Path` when idle time is generated during the extension. If idle time is not generated and the service start time or departure time does not violate the time

window of j , then $\Delta' = \min\{\Delta, (b_j - t')\}$ for forward labels and $\Delta' = \min\{\Delta, [b_{DC} - (a_j + s_j^k)] - t'\}$ for backward labels.

So as to increase the efficiency of computational implementations when it is not necessary to know the service start time or departure time of each vertex of the minimum cost path to generate a column for LRMP, it is not necessary to keep and recursively update them along the path when idle time is generated and there is consumption of Δ , that is, it is not necessary to execute the pseudocode lines {6,7,8} and {15,16,17} in the Update_Forward_Path and Update_Backward_Path algorithms and to store new service start times or departure times.

Since $0 \leq \Delta' \leq b_{DC}$, there is no need to verify whether the extension feasibility is lost in relation to this resource.

5.4.7. Elementary Path Constraints

To ensure that the optimal path is elementary, that is, to ensure that the optimal path does not contain cycles, Beasley and Christofides (1989) proposed adding a binary resource to each vertex i of the digraph. There is only one unit available of this resource that is integrally consumed when vertex i is visited. A vector S with $(|N| + 2)$ positions is initialized with 0 and used to indicate the consumption of this resource. When the feasible path is extended along arc (i, j) , the updating rule is:

$$S'[v] = \begin{cases} S[v] + 1, & v = j \\ S[v], & v \neq j \end{cases}. \text{ The label created is only feasible if: } S'[v] \leq 1, \forall v \in \{N \cup \{O_1, O_2\}\}.$$

It should be detached that the visit vector S does not contain any information on the vertices visit order.

5.4.8. Multiple Routes

If only one route is considered, the reduced digraph $G^* = (N \cup O_1 \cup O_2, A^*)$ is sufficient for the formulation. When more than one route is allowed up to a limit of R routes, the natural strategy is to replicate digraph G^* R times, obtaining digraph G in which a path from O_1 to O_{R+1} represents a workday. However, for computational implementation, it is not necessary to replicate digraph G^* R times because arcs linking destination O_2 to the origin O_1 and vice versa (backward direction) are used, together with specific extension rules for these arcs. For route control, a route r indicator is

added to each label. At first, an initial forward label is added to origin O_1 with $r = 1$ and an initial backward label to destination O_2 with $r = R$.

Forward and backward extension procedures are executed until all of the forward and backward labels have been extended and the list of pending vertices is empty. At this moment, all of the forward and dominant labels in O_2 are extended to O_1 according to a special rule. The same situation occurs for the backward and dominant labels of O_1 , which are extended to O_2 according to a special second rule. Among other things, during this extension, the route indicator of the forward labels is incremented as the route indicator of the backward labels is decremented; thus, a new cycle for the application of the extension procedures begins.

This process is repeated until $r = R$ in the forward labels and $r = 1$ in the backward labels. The cost and time for travelling the arc to return to the beginning of a new route are equal to 0.

The extension of a forward label from O_2 to O_1 is made according to the rules $C' = C$, $q' = 0$, $t' = t$, $w' = w$, $r' = r + 1$, $\Delta' = \Delta$, and $S'[v] = \begin{cases} 0, & v = O_2 \\ S[v], & v \neq O_2 \end{cases}$.

The extension of a backward label from O_1 to O_2 is made according to the rules $C' = C$, $q' = 0$, $r' = r - 1$, $\Delta' = \Delta$, and $S'[v] = \begin{cases} 0, & v = O_1 \\ S[v], & v \neq O_1 \end{cases}$. If the backward label does not visit any customer along route r , then $t' = t$ and $w' = w$. If the backward label visits at least one customer along route r , then $t' = t + s_{DC}^k$ and $w' = w + s_{DC}^k$.

5.4.9. Lexicographic Ordering of Labels

The dominant labels are kept organized in increasing lexicographic order along all of the iterations as described in Ceselli et al. (2009). A label L_1 is considered lexicographically smaller than a label L_2 if $w_1 < w_2$ or $w_1 = w_2$ and $vstQty_1 < vstQty_2$. As the workload resource is minimized in a label, it is a significant resource for the dominance rule, and for this reason, it is selected as the major ordering criterion, followed by the number of vertices served. In short, whenever a vertex is selected so that its labels not yet treated are extended, the selection and extension of labels occurs in increasing lexicographic order.

5.4.10. Bounding on Resource

Based on the strategy adopted by Righini and Salani (2008) to prevent the proliferation of labels in both directions, a critical resource is selected such that labels consuming over 50% of the resource are eliminated from each of the two directions. In the present problem, the critical resource is time; thus, a label is discarded when $t' > 0.5 \cdot (b_{DC} - a_{DC})$ during an extension.

5.4.11. Treatment of Cuts

It is usual to use arc flow variable cuts of the type $x_{ijr}^k = 0$ and $x_{ijr}^k = 1$ in a branch-and-price procedure to solve a vehicle routing problem, as in the present study. Given that the arc flow variables are specific for each vehicle, the arc flow variable cuts are part of the constraints of the pricing subproblem and have to be adequately treated by the dynamic programming algorithm.

In addition to arc flow variable cuts, multiple route cuts, defined by constraint (15), were presented in Section 4.2. These cuts are also part of the constraints of the pricing subproblem and must also be adequately treated by the dynamic programming algorithm.

5.4.11.1. Arc Flow Variable Cuts

There are two types of arc flow variable cuts: $x_{ijr}^k = 0$ (arc (i, j) in route r is forbidden) and $x_{ijr}^k = 1$ (arc (i, j) in route r is mandatory).

Cuts $x_{ijr}^k = 0$ may be easily treated in a preprocessing that eliminates from the digraph all the arcs related to these cuts or treated during the label extension by forbidding the extension from i to j in route r after an arc flow variable cut search with result $x_{ijr}^k = 0$.

Cuts $x_{ijr}^k = 1$ are treated by an increasing and unlimited resource, the value of which is initialized with 0. Whenever a cut $x_{ijr}^k = 1$ is traveled, the resource is incremented in one unit. This resource is defined by $cutQty$ and contains the number of cuts $x_{ijr}^k = 1$ traveled in the label path. The visit vector ensures that arc (i, j) of route r will no longer be traveled and, as a consequence, this cut cannot be used again to increment the value of resource $cutQty$. At the end of the execution of the label extension process in all of the workday routes, the number of cuts equal to 1 the optimal solution of the problem must have are counted, seeing that all these arcs have to be compulsorily traveled.

The number of cuts of the optimal solution is always the largest possible due to the use of resource *cutQty* in the dominance rule. With this, if the number of cuts $x_{ijr}^k = 1$ of the solution obtained is smaller than the total number of existing cuts $x_{ijr}^k = 1$; there is no optimal solution in this case. The optimal solution must contain all the cuts $x_{ijr}^k = 1$ which is verified by the value of *cutQty*.

5.4.11.2. Multiple Route Cuts

The existence of multiple route cuts requires a particular feasibility test during the label extension process so that the valid inequalities (15) are not violated.

In the forward direction, the extension of a label $(\mathbf{R}, C, r, O_r)$, associated to route origin O_r , to vertex $j \neq O_{r+1}$ when $r \geq 2$ is only feasible if in the previous route, $r - 1$, the label path does not travel the arc (O_{r-1}, O_r) .

In the backward direction, the extension of a label $(\mathbf{R}, C, r, O_{r+1})$, associated to route destination O_{r+1} , to vertex $j = O_r$ when $r \leq (R - 1)$ is only feasible if in the previous route, $r + 1$, the label path travels the arc (O_{r+1}, O_{r+2}) .

Besides the required feasibility tests during the label extension process in both forward and backward directions, particular feasibility tests are also required when combining a forward path with a backward path to form a workday. These tests are described in details in Section 5.4.15: Forward-backward Joint Feasibility Tests and Search for Optimal Solution.

5.4.12. Simplified Algorithm for Problems without Arc Flow Variable Cuts

The default dynamic programming algorithm developed for this study allows the forward extension in route r along the fictitious route arc (O_r, O_{r+1}) , as well as the backward extension in route r along the fictitious route arc (O_{r+1}, O_r) .

The extension along the fictitious route arc in the backward direction creates paths that are physically equivalent but located in different routes, as depicted in figure 16.

Figure 16: equivalent backward labels in different routes

In the forward extension, the replication of physically equal paths do not occur because of the existence of the multiple route cuts treatment described in Section 5.4.11.2: Multiple Route Cuts.

The existence of these equivalent paths in the backward direction are required when there are arc flow variable cuts and, hence, the algorithm has to ensure that any cut in any route will be checked.

However, when there are no arc flow variable cuts as in the root node of a branch-and-bound tree, it is possible to simplify the algorithm and increase its computational efficiency by forbidding the extension along the fictitious route arc in both forward and backward directions. Under those circumstances, an optimal solution may be the joint of paths between different routes with, if required, additional fictitious routes in the end of the resulting path to fulfill the R routes.

5.4.13. Dominance Rule

The efficiency of the dynamic programming algorithm depends mainly on its capacity to identify and to eliminate labels that will certainly not be a part of the optimal solution. This prevents these undesired labels from generating an unnecessary proliferation of labels that largely contribute to increasing the total computational cost. To conduct this identification and elimination of labels that are certainly not a part of the optimal solution, there are dominance tests that are always executed when a label is extended so that the labels at each vertex are non-dominated.

According to the dominance rule, label L_1 dominates label L_2 if all of the expressions below are true: $C_1 \leq C_2$; $q_1 \leq q_2$; $t_1 \leq t_2$; $w_1 \leq w_2$; $t_1 - w_1 \geq t_2 - w_2$; $\Delta_1 \geq \Delta_2$; $r_1 = r_2$; $vstQty_1 \leq vstQty_2$, $cutQty_1 \geq cutQty_2$ and $S_1[v] \leq S_2[v]$, $\forall v \in \{N \cup \{O_1, O_2\}\}$. The simplified version of the algorithm, that is designed for problems without arc flow variable cuts, does not have the test $cutQty_1 \geq cutQty_2$ in the dominance rule.

It is worth highlighting the important role of component $(t - w)$ which is responsible for maximizing the total amount of resource Δ consumed in a path. Whenever Δ is consumed, the difference between t and w increases which is desirable because it corresponds to a reduction in the potential to generate future idle time and/or an increase in the capacity to absorb future idle time.

The following figure illustrates an example in which component $(t - w)$ is required to find the optimal path DC-2-1-4-3-5-DC of cost 2. In this example: $W_{max,k} = 7$, $s_{DC} = 0$ and $[a_{DC}, b_{DC}] = [0, 20]$.

Figure 17: Example where component $(t-w)$ is required in the dominance rule.

There are 4 different paths from DC to vertex 3 which are fully described in table 2:

Table 2: Four different paths from DC to vertex 3.

Path	t	w	Δ	c	$S[1]$	$S[2]$	$S[3]$	$S[4]$	$t - w$
DC-2-1-4-3	13	4	0	0	1	1	1	1	9
DC-1-2-3	11	3	0	0	1	1	1	0	8
DC-2-3	11	2	0	2	0	1	1	0	9
DC-1-4-3	13	3	7	2	1	0	1	1	10

Paths DC-2-3 and DC-1-4-3 will not be analyzed due to their higher cost and the known optimal cost.

Regarding paths DC-2-1-4-3 and DC-1-2-3, note that without considering component $(t - w)$ in the dominance rule, path DC-2-1-4-3 would be dominated by path DC-1-2-3 and eliminated from vertex 3 label list. However, path DC-1-2-3 generates more idle time than path DC-2-1-4-3 when reaching customer 5 and it is not able to reach the final destination consuming an amount of workload resource less than or equal to 7 units. Figures 18 and 19 shows each path label in detail.

Figure 18: Path DC-2-1-4-3-5-DC labels in the example given.

Figure 19: Path DC-1-2-3-5-DC labels in the example given.

Therefore, component $(t - w)$ in the dominance rule is required to avoid eliminating a path which has lower potential to generate idle time and, in case it is generated, higher capacity to absorb it.

5.4.14. Heuristic Labeling Algorithm

Aiming at increasing the convergence speed of the column generation technique, columns are added by a heuristic labelling algorithm until no new column with negative reduced cost is found. The subproblem is then solved to optimality by the exact labelling algorithm.

The exact labelling algorithm is typically made heuristic by removing from its dominance rule combinatorial nature resources, which reduces the number of labels generated, as described by Ceselli et al. (2009). In the present study, the Heuristic Labelling Algorithm uses a relaxed dominance rule that does not have the visit vector, S , the capacity to reduce idle time, Δ , and component $(t - w)$.

It was observed that the initial use of a heuristic labeling algorithm makes several columns with negative reduced cost be rapidly added to the LRMP at the beginning, reducing the total resolution time.

5.4.15. Forward-backward Joint Feasibility Tests and Search for Optimal Solution

After the end of the label extension process for all of the routes, the forward labels are combined with the backward labels to form paths from O_1 to O_{R+1} . The optimal solution is determined after all of the feasible combinations between a forward label and backward label at each possible pair of vertices (i, j) are verified. For this purpose, there are nine joint feasibility tests that are applied to ensure that the combination of the two labels, L_1 forward in i and L_2 backward in j , creates a feasible path from O_1 to O_{R+1} . The nine tests are described below:

- 1) $r_1 = r_2$;
- 2) There is arc (i, j) ;
- 3) Cut $x_{ijr_1}^k = 0$ does not exist;
- 4) The value of $cutQty_1$ added to $cutQty_2$, plus 1 if cut $x_{ijr_1}^k = 1$ exists, must be equal to the total number of mandatory cuts existing in the subproblem of vehicle k ;
- 5) $S_1[v] + S_2[v] \leq 1 \quad \forall v \in N$
- 6) $q_1 + q_2 \leq 1$;
- 7) $t_i + s_i^k + \tilde{t}_{ij}^k + t_j \leq b_{DC}$;
- 8) **If $R \geq 2$ then**
 //Multiple route cut verification
 If $r_1 = 1$ then
 If $i = O_1$ and $j = O_2$ then
 (O_2, O_3) must be traveled in the backward path
 End

```

End
If  $r_1 > 1$  and  $r_1 < R$  then
    If  $i = O_{r_1}$  and  $j \neq O_{r_1+1}$  then
         $(O_{r_1-1}, O_{r_1})$  must not be traveled in the forward path
    End
    If  $i = O_{r_1}$  and  $j = O_{r_1+1}$  then
         $(O_{r_1+1}, O_{r_1+2})$  must be traveled in the backward path
    End
End
If  $r_1 = R$  then
    If  $i = O_R$  and  $j \neq O_{R+1}$  then
         $(O_{R-1}, O_R)$  must not be traveled in the forward path
    End
End
End
9)  $t_{idle} = [b_{DC} - (t_j + s_j^k)] - (t_i + s_i^k + \tilde{t}_{ij}^k)$ 
 $t_{idle} \geq 0$  must be true.
If  $t_{idle} = 0$  then
     $w_i + w_j + [(b_{DC} - t_j) - t_i] \leq W_{max,k}$  must be true.
Else
    Let  $\Delta = \Delta_1 + \Delta_2$ 
    If  $\Delta \geq t_{idle}$  then
         $w_i + w_j + [(b_{DC} - t_j) - t_i] - t_{idle} \leq W_{max,k}$  must be true.
    Else
         $w_i + w_j + [(b_{DC} - t_j) - t_i] - \Delta \leq W_{max,k}$  must be true.
    End
End

```

In the simplified version of the algorithm, that is designed for problems without arc flow variable cuts, there are only seven joint feasibility tests that are applied to ensure that the combination of the two labels, L_1 forward in i and L_2 backward in j , creates a feasible path. These tests are defined as follows:

- 1) $r_1 \leq r_2$;
- 2) There is arc (i, j) ;
- 3) $S_1[v] + S_2[v] \leq 1 \quad \forall v \in N$
- 4) $q_1 + q_2 \leq 1$;
- 5) $t_i + s_i^k + \tilde{t}_{ij}^k + t_j \leq b_{DC}$;
- 6) **If** $R \geq 2$ **then**
 - //Multiple route cut verification
 - If** $i = O_1$ and $j = O_2$ **then**
 - $r_2 = R$ must be true
 - End**
- End**
- 7) $t_{idle} = [b_{DC} - (t_j + s_j^k)] - (t_i + s_i^k + \tilde{t}_{ij}^k)$
 $t_{idle} \geq 0$ must be true.
If $t_{idle} = 0$ **then**
 - $w_i + w_j + [(b_{DC} - t_j) - t_i] \leq W_{max,k}$ must be true.**Else**
 - Let $\Delta = \Delta_1 + \Delta_2$
 - If** $\Delta \geq t_{idle}$ **then**
 - $w_i + w_j + [(b_{DC} - t_j) - t_i] - t_{idle} \leq W_{max,k}$ must be true.
 - Else**
 - $w_i + w_j + [(b_{DC} - t_j) - t_i] - \Delta \leq W_{max,k}$ must be true.
 - End**
- End**

If the label combination in the simplified algorithm creates a path with less than R routes, then fictitious routes are added to the end of the path so that the resulting path (workday) has exactly R routes.

For owned vehicles, the resulting cost C of the combination between labels L_1 and L_2 is calculated as: $C = C_1 + C_2 + c_{ij}$.

For chartered vehicles, the resulting cost C is computed as:

$$C = \begin{cases} C_1 + C_2 + c_{ij} + FC^k & \text{if } \exists v \in N | S_1[v] + S_2[v] = 1 \\ C_1 + C_2 + c_{ij}, & \text{otherwise} \end{cases}$$

The computational cost of searching for an optimal combination between a forward label and a backward label in each route is significant when compared to the total computational cost of the algorithm. When searching for an optimal solution, a very simple but rather efficient approach to reduce the computational cost is to first compute the cost of the label combination and then test the joint feasibility, only if its cost is lower than the lowest cost already found.

5.4.16. State-Space Relaxation and Decremental State-Space Relaxation

State-space relaxation was introduced by Christofides et al. (1981). The state-space, Ψ , explored by the exact dynamic programming algorithm is projected onto a lower dimensional space, Γ , so that each state in Γ retains the minimum cost among those of its corresponding states in Ψ (assuming the objective function must be minimized). In space Γ , the number of states to be explored is drastically reduced; the drawback is that some original state corresponding to an infeasible solution in Ψ may be projected onto a state corresponding to a feasible solution in Γ ; therefore, the search in the relaxed state-space does not guarantee finding an optimal solution but rather a lower bound.

The state-space relaxation algorithm considered in this study consists of mapping each state $(\mathbf{R}, C, r, i)$ onto a new state $(\hat{\mathbf{R}}, C, r, i)$ in which the visit vector S is removed from the resource vector $\mathbf{R}$ by being mapped as the existing resource $vstQty$ ($vstQty = \sum_{i \in N \cup \{O_1, O_2\}} S[i]$). Except for the removal of the visit vector S , there are no differences when compared to the exact dynamic programming algorithm dominance rule.

The considered mapping makes a large difference with respect to the exact dynamic programming algorithm in which the visit vector S yields an exponential number of possible states. Because the state (label) does no longer keeps information about the set of already visited vertices, cycles are allowed; therefore, the path is guaranteed to be feasible with respect to the resource constraints but it is not guaranteed to be elementary.

As opposed to the state-space relaxation, the decremental state-space relaxation pursues a compromise between forbidding and allowing multiple visits to the vertices, i.e., a compromise between the exact dynamic programming algorithm and state-space relaxation algorithm. This compromise is accomplished by identifying some vertices as

critical according to the solution obtained and by preventing multiple visits to critical vertices in subsequent runs while allowing multiple visits to non-critical vertices.

In summary, the decremental state-space relaxation algorithm runs iteratively the state-space relaxation procedure in which each label also has a binary vector $\hat{S}$ playing the same role as S in the exact dynamic programming algorithm: if $\hat{S}[j] = 1$, then j is critical, and $\hat{S}[j] = 0$ otherwise. Consequently, the extension of a label $(\hat{R}, C, r, i)$ from vertex i to vertex j is feasible with regard to $\hat{S}$ only if $\hat{S}[j] = 0$ or if $\hat{S}[j] = 1$ and j is not visited along the path $(\hat{R}, C, r, i)$.

Every time a solution with cycles is produced by the modified state-space relaxation procedure, the vertices visited more than once are marked as critical, and the algorithm restarts. The optimal solution, which is guaranteed to be feasible with respect to the resource constraints and to be elementary, is obtained when a solution with no cycles is produced.

It is worth highlighting that the decremental state-space relaxation is used only when optimality is required, i.e., only when the dominance rule is complete (includes the visit vector S). The heuristic labelling algorithm is based on the regular bidirectional dynamic programming algorithm.

5.5. Branch-and-Price

The Column Generation algorithm previously described is executed in each tree node under the branch-and-bound framework. In the next sections, the search strategy, initial upper bound and branching strategy used are described.

5.5.1. Search Strategies

Three search strategies were implemented: Depth First, Best-Node First and Mixed (Depth First followed by Best-Node First). The Depth First strategy, in experiments conducted, had to solve a larger quantity of nodes than the Best-Node First strategy or Mixed until optimality was obtained and, for this reason, it is not used. Analogously, as a tabu search metaheuristic that returns a good initial solution is used before the branching starts, there is no advantage in using the Mixed strategy seeing that, in most cases, the integer solution found when initially deepening the search in the tree is worse than the initial solution obtained with the metaheuristic. Due to the observations

presented before, the search for the best bound (Best-Node First) strategy is always used.

5.5.2. Initial Upper Bound

Constructive heuristic and tabu search are used at the root node aiming at starting the branching with a good upper bound. With this, the number of prunings by bound increases along the iterations which significantly reduces the total branching time to optimality. In the other tree nodes, a constructive heuristic is used with a tabu search restricted to a few iterations to generate an initial good solution to the LRMP (initial columns of the LRMP) at a low computational cost.

5.5.3. Branching Strategy

The branching strategy used for searching the optimal solution in the branch-and-price tree was to branch in the arc flow variables. In each tree node, the arc flow variable x_{ijr}^k with its fractional part closer to 0.5 in the optimal solution is selected. After selecting the flow variable, two child nodes of the examined node are created in which $x_{ijr}^k = 0$ in one and $x_{ijr}^k = 1$ in the other. It is worth stressing that not only cut $x_{ijr}^k = 1$ is added to the second child node. The fact of imposing $x_{ijr}^k = 1$, also implies that: in the same route, $x_{jir}^k = 0$; in the other routes of vehicle k , $x_{ijr}^k = x_{jir}^k = 0$; and, for all the other vehicles, all the arcs arriving at and leaving from i or j in any route are forbidden. It should be observed that, regarding the other vehicles, it is not allowed to forbid all the arcs that leave from or arrive at the vertices corresponding to the depot because these vertices must be visited in a workday.

5.6. Problem Lower Bounding

The following sections describe alternative methods to find lower bounds.

5.6.1. Alternative Lower Bounding

Many real-life applications of the vehicle routing and scheduling problem require maximum resolution times of few hours so that warehouse operations and the beginning of delivery routes are not delayed; e. g., a beverage company daily distribution

operation of goods to customers which placed orders on the day before and must be served during the current day.

It should be highlighted that the vehicle routing and scheduling problem considered in this study is a variant of the classical vehicle routing problem, which is NP-hard in the strong sense (Archetti et al., 2011); therefore, the problem studied is also NP-hard and obtaining optimal solutions for real instances of many customers and vehicles may not be tractable.

From the observations above, one may be interested in solving only the root node of the branch-and-price tree in order to obtain a valid lower bound which allows a precise evaluation of the maximum error between the heuristic solution and an optimal solution, i.e., to check the quality of the heuristic solution.

In order to obtain valid lower bounds, as described in Section 5.3, IP formulation was firstly rewritten as a set covering problem over the set of all feasible workdays of all vehicles. This set covering problem was denoted master problem or MP and it was equivalent to IP. Then, MP binary variables, $\lambda_p^k \in \{0,1\}$, were relaxed to $\lambda_p^k \geq 0$ to obtain the linear master problem, LMP (linear relaxation of the master problem). LMP were solved then by applying the column generation technique where the pricing subproblem was an elementary shortest path problem with resource constraints.

Computational results from Chapter 6 show that the lower bound obtained by solving the pricing subproblem as an elementary path problem, i.e., assuring elementary conditions; yielded excellent lower bounds. However, the computational cost required to find an elementary shortest path may be prohibitive for large instances.

One way to overcome this difficulty is to relax, besides the MP binary variables λ_p^k which are responsible for selecting columns, the pricing subproblem by allowing the existence of cycles in solutions (columns). In other words, it is possible to disregard the elementary path constraints described in Section 5.4.7 and remove the visit vector S from the dominance rule described in Section 5.4.12. The resulting pricing subproblem after removing elementary path constraints, as stated above, is a shortest path problem with resource constraints, SPPRC.

As the SPPRC is a relaxed version of the ESPPRC, the lower bound obtained by solving the SPPRC is worse than that obtained by solving the ESPPRC. However, the computational cost required to find a shortest path is usually lower than that required to

find an elementary shortest path due to the removal of the visit vector S from the dominance rule, which reduces the number of states (or labels) generated.

In both cases presented above, it is also possible to obtain a weaker lower bound at a lower computational cost by considering the root node column generation lower bound instead of the linear restricted master problem (LRMP) optimal solution.

All valid lower bounds considered in this research are listed below:

- 1) Branch-and-price tree root node optimal solution with the elementary shortest path problem (ESPPRC) as the column generation subproblem;
- 2) Branch-and-price tree root node optimal solution with the shortest path problem (SPPRC) as the column generation subproblem;
- 3) Branch-and-price tree root node column generation lower bound with the ESPPRC as the column generation subproblem;
- 4) Branch-and-price tree root node column generation lower bound with the SPPRC as the column generation subproblem.

5.6.2. Pseudo Lower Bound

Even if the pricing subproblem is relaxed by allowing cycles on solutions (SPPRC), the computational cost required to find the optimal solution for the shortest path problem (pricing subproblem) for very large instances may not be tractable.

One way to overcome this challenge is to use the near-optimal heuristic labelling algorithm described in Section 5.4.14 to generate columns. Because this heuristic labelling algorithm is a near-optimal algorithm, the column generation primal bound when no columns with negative reduced costs are found by the heuristic algorithm is very close or even equal to the optimal LMP objective value, which is a valid lower bound for the IP.

It is defined herein as pseudo-lower bound the column generation primal bound when no columns with negative reduced costs can be found by the heuristic labelling algorithm.

Although the pseudo-lower bound is not a valid lower bound, its value is very close to the actual lower bound that would be obtained; therefore, it allows one to foresee the error between an incumbent solution and an optimal solution.

6. Computational Results

Solomon's benchmark vehicle routing problem sets (Solomon's Problem Sets url) R1, C1, RC1, R2, C2 and RC2 were adapted to the context of the problem studied. With regard to these instances, problem sets R1, C1 and RC1 have a short scheduling horizon and allow only a few customers per route. In contrast, problem sets R2, C2 and RC2 have a long scheduling horizon permitting many customers to be serviced by the same vehicle. Besides, the problem sets also differ with respect to the width of the time windows as R1, C1 and RC1 have very tight time windows, while the remaining sets have time windows which are hardly constraining. Therefore, the computational cost of solving R2, C2 and RC2 is significantly higher than the computational cost of solving R1, C1 and RC1 instances.

A customer has demand for one product only, as in Solomon's instances, on the grounds that it is possible to convert demands for different products in a demand for a particular one based on volume ratios.

The maximum workday duration for any driver is $\left\lfloor \frac{9.5}{16} \cdot (b_{DC} - a_{DC}) \right\rfloor$, where $[a_{DC}, b_{DC}]$ is the depot time window, 9.5 hours is the maximum workday duration in the real-life context analyzed and 16 hours is the delivery operation planning horizon.

Based on Solomon's instance service time at each customer, defined as s_i , and considering a depot loading time $s_{DC} = 10$ for R1, C1, RC1, R2, and RC2 instances and $s_{DC} = 90$ for C2 instances, then the service/loading times are computed as $s_i^k = 1.5s_i$ for a truck, $s_i^k = 1.25s_i$ for a LCV and $s_i^k = s_i$ for a van.

In every instance, customer 2 can only be served by a specific truck, and customer 3 can only be served by another specific truck.

The tabu search is defined by the following parameters:

- 1) Short-term memory size = θ , which is dynamically updated by the algorithm;
- 2) Number of iterations without improvement that triggers diversification = 100;
- 3) Number of random moves required to diversify a solution = 60;
- 4) Maximum number of attempts made to find a random feasible move during the diversification process = 1000;
- 5) Number of diversifications = 1000.

Regarding the implemented BoxPen stabilization method, the linear penalty for leaving the box is initially set to 0.001, and the box size around each dual variable is initially set to 0.2. If a subproblem is able to find a negative reduced cost path, then the linear

penalty is increased by 10%. However, when there are no more such columns, the penalty is significantly decreased (divided by 1000), and every dual box is re-centered on the corresponding last dual value. This procedure guarantees column generation convergence. Furthermore, the initial dual solution is defined with potentially good initial dual values based on some knowledge of the problem, and consequently, they are estimated and computed as $\pi_i = \min_{j \in N} \{d_{ij}\}$ and $\sigma_k = VC_k \cdot \min_{j \in N} \{d_{O_1j}\}$.

The fleet is divided into van-type vehicles, LCVs (light cargo vehicles) and trucks. The capacities of a truck, a LCV and a van are 200 (as in Solomon's instances), 120 and 40, respectively. The costs per unit of distance run by a truck, LCV and van are 1, 0.75 and 0.5, respectively. The average speeds of a truck, LCV and van are 1, 1.333 and 1.666, respectively. The differences with regard to average displacement speed among different types of vehicles are, primarily, based on parking time, vehicle manoeuvring time in congested streets along the trip and traffic movement restrictions for trucks. Note that with a speed equal to 1, travel times are equal to the corresponding Euclidean distances, as in Solomon's instances.

In each branch-and-price tree node, except for the root node, a tabu search with the same parameters as above is used, except those concerning the number of diversifications, which is reduced to 10.

A workstation with an Intel® Core™ i7 2.80GHz processor and 16374-Mb RAM memory was used to run the program developed in Java. CPLEX™ 12.2 is used to solve each LRMP and find a feasible initial solution when the constructive heuristic insertion procedure fails to find one. The Java software developed is multithreaded.

In every instance of every scenario, CPLEX™ was not required to find a feasible solution at the root node.

6.1. First Set of Scenarios: Small Instances

The adapted Solomon's vehicle routing problem instances R1, C1, RC1, R2, C2 and RC2 with the first 20 customers are considered small instances in this study and they were solved to optimality in cases with 1, 2 and 3 routes in a workday.

In the available fleet for R1, C1 and RC1 instances of 20 customers there are 15 owned vehicles: 5 trucks, 5 LCVs and 5 vans. Regarding R2, C2 and RC2 instances of 20 customers, there are 10 owned vehicles in the available fleet: 4 trucks, 4 LCVs and 2 vans.

The final results obtained for instances of 20 customers are presented within the next tables (from Table 3 to Table 8), which display the initial solutions found by the constructive heuristic and allow a comparison of the solutions obtained by the heuristic, the lower-bounds and the optimal solutions to be made.

Table 3: Results obtained for R1 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
R101	1	356.242	36	353.592	1	356.242	269	2249
R102	1	315.161	43	308.821	1	315.161	460	3279
R103	1	300.486	55	289.578	1	300.486	942	5187
R104	1	290.321	56	286.759	1	290.321	1186	6499
R105	1	320.480	45	320.480	1	320.480	46	1
R106	1	303.291	48	293.106	1	303.291	912	5837
R107	1	291.302	56	288.467	1	291.302	214	1011
R108	1	290.321	56	286.354	1	290.321	2286	11771
R109	1	301.360	45	292.892	1	301.360	1790	11763
R110	1	290.321	52	287.454	1	290.321	216	1039
R111	1	291.302	55	290.396	1	291.302	71	81
R112	1	290.321	57	286.354	1	290.321	2241	11159
		303.409	50	298.688	1	303.409	886	4990
R101	2	351.901	67	348.664	1	351.901	234	717
R102	2	313.691	77	306.131	1	313.691	696	2791
R103	2	300.486	86	289.518	1	300.486	7094	26819
R104	2	290.321	86	286.759	1	290.321	2085	6943
R105	2	320.480	77	320.127	1	320.480	118	207
R106	2	303.291	80	293.106	2	303.291	2434	9479
R107	2	291.302	92	288.467	1	291.302	319	869
R108	2	290.321	89	286.354	1	290.321	3605	10669
R109	2	301.360	79	292.892	1	301.360	3370	14627
R110	2	290.321	86	287.454	1	290.321	258	769
R111	2	291.302	93	290.396	1	291.302	115	51
R112	2	290.321	94	286.354	1	290.321	6467	17965
		302.925	84	298.019	1	302.925	2233	7659
R101	3	351.901	89	348.664	1	351.901	260	779
R102	3	313.691	97	306.131	2	313.691	1239	4563
R103	3	300.486	111	289.518	1	300.486	7827	24233
R104	3	290.321	107	286.759	3	290.321	2353	5741
R105	3	320.480	98	320.127	1	320.480	144	219
R106	3	303.291	104	293.106	6	303.291	2828	8305
R107	3	291.302	122	288.467	1	291.302	421	869
R108	3	290.321	113	286.354	5	290.321	6902	15359
R109	3	301.360	97	292.892	1	301.360	5731	15533
R110	3	290.321	115	287.454	1	290.321	356	873
R111	3	291.302	121	290.396	1	291.302	148	59
R112	3	290.321	119	286.354	1	290.321	6723	16987
		302.925	108	298.019	2	302.925	2911	7793

Table 4: Results obtained for R2 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
R201	1	321.624	51	318.572	1	321.624	59	37
R202	1	294.230	62	288.223	1	294.230	87	127
R203	1	285.318	74	281.365	15	285.318	624	109
R204	1	269.632	79	264.018	430	269.632	34351	199
R205	1	275.590	58	273.657	1	275.590	71	35
R206	1	272.054	74	269.999	1	272.054	88	25
R207	1	272.054	78	268.901	19	272.054	1066	89
R208	1	260.517	81	260.517	1417	260.517	1498	1
R209	1	272.699	70	267.426	1	272.699	112	135
R210	1	285.318	72	281.876	2	285.318	108	65
R211	1	264.974	74	263.208	29	264.974	168	9
		279.455	70	276.160	174	279.455	3476	76
R201	2	316.940	82	313.861	1	316.940	100	81
R202	2	282.141	99	282.141	1	282.141	100	1
R203	2	278.189	114	277.389	493	278.189	1789	17
R204	2	269.632	110	264.018	3531	*	*	*
R205	2	275.590	93	273.657	1	275.590	104	27
R206	2	272.054	109	269.999	2	272.054	173	37
R207	2	272.054	117	268.901	114	272.054	6728	67
R208	2	260.517	116	260.517	7315	260.517	7431	1
R209	2	272.699	103	267.426	4	272.699	286	139
R210	2	278.189	108	277.874	9	278.189	205	23
R211	2	264.974	109	263.208	247	264.974	887	11
		276.634	105	274.454	1065	277.335	1780	40
R201	3	316.940	105	313.303	1	316.940	152	115
R202	3	282.141	123	282.141	7	282.141	130	1
R203	3	278.189	141	277.389	1493	278.189	5796	13
R204	3	269.632	143	264.018	11435	*	*	*
R205	3	275.590	116	273.657	5	275.590	250	23
R206	3	272.054	138	269.999	22	272.054	385	19
R207	3	272.054	141	268.901	628	272.054	23230	87
R208	3	260.517	142	260.517	25548	260.517	25690	1
R209	3	272.699	131	267.426	78	272.699	2607	147
R210	3	278.189	133	277.874	119	278.189	680	9
R211	3	264.974	113	263.208	220	264.974	635	7
		276.634	130	274.403	3596	277.335	5955	42

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 5: Results obtained for C1 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
C101	1	252.932	45	241.579	1	252.932	304	1923
C102	1	242.288	62	228.388	1	*	*	*
C103	1	237.447	72	226.325	1	*	*	*
C104	1	226.745	72	218.086	4	*	*	*
C105	1	241.933	46	231.737	1	241.933	598	3511
C106	1	251.060	46	241.311	1	251.060	730	2527
C107	1	239.681	50	229.569	1	239.681	5654	26851
C108	1	237.990	57	226.457	1	*	*	*
C109	1	232.788	61	218.811	1	*	*	*
		240.318	57	229.140	1	246.402	1822	8703
C101	2	252.932	77	241.579	1	252.932	513	2149
C102	2	242.288	95	228.388	2	*	*	*
C103	2	237.447	109	226.325	3	*	*	*
C104	2	226.745	110	218.086	20	*	*	*
C105	2	241.933	84	231.737	2	241.933	867	2851
C106	2	251.060	80	241.311	1	251.060	4241	3061
C107	2	239.681	82	229.569	1	239.681	10470	38167
C108	2	237.990	95	226.457	2	*	*	*
C109	2	232.788	95	218.811	5	*	*	*
		240.318	92	229.140	4	246.402	4023	11557
C101	3	252.932	102	241.579	1	252.932	393	1187
C102	3	242.288	143	228.388	3	*	*	*
C103	3	237.447	157	226.325	10	*	*	*
C104	3	226.745	155	218.086	78	*	*	*
C105	3	241.933	111	231.737	4	241.933	1234	2951
C106	3	251.060	101	241.311	2	251.060	11663	2941
C107	3	239.681	111	229.569	12	239.681	10754	26399
C108	3	237.990	137	226.457	4	*	*	*
C109	3	232.788	136	218.811	14	*	*	*
		240.318	128	229.140	14	246.402	6011	8370

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 6: Results obtained for C2 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
C201	1	261.897	38	260.913	1	261.897	44	7
C202	1	242.171	66	230.677	45	242.171	4853	509
C203	1	238.909	82	222.853	2172	238.909	337273	2531
C204	1	234.754	88	220.585	6788	*	*	*
C205	1	256.385	47	256.300	3	256.385	64	9
C206	1	256.385	54	256.153	8	256.385	107	9
C207	1	256.216	65	251.410	109	256.216	12668	167
C208	1	256.385	60	249.610	88	256.385	3480	187
		250.388	63	243.563	1152	252.621	51213	488
C201	2	261.897	63	260.913	1	261.897	77	13
C202	2	242.171	96	230.677	264	242.171	17815	521
C203	2	238.909	115	222.853	25041	*	*	*
C204	2	234.754	129	220.585	126749	*	*	*
C205	2	256.385	72	256.300	18	256.385	121	9
C206	2	256.385	85	256.153	115	256.385	332	11
C207	2	256.216	98	251.410	4420	256.216	33383	141
C208	2	256.385	95	249.610	1404	256.385	10548	249
		250.388	94	243.563	19751	254.907	10379	157
C201	3	261.897	79	260.913	2	261.897	94	9
C202	3	242.171	114	230.677	385	242.171	29279	507
C203	3	238.909	142	222.853	58299	*	*	*
C204	3	234.754	144	*	*	*	*	*
C205	3	256.385	92	256.300	51	256.385	184	9
C206	3	256.385	108	256.153	367	256.385	691	9
C207	3	256.216	118	251.410	12231	256.216	71220	153
C208	3	256.385	117	249.610	6582	256.385	26125	241
		250.388	114	246.845	11131	254.907	21266	155

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 7: Results obtained for RC1 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
RC101	1	416.368	44	414.646	1	416.368	67	59
RC102	1	411.490	57	404.859	1	411.490	1275	6133
RC103	1	398.287	68	394.937	1	398.287	226	541
RC104	1	393.771	70	387.693	1	393.771	865	1657
RC105	1	411.902	54	411.491	1	411.902	69	53
RC106	1	394.423	54	389.591	1	394.423	128	439
RC107	1	389.579	66	383.105	1	389.579	3131	8095
RC108	1	389.579	80	383.105	1	389.579	8497	11991
		400.675	61	396.178	1	400.675	1782	3621
RC101	2	416.368	78	414.646	1	416.368	103	59
RC102	2	411.490	102	404.859	2	411.490	679	2297
RC103	2	398.287	111	394.937	1	398.287	260	313
RC104	2	393.771	117	387.693	2	393.771	1070	1545
RC105	2	411.902	94	411.491	1	411.902	116	45
RC106	2	394.423	97	389.591	1	394.423	163	233
RC107	2	389.579	101	383.105	1	389.579	4428	7641
RC108	2	389.579	129	383.105	6	389.579	10028	10597
		400.675	103	396.178	2	400.675	2106	2841
RC101	3	416.368	95	414.646	1	416.368	133	59
RC102	3	411.490	120	404.859	2	411.490	540	1135
RC103	3	398.287	132	394.937	1	398.287	210	115
RC104	3	393.771	143	387.693	2	393.771	1364	2087
RC105	3	411.902	121	411.491	1	411.902	147	37
RC106	3	394.423	123	389.591	1	394.423	225	285
RC107	3	389.579	130	383.105	4	389.579	5103	6991
RC108	3	389.579	157	383.105	2	389.579	11115	10789
		400.675	128	396.178	1	400.675	2355	2687

Table 8: Results obtained for RC2 instances with the first 20 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Root Node Cost	Root Node Time (s)	Optimal Cost	Total Time (s)	Tree Size
RC201	1	372.045	59	362.729	1	372.045	238	823
RC202	1	328.640	66	328.640	1	328.640	68	1
RC203	1	320.444	77	317.506	77	320.444	1150	35
RC204	1	312.805	71	305.801	1882	312.805	66603	85
RC205	1	332.325	64	331.508	1	332.325	87	9
RC206	1	335.180	75	326.728	1	335.180	180	259
RC207	1	313.940	68	302.221	225	313.940	5228	491
RC208	1	311.822	74	300.103	742	311.822	189982	1695
		328.400	69	321.905	366	328.400	32942	425
RC201	2	372.045	96	362.729	1	372.045	321	719
RC202	2	328.640	103	328.640	5	328.640	108	1
RC203	2	320.444	120	317.506	431	320.444	5737	23
RC204	2	312.805	119	305.801	10004	*	*	*
RC205	2	332.325	100	331.508	2	332.325	172	7
RC206	2	335.180	113	326.728	8	335.180	656	197
RC207	2	313.940	103	302.221	1670	313.940	21518	437
RC208	2	311.822	110	300.103	1630	*	*	*
		328.400	108	321.905	1719	333.762	4752	231
RC201	3	372.045	124	362.729	3	372.045	538	719
RC202	3	328.640	130	328.640	8	328.640	139	1
RC203	3	320.444	150	317.506	1856	320.444	15464	29
RC204	3	312.805	144	305.801	30541	*	*	*
RC205	3	332.325	124	331.508	21	332.325	372	11
RC206	3	335.180	139	326.728	72	335.180	3868	187
RC207	3	313.940	120	302.221	5333	313.940	87983	471
RC208	3	311.822	149	300.103	12436	*	*	*
		328.400	135	321.905	6284	333.762	18060	236

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

6.2. Second Set of Scenarios: Large Instances

The adapted Solomon's vehicle routing problem instances with the first 50 customers are considered in this study large instances, which were solved with 1, 2 and 3 routes in a workday and with three different fleet mixes. In the first fleet mix considered (scenario L1) there are no chartered vehicles available, in the second fleet mix (scenario L2) there are chartered vehicles available with high associated fixed costs and, in the third fleet mix (scenario L3), there are chartered vehicles available with competitive and expensive associated fixed costs.

In the available fleet for R1, C1 and RC1 instances of 50 customers there are 20 vehicles: 15 trucks, 5 LCVs and 5 vans. Regarding R2, C2 and RC2 instances of 50 customers, in the available fleet there are 15 vehicles: 5 trucks, 5 LCVs and 5 vans.

As the computational cost of solving the problem to optimality in the second set of scenarios is high, only the root node column generation lower bound is computed and, therefore, no branching is performed.

The final results obtained for 50-customer instances are presented in tables of Sections 6.2.1 Scenario L1, 6.2.2 Scenario L2 and 6.2.3 Scenario L3 with the corresponding lower bound cells filled with the LRMP optimal solution for R1, C1 and RC1 instances and with the first valid lower bound, i.e., after one iteration with guaranteed optimality on subproblems, for R2, C2 and RC2 instances.

6.2.1. Scenario L1: No Chartered Vehicle Available

The final results obtained for instances with no chartered vehicles in the available fleet are presented in the following tables, which display the cost of the solutions found by the heuristic and allow a comparison of the solutions obtained by the heuristic and the corresponding lower-bounds to be made.

Table 9: Results obtained for R1 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
R101	1	967.497	235	955.531	2	1.237	45
R102	1	805.559	323	770.943	4	4.297	47
R103	1	657.448	375	642.114	20	2.332	84
R104	1	606.484	436	599.376	34	1.172	63
R105	1	742.452	283	722.445	4	2.695	86
R106	1	679.535	357	665.543	11	2.059	77
R107	1	604.177	394	598.596	17	0.924	63
R108	1	594.898	440	586.373	58	1.433	69
R109	1	624.043	342	623.274	5	0.123	65
R110	1	612.588	401	599.062	12	2.208	81
R111	1	606.677	402	601.543	13	0.846	56
R112	1	604.155	453	588.483	22	2.594	63
		675.459	370	662.774	17	1.827	67
R101	2	978.249	437	955.531	1	2.322	46
R102	2	796.028	549	770.479	6	3.210	85
R103	2	656.540	610	642.114	42	2.197	89
R104	2	607.629	683	599.214	197	1.385	71
R105	2	753.951	507	721.848	3	4.258	89
R106	2	677.576	598	663.198	20	2.122	103
R107	2	604.177	644	598.596	46	0.924	98
R108	2	596.087	647	584.634	255	1.921	89
R109	2	627.399	582	623.274	8	0.657	82
R110	2	612.588	652	599.062	23	2.208	91
R111	2	608.576	658	600.540	29	1.320	96
R112	2	599.212	710	588.442	46	1.797	66
		676.501	606	662.244	56	2.027	84
R101	3	962.944	549	955.531	1	0.770	41
R102	3	803.934	679	770.479	5	4.161	49
R103	3	656.540	761	642.114	60	2.197	92
R104	3	608.651	844	599.214	314	1.550	91
R105	3	742.080	646	721.848	4	2.726	93
R106	3	669.851	750	663.198	25	0.993	100
R107	3	604.177	811	598.596	53	0.924	67
R108	3	598.581	796	584.634	432	2.330	94
R109	3	625.534	733	623.274	10	0.361	50
R110	3	613.450	816	599.062	33	2.345	89
R111	3	606.677	823	600.540	42	1.012	92
R112	3	601.904	893	588.442	75	2.237	67
		674.527	758	662.244	88	1.801	77

Table 10 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for R1 instances. Table 10 also presents the average workday cost of each vehicle type regarding the solutions generated.

Table 10: Average workday cost and average number of vehicles used for R1 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	3.083	41.376	5.000	61.021	4.917	48.173
2	2.917	43.836	4.917	60.977	4.917	49.618
3	3.000	42.403	5.000	60.996	5.000	47.517

Table 11: Results obtained for C1 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
C101	1	531.162	320	503.426	3	5.222	72
C102	1	513.574	411	478.019	10	6.923	70
C103	1	505.821	470	463.480	146	8.371	95
C104	1	486.985	494	451.562	158	7.274	86
C105	1	515.703	339	480.643	4	6.798	72
C106	1	531.099	334	496.427	4	6.528	69
C107	1	499.371	367	468.609	5	6.160	70
C108	1	496.772	412	462.114	8	6.977	72
C109	1	495.884	432	455.221	53	8.200	98
		508.486	398	473.278	44	6.939	78
C101	2	530.246	557	500.974	4	5.520	81
C102	2	509.417	673	478.019	12	6.164	72
C103	2	509.642	738	463.480	339	9.058	108
C104	2	488.921	726	451.562	866	7.641	116
C105	2	512.083	581	480.643	6	6.140	81
C106	2	530.471	571	494.817	5	6.721	82
C107	2	504.975	606	468.609	9	7.202	89
C108	2	507.202	669	462.114	24	8.890	92
C109	2	497.580	685	455.221	415	8.513	105
		510.060	645	472.827	187	7.316	92
C101	3	526.653	722	500.974	5	4.876	82
C102	3	510.375	852	478.019	18	6.340	64
C103	3	497.734	930	463.480	812	6.882	95
C104	3	488.322	913	451.562	3886	7.528	116
C105	3	511.210	741	480.643	8	5.979	81
C106	3	525.619	730	494.817	6	5.860	75
C107	3	505.964	781	468.609	12	7.383	85
C108	3	505.634	856	462.114	57	8.607	81
C109	3	496.313	875	455.221	1079	8.279	121
		507.536	822	472.827	654	6.859	89

Table 12 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for C1 instances.

Table 12: Average workday cost and average number of vehicles used for C1 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	2.556	44.843	5.000	59.377	4.556	21.537
2	2.333	47.649	5.000	58.122	4.444	24.895
3	2.111	47.471	5.000	57.473	4.667	25.795

Table 13: Results obtained for RC1 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
RC101	1	1080.071	230	1026.517	3	4.958	56
RC102	1	1069.809	311	986.974	10	7.743	64
RC103	1	1046.486	333	950.419	45	9.180	66
RC104	1	998.308	325	913.479	332	8.497	78
RC105	1	1056.604	292	973.634	9	7.853	62
RC106	1	1014.810	300	921.185	12	9.226	67
RC107	1	956.531	315	893.878	55	6.550	76
RC108	1	991.799	323	893.142	133	9.947	80
		1026.802	304	944.904	75	7.994	69
RC101	2	1076.665	437	1026.517	2	4.658	62
RC102	2	1056.242	527	986.974	6	6.558	60
RC103	2	1050.128	558	950.419	48	9.495	68
RC104	2	991.781	525	913.479	403	7.895	81
RC105	2	1066.918	522	973.634	7	8.743	55
RC106	2	1013.631	537	921.185	17	9.120	70
RC107	2	987.725	530	893.878	55	9.501	82
RC108	2	971.603	512	893.142	165	8.075	76
		1026.837	518	944.904	88	8.006	69
RC101	3	1077.192	547	1026.517	2	4.704	62
RC102	3	1062.957	647	986.974	6	7.148	57
RC103	3	1049.131	684	950.419	29	9.409	67
RC104	3	982.823	649	913.479	279	7.056	78
RC105	3	1057.541	653	973.634	8	7.934	59
RC106	3	1011.733	657	921.185	11	8.950	68
RC107	3	993.495	654	893.878	56	10.027	76
RC108	3	992.275	629	893.142	144	9.990	75
		1028.393	640	944.904	67	8.152	68

Table 14 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for RC1 instances.

Table 14: Average workday cost and average number of vehicles used for RC1 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	5.125	78.582	5.000	70.563	5.000	54.334
2	4.875	79.117	5.000	70.335	5.000	57.887
3	4.875	78.190	5.000	71.418	5.000	57.984

Table 15: Results obtained for R2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
R201	1	629.643	323	617.533	3	1.923	56
R202	1	577.982	420	546.084	689	5.519	81
R203	1	546.259	980	525.949	65192	3.718	67
R204	1	504.971	1050	*	*	*	*
R205	1	579.811	465	567.687	41	2.091	67
R206	1	545.655	532	516.596	571	5.326	80
R207	1	522.382	557	*	*	*	*
R208	1	508.412	1090	*	*	*	*
R209	1	543.668	833	526.939	814	3.077	75
R210	1	556.702	532	532.627	493	4.325	76
R211	1	514.903	985	489.296	4843	4.973	92
		548.217	706	540.339	9081	3.869	74
R201	2	632.400	492	617.533	9	2.351	88
R202	2	584.674	687	550.202	2695	5.896	81
R203	2	543.985	771	*	*	*	*
R204	2	516.083	832	*	*	*	*
R205	2	582.968	610	566.487	216	2.827	84
R206	2	554.530	722	516.767	2836	6.810	105
R207	2	523.277	771	*	*	*	*
R208	2	509.368	851	*	*	*	*
R209	2	552.351	1224	519.866	2944	5.881	94
R210	2	559.125	655	530.574	2736	5.106	87
R211	2	524.933	776	491.795	22175	6.313	105
		553.063	763	541.889	4802	5.026	92
R201	3	639.635	608	617.533	18	3.455	96
R202	3	584.801	848	560.172	7223	4.212	88
R203	3	546.259	871	*	*	*	*
R204	3	514.158	1002	*	*	*	*
R205	3	582.524	844	562.628	871	3.415	89
R206	3	554.304	848	509.220	13876	8.133	98
R207	3	531.691	880	*	*	*	*
R208	3	507.904	1039	*	*	*	*
R209	3	550.108	854	517.297	7351	5.964	101
R210	3	559.496	859	534.958	18095	4.386	91
R211	3	524.667	954	494.806	92018	5.691	109
		554.141	873	542.373	19922	5.037	96

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 16 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for R2 instances.

Table 16: Average workday cost and average number of vehicles used for R2 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	2.000	100.105	3.273	78.246	2.273	37.935
2	2.000	97.454	3.182	84.648	2.273	43.938
3	2.000	96.928	3.000	81.445	2.909	36.768

Table 17 presents the average number of routes performed by vehicle type for R2 instances.

Table 17: Average number of routes performed by each type of vehicle in R2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Routes Performed by Truck	Routes Performed by LCV	Routes Performed by Van
R201	1	1.000	1.000	1.000
R202	1	1.000	1.000	1.000
R203	1	1.000	1.000	1.000
R204	1	1.000	1.000	1.000
R205	1	1.000	1.000	1.000
R206	1	1.000	1.000	1.000
R207	1	1.000	1.000	1.000
R208	1	1.000	1.000	0.000
R209	1	1.000	1.000	1.000
R210	1	1.000	1.000	1.000
R211	1	1.000	1.000	1.000
		1.000	1.000	0.909
R201	2	1.000	1.000	1.000
R202	2	1.000	1.000	1.250
R203	2	1.000	1.000	1.000
R204	2	1.000	1.500	1.000
R205	2	1.000	1.000	1.000
R206	2	1.000	1.000	1.000
R207	2	1.000	1.000	1.000
R208	2	1.000	1.500	1.000
R209	2	1.000	1.000	1.000
R210	2	1.000	1.000	1.000
R211	2	1.000	1.000	2.000
		1.000	1.091	1.114
R201	3	1.000	1.000	1.600
R202	3	1.000	1.000	1.200
R203	3	1.000	1.000	1.000
R204	3	1.000	1.000	1.000
R205	3	1.000	1.000	1.000
R206	3	1.000	1.000	1.000
R207	3	1.000	1.000	1.500
R208	3	1.000	1.500	1.000
R209	3	1.000	1.000	1.000
R210	3	1.000	1.000	1.000
R211	3	1.000	1.000	1.500
		1.000	1.045	1.164

Table 18: Results obtained for C2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
C201	1	528.431	286	514.776	22	2.584	75
C202	1	486.660	401	458.484	139	5.790	85
C203	1	473.024	485	449.026	1567	5.073	84
C204	1	470.394	545	452.567	151924	3.790	119
C205	1	522.224	348	505.535	189	3.196	91
C206	1	515.914	372	502.098	4319	2.678	103
C207	1	504.364	384	492.616	31350	2.329	111
C208	1	502.484	396	482.892	6519	3.899	98
		500.437	402	482.249	24504	3.667	96
C201	2	528.390	483	514.776	60	2.577	94
C202	2	486.004	600	457.721	512	5.819	91
C203	2	479.308	693	428.482	7937	10.604	87
C204	2	479.041	768	*	*	*	*
C205	2	520.874	485	505.046	6974	3.039	95
C206	2	516.942	563	490.882	20468	5.041	113
C207	2	504.901	586	*	*	*	*
C208	2	507.864	627	*	*	*	*
		502.916	601	479.381	7190	5.416	96
C201	3	528.753	599	512.485	95	3.077	95
C202	3	487.871	725	455.116	1367	6.714	91
C203	3	478.752	847	430.525	9806	10.073	89
C204	3	473.981	956	*	*	*	*
C205	3	524.282	601	505.920	27062	3.502	100
C206	3	517.138	694	493.004	105185	4.667	116
C207	3	510.367	711	*	*	*	*
C208	3	518.552	753	*	*	*	*
		504.962	736	479.410	28703	5.607	98

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 19 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for C2 instances.

Table 19: Average workday cost and average number of vehicles used for C2 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	3.500	90.319	2.625	66.827	1.000	15.860
2	3.500	87.577	2.500	68.250	1.500	23.317
3	3.750	88.612	2.375	66.758	1.125	20.635

Table 20 presents the average number of routes performed by vehicle type for C2 instances.

Table 20: Average number of routes performed by each type of vehicle in C2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Routes Performed by Truck	Routes Performed by LCV	Routes Performed by Van
C201	1	1.000	1.000	1.000
C202	1	1.000	1.000	1.000
C203	1	1.000	1.000	1.000
C204	1	1.000	1.000	1.000
C205	1	1.000	1.000	1.000
C206	1	1.000	1.000	1.000
C207	1	1.000	1.000	1.000
C208	1	1.000	1.000	1.000
		1.000	1.000	1.000
C201	2	1.000	1.000	1.000
C202	2	1.000	1.000	1.000
C203	2	1.000	1.000	1.000
C204	2	1.000	1.000	2.000
C205	2	1.000	1.000	1.000
C206	2	1.000	1.000	1.000
C207	2	1.000	1.000	1.000
C208	2	1.000	1.000	1.000
		1.000	1.000	1.125
C201	3	1.000	1.000	1.000
C202	3	1.000	1.000	1.000
C203	3	1.000	1.000	1.000
C204	3	1.000	1.000	1.000
C205	3	1.000	1.000	1.000
C206	3	1.000	1.000	1.000
C207	3	1.000	1.000	1.000
C208	3	1.000	1.000	1.000
		1.000	1.000	1.000

Table 21: Results obtained for RC2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
RC201	1	775.681	325.542	705.222	13	9.084	78
RC202	1	697.083	392.403	669.293	6132	3.987	96
RC203	1	674.266	488.858	*	*	*	*
RC204	1	606.261	503.163	*	*	*	*
RC205	1	711.411	387.067	653.898	657	8.084	90
RC206	1	661.163	417.535	639.695	1192	3.247	119
RC207	1	626.756	467.814	579.000	46455	7.620	99
RC208	1	574.287	456.005	*	*	*	*
		665.864	429.798	649.422	10890	6.404	96
RC201	2	778.741	538.591	708.907	319	8.968	106
RC202	2	708.350	579.260	*	*	*	*
RC203	2	695.183	704.509	*	*	*	*
RC204	2	606.630	780.233	*	*	*	*
RC205	2	718.351	653.907	653.414	8879	9.040	115
RC206	2	674.606	636.154	635.542	12527	5.791	151
RC207	2	638.869	740.468	*	*	*	*
RC208	2	578.607	670.646	*	*	*	*
		674.917	662.971	665.954	7242	7.933	124
RC201	3	777.245	691.752	705.556	1665	9.223	107
RC202	3	697.859	773.616	*	*	*	*
RC203	3	686.200	901.680	*	*	*	*
RC204	3	608.873	961.180	*	*	*	*
RC205	3	718.984	750.520	653.023	83731	9.174	102
RC206	3	672.365	821.591	*	*	*	*
RC207	3	636.838	931.521	*	*	*	*
RC208	3	578.607	830.795	*	*	*	*
		672.121	832.832	679.290	42698	9.199	105

Value of cells marked with * were not computed due to a large overall computing time or out of memory error.

Table 22 displays the average number of trucks, LCVs and vans used in the solutions generated by the heuristic procedure for RC2 instances.

Table 22: Average workday cost and average number of vehicles used for RC2 instances.

Route Qty.	Number of Trucks Used	Truck Workday Cost	Number of LCVs Used	LCV Workday Cost	Number of Vans Used	Van Workday Cost
1	4.000	117.718	2.250	61.101	0.625	22.551
2	3.875	126.338	2.500	63.661	0.250	11.397
3	3.625	129.631	2.500	63.668	0.750	18.543

Table 23 presents the average number of routes performed by vehicle type for RC2 instances.

Table 23: Average number of routes performed by each type of vehicle in RC2 instances with the first 50 customers and only owned vehicles available.

Instance	Routes Allowed	Routes Performed by Truck	Routes Performed by LCV	Routes Performed by Van
RC201	1	1.000	1.000	1.000
RC202	1	1.000	1.000	1.000
RC203	1	1.000	1.000	1.000
RC204	1	1.000	1.000	0.000
RC205	1	1.000	1.000	1.000
RC206	1	1.000	1.000	0.000
RC207	1	1.000	1.000	0.000
RC208	1	1.000	0.000	0.000
		1.000	0.875	0.500
RC201	2	1.000	1.000	1.000
RC202	2	1.000	1.000	1.000
RC203	2	1.000	1.000	0.000
RC204	2	1.333	1.000	0.000
RC205	2	1.000	1.000	0.000
RC206	2	1.000	1.000	0.000
RC207	2	1.000	1.000	0.000
RC208	2	1.200	0.000	0.000
		1.067	0.875	0.250
RC201	3	1.000	1.000	0.000
RC202	3	1.000	1.000	1.000
RC203	3	1.000	1.000	1.000
RC204	3	1.333	1.000	0.000
RC205	3	1.000	1.000	0.000
RC206	3	1.000	1.000	1.000
RC207	3	1.000	1.000	1.000
RC208	3	1.200	0.000	0.000
		1.067	0.875	0.500

In order to remark the importance of minimizing idle time and, consequently, the necessity of recomputing the service time at the distribution center, some problems of scenario L1 were solved and the corresponding solutions are presented in details. The following notation “ $\rightarrow i(t_{ir}^k)$ ” is used to represent vehicle k reaching vertex i to start the service at t_{ir}^k .

- R201 with 50 customers and $R = 3$:

Figure 20: R201 solution example.

Heuristic solution cost = 635.7970915492724

Owned truck

Workday cost = 31.048200038119788

1(235.0) → 28(255.0) → 2(707.0) → 1(737.2)

Owned truck

Workday cost = 91.30551909982545

1(150.5) → 3(183.5) → 40(224.0) → 24(247.6) → 22(280.6) → 27(588.0) → 1(614.1)

Owned LVC

Workday cost = 83.22484975896433

1(140.3) → 43(171.9) → 16(191.4) → 15(215.7) → 39(302.0) → 45(322.6) →
17(339.6) → 38(606.0) → 14(690.0) → **1(710.8)**

Owned LVC

Workday cost = 77.21474722449835

1(249.3) → 32(275.0) → 31(296.5) → 21(581.0) → 33(660.0) → 49(773.0) → **1(806.3)**

Owned LVC

Workday cost = 88.28743972004514

1(114.5) → 6(142.4) → 46(164.0) → 48(190.0) → 37(208.0) → 50(501.0) → 11(577.0)
→ **1(608.6)**

Owned van

Workday cost = 42.413435847838926

1(420.2) → 41(437.0) → 23(456.4) → 42(469.0) → 44(620.0) → **1(650.5)**

Owned van

Workday cost = 98.99014986715123

1(220.4) → 13(239.4) → 29(255.0) → **1(268.7)** → 12(298.9) → 20(313.1) → 8(331.0)
→ **1(353.7)** → 10(400.0) → 35(529.0) → 36(686.0) → **1(720.6)**

Owned van

Workday cost = 49.236355816017436

1(285.0) → 30(313.0) → 25(704.0) → 5(724.8) → 26(817.0) → **1(847.1)**

Owned van

Workday cost = 28.627860347636435

1(175.1) → 34(200.0) → 4(527.0) → 51(547.0) → **1(567.1)**

Owned van

Workday cost = 45.4485338291754

1(400.5) → 7(417.2) → 19(434.0) → 9(450.2) → 47(515.0) → 18(733.0) → **1(761.2)**

- R207 with 50 customers and $R = 3$:

Figure 21: R207 solution example

Heuristic solution cost = 537.7630867853705

Owned truck

Workday cost = 127.11350955291392

1(235.7) → 28(255.7) → 2(281.5) → 31(307.6) → 21(329.7) → 33(355.4) → 12(386.0)
→ 50(415.3) → 20(442.3) → 11(485.0) → 32(508.0) → 1(540.5)

Owned truck

Workday cost = 38.614321019555554

1(252.8) → 41(279.0) → 3(303.4) → **1(336.4)**

OwnedLCV

Workday cost = 58.23098716296403

1(328.1) → 7(349.0) → 6(369.0) → 17(389.9) → 45(407.0) → 15(423.7) → 38(444.6)
→ 14(639.0) → **1(659.8)**

OwnedLCV

Workday cost = 69.46557588457904

1(214.4) → 24(254.0) → 40(273.0) → 26(716.0) → 5(736.0) → 27(759.1) → 29(777.6)
→ **1(794.9)**

OwnedLCV

Workday cost = 73.14951658101906

1(145.3) → 8(173.7) → 49(191.7) → 48(209.0) → 37(226.9) → 47(248.4) → 46(269.0)
→ 9(323.0) → 19(343.3) → **1(367.6)**

OwnedLCV

Workday cost = 82.86344094130592

1(202.1) → 51(227.3) → 4(246.0) → 34(264.0) → 10(329.0) → 36(630.0) → 35(650.1)
→ 30(672.4) → 25(690.2) → 13(713.9) → **1(737.7)**

Owned van

Workday cost = 30.566068221881427

1(295.1) → 22(315.9) → 23(332.0) → 42(344.5) → **1(371.8)**

Owned van

Workday cost = 57.75966742115163

1(239.6) → 43(265.0) → 16(280.5) → 44(294.9) → 39(315.7) → 18(682.0) → **1(710.2)**

6.2.2. Scenario L2: Expensive Chartered Vehicles Available

In scenario L2, some of the vehicles of the available fleet are chartered vehicles with high fixed costs only.

In order to check the heuristic performance when solving the fleet size and mix problem, only R1, C1 and RC1 were solved on the grounds that, in instances R2, C2 and RC2 with the same customers and fleet available, only a few of the vehicles available are used.

For R1, C1 and RC1 instances, the owned fleet consists of 10 vehicles: 6 trucks, 2 LCVs and 2 vans. Therefore, the available fleet of chartered vehicles consists of the remaining 15 vehicles: 9 trucks, 3 LCVs and 3 vans.

The charter costs of the trucks in R1 instances are: 60, 60, 60, 70, 70, 70, 80, 80 and 80. The charter costs of the LCVs are: 80, 85 and 90. The charter costs of the vans are: 70, 75 and 80. These charter cost values are substantially higher than the average cost of an owned vehicle workday of the same type (see Table 10 for comparisons). A chartered truck of fixed cost equal to 60 is the vehicle that serves customer 2 (one of the customers with accessibility restrictions) and another chartered truck of fixed cost equal to 60 serves customer 3.

The results obtained for R1-50-customer instances are presented in Table 24.

Table 24: Results obtained for R1 instances with the first 50 customers and expensive chartered vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
R101	1	1109.684	516	1033.982	2	6.822	35
R102	1	926.757	812	858.302	10	7.387	40
R103	1	811.291	1100	762.503	61	6.014	43
R104	1	779.998	1348	696.813	638	10.665	50
R105	1	913.977	816	822.983	6	9.956	39
R106	1	857.103	1049	759.855	33	11.346	46
R107	1	804.082	1235	725.085	296	9.824	43
R108	1	758.346	1331	688.809	2750	9.170	49
R109	1	812.147	1132	726.839	36	10.504	50
R110	1	776.761	1324	698.789	156	10.038	49
R111	1	785.563	1313	702.197	207	10.612	48
R112	1	752.610	1324	680.943	743	9.522	47
		840.693	1108	763.092	411	9.322	45
R101	2	1093.234	824	1033.961	1	5.422	42
R102	2	945.663	1337	858.112	17	9.258	49
R103	2	858.931	1773	761.276	132	11.369	49
R104	2	784.036	2157	687.818	2694	12.272	85
R105	2	914.790	1222	812.776	8	11.152	45
R106	2	861.087	1734	758.237	76	11.944	55
R107	2	818.523	1767	720.311	668	11.999	61
R108	2	728.187	1949	681.170	3531	6.457	82
R109	2	813.333	1813	720.625	73	11.399	52
R110	2	784.186	2101	695.267	323	11.339	48
R111	2	793.833	2108	696.760	490	12.228	82
R112	2	764.724	2480	674.724	1658	11.769	61
		846.711	1772	758.420	806	10.551	59
R101	3	1116.837	1087	1033.961	2	7.421	39
R102	3	945.414	1773	858.112	17	9.234	42
R103	3	875.276	2360	761.276	198	13.024	48
R104	3	789.745	2537	687.818	2229	12.906	85
R105	3	923.956	1774	812.776	6	12.033	47
R106	3	865.962	2246	758.237	122	12.440	48
R107	3	812.960	2321	720.311	1098	11.397	56
R108	3	760.223	2442	681.170	5021	10.399	89
R109	3	813.805	2424	720.625	106	11.450	61
R110	3	790.291	2746	695.267	499	12.024	59
R111	3	790.538	2742	696.760	575	11.863	54
R112	3	759.898	2508	674.724	8146	11.209	56
		853.742	2247	758.420	1502	11.283	57

The charter costs of the trucks in C1 instances are: 70, 70, 70, 80, 80, 80, 90, 90 and 90. The charter costs of the LCVs are: 80, 85 and 90. The charter costs of the vans are: 45, 50 and 55. These charter cost values are substantially higher than the average cost of an owned vehicle workday of the same type (see Table 12). A chartered truck of fixed cost

equal to 70 is the vehicle that serves customer 2 (one of the customers with accessibility restrictions) and another chartered truck of fixed cost equal to 70 serves customer 3. The depot service time considered is $s_{DC} = 10$. The results obtained for C1-50-customer instances are presented in Table 25.

Table 25: Results obtained for C1 instances with the first 50 customers and expensive chartered vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
C101	1	636.396	1013	583.217	5	8.356	45
C102	1	582.411	1368	526.243	244	9.644	51
C103	1	568.065	1561	508.123	636	10.552	57
C104	1	556.137	1353	500.283	2492	10.043	89
C105	1	598.682	1102	550.087	12	8.117	46
C106	1	625.550	1085	575.290	9	8.035	51
C107	1	586.853	1205	532.755	36	9.218	78
C108	1	562.344	1477	509.330	86	9.427	81
C109	1	564.368	1494	504.379	488	10.629	90
		586.756	1295	532.190	445	9.336	65
C101	2	564.129	1856	502.412	32	10.940	104
C102	2	504.764	2277	452.357	207	10.382	93
C103	2	496.010	2312	438.129	2280	11.669	89
C104	2	487.306	2604	433.221	2496	11.099	96
C105	2	529.838	1929	474.623	50	10.421	101
C106	2	570.380	1858	489.715	88	14.142	94
C107	2	514.701	2072	461.068	82	10.420	94
C108	2	503.956	2301	440.812	170	12.530	87
C109	2	484.868	2517	432.949	825	10.708	85
		517.328	2192	458.365	692	11.368	94
C101	3	557.720	2489	482.207	46	13.540	99
C102	3	499.529	3042	423.664	27851	15.187	113
C103	3	450.299	3254	413.640	78459	8.141	89
C104	3	463.160	3055	408.487	46263	11.804	92
C105	3	498.837	2603	437.965	344	12.203	104
C106	3	553.131	2262	459.361	94	16.953	94
C107	3	501.343	2681	420.728	1049	16.080	106
C108	3	484.462	3075	412.062	12542	14.944	102
C109	3	448.681	3455	408.763	26079	8.897	93
		495.240	2879	429.653	21414	13.083	99

The charter costs of the trucks in RC1 instances are: 100, 100, 100, 120, 120, 120, 130, 130 and 130. The charter costs of the LCVs are: 90, 95 and 100. The charter costs of the vans are: 75, 80 and 85. These charter cost values are substantially higher than the average cost of an owned vehicle workday of the same type (see Table 14). A chartered truck of fixed cost equal to 100 is the vehicle that serves customer 2 (one of the

customers with accessibility restrictions) and another chartered truck of fixed cost equal to 100 serves customer 3.

The results obtained for RC1-50-customer instances are presented in Table 26.

Table 26: Results obtained for RC1 instances with the first 50 customers and expensive chartered vehicles available.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
RC101	1	1255.660	626	1178.503	4	6.145	47
RC102	1	1218.069	830	1135.172	16	6.806	61
RC103	1	1220.326	910	1117.366	74	8.437	51
RC104	1	1174.657	835	1075.776	231	8.418	54
RC105	1	1227.929	821	1132.050	17	7.808	57
RC106	1	1174.756	872	1074.715	18	8.516	53
RC107	1	1124.002	884	1054.855	137	6.152	67
RC108	1	1173.954	804	1054.746	268	10.154	65
		1196.169	823	1102.898	95	7.804	57
RC101	2	1249.045	1043	1177.690	2	5.713	48
RC102	2	1193.814	1192	1126.222	11	5.662	45
RC103	2	1202.710	1460	1109.859	97	7.720	70
RC104	2	1180.994	1338	1065.871	315	9.748	61
RC105	2	1241.599	1336	1124.255	20	9.451	65
RC106	2	1175.224	1329	1070.945	27	8.873	61
RC107	2	1178.164	1457	1047.554	151	11.086	75
RC108	2	1167.800	1236	1045.854	451	10.442	75
		1198.669	1299	1096.031	134	8.587	63
RC101	3	1254.491	1384	1177.690	2	6.122	44
RC102	3	1214.813	1553	1126.222	13	7.293	51
RC103	3	1179.662	1945	1109.859	101	5.917	69
RC104	3	1168.050	1789	1065.871	360	8.748	71
RC105	3	1194.733	1761	1124.255	13	5.899	53
RC106	3	1184.573	1717	1070.945	24	9.592	72
RC107	3	1150.661	1907	1047.554	125	8.961	70
RC108	3	1146.393	1615	1045.854	394	8.770	68
		1186.672	1709	1096.031	129	7.663	62

6.2.3. Scenario L3: Expensive and Inexpensive Chartered Vehicles Available

Scenario L3 has the same fleet size and fleet mix of scenario L2. However, some of the charter costs are competitive, i.e. substantially lower than the average cost of an owned vehicle workday of the same type, while the remaining charter costs are expensive, as in scenario L2.

In order to check the heuristic performance when solving the fleet size and mix problem, only R1, C1 and RC1 were solved on the grounds that, in instances R2, C2

and RC2 with the same customers and fleet available, only a few of the vehicles available are used.

As a consequence, for R1 instances, the charter costs of the trucks are: 60, 60, 60, 70, 24, 23, 22, 21 and 20. The charter costs of the LCVs are: 80, 45 and 40. The charter costs of the vans are: 70, 75 and 30. A chartered truck with a fixed cost equal to 60 is the vehicle that serves customer 2 and another chartered truck of fixed cost equal to 60 serves customer 3.

The results obtained for R1-50-customer instances are presented in Table 27.

Table 27: Results obtained for R1 instances with the first 50 customers and varied charter costs.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
R101	1	755.672	516	696.734	2	7.799	37
R102	1	640.887	826	579.038	8	9.651	34
R103	1	562.665	1193	507.692	47	9.770	37
R104	1	511.865	1421	453.616	419	11.380	49
R105	1	609.214	813	538.191	5	11.658	38
R106	1	558.153	1071	508.065	21	8.974	44
R107	1	557.871	1281	479.502	298	14.048	51
R108	1	516.424	1403	453.616	682	12.162	47
R109	1	541.104	1184	488.388	28	9.742	41
R110	1	507.677	1340	463.478	139	8.706	41
R111	1	540.368	1331	463.107	176	14.298	52
R112	1	515.575	1395	450.728	282	12.578	49
		568.123	1148	506.846	176	10.897	43
R101	2	737.898	815	696.734	3	5.579	32
R102	2	651.152	1402	579.038	10	11.075	36
R103	2	546.503	1650	507.663	94	7.107	35
R104	2	531.225	2212	453.616	671	14.609	57
R105	2	606.518	1344	536.475	5	11.548	40
R106	2	585.571	1602	508.065	52	13.236	40
R107	2	544.530	2021	479.225	284	11.993	48
R108	2	537.503	2014	453.616	2944	15.607	56
R109	2	558.856	1874	487.461	55	12.775	48
R110	2	547.018	2134	462.650	211	15.423	48
R111	2	528.008	2149	463.099	263	12.293	47
R112	2	531.155	2028	450.728	566	15.142	55
		575.495	1770	506.531	430	12.199	45
R101	3	755.998	1172	696.734	1	7.839	35
R102	3	654.895	1828	579.038	15	11.583	39
R103	3	572.435	2155	507.663	223	11.315	42
R104	3	534.474	3011	453.616	1354	15.129	55
R105	3	591.476	1801	536.475	5	9.299	39
R106	3	576.631	2315	508.065	41	11.891	39
R107	3	552.301	2664	479.225	357	13.231	56
R108	3	513.543	2889	453.616	1306	11.669	46
R109	3	545.597	2519	487.461	110	10.655	50
R110	3	543.704	2817	462.650	457	14.908	47
R111	3	532.843	2839	463.099	318	13.089	50
R112	3	528.615	2612	450.728	1482	14.734	52
		575.209	2385	506.531	472	12.112	46

For C1 instances, the charter costs of the trucks are: 70, 70, 70, 80, 24, 23, 22, 21 and 20. The charter costs of the LCVs are: 80, 45 and 40. The charter costs of the vans are: 45, 50 and 5. A chartered truck with a fixed cost equal to 70 is the vehicle that serves customer 2 and another chartered truck of fixed cost equal to 70 serves customer 3.

The results obtained for C1-50-customer instances are presented in Table 28.

Table 28: Results obtained for C1 instances with the first 50 customers and varied charter costs.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
C101	1	382.894	1077	361.690	8	5.538	51
C102	1	368.609	1399	349.917	285	5.071	46
C103	1	368.811	1681	344.692	466	6.540	44
C104	1	365.509	1468	341.275	1083	6.630	57
C105	1	375.689	1135	352.123	16	6.273	42
C106	1	386.437	1140	358.858	11	7.137	44
C107	1	367.256	1238	348.642	29	5.068	39
C108	1	365.464	1544	345.352	71	5.503	47
C109	1	370.116	1580	341.694	305	7.679	50
		372.309	1362	349.360	253	6.160	47
C101	2	372.151	1728	342.783	26	7.891	133
C102	2	349.666	2413	330.211	1969	5.564	65
C103	2	353.656	2837	325.592	7890	7.935	63
C104	2	357.497	2817	322.913	8546	9.674	96
C105	2	367.066	1987	333.191	55	9.229	64
C106	2	358.882	1917	339.713	31	5.341	61
C107	2	357.734	2087	329.497	85	7.893	49
C108	2	351.749	2446	326.397	300	7.207	61
C109	2	354.741	2650	323.258	3025	8.875	71
		358.127	2320	330.395	2437	7.734	74
C101	3	372.394	2642	342.369	25	8.063	64
C102	3	362.622	3230	330.211	4750	8.938	199
C103	3	348.959	3498	325.592	39766	6.696	219
C104	3	351.099	3380	322.913	51001	8.028	151
C105	3	358.330	2669	333.191	88	7.016	50
C106	3	375.850	2362	339.681	181	9.623	67
C107	3	354.707	2811	329.497	183	7.107	58
C108	3	354.133	3010	326.397	693	7.832	56
C109	3	344.433	3600	323.258	7126	6.148	73
		358.059	3022	330.345	11535	7.717	104

For RC1 instances, the charter costs of the trucks are: 100, 100, 100, 120, 44, 43, 42, 41 and 40. The charter costs of the LCVs are: 90, 45 and 50. The charter costs of the vans are: 75, 80 and 35. A chartered truck with a fixed cost equal to 100 is the vehicle that serves customer 2 and another chartered truck of fixed cost equal to 100 serves customer 3.

The results obtained for RC1-50-customer instances are presented in Table 29.

Table 29: Results obtained for RC1 instances with the first 50 customers and varied charter costs.

Instance	Routes Allowed	Heuristic Cost	Heuristic Time (s)	Lower Bound	Lower Bound Time (s)	Heuristic Error (%)	Column Generation Iterations
RC101	1	962.813	579	889.575	3	7.607	41
RC102	1	928.875	799	843.852	19	9.153	47
RC103	1	930.875	886	832.023	39	10.619	44
RC104	1	907.371	821	802.406	203	11.568	45
RC105	1	923.214	805	844.059	16	8.574	42
RC106	1	936.733	851	814.057	12	13.096	38
RC107	1	906.968	925	794.958	157	12.350	45
RC108	1	898.700	813	794.223	270	11.625	45
		924.444	810	826.894	90	10.574	43
RC101	2	956.324	1002	889.575	2	6.980	39
RC102	2	924.211	1291	843.852	18	8.695	45
RC103	2	930.792	1433	832.023	37	10.611	38
RC104	2	906.304	1335	802.406	155	11.464	38
RC105	2	928.456	1338	844.059	12	9.090	42
RC106	2	921.281	1416	814.057	10	11.639	37
RC107	2	903.178	1428	794.958	84	11.982	43
RC108	2	905.374	1308	794.223	283	12.277	46
		921.990	1319	826.894	75	10.342	41
RC101	3	954.226	1339	889.575	2	6.775	36
RC102	3	935.850	1709	843.852	18	9.830	48
RC103	3	937.447	1867	832.023	54	11.246	39
RC104	3	896.178	1742	802.406	160	10.464	41
RC105	3	944.331	1740	844.059	13	10.618	43
RC106	3	937.604	1848	814.057	10	13.177	39
RC107	3	926.345	1873	794.958	89	14.183	43
RC108	3	911.103	1729	794.223	268	12.828	44
		930.386	1731	826.894	77	11.140	42

7. Conclusions and Future Work

In this study, a mathematical formulation with arc flow variables that is able to solve a complex vehicle routing and scheduling problem was presented. A constructive heuristic algorithm and a learning metaheuristic algorithm based on tabu search, which can generate good solutions with low computational cost, were also presented.

Regarding the heuristic performance in the first set of scenarios, almost every R1, C1 and RC1 instance with 20 customers, despite the number of routes in a workday, were solved to optimality by the heuristic procedure at a low computational cost.

An excellent heuristic performance in the same set of scenarios is noteworthy for R2, C2 and RC2 instances, even though the optimal solution of two of the R2 instances, five of the C2 instances and four of the RC2 instances could not be computed in an acceptable period of time and it could only be possible to foresee a small error based on the corresponding lower bound in these cases.

In the second set of scenarios, the heuristic algorithm yielded a good solution at a low computational cost for every instance, despite the fleet cost structure. An excellent heuristic performance (5.73% average heuristic error) was observed in the second set of scenarios for instances with only owned vehicles in the available fleet (L1 instances). The existence of chartered vehicles in the available fleet slightly increased the average heuristic error to 9.88% among L2 instances and 9.87% among L3 instances. Unfortunately, the computational cost of finding the lower bound of some instances with 50 customers was too high, and it was not possible to assess the heuristic performance in these cases.

Regarding the increase of average heuristic error in the second set of scenarios when compared to the first set of scenarios, it is not possible to assess whether it is primarily a consequence of a worse performance of the heuristic procedure with the parameters used or a loss in quality of the lower bounds obtained.

It should also be noted that the standard deviation of heuristic resolution times for instances with one, or two or three routes in a workday was very low in both sets of scenarios, which, together with near-optimal results and the low computational cost observed, makes this heuristic solution procedure robust, efficient and effective to the daily routing operations.

As expected, the possibility for a vehicle to perform two or three instead of one route in a workday yielded a reduction in the average cost of operation for many instances.

Aiming at evaluating the performance of the heuristic developed and to solve small and medium size instances to optimality, it was presented a branch-and-price algorithm based on a strategy of branching on IP formulation arc flow variables.

It was also presented a bidirectional dynamic programming algorithm to solve the pricing subproblem that is able to deal with multiple routes in a workday, limitation on driver work hours, fixed and variable costs, accessibility restrictions on customers, among others.

It was noteworthy the existence of a strong dependence between total time of branching and particular characteristics of the instance. An example of that dependence was the increase of the total time required from 59s (R201) to 34351s (R204) among R2 instances with one route in a workday in the first set of scenarios.

Also noteworthy is that the lower bound obtained from the column generation optimal solution at the root node of the branch-and-price tree, in which the subproblem was an elementary shortest path problem, is excellent.

In the second set of scenarios, the lower bound of the column generation at the root node, obtained after solving several iterations with the heuristic labeling algorithm and after solving the first iteration with guaranteed optimality in subproblems, was also excellent. This occurs because this lower bound is obtained near the end of the tailing-off effect, which is characteristic of the column generation convergence, despite the known dual bound oscillation. Using this procedure, the majority of instances of the second set of scenarios could be solved in an acceptable time and with maximum errors with regard to optimal solutions of 5.73%, 9.88% and 9.87% on average for L1, L2 and L3 scenarios, respectively.

It should be emphasized that the vehicle routing problem considered in this study is a variant of the classical vehicle routing problem, which is NP-hard in the strong sense (Archetti et al., 2011). Therefore, the problem studied is also NP-hard. Given that the heuristic error is very low for almost every 50-customer instance and that the computational cost to optimality is high for some 20-customer instances, the best approach to solve large instances of the problem presented herein is the use of the proposed heuristic algorithm. For small and some medium size instances, the branch-and-price algorithm proposed may be employed.

Future work related to this study may include the discovery of some more good valid cuts that may be applied to the proposed branch-and-price procedure in order to develop a robust branch-and-price-and-cut procedure.

Alternative methods to solve the linear relaxation at each node of the branch-and-bound tree may also be addressed, particularly, those based on Lagrangian relaxation.

A branch-and-bound algorithm to solve to optimality the elementary shortest path problem with resource constraints may also be developed, by exploiting the lower bound given by the bounded bidirectional dynamic programming algorithm with state-space relaxation.

Regarding the problem studied, new similar variations including variable costs for chartered vehicles, lunch and dinner breaks, overtime penalties, among others may also be considered.

8. References

- Archetti, C., Feillet, D., Gendreau, M., Speranza, M. G. (2011). Complexity of the VRP and SDVRP. *Transportation Research Part C: Emerging Technologies*, Volume 19, Issue 5, Pages 741–750.
- Azi, N., Gendreau, M., Potvin, J.-Y. (2007). An exact algorithm for a single vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, Volume 178, Pages 755–766.
- Azi, N., Gendreau, M., Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, Volume 202, Pages 756–763.
- Ballou, R. H. (2004). *Business Logistics/Supply Chain Management*. Englewood Cliffs NJ. Prentice-Hall (5th ed.).
- Beasley, J. E., Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, Volume 19, Issue 4, Pages 379–394.
- Bolduc, M. C., Renaud, J., Boctor, F. (2007). A heuristic for the routing and carrier selection problem. *European Journal of Operational Research*, Volume 183, Issue 2, Pages 926–932. doi:10.1016/j.ejor.2006.10.013.
- Bolduc, M. C., Renaud, J., Boctor, F., Laporte, G. (2008). A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society*, Volume 59, Pages 776–787.
- Brandão, J. C. S., Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, Volume 100, Issue 1, Pages 180–191.
- Brandão, J. C. S., Mercer, A. (1998). The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, Volume 49, Pages 799–805.

Brandão, J. (2009). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, Volume 195, Issue 3, Pages 716–728.

Brandão, J. C. S. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers and Operations Research*, Volume 38, Issue 1, Pages 140–151.

Bräysy, O., Gendreau, M. (2005). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science*, Volume 39, Issue 1, Pages 119–139.

Ceselli, A., Righini, G., Salani, M. (2009). A Column Generation Algorithm for a Rich Vehicle-Routing Problem. *Transportation Science*, Volume 43, Issue 1, Pages 56–69.

Christofides, N., Mingozzi, A., Toth, P. (1981). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, Volume 11, Issue 2, Pages 145–164.

Chu, C. W. (2005). A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, Volume 165, Issue 3, Pages 657–667.

Clarke, G., Wright, J.W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, Volume 12, Number 4, Pages 568–581.

Cordeau, J. F., Laporte, G., Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, Volume 52, No. 8, Pages 928–936.

Cordeau, J. F., Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B: Methodological*, Volume 37, No. 6, Pages 579–594.

Côté, J. F., Potvin, J. Y. (2009). A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. *European Journal of Operational Research*, Volume 198, Issue 2, Pages 464–469.

Dantzig, G., Ramser, J. (1959). The truck dispatching problem. *Management Science*, Volume 6, Pages 80–91.

Dejax, P., Feillet, D., Gendreau, M., Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Wiley Periodicals, Inc. Networks*, Volume 44, Issue 3, Pages 216–229.

Desaulniers, G., Lavigne, J., Soumis, F. (1998). Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research*, Volume 111, Issue 3, Pages 479–494.

Desaulniers, G., Villeneuve, D. (2000). The Shortest Path Problem with Time Windows and Linear Waiting Costs. *Transportation Science*, Volume 34, No. 3, Pages 312–319.

Desrochers, M., Lenstra, J. K., Savelsbergh, M. W. P., Soumis, F. (1988). Vehicle routing with time windows: optimization and approximation. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.). North-Holland, Amsterdam, Pages 65–84.

Desrochers, M., Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, Volume 26, Pages 191–212.

Desrochers, M., Desrosiers, J., Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* Volume 40, No. 2, Pages 342–354.

Desrosiers, J., Sauvé, M., Soumis, F. (1988). Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. *Management Science*, Volume 34, No. 8, Pages 1005–1022.

Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F. (1995). Time constrained routing and scheduling in network routing, Ball M. O. et al. eds., *Handbooks in Operations Research and Management Science*, Elsevier Science.

Desrosiers, J., Lübbecke, M. E. (2005). Column generation. Chapter 1: A Primer in Column Generation. Edited by: Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon. GERAD 25th anniversary series. Springer, USA.

Diaby, M., Ramesh, R. (1995). The distribution problem with carrier service: a dual based approach. *ORSA Journal on Computing*, Volume 7, No. 1, Pages 24–35.

Fleischmann, B. (1990). The vehicle routing problem with multiple use of vehicles. Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany.

Garcia, B.-L., Potvin, J.-Y., Rousseau, J.-M. (1994). A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers and Operations Research*, Volume 21, Issue 9, Pages 1025–1033.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and post-optimization procedures for the traveling salesman problem. *Operations Research*, Volume 40, No. 6, Pages 1086–1094.

Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, E. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, Volume 26, Pages 1153–1173.

Gendreau, M. (2003). An introduction to tabu search. Glover, F., Kochenberger, G. A. eds. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, MA.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, Volume 13, Issue 5, Pages 533–549.

Glover, F. (1989). Tabu Search – Part I. *Operations Research Society of America Journal on Computing*, Volume 1, No. 3, Pages 190–206.

Glover, F. (1990). Tabu Search – Part II. *Operations Research Society of America Journal on Computing*, Volume 2, No. 1, Pages 4–32.

Glover, F., Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston, MA.

Hall, R. (2006). On the road to integration. *OR/MS Today*, Volume 33, Pages 50–57.

Hertz, A., Taillard, E., Werra, D. de. (1997). *Tabu Search*. Aarts, E., Lenstra, J. K. eds. *Local Search in Combinatorial Optimization*. John Wiley and Sons, Chichester, UK, Pages 121–136.

Kim, S., Chang, K.-N., Lee, J.-Y. (1995). A descent method with linear programming subproblems for nondifferentiable convex optimization. *Mathematical Programming*, Volume 71, No. 1, Pages 17– 28.

Lenstra, J. K., Rinnooy, K. (1981). Complexity of the vehicle routing and scheduling problems. *Networks*, Volume 11, Issue 2, Pages 221–227.

Li, X., Tian, P., Aneja, Y. (2010). An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *Transportation Research, Part E: Logistics and Transportation Review*, Volume 46, Issue 6, Pages 1111–1127.

Liu, S., Huang, W., Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Volume 45*, Pages 434–445.

Lübbecke, M. E., Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, Volume 53, No. 6, Pages 1007–1023.

Marsten, R. E., Hogan, W. W., Blankenship, J. W. (1975). The BOXSTEP method for large-scale optimization. *Operations Research*, Volume 23, No. 3, Pages 389–405.

Merle, O. du, Villeneuve, D., Desrosiers, J., Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, Volume 194, No. 1–3, Pages 229–237.

Mitroff, I. I., Betz, F., Pondy, L. R., Sagasti, F. (1974). On managing science in the systems age: two schemas for the study of science as a whole systems phenomenon. *Interfaces*, Volume 4, No. 3, Pages 46–58.

Moura, A., Oliveira, J.F. (2005). A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, Volume 20, Issue 4, Pages 50–57.

Ochi, L., Vianna, D., Drummond, L. M. A., Victor, A. (1998a). An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. *Lecture Notes in Computer Science*, Volume 1391, Pages 187–195.

Ochi, L., Vianna, D., Drummond, L. M. A., Victor, A. (1998b). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems*, Volume 14, Pages 285–292.

Olivera, A., Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers and Operations Research* 34, Pages 28–47.

Petch, R. J., Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, Volume 133, Pages 69–92.

Petch, R. J., Salhi, S. (2007). A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, Volume 6, Issue 4, Pages 591–613.

Prins, C. (2002). Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, Volume 1, Pages 135–150.

Prins, C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, Volume 22, Pages 916–928.

Righini, G., Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Elsevier, Discrete Optimization*, Volume 3, Pages 255–273.

Righini, G., Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Wiley Periodicals, Inc. NETWORKS*, Volume 51 (3), Pages 155–170.

Rousseau, L.-M., Gendreau, M., Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, Volume 35, Issue 5, Pages 660–668.

Scheithauer, G. (1992). Algorithm for the container loading problem. In: *Operational Research Proceedings*, 1991. Springer, Berlin, Pages 445–452.

Seixas, M. P., Mendes, A. B. (2013). Column Generation for a Multitrip Vehicle Routing Problem with Time Windows, Driver Work Hours, and Heterogeneous Fleet. *Mathematical Problems in Engineering*, Volume 2013, Article ID 824961, 13 pages, doi:10.1155/2013/824961.

Solomon's Problem Sets. Available at: <http://w.cba.neu.edu/~msolomon/problems.htm>

Solomon, M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, Volume 35, No. 2., Pages 254–265.

Solomon, M., Desrosiers, J. (1988). Time window constrained routing and scheduling problems. *Transportation Science*, Volume 22, Pages 1–13.

Tan, K. C., Chew, Y. H., Lee, L. H. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, Volume 172, Pages 855–885.

Taillard, É. D., Laporte, G., Gendreau, M. (1996). Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, Volume 47, Pages 1065–1070.

Taillard, É. D. (1999). A heuristic column generation method for heterogeneous fleet. *RAIRO - Operations Research*. Volume 33, Issue 01, Pages 1–14.

UPS. Driving success: Why the UPS model for managing 103500 drivers is a competitive advantage. Available at:

<http://pressroom.ups.com/mediakits/popups/factsheet/0,1889,1201,00.html>.

Volgenant, T., Jonker, R. (1987). On some generalizations of the travelling-salesman problem. *Journal of the Operational Research Society*, Volume 38, Issue 11, Pages 1073–1079.

Wolsey, L. (1998). *Integer programming*, Wiley-Interscience, USA.

Zäpfel, G., Bögl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, Volume 113, Pages 980–996.