ULISSES EDUARDO FERREIRA DA SILVEIRA

HEURISTIC APPROACHES TO THE DOUBLE VEHICLE ROUTING PROBLEM WITH MULTIPLE STACKS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2017

ULISSES EDUARDO FERREIRA DA SILVEIRA

HEURISTIC APPROACHES TO THE DOUBLE VEHICLE ROUTING PROBLEM WITH MULTIPLE STACKS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 06 de março de 2017.

[illegible]
Mauro Nacif Rocha

[illegible]
Heleno do Nascimento Santos

[illegible]
André Gustavo dos Santos
Orientador

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T

Silveira, Ulisses Eduardo Ferreira da, 1989-

S587h Heuristic approaches to the double vehiche routing problem
2017 with multiple stacks / Ulisses Eduardo Ferreira da Silveira. – Viçosa, MG, 2017.

vii, 66f. : il. ; 29 cm.

Inclui apêndice.

Orientador: André Gustavo dos Santos.

Dissertação (mestrado) - Universidade Federal de Viçosa.

Referências bibliográficas: f.45-47.

1. Heurística. 2. Programação heurística. 3. Veículos a motor. 4. Levantamento e carregamento. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22 ed. 004

“A ciência é, portanto, uma perversão de si mesma,
a menos que tenha como fim último melhorar a humanidade.”
(Nikola Tesla)

Acknowledgments

Agradeço primeiramente à Deus por ter me dado forças para continuar uma carreira com tantas mudanças e por ter colocado pessoas tão boas no meu caminho.

Agradeço à minha esposa Elisa por tantos anos compartilhados. Por sempre acreditar em meus sonhos e pelas sugestões valiosas.

Agradeço aos meus pais Jesiel e Rosa por todo suporte financeiro e por serem meu maior exemplo, se abdicando de prazeres e conforto em prol de uma melhor educação para os filhos.

Agradeço à minha irmã Deborah por me acompanhar desde pequeno e ao cunhado Radical pelo carinho.

Agradeço aos meus segundos pais Ney e Cássia pelas orientações de carreira.

Agradeço ao Prof. Dr. André Gustavo dos Santos pela oportunidade e orientação.

Agradeço à Universidade Federal de Viçosa pelo espaço primoroso e por fornecer tão grande conhecimento intelectual por meio de seu corpo docente.

Agradeço à CAPES pelo aporte financeiro deste trabalho.

Agradeço aos meus colegas de Mestrado que contribuíram para a conclusão deste trabalho, em especial ao Jonatas Chagas e ao Marcelo Pinheiro por compartilharem seus conhecimentos.

Agradeço aos tantos outros amigos feitos em Viçosa, pela amizade e convivência. Em especial aos doutos colegas de República dos Frávios, aos amigos do louvor, do coral e do diaconato na Igreja Presbiteriana de Viçosa e, por fim,aos amigos da graduação em Engenharia de Alimentos e do mestrado em Ciênca da Computação.

Contents

Abstract **vii**

Resumo **viii**

1 Introduction **1**
1.1 The problem and its importance 2
1.2 Objectives 3
1.2.1 Specific Objectives 3
1.3 Chapters 4

2 Pickup and Delivery problems **5**
2.1 Double Traveling Salesman Problem with Multiple Stacks 6
2.1.1 Mathematical formulation 7
2.2 Double Vehicle Routing Problem with Multiple Stacks 8
2.2.1 Mathematical formulation 9
2.2.2 Modification in the orders offering 12
2.2.3 Mathematical model based on a *score* value 12

3 Methods **15**
3.1 Solution representation 15
3.2 Evaluation function for the DVRPMS with Surplus Demand 16
3.3 Initial solution generation procedure 16
3.4 DVRPMS neighboorhood structures 17
3.4.1 route-swap 17
3.4.2 complete-swap 18
3.4.3 instack-swap 19
3.4.4 r-stack-permutation 19

3.4.5 route-exchange 20
3.5 DVRPMS with Surplus Demand neighboorhood structures 20
3.5.1 swap 21
3.5.2 ILS perturbation 22
3.6 Metaheuristics applied to the DVRPMS 22
3.6.1 ILS approach 23
3.6.2 VND approach 24
3.6.3 SA approach 24
3.7 Metaheuristics applied to the DVRPMS with Surplus Demand 24
3.7.1 Local search 26
3.7.2 SA+LS approach 26
3.7.3 ILS+LS approach 27
3.7.4 TS+LS approach 28

4 Computational Experiments **30**
4.1 Instances 30
4.1.1 Calibration 31
4.2 Results 32
4.2.1 DVRPMS 32
4.2.2 DVRPMS with Surplus Demand 38

5 Conclusions **43**
5.1 Publications 43
5.2 Further works 44

Appendix A Parameter Calibration **48**
A.1 TS+LS parameter calibration 53
A.2 SA+LS parameter calibration 55
A.3 ILS+LS parameter calibration 62
A.4 DVRPMS-ILS parameter calibration 66

Abstract

SILVEIRA, Ulisses Eduardo Ferreira da, M.Sc., Universidade Federal de Viçosa, March, 2017. **Heuristic Approaches to the Double Vehicle Routing Problem with Multiple Stacks**. Adviser: André Gustavo dos Santos

In a world, that in a fast pace, has become increasingly needed in consumable and non-consumable goods, the logistics in the transportation of these products has been put to the test, being one of the most important stages in the relationship between the pro-duction process and the end user. It is said that at least 30% of the costs between the industry and the end user are solely determined by the cost of transportation. A novel problem arose followed by a question that was encountered in a real-life scenario. The Double Routing Vehicle Problem with Multiple Stacks (DVRPMS) consists in a Dou-ble Traveling Salesman Problem with Multiple Stacks (DTSPMS) with multiple vehicles. Both problems appeared for the urgent need of optimizing intermodal transportation in the european context. It consists in gathering costumer inquires from a pickup region and loading them in a set of stacks inside a container that must not be rearranged for security reasons. The container moves to a delivery region and the items gathered must be delivered according the last-in-first-out policy of the stacks. In this work, four heuristics were proposed based on the Iterated Local Search (ILS), Variable Neighborhood Descent and Simulated Annealing (SA) metaheuristics. The DVRPMS was extended to a modi-fied version where the items offered are bigger than the vehicle fleet capacities. An exact model approach is proposed and three other heuristics, based on the ILS, SA and Tabu Search are proposed and tested. The approaches presented in this work were tested by computational experiments and a statistical analysis was made to chose the best com-bination of parameters. Good results were found, providing a better average than the current literature.

Resumo

SILVEIRA, Ulisses Eduardo Ferreira da, M.Sc., Universidade Federal de Viçosa, março de 2017. **Abordagens Heurísticas para o Problema do Roteamento Duplo com Múltiplas Pilhas**. Orientador: André Gustavo dos Santos

Em um mundo que, em tempo acelerado, tem-se tornado cada vez mais necessitado em bens consumíveis e não consumíveis, a logística no transporte destes produtos tem sido colocada à prova, sendo uma das etapas mais importantes da relação entre o processo de produção e o usuário final. Cerca de um terço dos custos logísticos desta relação (produção-usuário) são determinados pelo custo de transporte dos bens, tornando essa operação muito importante. Um novo problem surgiu a partir de um entrave encontrado numa situação prática. O Problema do Roteamento de Veículos Duplo com Múltiplas Pilhas (DVRPMS) consiste em um Problema do Caixeiro Viajante com Múltiplas Pilhas (DTSPMS) com múltiplos veículos. Os dois problemas apareceram pela necessidade urgente em otimizar o transporte intermodal em um contexto europeu. Consiste em coletar pedidos em uma região de coleta e carregá-los num conjunto de pilhas dentro de um container, que não deve ser rearranjado por razões de segurança. O container então é levado a região de entrega e os itens são entregues seguindo a política das pilhas, em que os itens do topo, coletados por último, devem ser entregues primeiro. Nesta dissertação, quatro heurísticas baseadas na busca local iterativa (ILS), na descida em vizinhança variável (VND) e no recozimento simulado (SA) são propostas. O problema foi estendido para uma versão onde há uma oferta de itens maior do que a capacidade da frota de veículos. Um método exato é proposto juntamente com outras três heurísticas baseadas no ILS, SA e na busca tabu (TS). Os algoritmos propostos foram testados em experimentos computacionais e análises estatísticas foram feitas com intenção de encontrar a melhor combinação de parâmetros para estas heurísticas. Os resultados encontrados foram bons, tendo encontrado melhores médias que a literatura atual.

Chapter 1

Introduction

In a fast-paced world that has become increasingly in need of consumable and non-consumable goods, logistics in the transportation of these products has been put to the test, being one of the most important stages in the relationship between the production process and the end user. About one-third of the logistics costs of this relationship (production-user) are determined by the cost of the transportation of these goods, thus this operation is considered of high importance, as shown by Yung-yu Tseng and Taylor [2010]

The large share that transportation has in the distribution costs makes its use increasingly exploited by producers of different sizes and sectors. The expansion of the logistics sector has stood out every year. The UK Department of Transport considers that the logistics business market is heavily contested and is operated, in the majority of cases, by small businesses operating at very low profit margins (1-3%), therefore any research that contributes for an increase in the profits of this sector will have as consequence the *"reduction of the costs of production and transportations of goods and therefore reduced prices for the consumer"* (UK Department for Transport [2011]).

According to Frazzon [2009], the logistics industry, when considered all over the world, handled approximately 5.5 trillion Euros or about 13.8% of the Global Gross Domestic Product, which shows that transportation logistics are not only an important part of the distribution of goods in local trades, but also an extremely important field in our increasingly globalized world economy.

Europe, as a continent of consolidated economy and densely populated, plays an important part in these logistic studies. The 2011 annual report from the European road freight transport industry shows that half of the goods is carried by land on highways of less than 50 km, and three-quarters, or 75%, of all goods are carried on highways of less than 150 km (European Commission for Mobility and Transport [2010]). Part of this traffic is carried out via intermodal transport, the one in which a container is loaded and the products travel by different road modes (railroad, sea and truck-container). The main objective of this transport is the reduction of the gases that cause the greenhouse effect and has been subject of intense research.

After finding a situation where the container should have not be changed when moving between two regions, a software development team called for research on a novel problem. Published in Petersen and Madsen [2009], the DTSPMS was then proposed, with an exact model and four heuristics based on the Iterated Local Search (ILS), Simulated Annealing (SA), Tabu Search (TS) and a Large Neighboorhood Search (LNS). The LNS heuristic performed the best, averaging 3% away from optimality for instances with 66 orders. The DTSPMS was extended in a more complex form and this is the problem that will be treated in this dissertation.

1.1 The problem and its importance

The Double Vehicle Routing Problem with Multiple Stacks (DVRPMS) arose in its simplest form, with the Double Traveling Saleman Problem with Multiple Stacks (DTSPMS), proposed by Petersen and Madsen [2009] when a software company that set up routes in its intermodal traffic encountered this problem with one of its customers.

The problem basically consists in the collection and delivery of products that are carried out between different regions, without the loading being interfered with. This is due to the fact that the regions are separated (by different road routes) and the necessity of not changing the content of the containers, which is a characteristic of the intermodal traffic. The interior of these containers are filled with standard 3 x 11 euro pallets that fill the area of a 40-foot container.

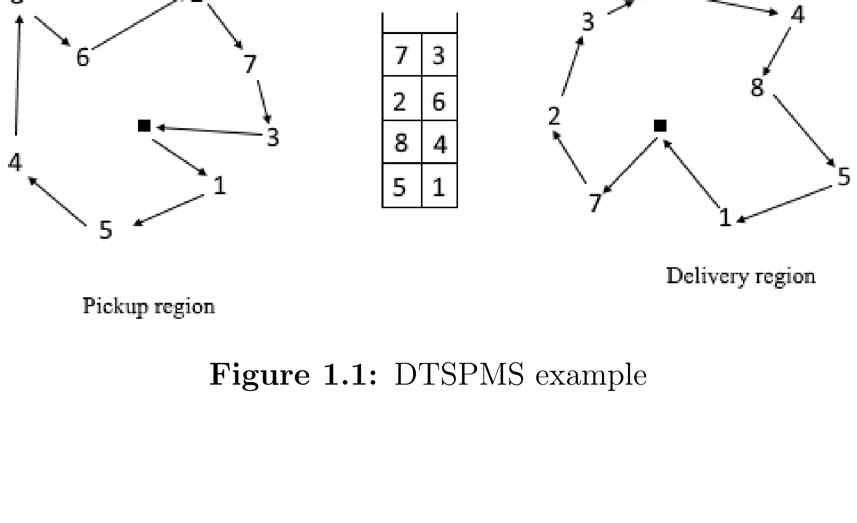


Figure 1.1: DTSPMS example

Figure 1.1 shows an example of a DTSPMS solution. There are two regions, the pickup and delivery regions. In the depot, which is indicated as 0, a 2 x 8 capacity container is located. The vehicle has its loading stack shown in the center of the figure. The truck should then leave the warehouse in the pickup region and go through all the cities, numbered from 1 to 16, collecting the products, respective to each city. Each city in the pickup region has a corresponding city in the delivery region. The vehicle must deliver the loading by following an optimal route within the two regions, without changing the order of the stacks, where the LIFO (Last-in, First-out) policy must be respected. In this policy, the last item placed on the stack must be the first one to be withdrawn.

Optimizing the pickup and delivery routes were necessary because they would reduce the transport costs. This follows the increansing tendency for the intermodality of the transport of goods, especially in Europe. According to Petersen and Madsen [2009], the problem may seem too theoretical, but it has been found in this particular real-life situation, which makes it important. The degree of difficulty of the problem is mainly due to the conditions of precedence given by the stack loading constraints. Following the tradition in the Operational Research researches, the problem was extended to the DVRPMS by Iori and Riera-Ledesma [2015]. The extension basically consists of determining the routes having now a fleet of vehicles available, unlike a single vehicle of the DTSPMS.

We decided to work on the DVRPMS because of its importance for the industry and for the intermodality trend. The use of metaheuristics in solving the problem allows the determination of optimal or near-optimal solutions in less time than the exact algorithm. This is of the utmost importance since the urgency is recurrent in these specific transport cases.

1.2 Objectives

The general objective of the work is to propose heuristics for the DVRPMS with the purpose of generating optimal or near optimal routes that minimize the distance traveled by the vehicles, that does not violate the LIFO policy of the stacks of collected items and that does not exceed an appropriate time of analysis.

1.2.1 Specific Objectives

Specifically, we aim to:

- Perform a bibliographic review of the current scenario of collection and delivery problems, specifically those with some dependency between the two regions.
- Propose heuristics based on metaheuristics capable of generating approximate solutions to the already known optimal instances in a smaller computational time when compared to those generated by the exact algorithm in the literature.
- Compare and corroborate the solutions determined by the heuristics with the optimal solutions already existent in the literature so that it is possible to determine the best approaches to the problem.

1.3 Chapters

This dissertation is divided into five chapters. The first chapter was an introduction to the chosen problem, its importance and the objectives of possible findings. The second chapter presents a broad view on the current state of the art of pickup and delivery problems, specially the double traveling salesman with multiple stacks and the double vehicle routing with multiple stacks. The third chapter shows the methods proposed to solve the DVRPMS and a proposed DVRPMS with Surplus Demand, such as solution representation, evaluation function and neighboorhood structures. In the fourth chapter, tests results are shown. Good results were obtained for the DVRPMS which led to further investigations on the proposed modified version of this problem. Future works suggestions can be seen in the fifth and last chapter, as well as a conclusion on the findings presented in this dissertation.

Chapter 2

Pickup and Delivery problems

This chapter aims to present the class of problems in which the DVRPMS is inserted. In each section, the formalization and mathematical formulation of DTSPMS, DVRPMS and DVRPMS with a modification in the supply of items is also presented. It was interesting to modify the problem because the proposed algorithms were satisfactory in resolving the instances of the basic version known in the literature.

In recent years, a number of different formulations related to pickup and delivery (PDP) problems have been raised. Pickup and delivery problems raised a lot of interest in the past few years because they are considered to be difficult to solve (belonging to the class of NP-hard problems). When taken in conjunction with other classic packing, routing or time-constrained problems they become even more complex and they attract greater research interest. Some of these problems are not only seen in the theoretical field, but, to a great extent, they are seen in real-life situations. The practicity of these problems can be verified in scheduling of sea ports and other routes of distribution. It can also be seen in production scheduling.

In the context of these problems, some authors denote specific vehicle routing problems where the pickup and delivery operations are carried out at different points among each other as Vehicle Routing Problem with Pickup and Delivery (VRPPD).

Parragh et al. [2008] divided the problems of this class into two subclasses. Those in which the sites (or points) are non-paired, that is, when one pickup point satisfies any other delivery point, and another class with paired locations, when there are dependencies between the locations of these two regions. The DVRPMS is inserted in the last case.

Some problems are predecessors to the ones treated in this dissertation. The Multi-vehicle Pickup and Delivery Problem mentioned above (VRPPD), the general Pickup and Delivery Problem (PDP), the Dial-A-Ride-Problem (DARP) and the Vehicle Routing Problem with Time Windows (VRPTW). The latter, proposed by Cordeau et al. [2001]

presents the formulation of three indices that has this common objective function:

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

where K represents the set of vehicles and A the set of city arcs. c defines the cost of crossing a given arc, and x is the binary variable that defines whether the vehicle k traverses a given arc (i, j).

Most pickup and delivery problems are based on the formulation of three indices, also seen in Petersen and Madsen [2009] and Iori and Riera-Ledesma [2015]. In the next section, the formulations of these problems that are the object of study of this dissertation are presented.

2.1 Double Traveling Salesman Problem with Multiple Stacks

Presented in Petersen and Madsen [2009], this problem was verified in a practical situation of European industry context. Goods produced in a given region, called a pickup region, should be delivered to another region, called the delivery region.

Items should be placed in containers and arranged in multiple queues with stacks of a given height. These containers are attached to a carrier vehicle that collects the items in these cities. Containers are usually moved to another region by sea or rail. The vehicles then depart from the delivery depot to distribute the items to their respective cities in the delivery region. The loading of these vehicle must comply with a LIFO policy of the stacks located in the container, and, throughout the path, the loading can not be modified, in any of the steps.

Although it arose from a necessity, some studies on this subject had already been proposed. Hernández-Pérez and Salazar-González [2004] proposes an exact model to the classic problem of the traveling salesman in which the regions of pickup and delivery are different from each other.

Cordeau et al. [2010] proposes a model based on the Branch & Cut technique for the traveling salesman problem with LIFO loading in pickup and delivery regions. By following Cordeau et al. [2010], the non-movement of the container's cargo became of great interest, because the subject on transportation might be corrosive, fragile or of heavy weight. Not handling these products makes the work simpler and reduce costs during transportation.

2.1.1 Mathematical formulation

The formulation below is given by Petersen and Madsen [2009]. This model was named DTSPMS, and from it, the exact algorithm was generated.

Formally, let G be the graph that contains the set of vertex V^G and the set of edges (or arcs) E^G. $G \in \{P, D\}$ with P related to the pickup region and D to the delivery region. Each region has a depot vertex $v_0^G \in \{P, D\}$ and a set of customer inquiries $v_1^G, \ldots, v_n^G$. Each arc i, j has a correspondent and synmetric cost c_{ij}^G.A set of items $1, \ldots, n$ is defined and the item related to a given vertex ($v_i^P \in V^P$) must be delivered to a given vertex at $v_i^D \in V^D$. Let V_C^G represent the set that contains all customer nodes in graph G as in $V_C^G = V^G \backslash \{v_0^G\}$. The model uses the following decision variables:

- x_{ij}^G : binary variable for each arc (i, j), set to 1 if the arc (i, j) is used in graph $G_{\forall i,j \in V^G}$ and 0 otherwise.
- y_{ij}^G : binary variable set to 1 if vertex v_i^G is visited before vertex v_j^G, $\forall i, j \in V_C^G$
- z_{ir}^G : binary variable set to 1 if item i is placed in row r, $\forall i \in V_C^G$ and $r = 1, \ldots, R$. in which $G \in P, D$.

Objective function:

$$\min \sum_{\substack{i,j \in V^G \\ G \in \{P,D\}}} c_{ij}^G x_{ij}^G \tag{2.1}$$

subject to:

$$\sum_i x_{ij}^G = 1, \qquad \forall j \in V^G \tag{2.2}$$

$$\sum_j x_{ij}^G = 1, \qquad \forall i \in V^G \tag{2.3}$$

$$y_{ij}^G + y_{ji}^G = 1, \qquad \forall i, j, G, i \neq j \in V^G \tag{2.4}$$

$$y_{ik}^G + y_{kj}^G \leq y_{ij}^G + 1 \qquad \forall i, j, k, G, \tag{2.5}$$

$$x_{ij}^G \leq y_{ij}^G \qquad \forall i, j, G, \tag{2.6}$$

$$y_{ij}^P + z_{ir} + z_{jr} \leq 3 - y_{ij}^D \qquad \forall i, j, r = 1, \ldots, R, \tag{2.7}$$

$$\sum_r z_{ir} = 1, \qquad \forall i, \tag{2.8}$$

$$\sum_i z_{ir} = L, \qquad \forall r = 1, \ldots, R, \tag{2.9}$$

$$x, y, z \in \mathbb{B} \tag{2.10}$$

The objective function (2.1) minimizes the sum of weights of the used edges (i, j).

Constraints (2.2) and (2.3) state that the vehicle must enter and leave each node exactly once. Constraint (2.4) ensures a precedence between a pair of nodes. The precedence is determined by the binary variable y, either if i is visited before j or j before i. Constraint (2.5) enforces a transitive property where: if i comes before k and k before j, then i is also visited before j. Constraint (2.6) defines the precedence variable according the way in which the items are gathered. Constraint (2.7) states that if two items are placed in the same row, the first item picked must be the last one delivered. This contraint ensures that the problem follows the LIFO policy. Constraint (2.8) ensure that each item is assigned to only one row, and constraint (2.9) enforces the stack capacity, given by lenght L. Constraint (2.10) enforces that variables x, y and z are binary variables.

In addition to the above cited exact model proposed in Petersen and Madsen [2009], other authors worked in the DTSPMS. New neighboorhoods and a heuristic based on a Hybrid Variable Neighboorhood Search were proposed by Felipe et al. [2009a] and Felipe et al. [2009b]. Better results were found when compared to the Large Neighboorhood Search proposed by Petersen and Madsen [2009].

A few years later, an exact algorithm based on the branch-and-bound algorithm was proposed by Martínez et al. [2013] outperforming the first exact model proposed by Petersen and Madsen [2009]. Efficient algorithms were proposed in Casazza et al. [2012] and a competitive heuristic based on a dynamic programming local search was proposed by Urrutia et al. [2015]. The exact model proposed by Barbato et al. [2016] that solves instances previously unsolved in the literature is the last report on the subject.

2.2 Double Vehicle Routing Problem with Multiple Stacks

In the same way TSP problems were extended to VRP problems, the DTSPMS was extended to the DVRPMS in Iori and Riera-Ledesma [2015]. While the DTSPMS involves a single vehicle, in the DVRPMS a fleet of vehicles is available.

Figure 2.1 shows an example of solution for the DVRPMS. The depots are indicated by the large central dot on both pickup and delivery regions. The orders are numbered from 1 to 16, and for each order there is a specific pickup location and a respectively associated location in the delivery region. For each depot, there is a fleet of four vehicles (containers), each one with two stacks of height two. The stacks are shown in the center of the figure. Their items can not be rearranged. Each vehicle must leave the depot in

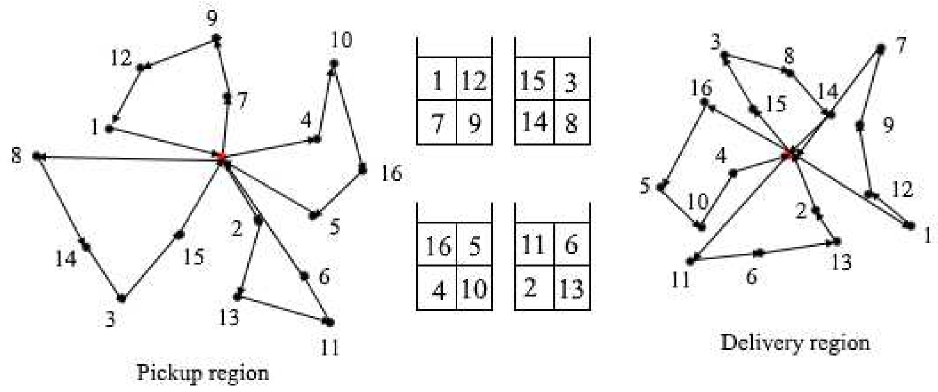


Figure 2.1: DVRPMS example

the pickup region and return to the same depot when its loading is completely filled. The containers are then shipped to the delivery depot. Then the vehicles deliver the items on the respective cities of each order according to its loading. The cost of all routes performed by the fleet on both regions must be optimum, having the minimum possible value.

2.2.1 Mathematical formulation

The problem was formally described in Iori and Riera-Ledesma [2015] and it is shown in this subsection. Let $I = \{1, 2, \ldots, n\}$ be the set of customer inquiries carried by the vehicles in the pickup and delivery regions. Let also $V_0^P = \{1^P, 2^P, \ldots, n^P\}$ be the set of customers related to the pickup regions. Following the region dependences, each request $i \in I$ corresponds to its $i^T \in V_0^T$ vertex, given that T refers to any of the two regions.

It is possible to represent the DVRPMS as a directed graph $G = (V, A)$, where V is the vertex set given by $V = V^P \cup\ V^D$, where $V^P = \{0^P\} \cup V_0^P$ and $V^D = \{0^P\} \cup V_0^D$. The members 0^P and 0^D are the depots for the pickup and delivery regions and members V_0^P and V_0^D are, respectively, the sets of vertices excluding the depots for the pickup and delivery regions. Likewise, the arc set is given by $A = A^P \cup\ A^D$, where $A^P = \{(i^P, j^P) \in V^P \times V^P | i^P \neq j^P\}$ and $A^D = \{(i^D, j^D) \in V^D \times V^D | i^D \neq j^D\}$. For each arc $(i, j) \in A^P$ and $(i, j) \in A^D$ there is an associated routing cost of c_{ij}^P and c_{ij}^D, respectively.

To avoid confusion in the index representation, $T \in \{P, D\}$ was used to refer to properties applied to both pickup and delivery regions. The respective indexes P and D were used to highlight a property that is exclusive for the pickup and delivery region, respectively.

The set of vehicles that are strictly allocated in the pickup depot is defined by K. Each vehicle $k \in K$ will receive the goods in r_k stacks of height l_k. For a proper designation of the vertices and arcs traveled by the $k \in K$ vehicles, $Q_k = \{p_1, p_2, \ldots, p_q\} \subseteq V^P$ was defined to represent the vertex set covered by the vehicle $k \in K$ in the pickup region and $A(Q_k) = \{(p_1, p_2), (p_2, p_3), \ldots, (p_{q-1}, p_q)\}$ was defined to represent the arc set, where each vehicle $k \in K$ travels in the pickup region. Likewise, for the delivery region, $F_k = \{d_1, d_2, \ldots, d_f\} \subseteq V^D$ represents the set of visited vertices and $A(F_k) = \{(d_1, d_2), (d_2, d_3), \ldots, (d_{f-1}, d_f)\}$ the set of covered arcs.

Due to the premisse that each vehicle k is able to carry an amount of $r_k l_k$ products and to the obligation of precedence on an arc (i, j) in view of the LIFO policy, the pair (Q_k, F_k), is called load-infeasible if there is no load that meets the above mentioned specifications. $\mathcal{R}_k$ is denoted as an unfeasible set of pairs for each vehicle $k \in K$. The definition of these nonviable paths were important to handle the mathematical constraints added to the relaxed model. This constraints are not presented in this dissertation and can be seen in Iori and Riera-Ledesma [2015].

Furthermore, it was defined as C_k^P a cycle of one vehicle $k \in K$ that leaves the pickup depot and travels through the cities of this region, and $I(C_k^P)$ as the set of inquiries collected in the pickup process. Likewise, C_k^D and $I(C_k^D)$ are, respectively, the cycles in the delivery region and the set of products distributed in the delivery region.

With all vehicles $k \in K$ there is a set of $|K|$ routes C_k that, to be considered as a feasible solution, must satisfy four conditions. The first one is the paring constraint. Each product from the pickup region must be delivered to its respective pair in the delivery region, i.e., $I(C_k^P) = I(C_k^D)$. Secondly, at most $r_k l_k$ customers must be served, i.e., $|I(C_k^T)| \leq r_k l_k$. Thirdly, each route must support the loading of its respective container or $C_k \notin \mathcal{R}_k$. Finally, each customer must be served by only one route, i.e., $\bigcup_{k \in K} I(C_k^T) = I$ and $I(C_j^T) \cup I(C_k^T) = \emptyset$ for $j, k \in K, j \neq k$ and $T \in \{P, D\}$.

The model below is given by Iori and Riera-Ledesma [2015]. This relaxed model was named DVRPMS0, and from it, all of the exact algorithms were generated. Let:

- $x_{i,j}^{Tk}$: binary variable for each arc $(i, j) \in A^T$, $T \in \{P, D\}$ set to 1 if the vehicle k crosses the arc (i, j) and 0 otherwise.
- y_i^{Tk} : binary variable set to 1 if the vehicle k visits the customer in i of region T and 0 otherwise.

Objective function:

$$\min \sum_{T \in \{P,D\}} \sum_{(i,j) \in A^T} c_{ij}^T \sum_{k \in K} x_{ij}^{Tk} \tag{2.11}$$

subject to:

$$\sum_{j\in V^T} x_{ij}^{Tk} - y_i^{Tk} = 0, \qquad i \in I, T \in \{P,D\}, k \in K \tag{2.12}$$

$$\sum_{j\in V^T} x_{ij}^{Tk} - \sum_{j\in V^T} x_{ji}^{Tk} = 0, \qquad i \in I, T \in \{P,D\}, k \in K \tag{2.13}$$

$$\sum_{j\in V_0^T} x_{0j}^{Tk} = 1, \qquad T \in \{P,D\}, k \in K \tag{2.14}$$

$$\sum_{i\in S}\sum_{j\in V^T\setminus S} x_{ij}^{Tk} \geq y_\delta^{Tk}, \qquad \delta \in S, T \in \{P,D\}, S \subseteq V_0^T, k \in K \tag{2.15}$$

$$\sum_{i\in I} y_i^{Tk} \leq l_k r_k, \qquad T \in \{P,D\}, k \in K \tag{2.16}$$

$$y_i^{Pk} - y_i^{Dk} = 0, \qquad i \in I, k \in K \tag{2.17}$$

$$\sum_{k\in K} y_i^{Tk} = 1, \qquad i \in I, T \in \{P,D\} \tag{2.18}$$

$$x_{ij}^{Tk} \in \{0,1\} \qquad (i,j)^T \in A^T, T \in \{P,D\}, k \in K \tag{2.19}$$

$$y_i^{Tk} \in \{0,1\} \qquad i \in I, T \in \{P,D\}, k \in K \tag{2.20}$$

The objective function (2.11) minimizes the cost of all arcs (i,j) traversed by the vehicles $k \in K$ in both regions P and D. The constraint (2.12) and the bound (2.20) defines the variable y and forces only one arc to enter a customer city i. The constraint (2.13) establishes the conservation flow of each vertex. The constraint (2.14) forces each vehicle to leave and arrive a depot only once. The connectivity constraint in equation (2.15) ensures that each customer is connected to the depot by a path. The constraint in equation (2.16) guarantees that the vehicle is filled according to its capacity. A pairing constraint in (2.17) forces each container to deliver the goods gathered in the pickup region to its respective customer in the delivery region. The constraint (2.18) ensures that each customer is served only by one vehicle. Equations (2.19) and (2.20) assures that the variables $x_{i,j}^{Tk}$ and y_i^{Tk} are binary. Albeit having one constraint that controls how many items will be assign on each route (Equation 2.16), the relaxed model does not take account of the infeasible paths caused as a result of negleting the LIFO policy.

2.2.2 Modification in the orders offering

With the works produced by Petersen and Madsen [2009], Felipe et al. [2009a] and Martínez et al. [2013], the DTSPMS generated satisfactory results (up to 4% distant from the global optimum solution) for instances with 33 requests. Iori then generated new instances of the problem, now with a fleet of vehicles available. The exact mathematical model of DVRPMS solves small instances easily, but for large instances with 24 requests, the model does not resolve within one hour of execution. The heuristics proposed in this dissertation generated satisfactory results.

Therefore, one way forward is to extend the DVRPMS considering now that there is a spare supply. For example, let us consider that the set of items that compose the regions has 24 units (1 warehouse and 24 cities). A surplus demand is then added. There will be 29 items (rounded to the nearest integer) to be collected and delivered. The fleet is not modified, thus still containing 24 positions to be allocated, which requires that items be collected and delivered based on a priority index defined as *score*.

A real-life situation of this problem can be seen in a context where the number of orders are greater than the capacity of the fleet. The logistics manager must choose the items based on their priority, which can be designed based on urgency or other factors.

2.2.3 Mathematical model based on a *score* value

The score-based model is a mixture between the DTSPMS and the DVRPMS and can be formalized using the DVRPMS variables. With the given routing costs of the edges (i, j), called c_{ij} and the scores associated to the customers i, named s_i, this formalization was proposed:

- $x_{i,j}^{kT}$: binary variable set to 1 if the vehicle k crosses the arc (i, j) and 0 otherwise. T indicates that the properties of the variable that takes this index are valid for both regions P and D.

- y_{ij}^{T} : binary variable set to 1 if the vertex i is visited before vertex j in region T and 0 otherwise.

- z_{ir}^{k} : binary variable set to 1 if the item i is placed in row r of the vehicle k and 0 otherwise.

- w_{i}^{kT} : binary variable set to 1 if the vehicle k visits the customer i of region T and 0 otherwise.

$$\max \sum_{k\in K}\sum_{i\in V_c} s_i w_i^{kP} - \sum_{k\in K}\sum_{T\in\{P,D\}}\sum_{i\in V}\sum_{j\in V\setminus\{i\}} c_{ij}^T x_{ij}^{kT} \tag{2.21}$$

$$\sum_{j\in V_c} x_{0j}^{kT} = 1 \qquad \forall k\in K, \forall\, T\in\{P,D\} \tag{2.22}$$

$$\sum_{i\in V_c} x_{i0}^{kT} = 1 \qquad \forall k\in K, \forall\, T\in\{P,D\} \tag{2.23}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij}^{kT} = w_j^{kT} \qquad \forall k\in K, \forall\, T\in\{P,D\}, \forall j\in V_c \tag{2.24}$$

$$\sum_{j\in V\setminus\{i\}} x_{ij}^{kT} = \sum_{j\in V\setminus\{i\}} x_{ji}^{kT} \qquad \forall k\in K, \forall\, T\in\{P,D\}, \forall i\in V_c \tag{2.25}$$

$$\sum_{k\in K} w_i^{kT} \le 1 \qquad \forall\, T\in\{P,D\}, \forall i\in V_c \tag{2.26}$$

$$\sum_{r=1}^{R_k} z_{ir}^k = w_i^{kP} \qquad \forall k\in K, \forall i\in V_c \tag{2.27}$$

$$\sum_{i\in V_c} z_{ir}^k = L_k \qquad \forall k\in K, \forall r=1..R_k \tag{2.28}$$

$$w_i^{kP} = w_i^{kD} \qquad \forall k\in K, \forall i\in V_c \tag{2.29}$$

$$y_{ij}^T + y_{ji}^T = 1 \qquad \forall\, T\in\{P,D\}, \forall i\in V_c, \forall j\in V_c\setminus\{i\} \tag{2.30}$$

$$y_{il}^T + y_{lj}^T \le y_{ij}^T + 1 \qquad \forall\, T\in\{P,D\}, \forall l\in V_c, \forall i\in V_c, \forall j\in V_c\setminus\{i\} \tag{2.31}$$

$$x_{ij}^{kT} \le y_{ij}^T \qquad \forall k\in K, \forall\, T\in\{P,D\}, \forall i\in V, \forall j\in V\setminus\{i\} \tag{2.32}$$

$$y_{ij}^P + z_{ir}^k + z_{jr}^k \le 3 - y_{ij}^D \qquad \forall k\in K, \forall i\in V_c, \forall j\in V_c\setminus\{i\}, \forall r=1..R_k \tag{2.33}$$

$$x_{ij}^{kT} \in \{0,1\} \qquad \forall k\in K, \forall\, T\in\{P,D\}, \forall i\in V, \forall j\in V\setminus\{i\} \tag{2.34}$$

$$y_{ij}^T \in \{0,1\} \qquad \forall\, T\in\{P,D\}, \forall i\in V, \forall j\in V\setminus\{i\} \tag{2.35}$$

$$z_{ir}^k \in \{0,1\} \qquad \forall k\in K, \forall i\in V_c, \forall r=1..R_k \tag{2.36}$$

$$w_i^{kT} \in \{0,1\} \qquad \forall k\in K, \forall\, T\in\{P,D\}, \forall i\in V_c \tag{2.37}$$

The objective function (2.21) maximizes the difference between the scores of served cities and the cost to travel through the selected cities. Constraint (2.22) and (2.23) ensures that one vehicle must leave or arrive on the depot. Constraint (2.24) defines variable w_j^{kT} if a vehicle arrives at city j. Flow constraint (2.25) ensures that a vehicle arrives and leaves a city only once (this is also covered by (2.26)). Constraint (2.27) enforces that a item is allocated in only one row of a vehicle, and constraint (2.28) ensures that the rows do not exceed the vehicle capacity. Constraint (2.29) is a pairing constraint and ensures that a given pickup item have a correspondent delivery item. Constraint

(2.30) estabilishes a precedence relation between arc i, j. Constraint (2.31) ensures that if i is visited before l and l before j, then i is visited before j, set by the precedence constraint (2.32). Finally, constraint (2.33) ensures that if i is placed in the same row of item j, and i is picked before j, than i must be delivered after j. Bounds (2.34), (2.35), (2.36) and (2.37) define the binary variables.

This a new model proposed by Chagas and it combines the DTSPMS and DVRPMS models and introduces a new variable to check if the items of the same row are visited by a vehicle k, thus constraints, 2.26 and 2.29 does not appear in either DTSPMS or DVRPMS models. This model was not published yet and it was implemented through personal communications.

Chapter 3

Methods

Before detailing the heuristics in this chapter, the initial solution generation and the neighborhood structures are introduced and the algorithms applied to the known instances are showed. With an exception to the algorithms that use a special local search procedure (LS), all others share the same randomly initial solution generation.

3.1 Solution representation

A solution is represented by the distribution of $|I|$ items over the $|K|$ vehicle fleet. Each vehicle of this fleet has one container with a loading plan in accordance with its inherent configuration. For the DVRPMS, the routes are obtained selecting the nearest neighboor when a vehicle k leaves the depot. Regarding data structures, a struct named *vehicle* was defined, and all records were defined as follows: the size of its stacks (r and l), the routes (pickup and delivery), and the stack itself were declared as bidimensional vectors.

Figure 3.1 shows an example of a solution representation for three containers of size 2×4, 1×4 and 2×2. For the DVRPMS with Surplus Demand, the pickup and delivery routes for each vehicle are obtained by the evaluation function described on section 3.2. Essentially, data structures are the same for the DVRPMS with Surplus Demand, except for a new “ghost” vehicle, created to receive the unused items of a current solution and the fact that routes are not stored on the current solutions.

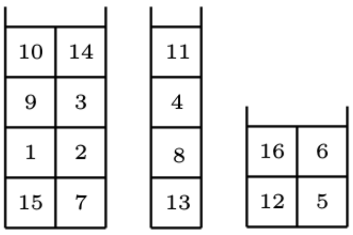


Figure 3.1: Example of a DVRPMS with Surplus Demand solution representation

3.2 Evaluation function for the DVRPMS with Surplus Demand

The evaluation function role is to generate pickup and delivery routes given a stack configuration. The function uses a greedy approach or an optimum approach depending on the stack configuration. These approaches are described below:

- *Greedy*: On each run, among the items at the top (or bottom) of each stack, the item that has the least impact on the delivery (or pickup) route costs is chosen. Generating the pickup and delivery routes by this strategy is $O(r^2l)$, because one needs to decide $r \times l$ times which of the r locations of the items on the top of the stack has the shortest distance to the next point.

- *Optimum:* Among all possible routes, the routes of pickup and delivery that have the smallest distance are chosen, i.e., the optimal pickup and delivery routes is selected, given a loading plan. Casazza et al. [2012] demonstrate this for the DTSPMS using the approach once for each vehicle. The complexity of this algorithm is $O((rl)^r)$. The exponential complexity of the algorithm makes it computationally infeasible to be performed in all solution evaluations. Therefore, obtaining the optimal routes is used only in specific cases.

3.3 Initial solution generation procedure

The problem considers that, for each instance, the number of items is equal to the capacity of the fleet, hence no truck is overloaded or underloaded. For the construction of a first feasible solution, a random array of size n is generated with the items given by I. The stacks are then filled from bottom up by removing the items of this random array, which generates random stacks for each vehicle $k \in K$.

For each vehicle $k \in K$ the pickup route is constructed based on the minimum distance between the items on top of the stack and the vertices that can be potentially visited in the region (according the precedence in the current stack), starting from the pickup depot. The delivery routes were generated by a nearest neighbor approach that not violated the LIFO policy.

In the DVRPMS with Surplus Demand an initial solution is created as follows. For each vehicle, $r \times l$ items of I are randomly selected. The container's stacks are loaded from bottom-up, similarly as it works on the DVRPMS. It should be noted that there is a "ghost" vehicle that receives the items that are unused in a given solution.

3.4 DVRPMS neighboorhood structures

The four neighborhood structures used in the DVRPMS algorithms are shown in this section. These structures were used in the local search procedures of the proposed algorithms in Section 3.6. The first two presented, *route-swap* and *complete-swap*, were based in the structures of Petersen and Madsen [2009] for the DTSPMS. The *instack-swap* structure was based on a structure defined in Felipe et al. [2009a]. The structure *route-exchange* was created for this problem and is based on a basic VRP neighborhood structure from Toth and Vigo [2001].

3.4.1 route-swap

The structure *route-swap* moves through the search space changing all pairs of consecutive items A and B, similarly to the classic 2-opt structure seen in Croes [1958]. The swap reorder the routes considering that two items cross each other. On account of this change in a pickup route, it is often necessary a swap in the delivery region to preserve the stack feasibility.

For example, if the shifted items A and B are located in the same stack, the position that these items A and B occupy in the delivery route must be changed as well, to preserve feasibility. On the other hand, if the shifted items A and B are located in different stacks, it is not necessary to change the position of the items in the delivery route, because it is still possible to travel through A and B maintaining the precedence.

It is noteworthy that the *route-swap* operations are performed in one vehicle at a time. Since all the items of a pickup route are examined as well as the viability of the same items in the delivery route, the size of this structure is $O(|I|)$ where $|I|$ is the number of items in the problem. An example can be seen in Figure 3.2.

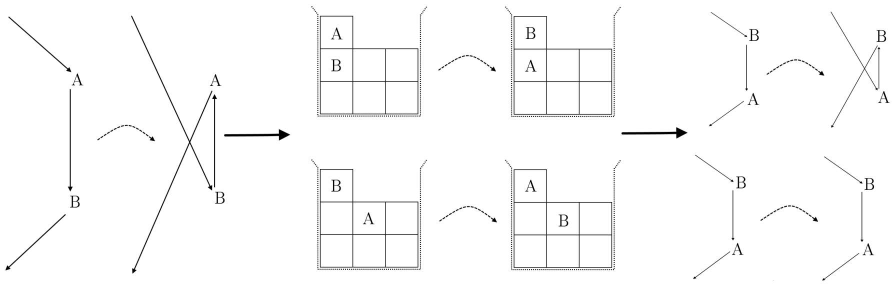


Figure 3.2: Route-swap example. The algorithm starts changing all consecutive items in the pickup route. Depending on the stack configuration, a change will be necessary on the delivery route.

3.4.2 complete-swap

The structure *complete-swap* moves through the search space changing all pairs of items A and B that are located in different stacks. Here, when a swap is performed between two items A and B in the pickup route, the corresponding items in the delivery route are swapped, to maintain feasibility.

Since all *complete-swap* operations are performed in one vehicle at a time and all pairs of items in different stacks are considered, the size of this structure is $O(\sum_k^K \frac{|I|_k(|I|_k-l)}{2}) \approx O(|I|_k^2)$, since for each set of items I on vehicle k, one has to check all pairs of items that are on a different stack, hence $|I|_k - l$. An example can be seen in Figure 3.3:

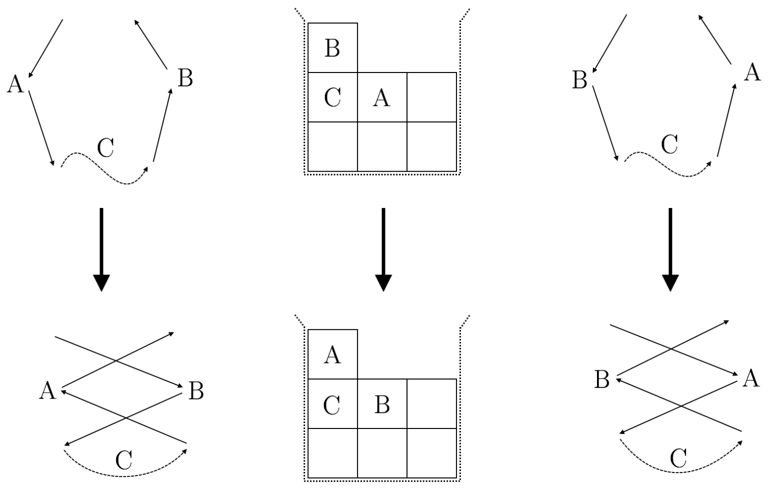


Figure 3.3: Complete-swap example. After changing a pair A,B on the stack, the algorithm fixes feasibility by swaping the items in the pickup route (left) and in delivery route(right)

3.4.3 instack-swap

The structure *instack-swap* moves through the search space changing all pairs A and B that are allocated in the same stack. On account of this change, in the pickup and delivery routes, it is necessary to swap both A and B to maintain the stack feasibility.

It is noteworthy that the *instack-swap* operations are performed in one vehicle at a time. One has to check for all items on the same stack, hence a number of items r are excluded from the search, that is, one that represents the items that are currently checking. Since all the items on the same stack are examined ($O(|I| - r)$) as well as the viability of the same items in the routes ($O(1)$), the size of this structure is $O(|I|)$. An example can be seen in Figure 3.4:

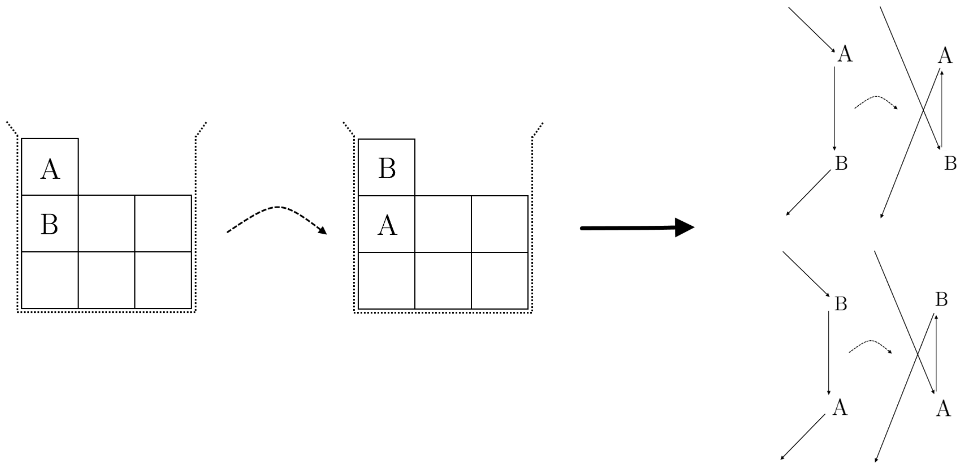


Figure 3.4: Instack-swap example. After the swap on the pair A, B, the algorithm fixes feasiblity by swaping the items on the pickup route (top) and in the delivery route (bottom)

3.4.4 r-stack-permutation

The structure *r-stack-permutation* moves through the search space randomly selecting a stack and the first r items from top-down allocated at this stack are selected to be permuted. r, not to be confused with the number of rows of a given stack, is a variable set to the permutation size in a given column of the stack. Since the maximum size of a column (l) according to the instances in the literature is 4 (see Table 4.1), the values accepted for r are 1, 2, 3 and 4. On account of this change, in the pickup and delivery routes, it is necessary to move the items according to its precedence sequence to maintain the stack feasibility.

It is noteworthy that the *r-stack-permutation* operations are performed in one vehicle at a time. Since all permutations of the selected r items are examined, the size of this structure is $O(r!)$. An example can be seen in Figure 3.5:

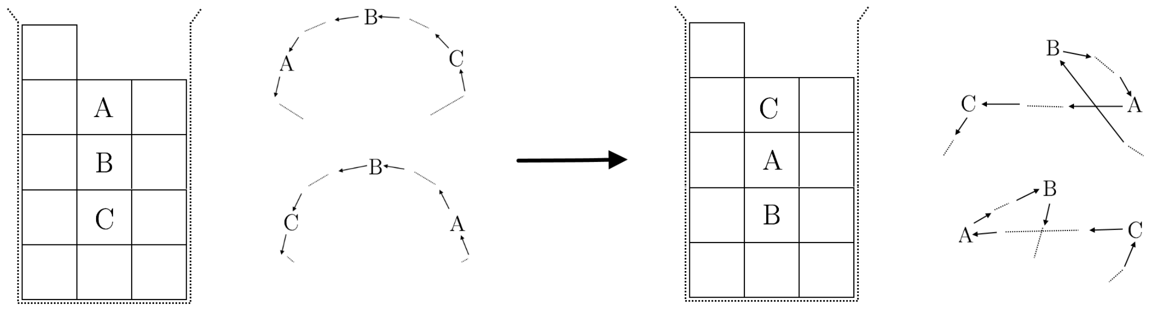


Figure 3.5: r-Stack-Permutation example. The algorithm randomly chooses a stack (the middle stack) and selects all r items from top-down ($r = 3$). All permutations are evaluated. One of the permutations is seen in the figure

3.4.5 route-exchange

The structure *route-exchange* covers the search space doing a swap in a random k set of items (one for each vehicle $k \in K$). For example, consider that there are three vehicles, each one with a pickup route. An item from the pickup route of vehicle 1 is randomly chosen and exchanged with an item from the pickup route of vehicle 2, an item from the pickup route of vehicle 2 is randomly chosen and inserted with an item from the pickup route of vehicle 3, until a cycle is completed. The same items are swapped in the stack and in the delivery route to preserve the solution's feasibility. The effects of this swap for the DVRPMS with Surplus Demand can be seen in Figure 3.6. Note that the DVRPMS with Surplus Demand uses a "ghost" vehicle that receives the orders that are not served by the current solution. Consequently, *route-exchange* is the only neighboorhood that verifies unused orders.

3.5 DVRPMS with Surplus Demand neighboorhood structures

The two neighborhood structures used specifically in the DVRPMS with Surplus Demand algorithms are shown in this section. These structures were used in the local search procedures of the ILS and SA.

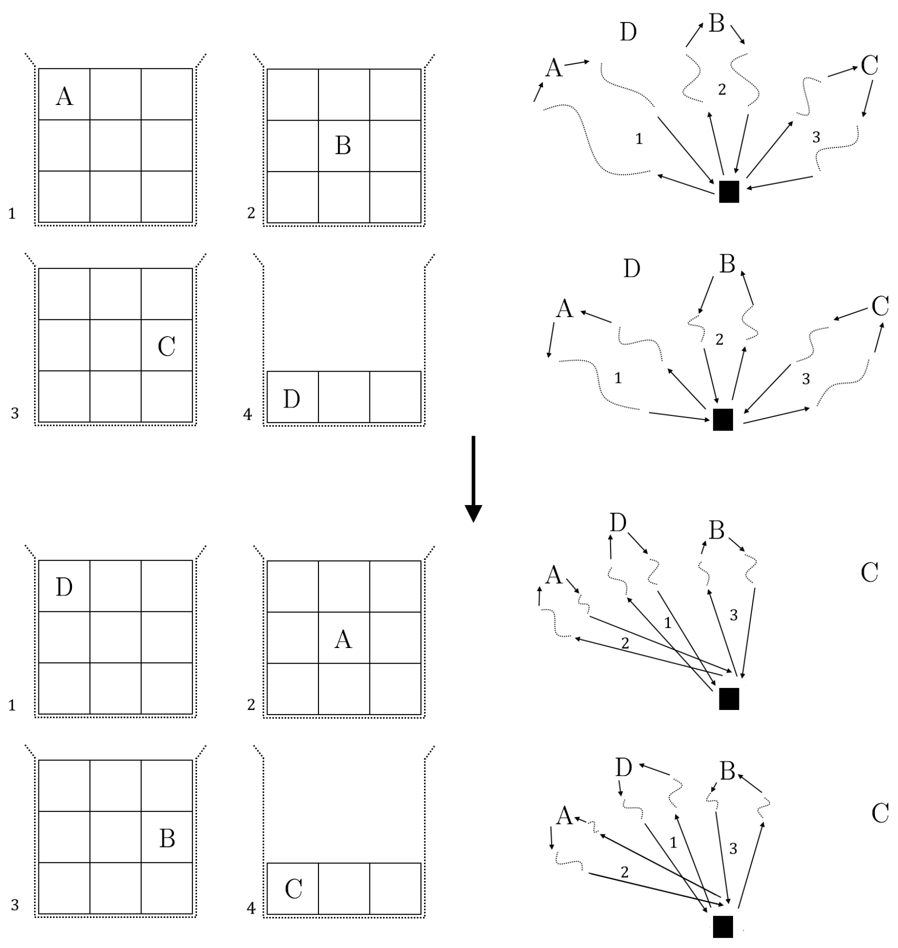


Figure 3.6: Route-exchange example. The algorithms chooses randomly one item of each vehicle and swaps it with the next vehicle following a cycle. At the top right there are the pickup and delivery routes. The routes after the changes are shown at the bottom right

3.5.1 swap

The neighbourhood structure set to explore the solution space for the DVRPMS with Surplus Demand is a simple switch of two items positions (swap). The products to be relocated may belong to the same container or to different containers, so the neighbourhood size is $O(|I|^2)$. Figure 3.7 shows a solution s using three heterogeneous vehicles (2×4, 1×4 and 2×2), and one of its neighbours s'. The set of all neighbors of a given solution s is represented by $N(s)$.

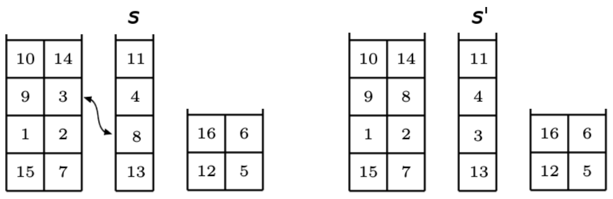


Figure 3.7: Simple swap example. A solution s and a neighbor s'. A simple swap is performed on a pair of items. The resulting solution is then passed to the the evaluation function and the routes are built

3.5.2 ILS perturbation

This perturbation was set to explore a wider portion of the solution space. A set of n items is randomly selected and the solution is rebuilt swaping all set items with a shuffle. In this algorithm, n becomes bigger when the number of iterations with no improvement grows. Figure 3.8 shows an example of this perturbation. In this case, $n = 5$, so 5 items are randomly chosen and the positions are changed after a shuffle

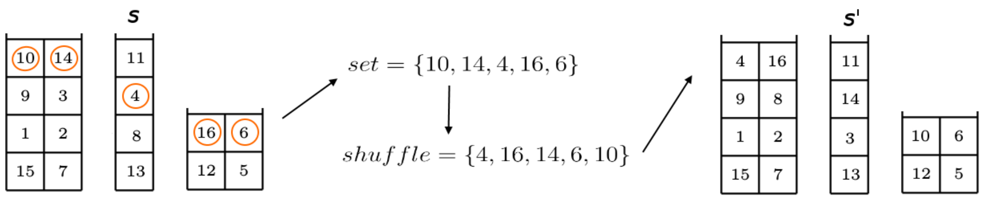

s
10
14
9
3
1
2
15
7
11
4
8
13
16
6
12
5
$set = \{10, 14, 4, 16, 6\}$
$shuffle = \{4, 16, 14, 6, 10\}$
s'
4
16
9
8
1
2
15
7
11
14
3
13
10
6
12
5

Figure 3.8: Perturbation example. A solution s and a solution s' after a perturbation. A randomly selected set of 5 items is shuffled and reinserted in the stacks.

3.6 Metaheuristics applied to the DVRPMS

Following Talbi [2009], metaheuristics are used to deal with large-scale and often difficult problems (commonly NP-complete) generating solutions in fairly small times without guaranteeing that the solution is optimal or close to it. In the last 30 years a great range of metaheuristics has been proposed.

For some large instances of this problem, none of the exact methods could solve the problem within one hour. To work around this problem, heuristics approaches that could rapidly give a solution close to the optimum were proposed for both the DVRPMS and DVRPMS with Surplus Demand. These heuristics were based on the Iterated Local Search (ILS), Simulated Annealing (SA) and Variable Neighborhood Descent (VND) meta-heuristics .

The ILS metaheuristic has been widely used in vehicle routing problems with favorable results, especially in highly restrictive problems. The VND metaheuristic is commonly used for vehicle routing problems to show how easily the solution converges to a local optimum when the local search is performed. The SA metaheuristic is often used when one wants to avoid choosing bad perturbations, and proved to be a great heuristic for this problem.

3.6.1 ILS approach

In their work, Lourenço et al. [2010] summarizes the iterative local search. The essence of the method lies in the iterativity that constructs the sequence of solutions generated by a initial heuristic inside the method. The organization of this iterativity provides better solutions in comparison if the changes were random.

The basic version of ILS was used for the normal DVRPMS. The algorithm starts with a local search procedure in the initial solution. When the current solution can not be improved by the neighborhood operators, the algorithm restarts with a perturbation. This is done to avoid being locked in a local optimum. The local search in the ILS algorithm was carried out by the structures *route-swap* and *complete-swap*. The current solution was chosen in a best improvement basis. When none of these operators provide a better solution, a perturbation was applied by the *route-exchange* structure. The perturbation seen in the previous section is only applied in the modified version of the DVRPMS. The ILS algorithm has a fixed running time of 10 seconds, which was set according tests seen in the Appendix. The algorithm is described by the Algorithm 1.

Algorithm 1 Iterated Local Search

```
1: Initialize with $init\_sol$.
2: $best\_sol \leftarrow init\_sol$
3: $current\_sol \leftarrow init\_sol$
4: while $time < max\_time$ do
5:    $route\_swap(current\_sol)$
6:    $complete\_swap(current\_sol)$
7:    $route\_exhange(current\_sol)$
8:    if $current\_sol < best\_sol$ then
9:       $best\_sol \leftarrow current\_sol$
10:   end if
11: end while
12: return $best\_sol$
```

3.6.2 VND approach

The VND is the simplest form of the VNS based heuristics, first shown in Mladenović et al. [2000]. The name comes from the fact that various neighboorhood structures perform the local search inside the algorithm. For the VND, for example, the changes are made in a deterministic way, and the final solution is the local optimum of all neighboorhoods structures. These neighborhood structures are normally arranged hierarchically according to its perturbation strength. In each iteration, if a better solution is found after a local search structure operator, the algorithm returns to the first perturbation until no improvement can be found. VND heuristics quickly converge to a local optimum and might not provide good solutions in VRP problems, but was still used in the problem to verify the behaviour of this algorithm in particular to the problem. The VND implementation is described in Algorithm 2.

3.6.3 SA approach

Inspired by a technique from metallurgy, this probabilistic metaheuristic tries to locate a good approximation to the optimum solution of the problem. For the DVRPMS, the SA algorithm begins with an initial solution that is perturbed with either *route-swap* or *route-exchange* neighborhoods. The probability of choosing a particular neighborhood is a parameter of the algorithm. Instead of picking the best neighbor, a random one is chosen. This neighbor can then be accepted depending on its value and current temperature according to the probability function seen in line 10 of Algorithm 3. The pseudocode in Algorithm 3 describes the above cited process.

Note that the better the value and the higher the temperature, the higher is the chance to chose the current solution.

3.7 Metaheuristics applied to the DVRPMS with Surplus Demand

For the DVRPMS with Surplus Demand, the ILS proposed for Algorithm 1 was used to compare how the solutions were related between these two different problems. Since there is a higher number of items to be picked and delivered in the modified version of the problem, a ghost vehicle set to the items that are not currently used in the solution was implemented. This approach was used in all algorithms of this section.

Algorithm 2 Variable Neighbourhood Descent

```
1: Initialize with $current\_sol \leftarrow init\_sol$.
2: $best\_sol \leftarrow init\_sol, neigh \leftarrow 1, n\_max \leftarrow 3$
3: while $time < max\_time$ and $neigh < n\_max$ do
4:     if $neigh = 1$ then
5:         $route\_swap(current\_sol)$
6:         if $F(current\_sol) < best\_sol$ then
7:             $best\_sol \leftarrow current\_sol$
8:             $neigh \leftarrow 1$
9:         else
10:            $neigh + +$
11:        end if
12:    end if
13:    if $neigh = 2$ then
14:        $complete\_swap(current\_sol)$
15:        if $F(current\_sol) < best\_sol$ then
16:            $best\_sol \leftarrow current\_sol$
17:            $neigh \leftarrow 1$
18:        else
19:            $neigh + +$
20:        end if
21:    end if
22:    if $neigh = 3$ then
23:        $route\_exhange(current\_sol)$
24:        if $F(current\_sol) < best\_sol$ then
25:            $best\_sol \leftarrow current\_sol$
26:            $neigh \leftarrow 1$
27:        else
28:            $neigh + +$
29:        end if
30:    end if
31: end while
32: return $best\_sol$
```

Algorithm 3 Simulated Annealing

```
1: $iter \leftarrow 0$
2: $best \leftarrow initialSolution$
3: $temperature \leftarrow initialTemp$
4: do
5:   $pert \leftarrow 0$
6:   $nSuc \leftarrow 0$
7:   do
8:     $candidate \leftarrow perturb(best)$
9:     $delta \leftarrow candidate.value() - best.value()$
10:    if$(delta <= 0$ $or$ $\mathbf{exp}(-delta/temperature) > random())$
11:      $best \leftarrow candidate$
12:      $nSuc \leftarrow nSuc + 1$
13:  while$(++pert < maxPert$ $and$ $nSuc < maxSuc)$
14:  $temperature \leftarrow updateTemperature()$
15: while$(++iter < maxIter)$
16: return $best$
```

3.7.1 Local search

A solution found by an algorithm should be submitted to the local search procedure described by Algorithm 4 either during the algorithm (ILS+LS) or at the end of the procedure (SA+LS). In the local search procedure, for each container of a given solution, the optimal pickup and delivery routes are obtained. The purpose of this procedure is to improve the value of the solution, trying to achieve a local minimum that was not found by the algorithms due to the possibility of not obtaining the best routes of each container.

Algorithm 4 Local Search (LS)

```
1: $s \leftarrow$ initial solution
2: repeat
3:   $N(s) \leftarrow$ generate the neighbors of $s$
4:   $s' \leftarrow$ first improvement neighbor $s' \in N(s)$
5:   if $f_{LS}(s') < f_{LS}(s)$ then
6:     $s \leftarrow s'$
7:   end if
8: until $s$ is a local optimum
9: return $s$
```

3.7.2 SA+LS approach

This SA approach differs from the last SA (the one used in the DVRPMS) because it now has a local search procedure that generates better solutions. It is used only for the

DVRPMS with Surplus Demand. The initial solution for this SA is generated randomly and is considered as the best solution so far. While the minimum temperature and the maximum number of iterations are not achieved, the algorithm randomly selects a solution nearby the current solution and accepts it according to a probability function given by the method, selecting it as the best solution so far if it is better than the last best solution. The algorithm is described in Algorithm 5. Then, the best solution proceeds to the Local Search procedure described in Algorithm 4.

As it was previously said in this dissertation, the SA+LS heuristic managed to achieve good results, and it was also applied to the Iori and Ledesma version of the DVRPMS.

Algorithm 5 DVRPMS with Surplus Demand Simulated Annealing (SA+LS)

```
1: $s \leftarrow$ generate a random solution
2: $best\_solution \leftarrow s$
3: $t \leftarrow t_0$
4: while t > $t_{min}$ do
5:     $it \leftarrow 0$
6:     while $it < num\_it$ do
7:         $N(s) \leftarrow$ generate the neighbors of $s$
8:         $s' \leftarrow$ select a random solution $s' \in N(s)$
9:         $\Delta \leftarrow f_{SA}(s') - f_{SA}(s)$
10:        if $\Delta < 0$ or random[0, 1] $< e^{-\Delta/t}$ then
11:            $s \leftarrow s'$
12:            if $f_{SA}(s') < f_{SA}(best\_solution)$ then
13:                $best\_solution \leftarrow s'$
14:            end if
15:        end if
16:        $it \leftarrow it + 1$
17:    end while
18:    $t \leftarrow t \times (1 - \alpha)$
19: end while
20: return $best\_solution$
```

3.7.3 ILS+LS approach

This ILS approach heavily differs from the last approach used on the DVRPMS because it now has a better local search procedure that generates better solutions. It is used only for the DVRPMS with Surplus Demand. In this ILS algorithm an initial solution is generated randomly and is considered as the best solution so far. While the iterations do not meet the stopping criteria, which is the maximum number of iterations with no improvement, the algorithm makes a perturbation, that depends on the current iteration and saves the best solution so far. The best solution proceeds to the local search procedure and if a better solution is found, the variable is updated. The algorithm is described in Algorithm 6.

Algorithm 6 DVRPMS with Surplus Demand Iterated Local Search (ILS+LS)

```
1: $s \leftarrow$ generate a random solution
2: $best\_solution \leftarrow s$
3: $it \leftarrow 0$
4: while $it < max_{it}$ do
5:     $pert_{size} \leftarrow 3$
6:     if $it > 30$ then
7:         $pert_{size} \leftarrow 0,3 * num_{itens}$
8:     end if
9:     if $it > 50$ then
10:        $pert_{size} \leftarrow 0,5 * num_{itens}$
11:    end if
12:    if $it > 70$ then
13:        $pert_{size} \leftarrow 0,7 * num_{itens}$
14:    end if
15:    if $it > 90$ then
16:        $pert_{size} \leftarrow 0,9 * num_{itens}$
17:    end if
18:    $s' \leftarrow$ perturbate $s \in N(s)$ according $pert_{size}$
19:    $s' \leftarrow LOCAL\_SEARCH\{s'\}$
20:    if $f_{SA}(s') < f_{SA}(best\_solution)$ then
21:        $best\_solution \leftarrow s'$
22:        $it \leftarrow 0$
23:    else
24:        $it \leftarrow it + 1$
25:    end if
26: end while
27: $best\_solution \leftarrow LOCAL\_SEARCH\{best\_solution\}$
28: return $best\_solution$
```

3.7.4 TS+LS approach

Following the work in Petersen and Madsen [2009], a tabu search was implemented to compare the general acceptance of the swap neighboorhood. It is only implemented for the DVRPMS with Surplus Demand.

The Tabu Search heuristic uses flexible memory structures to store data from spaces already trespassed. Due to the fact the Tabu Search uses intensively adaptive memory techiniques, the heuristic can usually generate good solutions, where good denotes solutions that are close to the optimum solutions or the global optimum solution itself, obtained in small computational time.

In the TS algorithm an initial solution is generated randomly and is considered as the best solution so far. While the iterations does not meet the stopping criteria, which is the maximum number of iterations with no improvement, a variable $current_{it}$ gets the current iteration. A function N that checks all neighboors (possible swaps) of a current solution s is then called. If a better solution is found, the algorithm updates the current solution, only if the particular swap that has better solution value was not performed in the last $tabu_{size}$ iterations. This is checked by equation $current_{it} - tabu_{size}$. The best solution proceeds to the local search procedure and if a better solution is found, the variable is updated and is marked as the best solution so far. This algorithm is described in Algorithm 7

Algorithm 7 DVRPMS with Surplus Demand Tabu Search (TS+LS)

```
1: $s \leftarrow$ generate a random solution
2: $best\_solution \leftarrow s$
3: $it \leftarrow 0$
4: $current_{it} \leftarrow 0$
5: $tabutable \leftarrow$ initialize with -INF
6: while $it < max_{it}$ do
7:     $current_{it} \leftarrow it + 1$
8:     $N(s) \leftarrow$ generate the neighbors of $s$
9:     $tabutable\{s\} \leftarrow current_{it}$
10:    $s' \leftarrow$ select the best solution $s' \in N(s)$
11:    while $tabutable\{s'\} \neq current_{it} - tabu_{size}$ do
12:        $s' \leftarrow$ select the next best solution $s' \in N(s)$
13:    end while
14:    if $f_{SA}(s') < f_{SA}(best\_solution)$ then
15:        $best\_solution \leftarrow s'$
16:        $it \leftarrow 0$
17:    else
18:        $it \leftarrow it + 1$
19:    end if
20: end while
21: $best\_solution \leftarrow LOCAL\_SEARCH\{best\_solution\}$
22: return $best\_solution$
```

Chapter 4

Computational Experiments

The heuristic approaches were implemented in C++ and 10 repetition tests were performed by an Intel i7 Desktop with 16Gb RAM and a 4.0 Ghz quad-core processor running Ubuntu 16.04 LTS. The exact model for the DVRPMS with Surplus Demand was implemented in C++ using CPLEX 12.5 libraries and was performed by an Intel i7 Desktop with 32Gb RAM and a 4.0 Ghz quad-core processor running Ubuntu 12.10. The latter configuration is a personal computer allocated in the computer science department. It has more RAM memory and was chosen to run the exact model.

A parameter calibration was made for Algorithms 1, 5, 6 and 7. The calibration results are shown in Appendix A. The parameter calibration for the 3 was set in da Silveira et al. [2015] and is shown in the Appendices for completeness.

4.1 Instances

The set of items in each test instance was randomly generated by Iori and Riera-Ledesma based on TSPLIB's `concorde33`. The items are arranged in an area of 100×100 and the depot is located at $(50, 50)$. Euclidean distances were rounded to the smallest integer and the created items sets were named `R05` to `R09`. In some of these instances, not all items are used. Therefore, the items coordinates were chosen in ascending order starting from the first item in the list.

The configurations from Iori and Riera-Ledesma [2015] are shown in Table 4.1. Each configuration was combined with each set of items, generating a total of 75 instances. The first column in Table 4.1 indicates the index of the container's configuration. The second column shows the number of vertices in the instance. The third column shows the number of items in the instance. The fourth column indicates the number of vehicles. Finally, the last column presents the stack configuration of each vehicle.

The first 65 instances of the DVRPMS have a known global optimum, found by the exact methods proposed in Iori and Riera-Ledesma [2015]. The same regions and stack configurations were used in the DVRPMS with surplus demand. To define the score for the problem, an uniform distribution was used to generate random values for each item in the DVRPMS original instances.

Table 4.1: Configuration of the containers for each type of instances

Type	$\|V\|$	$\|I\|$	$\|K\|$	$(r \times l)$
(a)	26	12	2	(2×3) (2×3)
(b)	26	12	2	(2×2) (2×4)
(c)	26	12	3	(2×2) (2×2) (2×2)
(d)	34	16	2	(2×4) (2×4)
(e)	34	16	3	(2×2) (2×3) (2×3)
(f)	34	16	4	(2×2) (2×2) (2×2) (2×2)
(g)	38	18	2	(2×3) (3×4)
(h)	38	18	3	(2×3) (2×3) (2×3)
(i)	38	18	4	(2×2) (2×2) (2×2) (2×3)
(j)	42	20	2	(2×4) (3×4)
(k)	42	20	3	(2×3) (2×3) (2×4)
(l)	42	20	4	(2×3) (2×3) (2×2) (2×2)
(m)	50	24	2	(3×4) (3×4)
(n)	50	24	3	(2×4) (2×4) (2×4)
(o)	50	24	4	(2×3) (2×3) (2×3) (2×3)

4.1.1 Calibration

The parameters selected for the simulated annealing approach for the DVRPMS was chosen as follows: the initial temperature is 800; the number of iterations is 2000; the number of perturbations is 200; the maximum number of successes is 180; the reduction factor is 1% and the probability of the *route-swap* neighborhood being chosen is 60%. The parameters were chosen so the algorithm would take at most 10 seconds for the instances tested, which is the stop condition in the ILS algorithm.

Other calibrated combinations can be seen in Appendix A. Four were the parameters selected for the calibrated Simulated Annealing applied to the modified version of the DVRPMS. The first one, num_it (the maximum number of iterations without improvement) was 300, the t_{min} (the final temperature or stopping criteria of the simulated annealing cooling process) was 6, α (the drop in the temperature on each iteration) was 0.002, and t (the initial temperature of the procedure) was 20.

Only one parameter was set to the ILS applied to the modified version of the DVRPMS. The maximum number of iterations without improvement allowed was set as 100. The percentages were set according the size of the iterations with no improvement. For each instance there is a different size of requests, making it hard to set a fixed size to produce a perturbation, hence only the maximum number of iterations without improvement was calibrated.

Two were the parameters selected for the calibrated Tabu Search applied to the modified version of the DVRPMS. The first one, max_{it}, i.e., the maximum number of iterations without improvement, was 100, and $tabu_{size}$, i.e, the value of the maximum accepted iteration in the tabu table history in which a swap can be performed, was 7.

4.2 Results

In this section, results for the DVRPMS version seen in Iori and Riera-Ledesma [2015] are shown, followed by all results for the modified version of the DVRPMS, where there is a surplus demand of requests.

4.2.1 DVRPMS

From left to right, table 4.2 shows the best solution found in the exact method proposed in Iori and Riera-Ledesma [2015]. The next colums shows the best found solution for the VND algorithm, followed by the computational time of this method and also a column BAO (Best Among Others) that receives $*$ if the best solution of the method is better or equal to the best solution among all other methods. These three columns (Best solution, computational time in seconds and BAO columns) are also shown for the ILS, SA and SA+LS methods. Table 4.3 shows a summary of these findings.

Table 4.2: All DVRPMS results

Instance			Iori & Riera-Ledesma			VND			ILS			SA			SA+LS		
ID	Region	Type	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
1	R05	(a)	738	2	*	927	<1s		739	10		741	3		738	13	*
2	R06		895	<1s	*	1002	<1s		898	10		897	3		895	13	*
3	R07		761	5	*	903	<1s		771	10		771	3		761	13	*
4	R08		848	<1s	*	992	<1s		860	10		857	3		851	13	
5	R09		771	<1s	*	909	<1s		778	10		791	3		776	14	
6	R05	(b)	716	1	*	925	<1s		719	10		721	3		716	3	*
7	R06		859	2	*	1081	<1s		873	10		888	3		866	3	
8	R07		733	3	*	927	<1s		737	10		737	3		733	3	*
9	R08		858	3	*	993	<1s		858	10	*	858	3	*	858	3	*
10	R09		738	1	*	975	<1s		760	10		775	3		741	3	
11	R05	(c)	855	<1s	*	973	<1s		860	10		858	3		855	5	*
12	R06		1011	<1s	*	1166	<1s		1011	10	*	1011	3	*	1011	5	*
13	R07		894	1	*	966	<1s		896	10		894	3	*	894	5	*
14	R08		990	1	*	1131	<1s		998	10		990	3	*	990	5	*
15	R09		852	<1s	*	992	<1s		855	10		852	3	*	852	5	*
16	R05	(d)	947	329	*	1281	<1s		990	10		976	3		948	1	
17	R06		1036	43	*	1325	<1s		1098	10		1093	3		1036	1	*
18	R07		925	46	*	1205	<1s		988	10		971	3		925	1	*
19	R08		1006	52	*	1313	<1s		1089	10		1060	3		1020	1	
20	R09		907	97	*	1156	<1s		951	10		974	3		925	1	
21	R05	(e)	1050	21	*	1300	<1s		1142	10		1087	4		1050	21	*
22	R06		1102	6	*	1402	<1s		1240	10		1114	4		1114	20	
23	R07		1063	29	*	1290	<1s		1168	10		1089	4		1063	20	*
24	R08		1117	10	*	1357	<1s		1215	10		1142	4		1126	21	
25	R09		1021	9	*	1234	<1s		1092	10		1046	4		1021	21	*
26	R05	(f)	1217	37	*	1477	<1s		1264	10		1223	4		1217	10	*

Table 4.2: All DVRPMS results

Instance			Iori & Riera-Ledesma			VND			ILS			SA			SA+LS		
ID	Region	Type	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
27	R06		1290	10	*	1640	<1s		1378	10		1290	4	*	1290	10	*
28	R07		1230	44	*	1506	<1s		1306	10		1230	4	*	1230	10	*
29	R08		1261	18	*	1490	<1s		1340	10		1262	4		1261	10	*
30	R09		1127	7	*	1363	<1s		1214	10		1134	4		1134	10	
31	R05	(g)	950	9	*	1439	<1s		1083	10		1099	3		950	19	*
32	R06		1012	27	*	1516	<1s		1168	10		1087	3		1024	19	
33	R07		932	67	*	1350	<1s		1069	10		1054	3		932	21	*
34	R08		1011	50	*	1475	<1s		1153	10		1128	3		1031	20	
35	R09		909	48	*	1257	<1s		1027	10		985	3		919	21	
36	R05	(h)	1147	60	*	1544	<1s		1292	10		1198	4		1147	33	*
37	R06		1165	8	*	1578	<1s		1304	10		1190	4		1177	33	
38	R07		1123	32	*	1549	<1s		1255	10		1136	4		1123	33	*
39	R08		1184	40	*	1483	<1s		1316	10		1214	4		1184	33	*
40	R09		1080	50	*	1424	<1s		1200	10		1111	4		1080	33	*
41	R05	(I)	1269	319	*	1666	<1s		1400	10		1292	4		1269	18	*
42	R06		1264	38	*	1710	<1s		1467	10		1282	4		1275	18	
43	R07		1261	193	*	1612	<1s		1379	10		1284	4		1261	18	*
44	R08		1310	504	*	1689	<1s		1417	10		1338	4		1310	18	*
45	R09		1157	62	*	1493	<1s		1297	10		1196	4		1157	18	*
46	R05	(j)	1012	115	*	1466	<1s		1186	10		1126	3		1012	6	*
47	R06		1018	24	*	1569	<1s		1284	10		1177	3		1018	6	*
48	R07		1047	290	*	1465	<1s		1249	10		1162	3		1054	6	
49	R08		1040	751	*	1527	<1s		1221	10		1237	3		1050	8	
50	R09		959	55	*	1467	<1s		1165	10		1126	3		977	7	
51	R05	(k)	1174	860	*	1536	<1s		1325	10		1234	4		1174	28	*
52	R06		1200	66	*	1778	<1s		1437	10		1289	4		1200	27	*
53	R07		1243	1471	*	1603	<1s		1449	10		1342	4		1243	28	*

Table 4.2: All DVRPMS results

Instance			Iori & Riera-Ledesma			VND			ILS			SA			SA+LS		
ID	Region	Type	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
54	R08		1196	915	*	1667	<1s		1360	10		1272	4		1206	28	
55	R09		1133	268	*	1502	<1s		1323	10		1206	4		1144	27	
56	R05	(l)	1293	1306	*	1648	<1s		1511	10		1317	5		1293	30	*
57	R06		1340	202	*	1866	<1s		1587	10		1394	4		1340	29	*
58	R07		1373	2843	*	1846	<1s		1596	10		1406	4		1373	30	*
59	R08		1305	460	*	1683	<1s		1527	10		1362	5		1305	29	*
60	R09		1252	832	*	1689	<1s		1429	10		1294	4		1258	30	
61	R05	(m)	1060	2326	*	1682	<1s		1367	10		1220	3		1073	14	
62	R06		1093	70	*	1745	<1s		1444	10		1324	4		1093	14	*
63	R07		1096	263	*	1607	<1s		1446	10		1325	3		1103	13	
64	R08		1120	1149	*	1772	<1s		1454	10		1300	3		1133	18	
65	R09		1034	735	*	1600	<1s		1355	10		1266	3		1047	20	
66	R05	(n)	1318	3600		1880	<1s		1547	10		1387	5		1262	3	*
67	R06		1289	3600		1843	<1s		1678	10		1438	5		1304	3	
68	R07		1350	3600		1871	<1s		1682	10		1429	5		1333	3	*
69	R08		1297	3600		1888	<1s		1633	10		1426	5		1297	3	*
70	R09		1239	3600		1810	<1s		1564	10		1346	5		1243	3	
71	R05	(o)	1518	3600		1978	<1s		1700	10		1412	5		1367	60	*
72	R06		1449	3600		2206	<1s		1834	10		1544	5		1448	60	*
73	R07		1677	3600		2078	<1s		1842	10		1553	5		1465	61	*
74	R08		1463	3600		2016	<1s		1779	10		1476	5		1444	61	*
75	R09		1570	3600		1817	<1s		1732	10		1425	5		1362	60	*

Table 4.3: Summary of results table

	Iori & Riera-Ledesma			VND			ILS			SA			SA+LS		
	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
Average	1103,3	266	65	1467,9	<1s	0	1248,3	10	2	1156,5	4	7	1098,1	18	48

The VND algorithm quickly converges to a local optimum. Knowing that the operator *route-exchange* did not perform well, it was inferred that this fast convergence is due to the many local minima of the DVRPMS. The algorithm served more as a guide to the quality of the initial solution than as a provider of a good final solution.

The algorithms performance depends on the number of items in each stack configuration. The instances were divided by size for comparison. The stacks can have 4, 6, 8 or 12 items per container. Each configuration type was divided according to the highest number on all containers, as seen in Table 4.4. This dependence can be seen in Table 4.5. All proposed algorithms had better results for small instances, and for the SA+LS, worse efficiency is reached when instance size grows. For the exact method, what matters is the number of total items, hence types m, n, o did not reach a global optimum.

Table 4.4: Configuration of the containers for each type of instances

Type	$\|V\|$	$\|I\|$	$\|K\|$	$(r \times l)$	*Size*
(a)	26	12	2	(2×3) (2×3)	Medium
(b)	26	12	2	(2×2) (2×4)	Large
(c)	26	12	3	(2×2) (2×2) (2×2)	Small
(d)	34	16	2	(2×4) (2×4)	Large
(e)	34	16	3	(2×2) (2×3) (2×3)	Medium
(f)	34	16	4	(2×2) (2×2) (2×2) (2×2)	Small
(g)	38	18	2	(2×3) (3×4)	Extra Large
(h)	38	18	3	(2×3) (2×3) (2×3)	Medium
(i)	38	18	4	(2×2) (2×2) (2×2) (2×3)	Medium
(j)	42	20	2	(2×4) (3×4)	Extra Large
(k)	42	20	3	(2×3) (2×3) (2×4)	Large
(l)	42	20	4	(2×3) (2×3) (2×2) (2×2)	Medium
(m)	50	24	2	(3×4) (3×4)	Extra Large
(n)	50	24	3	(2×4) (2×4) (2×4)	Large
(o)	50	24	4	(2×3) (2×3) (2×3) (2×3)	Medium

Table 4.5: Performance comparison using instances size

	Iori & Riera-Ledesma	VND	ILS	SA	SA+LS
	BAO	BAO	BAO	BAO	BAO
Small	100,0%	0,0%	10,0%	60,0%	90,0%
Medium	80,0%	0,0%	0,0%	0,0%	92,0%
Large	66,7%	0,0%	6,7%	6,7%	73,3%
Extra Large	100,0%	0,0%	0,0%	0,0%	33,3%

Both ILS and SA algorithm proved satisfactory for small instances. The ILS algorithm managed to find the best solution for two instances *R08(b)* and *R06(c)* while the SA algorithm found the best solution for seven instances, including *R08(b)* and *R06(c)*.

However, the ILS performed poorly in larger instances, due to the high probability of choosing a bad perturbation. Preliminary investigation had indeed showed that the *route-exchange* structure worsened the solution in more than 80% of the ILS iterations and that an heuristic based in the simulated annealing or tabu search would be a better choice. Table 4.6 shows the average results for these preliminary experiments. They were applied on container types a, c and e, for all 5 regions.

Table 4.6: Effects of the neighboorhood structures in the ILS.

Iterations	Route swap improved	Complete swap improved	Route Exchange improved	Route Exchange worsened
71951,5	97,00%	92,15%	17,31%	82,39%

Indeed, the SA algorithms were better for small instances. In three of these instances (*R05(o)*, *R07(o)* and *R09(o)*), the SA algorithm even managed to find better values than the upper-bounds known in the literature. Thus, it can be inferred that the SA algorithm tends to cover the poor choice of perturbation.

Finally, the SA+LS outperformed all algorithms. It managed to find 40 known global optima, and for 8 of the 10 instances with unkown optimum, the SA+LS outperformed the exact method. This heuristic approach had a better average solution when compared to either the exact method or the other heuristic approaches.

Regarding computational time, ILS runs for a fixed 10 seconds, while SA averages 4 seconds, which means SA runs roughly for the half amount of the ILS time. The VND is very fast, as already mentioned, reaching a local optimum in less than 1 second. The exact model needs a few seconds for the smaller instances but hundreds or thousands for the larger ones, not reaching the optimal solution within 1 hour for the larger set. The SA+LS had again the most successful results, averaging 18 seconds on all 75 instances.

In average, the best solutions found by VND, ILS and SA are respectively 33,0%, 11,6%, and 4,8% worse than the best known solution, but is found in less computational time. While the SA+LS is roughly 0,5% better than the exact method, also running in less computational time. Since specially the SA heuristics performed great for the DVRPMS, a generalization to the original problem is proposed. The DVRPMS with Surplus Demand was proposed to verify the quality of these approaches in situations where the amount of orders for the day exceeds the vehicles capacity.

4.2.2 DVRPMS with Surplus Demand

Table 4.8 shows in the first column the optimal solutions for the 15 instancesfound by the exact method and 65 instances with their upper bound and respective gaps. The second column provides the computational time of the exact model and the BAO column shows if the best solution of a method is better or equal to the best solution among all methods. The same columns are shown for the calibrated ILS+LS, SA+LS, TS+LS and the previously proposed ILS. In the latter, the neighboorhood structures instack-swap and r-stack-permutation were used. Table 4.7 shows a summary of these findings.

The exact model managed to find the global optimum in a feasible time for 15 of the 75 problem instances. For these 15 known global optima solutions, the only instance where the ILS+LS and SA+LS heuristics did not find the global optimum value was instance *R06(b)*. For 12 instances, the upper bound in the exact method was coincidentally also the best known solution of all methods. It might be the global optimum, but the model was not able to verify those 12 solutions in less than a hour. Among all proposed heuritic approaches, the ILS+LS proved to be the best, finding 70 best solutions when compared to all other solutions.

Even after including the neighboorhoods instack-swap and r-stack-permutation, the first proposed ILS was not able to find any of the optima solutions, which may again be due to bad perturbations and unoptimized structures inherited from the DTSPMS. The LS procedure plays a huge role in finding a good route for these vehicles.

In average, the ILS+LS, SA+LS and TS+LS are respectively 4,71%, 4,68% and 2,7% better on finding the best solution to the instances when compared to the exact model, while the ILS is 5,5% worse. Regarding computation time, when compared to the exact model, the fastest heuristic approach (ILS+LS) is 311 times faster than the exact model, finding 70 better or equal to the best solutions among all other methods in an average of 9,4 seconds.

Table 4.7: Summary of the DVRPMS with Surplus Demand findings

Model			ILS+LS Calibrated			SA+LS Calibrated			TS+LS Calibrated		
Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
2091,5	2892,5	27	2190,1	9,3	70	2189,4	9,9	62	2148,7	169,1	13

Table 4.8: All DVRPMS with Surplus Demand results

Instance			Model				ILS+LS Calibrated			SA+LS Calibrated			TS+LS Calibrated			ILS		
ID	Region	Type	Obj	Gap	t(s)	Opt	Best	Time (s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
1	R05	(a)	1401	0,00%	65	*	1401	4	*	1401	6	*	1394	2		1400	10	
2	R06		1436	0,00%	71	*	1436	3	*	1436	6	*	1432	2		1428	10	
3	R07		1462	0,00%	44	*	1462	3	*	1462	6	*	1462	2	*	1450	10	
4	R08		1304	0,00%	33	*	1304	4	*	1304	6	*	1300	2		1280	10	
5	R09		1436	0,00%	24	*	1436	4	*	1436	6	*	1431	2		1422	10	
6	R05	(b)	1424	0,00%	54	*	1424	1	*	1424	1	*	1424	3	*	1374	10	
7	R06		1417	0,00%	37	*	1412	1		1412	1		1417	3	*	1412	10	
8	R07		1482	0,00%	54	*	1482	1	*	1482	1	*	1482	3	*	1457	10	
9	R08		1314	0,00%	45	*	1314	1	*	1314	1	*	1296	3		1309	10	
10	R09		1480	0,00%	15	*	1480	1	*	1480	1	*	1456	3		1431	10	
11	R05	(c)	1317	0,00%	73	*	1317	1	*	1317	2	*	1317	1	*	1285	10	
12	R06		1288	0,00%	132	*	1288	1	*	1288	2	*	1274	1		1253	10	
13	R07		1337	0,00%	119	*	1337	1	*	1337	2	*	1337	1	*	1334	10	
14	R08		1208	0,00%	48	*	1208	1	*	1208	2	*	1208	1	*	1171	10	
15	R09		1337	0,00%	126	*	1337	1	*	1337	2	*	1337	1	*	1314	10	
16	R05	(d)	1925	6,00%	3600		1954	<1s	*	1954	1	*	1926	9		1845	10	
17	R06		2032	2,20%	3600	*	2032	<1s	*	2032	1	*	1982	9		1892	10	
18	R07		1995	2,60%	3600		2006	<1s	*	1995	1		1981	8		1844	10	
19	R08		1882	3,90%	3600		1896	<1s	*	1896	1	*	1862	8		1789	10	
20	R09		2044	1,60%	3600		2036	<1s		2049	1	*	2007	9		1901	10	
21	R05	(e)	1857	5,30%	3600	*	1857	8	*	1857	5	*	1816	3		1692	10	
22	R06		1913	4,10%	3600	*	1913	9	*	1913	5	*	1897	3		1757	10	

Table 4.8: All DVRPMS with Surplus Demand results

Instance			Model				ILS+LS Calibrated			SA+LS Calibrated			TS+LS Calibrated			ILS		
ID	Region	Type	Obj	Gap	t(s)	Opt	Best	Time (s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
23	R07		1854	4,80%	3600		1888	8	*	1888	5	*	1861	3		1707	10	
24	R08		1781	5,10%	3600		1792	8	*	1792	5	*	1761	3		1672	10	
25	R09		1915	4,40%	3600	*	1915	8	*	1915	5	*	1890	3		1771	10	
26	R05	(f)	1718	10,80%	3600	*	1718	3	*	1718	2	*	1697	1		1583	10	
27	R06		1795	9,70%	3600	*	1795	3	*	1795	2	*	1756	1		1641	10	
28	R07		1720	7,90%	3600	*	1720	3	*	1720	2	*	1720	1	*	1584	10	
29	R08		1660	9,90%	3600	*	1660	3	*	1660	3	*	1654	1		1540	10	
30	R09		1790	9,50%	3600	*	1790	3	*	1790	2	*	1790	1	*	1707	10	
31	R05	(g)	2221	4,40%	3600		2240	10	*	2240	6	*	2185	462		2083	10	
32	R06		2340	2,40%	3600		2349	11	*	2349	7	*	2271	463		2115	10	
33	R07		2298	2,80%	3600		2318	11	*	2318	7	*	2236	409		2092	10	
34	R08		2227	1,60%	3600	*	2227	11	*	2224	6		2192	494		2077	10	
35	R09		2319	3,00%	3600	*	2319	12	*	2319	5	*	2266	553		2127	10	
36	R05	(h)	2086	9,20%	3600		2104	13	*	2104	6	*	2083	4		1895	10	
37	R06		2155	9,50%	3600		2175	14	*	2175	6	*	2174	4		1945	10	
38	R07		2076	11,20%	3600		2119	13	*	2119	6	*	2062	4		1892	10	
39	R08		2011	10,10%	3600		2071	15	*	2071	6	*	2043	4		1882	10	
40	R09		2107	12,00%	3600		2148	15	*	2148	6	*	2148	5	*	1943	10	
41	R05	(i)	2001	11,10%	3600	*	2001	8	*	2001	3	*	1965	3		1782	10	
42	R06		2054	12,50%	3600		2070	7	*	2062	3		2070	3	*	1788	10	
43	R07		1920	15,10%	3600		1986	9	*	1975	3		1969	3		1735	10	
44	R08		1895	13,70%	3600		1963	6	*	1963	3	*	1920	3		1786	10	
45	R09		2017	13,60%	3600		2043	9	*	2043	3	*	2008	3		1830	10	

Table 4.8: All DVRPMS with Surplus Demand results

Instance			Model				ILS+LS Calibrated			SA+LS Calibrated			TS+LS Calibrated			ILS		
ID	Region	Type	Obj	Gap	t(s)	Opt	Best	Time (s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
46	R05	(j)	2621	6,40%	3600		2626	6	*	2608	4		2517	538		2356	10	
47	R06		2593	8,00%	3600		2645	6	*	2641	4		2565	615		2376	10	
48	R07		2491	12,80%	3600		2646	5	*	2646	3	*	2573	508		2345	10	
49	R08		2510	7,30%	3600		2578	6	*	2575	5		2511	554		2284	10	
50	R09		2557	9,30%	3600		2626	7		2629	5	*	2590	529		2352	10	
51	R05	(k)	2339	18,90%	3600		2494	15	*	2488	4		2416	9		2218	10	
52	R06		2381	15,50%	3600		2525	18	*	2517	4		2497	10		2171	10	
53	R07		2424	12,50%	3600		2484	16	*	2477	4		2446	9		2167	10	
54	R08		2351	14,40%	3600		2433	18	*	2433	4	*	2366	9		2186	10	
55	R09		2357	16,10%	3600		2484	16	*	2484	4	*	2432	9		2221	10	
56	R05	(l)	2288	17,40%	3600		2385	14	*	2374	4		2320	4		2071	10	
57	R06		2293	18,70%	3600		2381	14	*	2370	4		2379	4		2054	10	
58	R07		2238	18,70%	3600		2366	16	*	2366	4	*	2277	4		2046	10	
59	R08		2257	16,70%	3600		2315	16	*	2315	4	*	2251	4		2018	10	
60	R09		2285	17,40%	3600		2362	18	*	2362	4	*	2337	4		2075	10	
									*									
61	R05	(m)	3091	14,00%	3600		3269	10	*	3250	8		3214	1576		2890	10	
62	R06		2439	43,10%	3600		3250	14	*	3242	12		3164	1612		2847	10	
63	R07		3008	16,40%	3600		3259	12	*	3259	6	*	3157	1085		2818	10	
64	R08		3076	11,10%	3600		3199	14	*	3199	9	*	3096	1555		2869	10	
65	R09		2584	33,50%	3600		3252	17	*	3184	12		3111	1395		2853	10	
66	R05	(n)	2680	29,90%	3600		3095	1		3100	1	*	3007	20		2729	10	
67	R06		2751	24,00%	3600		3090	1	*	3087	1		3042	20		2663	10	

Table 4.8: All DVRPMS with Surplus Demand results

Instance			Model				ILS+LS Calibrated			SA+LS Calibrated			TS+LS Calibrated			ILS		
ID	Region	Type	Obj	Gap	t(s)	Opt	Best	Time (s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO	Best	t(s)	BAO
68	R07		2663	30,70%	3600		3037	1	*	3012	1		2976	18		2620	10	
69	R08		2775	22,00%	3600		3010	1	*	2978	1		2944	20		2666	10	
70	R09		2533	34,70%	3600		3027	1		3037	1	*	2955	20		2651	10	
71	R05	(o)	2700	26,10%	3600		2990	37	*	2990	6	*	2886	8		2614	10	
72	R06		2422	38,80%	3600		2956	34	*	2942	6		2904	8		2516	10	
73	R07		2708	25,60%	3600		2905	39	*	2905	6	*	2788	8		2396	10	
74	R08		2552	30,80%	3600		2891	39	*	2891	6	*	2806	8		2490	10	
75	R09		2645	26,30%	3600		2933	42	*	2933	6	*	2869	8		2454	10	

Chapter 5

Conclusions

The DVRPMS was a novel problem published in 2015 and based on previous problems, specially the DTSPMS, published in 2009. It all started with a real-life problem, regarding the logistics of using a vehicle that travels through different regions usually separated by sea. The container should not be rearranged for security reasons, so the vehicle should also have an optimum delivery route.

The heuristic approaches used in this dissertation proved to be a great alternative to the exact models for both DVRPMS and DVRPMS with Surplus Demand. For the DVRPMS, the SA+LS heuristic could find the best solution among all methods for 48 out of 75 instances. In average, it provided better results when compared to the exact model, outperforming the latter by roughly 0,5%.

The DVRPMS with Surplus Demand is a novel extension to the DVRPMS that was proposed in this dissertation. In regard of this problem, the ILS+LS heuristic approach found the best solution among all methods for 70 out of the 75 instances, averaging 9,4 seconds of computational time. Also, even with 100 iterations, the ILS+LS averaged only 9,4 seconds of computational time, which is desirable for a practical situation.

Additionally, even though the chosen ILS+LS combination obtained better results, some SA+LS combinations had better averages. Indeed, the ILS+LS combination and the better SA+LS combinations are in a group that does not statistically differ at 95% of confiability, according to the statistical tests made on Appendix A.

5.1 Publications

The papers generated by the work present on this dissertation were published and presented on:

- **U. E. F. Silveira**, M. P. L. Benedito, A. G. Santos, Heuristic approaches to double vehicle routing problem with multiple stacks, 2015, *15th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 231-236.

- J. B. C. Chagas, **U. E. F. Silveira**, M. P. L. Benedito and A. G. Santos, Simulated annealing metaheuristic for the Double Vehicle Routing Problem with Multiple Stacks, 2016, *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016, pp. 1311-1316.

5.2 Further works

The problem is still in its infancy and does not work for real-life size instances (containers of 11x3 feet). Therefore, the DVRPMS is open for further investigation of bigger instances and additional constraints, such as time dependable constraints or variable demands.

Even for the first problem explored (DTSPMS) there is no model or heuristic approach that could solve the problem for bigger instances of 132 requests. Since the local search procedure does not apply for the DTSPMS, a further modification on this procedure could be made to verify its quality when used to solve DTSPMS instances, because it performed reasonably well to the DVRPMS.

Bibliography

Barbato, M., Grappe, R., Lacroix, M., and Calvo, R. W. (2016). Polyhedral results and a branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. *Discrete Optimization*, 21:25 – 41.

Casazza, M., Ceselli, A., and Nunkesser, M. (2012). Efficient algorithms for the double traveling salesman problem with multiple stacks. *Computers & operations research*, 39(5):1044–1053.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). Vehicle routing problems with time windows. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Cordeau, J.-F., Iori, M., Laporte, G., and Salazar-González, J. J. (2010). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with lifo loading. *Network*, 55(1):46–59.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

da Silveira, U. E. F., Benedito, M. P. L., and dos Santos, A. G. (2015). Heuristic approaches to double vehicle routing problem with multiple stacks. In *15th International Conference on Intelligent Systems Design and Applications, ISDA 2015, Marrakech, Morocco, December 14-16, 2015*, pages 231–236.

European Commission for Mobility and Transport (2010). Road freight transport vademecum. Technical report.

Felipe, A., Ortuño, M. T., and Tirado, G. (2009a). The double traveling salesman problem with multiple stacks: A variable neighborhood search approach. *Comput. Oper. Res.*, 36(11):2983–2993.

Felipe, A., Ortuño, M. T., and Tirado, G. (2009b). New neighborhood structures for the double traveling salesman problem with multiple stacks. *TOP*, 17(1):190–213.

Frazzon, E. M. (2009). Sustainability and effectiveness in global logistic systems: [an approach based on a long-term learning process]. GITO-Verl, Berlin.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126 – 139. Graph Optimization {IV}.

Iori, M. and Riera-Ledesma, J. (2015). Exact algorithms for the double vehicle routing problem with multiple stacks. *Computers & Operations Research*, 63(C):83–101.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: Framework and applications handbook of metaheuristics. volume 146 of *International Series in Operations Research & Management Science*, chapter 12, pages 363–397. Springer US, Boston, MA.

Martínez, M. A. A., Cordeau, J.-F., Dell'Amico, M., and Iori, M. (2013). A branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. *INFORMS Journal on Computing*, 25(1):41–55.

Mladenović, J. B., Hansen, P., and Taillard, E. D. (2000). Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operations Research*, 48(3):444–460.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117.

Petersen, H. L. and Madsen, O. B. (2009). The double travelling salesman problem with multiple stacks - formulation and heuristic solution approaches. *European Journal of Operational Research*, 198(1):139 – 147.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation.* Wiley Publishing.

Toth, P. and Vigo, D., editors (2001). *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

UK Department for Transport (2011). The logistics growth review connecting people with goods. Technical report, London.

Urrutia, S., Milanés, A., and Løkketangen, A. (2015). A dynamic programming based local search approach for the double traveling salesman problem with multiple stacks. *International Transactions in Operational Research*, 22(1):61–75.

Yung-yu Tseng, W. L. Y. and Taylor, M. A. P. (2010). The role of transportation in logistics chain. *Proceedings of the Eastern Asia Society for Transportation Studies.*

Appendix A

Parameter Calibration

The results for the calibration of the proposed methods are shown in this appendix. Such results indicate the best parameters for each algorithm. Each algorithm (ILS+LS, SA+LS, TS+LS, ILS) had several combinations of parameters that needed to be analised. These combinations were defined in the hope that the computational time would not exceed one minute. The complete calibration process was the same for each algorithm that used the local search procedure (ILS+LS, SA+LS and TS+LS) while the ILS without a local search procedure was calibrated visualizing a simple plot graph of the average solutions through time.

In order to properly define the best parameters, a set of experiments was conducted and best results were chosen. A portion of the known instances was randomnly selected for each algorithm and different combinations of parameters were set for each one of these methods.

Because heuristic approaches based on the ILS, SA and TS are different in nature, each one received a different set of parameters, chosen according to preliminary tests that compared execution time of several combinations of parameters. For a more reliable result, each treatment was executed with 10 repetition tests.

The value that was given to the statistical analysis tools was the RPD. RPD is a metric commonly used to evaluate the quality of the solutions obtained by heuristic algorithms. Equation (A.1) describes how this metric was calculated, where f_{method} is the objective function value obtained by a given solution and f_{best} is the value of the best objective function value found among all solutions. The lower the RPD the better, because this means that compared solution is closer to the best known solution. All tests were conducted in the RStudio software on an Intel i7 Desktop and Ubuntu 16.04 LTS.

$$\text{RPD}_{method} = [(f_{method} - f_{best})/f_{best}] \times 100\% \tag{A.1}$$

A performance evaluation of these treatments was performed using the Additive Indicator metric. Using a reference set composed by non-dominated points each execution is compared to the reference set to be properly evaluated. To proceed to the statistical analysis, the average value obtained in all instances is defined as a dependent variable related to the pair treatment/repetition.

Table A.1 shows all combinations for all algorithms that use the local search procedure and their respective treatment names (varying from T001 to T134). In this table, it_n represents the number of iterations without improvement (same on both ILS and SA). It should be noted that for the SA, this repetitions are applied on the same temperature, while the ILS does not have temperature parameters. T_i is the initial temperature of the SA heuristic, α, the drop in the temperature on each iteration, T_f is the final temperature of the SA heuristic, n_{rep} is the maximum value of repetitions the tabu search algorithm will perform without any improvement and $tabu_s$ is the size of the tabu table.

Table A.1: Treatments and their respective parameters

Treatment	Alg.	it_n	T_i	α	it_n	T_f	n_{rep}	$tabu_s$
T001	ILS1	10	-	-	-	-	-	-
T002	ILS2	25	-	-	-	-	-	-
T003	ILS3	50	-	-	-	-	-	-
T004	ILS4	75	-	-	-	-	-	-
T005	ILS5	100	-	-	-	-	-	-
T006	SA1	-	10	0,002	100	6	-	-
T007	SA2	-	10	0,002	100	8	-	-
T008	SA3	-	10	0,002	100	10	-	-
T009	SA4	-	10	0,002	200	6	-	-
T010	SA5	-	10	0,002	200	8	-	-
T011	SA6	-	10	0,002	200	10	-	-
T012	SA7	-	10	0,002	300	6	-	-
T013	SA8	-	10	0,002	300	8	-	-
T014	SA9	-	10	0,002	300	10	-	-
T015	SA10	-	10	0,01	100	6	-	-
T016	SA11	-	10	0,01	100	8	-	-
T017	SA12	-	10	0,01	100	10	-	-
T018	SA13	-	10	0,01	200	6	-	-
T019	SA14	-	10	0,01	200	8	-	-
T020	SA15	-	10	0,01	200	10	-	-
T021	SA16	-	10	0,01	300	6	-	-

Table A.1: Treatments and their respective parameters

Treatment	Alg.	it_n	T_i	α	it_n	T_f	n_{rep}	$tabu_s$
T022	SA17	-	10	0,01	300	8	-	-
T023	SA18	-	10	0,01	300	10	-	-
T024	SA19	-	10	0,05	100	6	-	-
T025	SA20	-	10	0,05	100	8	-	-
T026	SA21	-	10	0,05	100	10	-	-
T027	SA22	-	10	0,05	200	6	-	-
T028	SA23	-	10	0,05	200	8	-	-
T029	SA24	-	10	0,05	200	10	-	-
T030	SA25	-	10	0,05	300	6	-	-
T031	SA26	-	10	0,05	300	8	-	-
T032	SA27	-	10	0,05	300	10	-	-
T033	SA28	-	12	0,002	100	6	-	-
T034	SA29	-	12	0,002	100	8	-	-
T035	SA30	-	12	0,002	100	10	-	-
T036	SA31	-	12	0,002	200	6	-	-
T037	SA32	-	12	0,002	200	8	-	-
T038	SA33	-	12	0,002	200	10	-	-
T039	SA34	-	12	0,002	300	6	-	-
T040	SA35	-	12	0,002	300	8	-	-
T041	SA36	-	12	0,002	300	10	-	-
T042	SA37	-	12	0,01	100	6	-	-
T043	SA38	-	12	0,01	100	8	-	-
T044	SA39	-	12	0,01	100	10	-	-
T045	SA40	-	12	0,01	200	6	-	-
T046	SA41	-	12	0,01	200	8	-	-
T047	SA42	-	12	0,01	200	10	-	-
T048	SA43	-	12	0,01	300	6	-	-
T049	SA44	-	12	0,01	300	8	-	-
T050	SA45	-	12	0,01	300	10	-	-
T051	SA46	-	12	0,05	100	6	-	-
T052	SA47	-	12	0,05	100	8	-	-
T053	SA48	-	12	0,05	100	10	-	-
T054	SA49	-	12	0,05	200	6	-	-
T055	SA50	-	12	0,05	200	8	-	-

Table A.1: Treatments and their respective parameters

Treatment	Alg.	it_n	T_i	α	it_n	T_f	n_{rep}	$tabu_s$
T056	SA51	-	12	0,05	200	10	-	-
T057	SA52	-	12	0,05	300	6	-	-
T058	SA53	-	12	0,05	300	8	-	-
T059	SA54	-	12	0,05	300	10	-	-
T060	SA55	-	15	0,002	100	6	-	-
T061	SA56	-	15	0,002	100	8	-	-
T062	SA57	-	15	0,002	100	10	-	-
T063	SA58	-	15	0,002	200	6	-	-
T064	SA59	-	15	0,002	200	8	-	-
T065	SA60	-	15	0,002	200	10	-	-
T066	SA61	-	15	0,002	300	6	-	-
T067	SA62	-	15	0,002	300	8	-	-
T068	SA63	-	15	0,002	300	10	-	-
T069	SA64	-	15	0,01	100	6	-	-
T070	SA65	-	15	0,01	100	8	-	-
T071	SA66	-	15	0,01	100	10	-	-
T072	SA67	-	15	0,01	200	6	-	-
T073	SA68	-	15	0,01	200	8	-	-
T074	SA69	-	15	0,01	200	10	-	-
T075	SA70	-	15	0,01	300	6	-	-
T076	SA71	-	15	0,01	300	8	-	-
T077	SA72	-	15	0,01	300	10	-	-
T078	SA73	-	15	0,05	100	6	-	-
T079	SA74	-	15	0,05	100	8	-	-
T080	SA75	-	15	0,05	100	10	-	-
T081	SA76	-	15	0,05	200	6	-	-
T082	SA77	-	15	0,05	200	8	-	-
T083	SA78	-	15	0,05	200	10	-	-
T084	SA79	-	15	0,05	300	6	-	-
T085	SA80	-	15	0,05	300	8	-	-
T086	SA81	-	15	0,05	300	10	-	-
T087	SA82	-	20	0,002	100	6	-	-
T088	SA83	-	20	0,002	100	8	-	-
T089	SA84	-	20	0,002	100	10	-	-

Table A.1: Treatments and their respective parameters

Treatment	Alg.	it_n	T_i	α	it_n	T_f	n_{rep}	$tabu_s$
T090	SA85	-	20	0,002	200	6	-	-
T091	SA86	-	20	0,002	200	8	-	-
T092	SA87	-	20	0,002	200	10	-	-
T093	SA88	-	20	0,002	300	6	-	-
T094	SA89	-	20	0,002	300	8	-	-
T095	SA90	-	20	0,002	300	10	-	-
T096	SA91	-	20	0,01	100	6	-	-
T097	SA92	-	20	0,01	100	8	-	-
T098	SA93	-	20	0,01	100	10	-	-
T099	SA94	-	20	0,01	200	6	-	-
T100	SA95	-	20	0,01	200	8	-	-
T101	SA96	-	20	0,01	200	10	-	-
T102	SA97	-	20	0,01	300	6	-	-
T103	SA98	-	20	0,01	300	8	-	-
T104	SA99	-	20	0,01	300	10	-	-
T105	SA100	-	20	0,05	100	6	-	-
T106	SA101	-	20	0,05	100	8	-	-
T107	SA102	-	20	0,05	100	10	-	-
T108	SA103	-	20	0,05	200	6	-	-
T109	SA104	-	20	0,05	200	8	-	-
T110	SA105	-	20	0,05	200	10	-	-
T111	SA106	-	20	0,05	300	6	-	-
T112	SA107	-	20	0,05	300	8	-	-
T113	SA108	-	20	0,05	300	10	-	-
T114	TS1	-	-	-	-	-	100	3
T115	TS2	-	-	-	-	-	100	5
T116	TS3	-	-	-	-	-	100	7
T117	TS4	-	-	-	-	-	100	9
T118	TS5	-	-	-	-	-	100	10
T119	TS6	-	-	-	-	-	100	15
T120	TS7	-	-	-	-	-	100	20
T121	TS8	-	-	-	-	-	200	3
T122	TS9	-	-	-	-	-	200	5
T123	TS10	-	-	-	-	-	200	7

Table A.1: Treatments and their respective parameters

Treatment	Alg.	it_n	T_i	α	it_n	T_f	n_{rep}	$tabu_s$
T124	TS11	-	-	-	-	-	200	9
T125	TS12	-	-	-	-	-	200	10
T126	TS13	-	-	-	-	-	200	15
T127	TS14	-	-	-	-	-	200	20
T128	TS15	-	-	-	-	-	300	3
T129	TS16	-	-	-	-	-	300	5
T130	TS17	-	-	-	-	-	300	7
T131	TS18	-	-	-	-	-	300	9
T132	TS19	-	-	-	-	-	300	10
T133	TS20	-	-	-	-	-	300	15
T134	TS21	-	-	-	-	-	300	20

In the next section, a set of three tables will be shown for each heuristic approach. The first shows the results found in the additive indicator metric and is composed by the treatments and their respective depedent variable values (Tables A.2 and A.4, A.6). The next table shows the Shapiro-Wilk p-values (so it is possible to check if these methods do not violate any statistical assumption, such as data normality), followed by the groups of treatments (if any difference can be infered) and their respective averages given by the Scott-Knott test (Tables A.3, A.5 and A.7). A boxplot is also shown to visualize the last table (Figures A.1, A.2, A.3 and A.4).

A.1 TS+LS parameter calibration

The TS+LS combinations were put in a script and ten repetition tests were made in 51 instances of the 75 known instances. The number was set to 51, for a feasible amount of time (instances with four vehicles ran for more than 10 minutes) The residuals were considered normal according to the Shapiro-Wilk test, and no difference could be inferred in all combinations. The combination $\{100, 7\}$ had the second lowest average and this combination was chosen, because the first lowest combination $\{300, 3\}$ runs for an excessive amount of time on the instances with four vehicles (more than 4000 seconds). Tables A.2 and A.3 refers to the TS+LS statistical data and Figure A.1 is the boxplot that refers to table A.3 data.

Table A.2: TS+LS statistical data

Treatment	Averages
TS001	3,811036
TS002	3,883321
TS003	3,571547
TS004	3,730748
TS005	3,808890
TS006	3,768393
TS007	3,734628
TS008	3,585761
TS009	3,756768
TS010	3,752020
TS011	3,697099
TS012	3,831125
TS013	3,681276
TS014	3,649213
TS015	3,554143
TS016	3,834498
TS017	3,646526
TS018	3,761541
TS019	3,778317
TS020	3,703896
TS021	3,673472

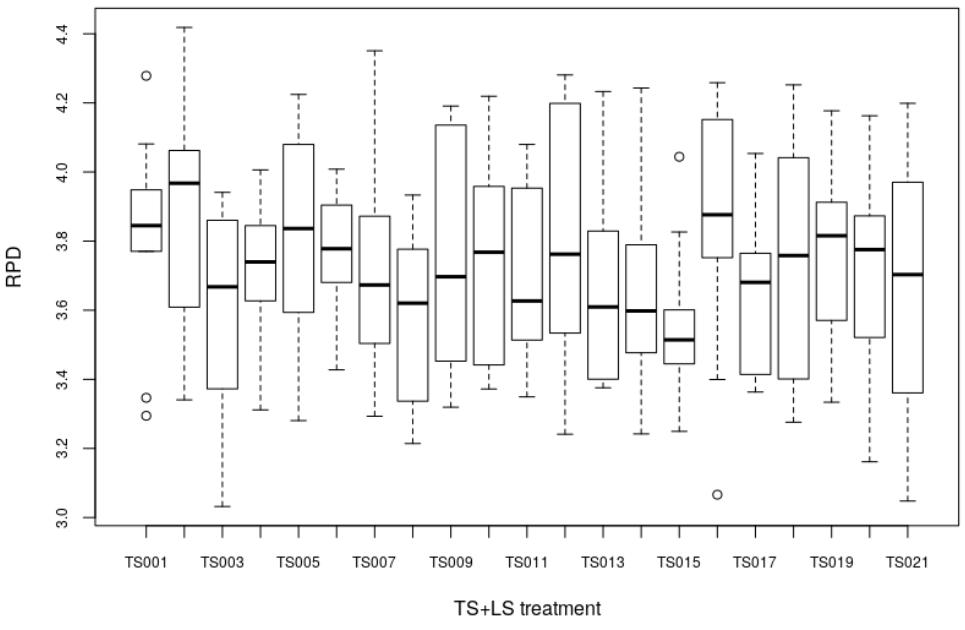


Figure A.1: TS+LS boxplot image

Table A.3: TS+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	0,1693	
Scott-Knott test		
	Groups	Averages
TS001	-	3,811036
TS002	-	3,883321
TS003	-	3,571547
TS004	-	3,730748
TS005	-	3,80889
TS006	-	3,768393
TS007	-	3,734628
TS008	-	3,585761
TS009	-	3,756768
TS010	-	3,75202
TS011	-	3,697099
TS012	-	3,831125
TS013	-	3,681276
TS014	-	3,649213
TS015	-	3,554143
TS016	-	3,834498
TS017	-	3,646526
TS018	-	3,761541
TS019	-	3,778317
TS020	-	3,703896
TS021	-	3,673472

A.2 SA+LS parameter calibration

The SA+LS combinations were put in a script and ten repetition tests were made in 60 instances of the 75 known instances. The number of instances was set to 60 to avoid fixing results when using all instances, which may mask the statistical data. The residuals were not considered normal according to the Shapiro-Wilk test, and differences could be inferred according to the Scott-Knott test and are shown in Table A.5. The combination $\{20, 0, 002, 300, 6\}$ had the lowest average on the lowest group, so this combination was chosen. Tables A.4 and A.5 refer to SA+LS statistical data and Figure A.2 is the boxplot that refers to table A.5 data.

Table A.4: SA+LS statistical data

Treatment	Averages
SA001	1,06886

Table A.4: SA+LS statistical data

Treatment	Averages
SA002	1,43269
SA003	4,21887
SA004	0,88843
SA005	1,08366
SA006	4,19446
SA007	0,66557
SA008	0,90133
SA009	4,24547
SA010	1,97331
SA011	2,45460
SA012	4,07564
SA013	1,60324
SA014	2,00693
SA015	4,20067
SA016	1,30599
SA017	1,70926
SA018	4,18087
SA019	3,15729
SA020	3,67309
SA021	4,20134
SA022	2,75253
SA023	3,22621
SA024	4,20091
SA025	2,44666
SA026	2,79827
SA027	4,35845
SA028	0,85467
SA029	1,02927
SA030	1,45623
SA031	0,72776
SA032	0,74747
SA033	1,07075
SA034	0,56849
SA035	0,64771

Table A.4: SA+LS statistical data

Treatment	Averages
SA036	0,91690
SA037	1,70391
SA038	2,00766
SA039	2,46814
SA040	1,28706
SA041	1,50951
SA042	2,02196
SA043	1,13721
SA044	1,26733
SA045	1,73629
SA046	2,83262
SA047	3,16351
SA048	3,62799
SA049	2,31775
SA050	2,68220
SA051	3,11916
SA052	2,01838
SA053	2,39303
SA054	2,92058
SA055	0,72118
SA056	0,85489
SA057	0,98954
SA058	0,51238
SA059	0,58461
SA060	0,79286
SA061	0,40736
SA062	0,48286
SA063	0,62843
SA064	1,45177
SA065	1,66529
SA066	1,94797
SA067	1,09874
SA068	1,17150
SA069	1,41345

Table A.4: SA+LS statistical data

Treatment	Averages
SA070	0,90085
SA071	1,08221
SA072	1,25560
SA073	2,48737
SA074	2,87110
SA075	3,10077
SA076	2,08539
SA077	2,21842
SA078	2,63982
SA079	1,82428
SA080	2,00145
SA081	2,33786
SA082	0,63045
SA083	0,74291
SA084	0,85727
SA085	0,43525
SA086	0,56444
SA087	0,66336
SA088	0,37431
SA089	0,42078
SA090	0,54401
SA091	1,25694
SA092	1,36293
SA093	1,54958
SA094	0,95663
SA095	1,03755
SA096	1,27833
SA097	0,78930
SA098	0,90904
SA099	1,09009
SA100	2,28237
SA101	2,55522
SA102	2,69015
SA103	1,84211

Table A.4: SA+LS statistical data

Treatment	Averages
SA104	2,04399
SA105	2,19341
SA106	1,66986
SA107	1,80729
SA108	1,88563

Table A.5: SA+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	$< 0{,}005$	
Scott-Knott test		
	Groups	Averages
SA027	a	4,35845
SA009	a	4,24547
SA003	a	4,21887
SA021	a	4,20134
SA024	a	4,20091
SA015	a	4,20067
SA006	a	4,19446
SA018	a	4,18087
SA012	a	4,07564
SA020	b	3,67309
SA048	b	3,62799
SA023	c	3,22621
SA047	c	3,16351
SA019	c	3,15729
SA051	c	3,11916
SA075	c	3,10077
SA054	d	2,92058
SA074	d	2,87110
SA046	d	2,83262
SA026	d	2,79827
SA022	d	2,75253
SA102	e	2,69015

Table A.5: SA+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	$< 0{,}005$	
Scott-Knott test		
	Groups	Averages
SA050	e	2,68220
SA078	e	2,63982
SA101	f	2,55522
SA073	f	2,48737
SA039	f	2,46814
SA011	f	2,45460
SA025	f	2,44666
SA053	g	2,39303
SA081	g	2,33786
SA049	g	2,31775
SA100	g	2,28237
SA077	h	2,21842
SA105	h	2,19341
SA076	i	2,08539
SA104	i	2,04399
SA042	i	2,02196
SA052	i	2,01838
SA038	i	2,00766
SA014	i	2,00693
SA080	i	2,00145
SA010	i	1,97331
SA066	i	1,94797
SA108	i	1,88563
SA103	j	1,84211
SA079	j	1,82428
SA107	j	1,80729
SA045	k	1,73629
SA017	k	1,70926
SA037	k	1,70391
SA106	k	1,66986
SA065	k	1,66529

Table A.5: SA+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	$< 0{,}005$	
Scott-Knott test		
	Groups	Averages
SA013	l	1,60324
SA093	l	1,54958
SA041	l	1,50951
SA030	m	1,45623
SA064	m	1,45177
SA002	m	1,43269
SA069	m	1,41345
SA092	n	1,36293
SA016	n	1,30599
SA040	n	1,28706
SA096	n	1,27833
SA044	n	1,26733
SA091	n	1,25694
SA072	n	1,25560
SA068	o	1,17150
SA043	o	1,13721
SA067	o	1,09874
SA099	o	1,09009
SA005	o	1,08366
SA071	o	1,08221
SA033	o	1,07075
SA001	o	1,06886
SA095	o	1,03755
SA029	o	1,02927
SA057	o	0,98954
SA094	p	0,95663
SA036	p	0,91690
SA098	p	0,90904
SA008	p	0,90133
SA070	p	0,90085
SA004	p	0,88843

Table A.5: SA+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	< 0,005	
Scott-Knott test		
	Groups	Averages
SA084	p	0,85727
SA056	p	0,85489
SA028	p	0,85467
SA060	p	0,79286
SA097	p	0,78930
SA032	q	0,74747
SA083	q	0,74291
SA031	q	0,72776
SA055	q	0,72118
SA007	q	0,66557
SA087	q	0,66336
SA035	q	0,64771
SA082	q	0,63045
SA063	q	0,62843
SA059	q	0,58461
SA034	r	0,56849
SA086	r	0,56444
SA090	r	0,54401
SA058	r	0,51238
SA062	r	0,48286
SA085	s	0,43525
SA089	s	0,42078
SA061	s	0,40736
SA088	s	0,37431

A.3 ILS+LS parameter calibration

The ILS+LS combination was put in a script and ten repetition tests were made in 60 instances of the 75 known instances. The residuals were considered not normal according the Shapiro-Wilk test, and differences could be inferred in some combinations. The value

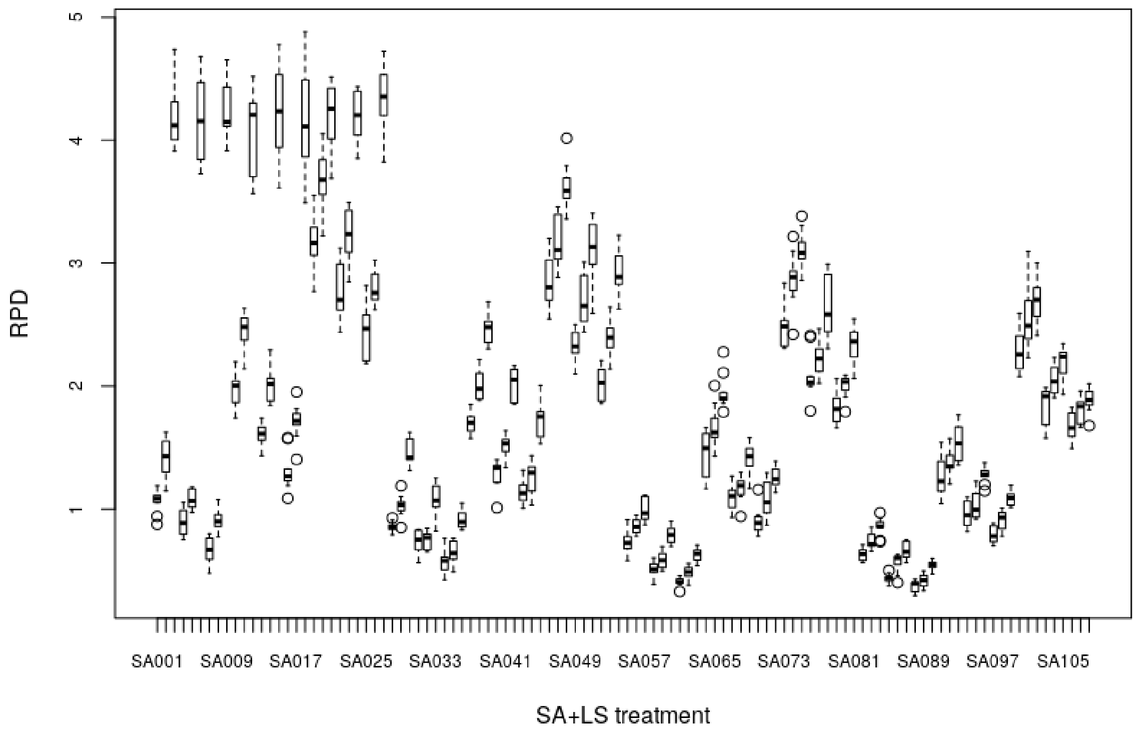


Figure A.2: SA+LS boxplot image

$max_{noimprovement} = 100$ had the lowest average so this value was chosen. In the end of this section, figure A.4 shows the boxplot of all algorithms tested. Even tough the ILS+LS had better results, the SA+LS combination had the better average, but they are in a group that does not differ in 95% of confiability, which might be a indication that the SA+LS is a more robust approach. Below tables A.6 and A.7 are shown followed by Figure A.3 referring to table A.7 data.

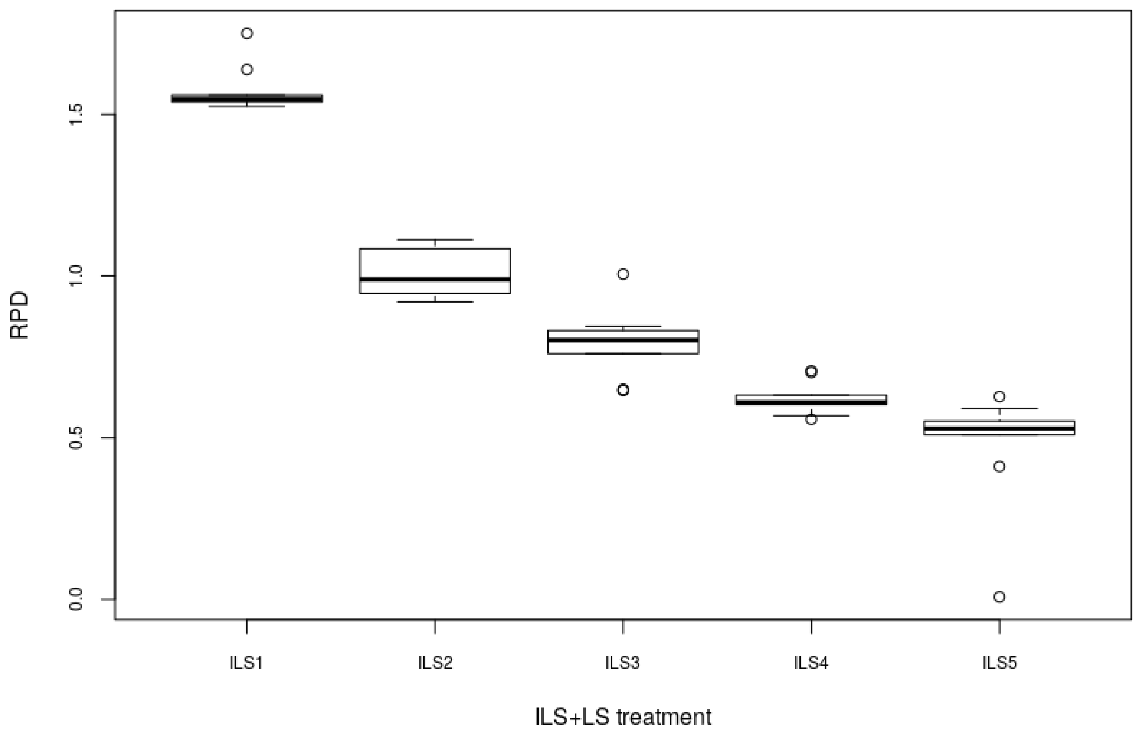
Table A.6: ILS+LS statistical data

Treatment	Averages
ILS001	1,57309
ILS002	1,00650
ILS003	0,79083
ILS004	0,62150
ILS005	0,48117

Table A.7: ILS+LS Shapiro-Wilk and Scott-Knott test results

Shapiro-Wilk test		
P-value	<0,005	
Scott-Knott test		
	Groups	Averages
ILS001	a	1,573085
ILS002	b	1,006496
ILS003	c	0,790826
ILS004	d	0,621496
ILS005	e	0,481167

Figure A.3: ILS+LS boxplot image



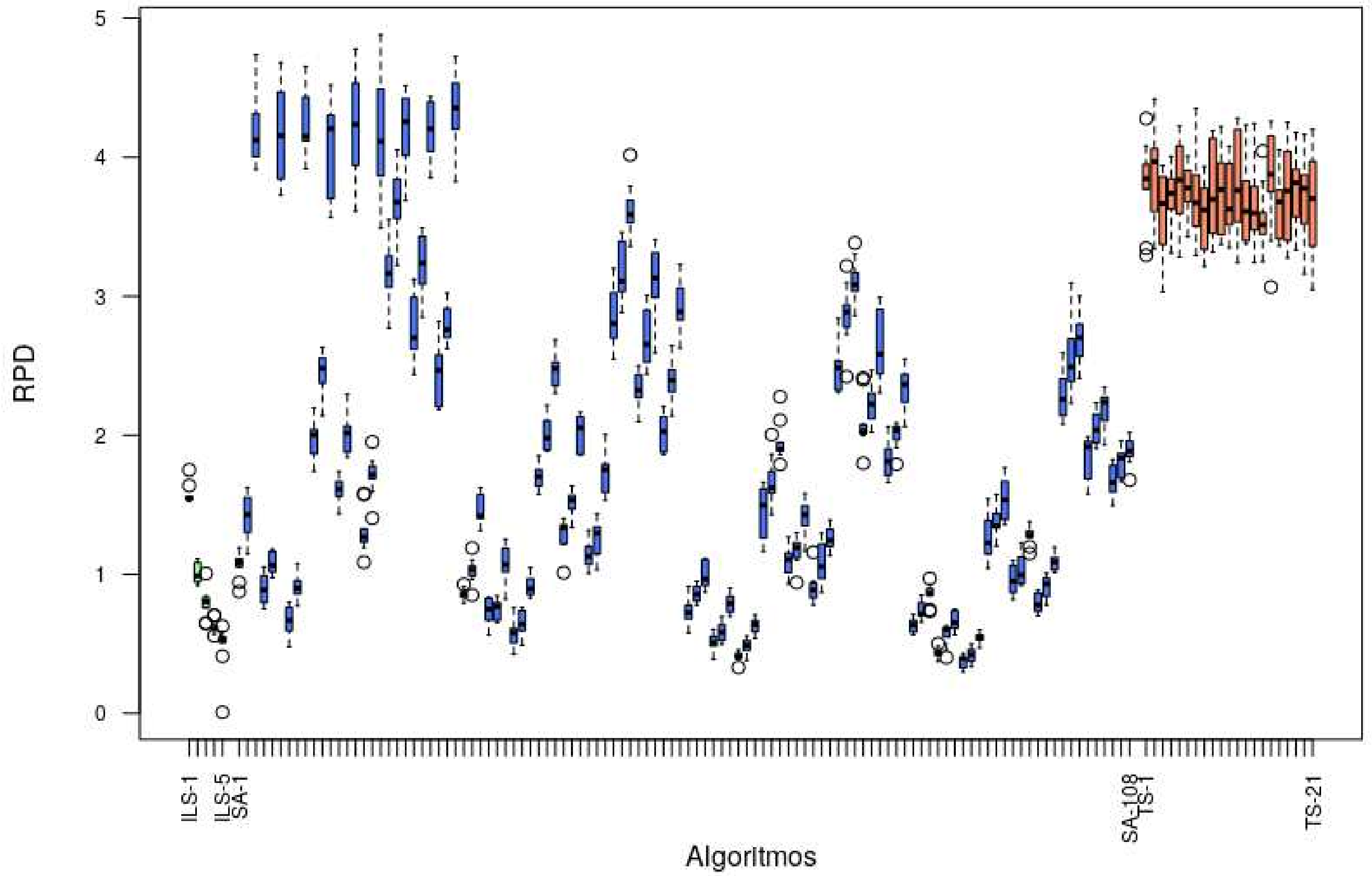


Figure A.4: All algorithms boxplot image

A.4 DVRPMS-ILS parameter calibration

The ILS used in the normal DVRPMS had only one parameter, which was the time in which the algorithm was run. 10 repetition tests were performed on all instances and a graph of the objective functions averages was ploted against time. The parameter time was chosen as 10 seconds since the solutions for greater times did not vary much, as can be seen in Figure A.5.

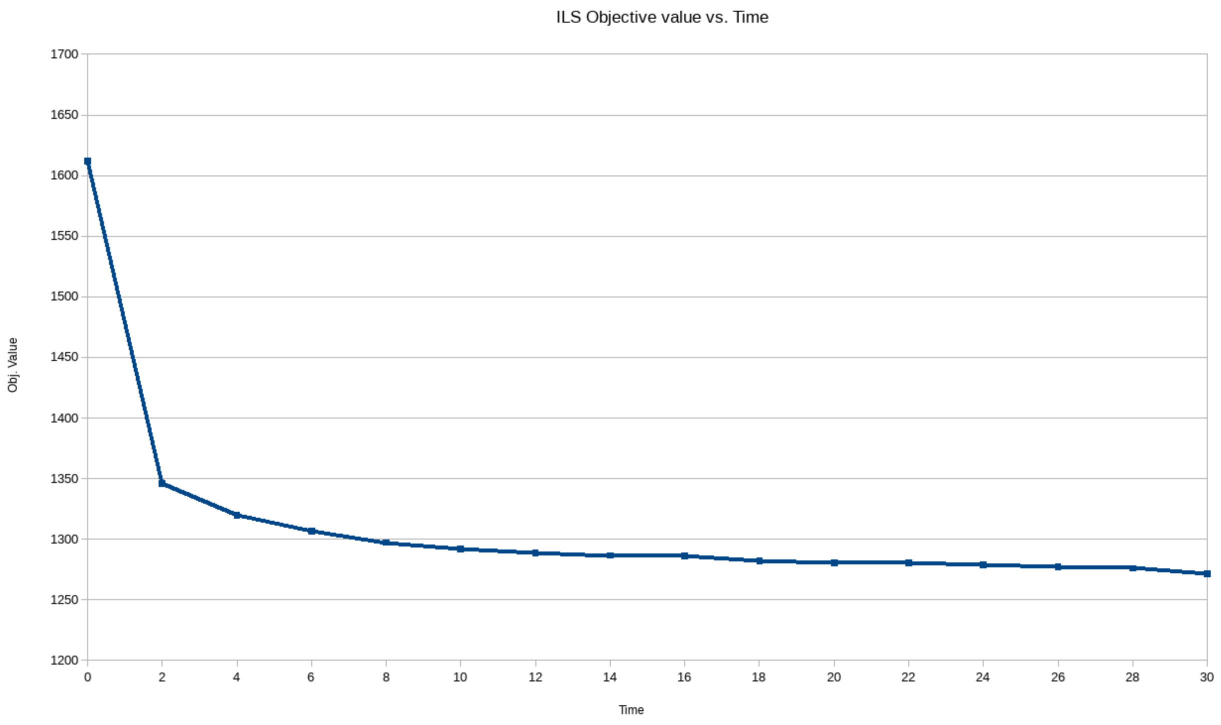


Figure A.5: DVRPMS-ILS graph