Universidade do Minho
Escola de Engenharia

Telmo Miguel Pires Pinto

Models and advanced optimization algorithms for the integrated management of logistics operations

março de 2016

Universidade do Minho
Escola de Engenharia

Telmo Miguel Pires Pinto

Models and advanced optimization algorithms for the integrated management of logistics operations

Tese de Doutoramento em Engenharia Industrial e de Sistemas

Trabalho realizado sob a orientação do
Professor Doutor Cláudio Manuel Martins Alves
e do
Professor Doutor José Manuel Vasconcelos Valério de Carvalho

março de 2016

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, 31st MARCH 2016

Full Name: Telmo Miguel Pires Pinto

Signature: [illegible]

This work was supported by the Portuguese Foundation of Science and Technology (FCT) through the grant SFRH/BD/73584/2010, funded by QREN - POPH - Typology 4.1 - co-funded by the European Social Fund and MEC National Funding.

POPH
QUALIFICAR É CRESCER.

QREN QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL
PORTUGAL 2007.2013

GOVERNO DA REPÚBLICA PORTUGUESA

UNIÃO EUROPEIA
Fundo Social Europeu

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Summary

In this thesis, we propose a set of algorithms regarding real combinatorial optimization problems in the context of transportation of goods. These problems consist in the combination of the vehicle routing problem with the two-dimensional bin-packing problem, which is also known as the vehicle routing problem with two-dimensional loading constraints. We also analyzed two related problems, namely the elementary shortest path and the vehicle routing problem with mixed linehauls and backhauls. In both problems, two-dimensional loading constraints are explicitly considered.

Two column generation based approaches are proposed for the vehicle routing problem with two-dimensional constraints. The first one relies on a branch-and-price algorithm with different branching schemes. A family of dual valid inequalities is also defined, aiming to accelerate the convergence of the algorithm. The second approach is based on a set of different heuristics strategies, which are applied to the reformulated model.

The elementary shortest path problem with two-dimensional constraints is addressed due to its importance in solving the subproblem of the column generation algorithms. To the best of our knowledge, we contribute with the first approach for this problem, through different constructive strategies to achieve feasible solutions, and a variable neighborhood search algorithm in order to search for improved solutions.

In what concerns the vehicle routing problem with mixed linehaul and backhauls and two-dimensional loading constraints, different variable neighborhood search algorithms are proposed. These algorithms explored various neighborhood structures, being some of those developed based on the features of the problem.

All the proposed methods were implemented and experimentally tested. An exhaustive set of computational tests was conducted, using, for this purpose, a large group of benchmark instances. In some cases, a large set of benchmark instances was adapted in order asses the quality of the proposed models. All the obtained results are presented and discussed.

Resumo

Nesta tese, propomos um conjunto de algoritmos sobre problemas reais de otimização combinatória no contexto do transporte de bens. Estes problemas consistem na combinação do problema de planeamento de rotas de veículos com o problema de empacotamento bidimensional, que também é conhecido como o problema de planeamento de rotas de veículos com restrições de carregamento bidimensional. Analisamos também dois problemas relacionados, nomeadamente o problema de caminho mais curto e o problema de planeamento de rotas veículos com entregas e recolhas indiferenciadas. Em ambos os problemas, são explicitamente consideradas restrições de carregamento bidimensional.

Duas abordagens baseadas em geração de colunas são propostas para o problema de planeamento de rotas de veículos com restrições de carregamento bidimensional. O primeiro baseia-se num algoritmo de partição e geração de colunas com diferentes estratégias de partição. Uma família de desigualdades duais válidas é também apresentada, com o objetivo de acelerar a convergência do algoritmo. A segunda abordagem baseia-se num conjunto de diferentes estratégias heurísticas, que são aplicadas ao modelo reformulado.

O problema do caminho mais curto com restrições de carregamento bidimensional é abordado devido à sua importância na resolução do subproblema dos aos algoritmos de geração de colunas. De acordo com o nosso conhecimento, contribuímos com a primeira abordagem para este problema, através de diferentes estratégias construtivas para obter soluções válidas, e um algoritmo de pesquisa em vizinhança variável, com o objetivo de encontrar soluções de melhor qualidade.

No que concerne ao problema de planeamento de rotas de veículos com entregas e recolhas indiferenciadas, diferentes algoritmos de pesquisa em vizinhança variável são propostos. Estes algoritmos exploram várias estruturas de vizinhança, sendo algumas destas desenvolvidas com base nas características do problema.

Todos os métodos propostos foram implementados e testados experimentalmente. Um extenso conjunto de testes computacionais foi efetuado, utilizando um grande grupo de instâncias descritas na literatura. Em alguns casos, um grande conjunto de instâncias descritas na literatura foi adaptado com o objetivo de avaliar a qualidade dos métodos propostos.

Acknowledgments

Writing this thesis was only possible with the help of several people and institutions, to whom I would like to express my deepest gratitude.

First of all, I would like to express my sincere gratitude to my supervisors, Professor Cláudio Alves and Professor Valério de Carvalho.

Professor Cláudio was undoubtedly an essential source of support for me. He was always thoughtful, close and available. He has given me many research opportunities and supported all my initiatives. He always believed in me and my abilities and has always included me in several projects, to which I am deeply thankful. I look to him as an example of rigour, effort, seriousness and enthusiasm, which I will always attempt to follow. His enthusiasm has always strengthened my motivation for continuing to strive to be a good researcher. I really hope he will be proud of my future work. I would like to express my appreciation for having been supervised by him, for every meeting and every conversation we have shared throughout these years. I hope this PhD is only the beginning of several future joint collaborations.

To Professor Valério, I would to thank for all the motivation for the area of operational research. Such motivation has captivated me since when I was his undergraduate student and played an important role in the pursuit of my career. I thank his supervision, availability and empathy throughout the elaboration of this thesis. I would like to express my appreciation of his company in many events and conferences to which he took me, turning our conversions in moments that I will remember with great fondness.

I would like to thank the companionship, help and support of my colleagues Elsa, Nuno, Raid and Rita, throughout the elaboration of this thesis.

To my friends, Alessandra, Bruno, Graça, João, Rui, Sofia, my deepest gratitude for the good humour that they have always brought to my life, even during my absence, and for their encouragement and companionship.

I would like to thank the Portuguese Foundation for Science and Technology, which supported this work through the grant SFRH/BD/73584/2010 (funded by QREN - POPH - Typology 4.1 - co-funded by MEC National Funding and the European Social Fund).

Finally, I would like to thank my family. My parents have always encouraged and helped me in my studies and work, always supporting my choices. To my sister, I thank the constant support and encouragement, always believing in me more than myself.

Braga, March 2016

Telmo Pires Pinto

”Dans la vie, il n’y a pas de solutions. Il y a des forces en marche: il faut les créer, et les solutions suivent.”

Antoine de Saint-Exupéry, *Vol de Nuit*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation and objectives

Vehicle routing plays a major role in the transportation and operations research field. The proposed approaches for this problem are not only relevant for the scientific community, but also for real-world situations since they impact significantly on the final cost of goods and commodities. It has been observed that the part corresponding to transportation represents from 10% to 20% of these final costs [142]. An efficient planning of routes seems very hard to achieve without the adequate analytical and technological tools [136]. The results of applying such tools are very much commented through the literature. In some cases, they may rise up to 20% as referred to in [142].

The economical waste that results from unnecessary or excess travel is also well documented. King and Mast [87] defined the excess travel as the difference between the real travel time and the potential travel time if all the travelled routes are optimal. They conclude that only in United States of America, this excess travel amounts to 7% of all the travel time. From all these observations, it seems clear that there is a true potential for improvements with both economic and environmental benefits [2].

The optimization problem that is behind this thematic is called the vehicle routing problem. It was proposed first in [45] as a generalization of the Travelling Salesman Problem (TSP). Applications of the vehicle routing problem are not restricted to logistics. The conceptual model that underlies this problem finds applications in diverse fields including computer networking and robotics.

Broadly speaking, the vehicle routing problem consists of finding the best set of routes of a fleet of vehicles that must visit a set of customers, taking into account operational constraints and human limitations resulting from the maximum time for the drivers work, or the compulsory breaks for example. Additional constraints can be added to this general problem. One of the most usual constraints that are related in the literature is concerned to the capacity of the vehicles. However, as will be stated below, this type of constraints are not sufficient to reflect the complexity of some real systems.

The objective of this thesis is to develop a set of optimization tools for a family of combinatorial optimization problems that apply to the field of transportation and supply chain management in general. The focus is on routing problems with a concern on the real constraints that apply to these problems. One of the issues that is typically neglected by those that addressed the general vehicle routing problems is related to the loading component of these problems, concerning the shapes of the loads. When the loads are small compared to the size of the vehicles, one may characterize them through a single measure such as weight or volume. Most of the contributions in the literature follow this approach. In these cases, the capacities of the vehicles are represented by a single value. On another hand, when the loads are large compared to the sizes of the vehicles, deciding how to place them in the vehicle becomes a part of the problem. The former approaches may produce infeasible solutions for these problems. Indeed, it may happen that a given load whose weight (for example) does not exceed the capacity of a vehicle does not fit in the vehicle because of its dimensions. The related optimization problem is known in the literature as then vehicle routing problem with loading constraints. This problem integrates two hard combinatorial optimization problems, resulting in a very challenging optimization problem. Indeed, the methods applied to this variant have to consider both the definition of the routes and the packing problem in either two- or three-dimensions.

1.2 Outline

The thesis is organized in 8 chapters, including the present one. In some cases, the applied methods in a given chapter rely on those presented in earlier ones. In the sequel, we describe the following chapters.

In Chapter 2, we provide a description of the problems that will be addressed in this thesis, and the context in which they arise. More precisely, we describe the vehicle routing problem and some of its variants. We give special focus to routing problems with loading constraints, and we completely define the problems that will be tackled in the following chapters. Since the vehicle routing problem with loading constraints result from the combination of routing with packing problems, we also provide a brief description of cutting and packing problems, and we present some variants of these problems dealing explicitly with constraints arising in the routing field.

Chapter 3 provides an exhaustive survey in the field of routing problems with loading constraints. Special emphasis is given to the developed approaches for the capacitated vehicle routing problem with two- or three-dimensional loading constraints. Some real-world applications of these problems are also presented as well as other problems that explicitly consider loading constraints. In the latter case, and among them, very different assumptions are taken into account, due to the different operational constraints arising in each problem.

In Chapter 4, we address the elementary shortest path problem with two-dimensional loading constraints. The objective of this problem is to find the shortest path in a graph, visiting each customer at most once. The cost of the edges may be negative and the demand of each node is composed by two-dimensional items. The overall approach relies on different constructive heuristics to generate feasible solutions, and on a variable neighborhood search algorithm to improve the former solutions. This approach can be seen as the first step to build a column generation algorithm, since the referred to above problem corresponds to the subproblem of the capacitated vehicle routing problem with two-dimensional loading constraints. To the best of our knowledge, this approach is the first one tackling this problem.

In Chapter 5, we present a branch-and-price algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. There are several column generation based approaches for many variants of the vehicle routing problem. In contrast, approaches for the variant with loading constraints through column generation are not quite explored. We apply the Dantzig-Wolfe decomposition to the original problem, resulting in a reformulated model and in a pricing subproblem which is solved using the approach described in Chapter 4. A family of dual inequalities is used, aiming to accelerate the convergence of the branch-and-price approach.

Chapter 6 addresses the same problem as in Chapter 5. We make use of the reformulated model obtained in that chapter in order to develop a set of heuristic approaches. These heuristics work in the formulation of the master problem, by iteratively selecting a feasible route, using either the solution provided by linear programming relaxation of the reformulated model, or the solution provided by mixed integer programming models after enforcing integrality in some variables.

In Chapter 7, we consider a pickup and delivery problem with two-dimensional loading constraints. More precisely, we address the capacitated vehicle routing problem with loading constraints and mixed linehauls and backhauls. The main difference between this problem and the one addressed both in chapters 5 and 6 is that customers are divided into two different sets: the ones which require a given demand, and the ones which provide a given supply. The demands can only be provided by the depot, which is also the only destination of items provided by backhaul customers. Both the demand and the supply are composed by two-dimensional items. We suggest an insertion heuristic for generating feasible solutions, and three different variants of the variable neighborhood search algorithm in order to search for improved solutions.

Finally, in Chapter 8, we draw some conclusions, and we present the main contributions provided by the work presented in this thesis. Additionally, we present some future research directions.

Chapter 2

Vehicle routing problems with loading constraints

Contents

2.1 The vehicle routing problem

The Vehicle Routing Problem (VRP) is one of the most well-known and studied combinatorial optimization problems. The first work related to VRP was published in 1959 [45] by Dantzig and Ramser. In their work, the authors presented a generalization of the Travelling Salesman Problem applied in the context of gasoline distribution. The authors considered both the return to a so-called terminal point (which is nowadays known as depot) and the demand associated to each point, except for the terminal point. The total demand is greater than the capacity of the vehicle and thus it is not possible to satisfy the demand of each point in a single visit. The authors called this problem as the truck dispatching problem. Since then, a huge number of contributions in this field appeared in the literature, reporting a wide diversity of variants and extensions, resulting in an increased closeness between the proposed methods and the real context in which these problems arise. This feature can justify the success of the VRP [106]. In fact, the application context of VRP is huge and it is not confined to the distribution of goods, since many contributions arise in different situations as the public transport, the waste and kerbside collection, or less frequently in robotic networks [122] and in agriculture [19, 18].

The VRP is a combinatorial optimization problem which consists in determining a set of routes, beginning and ending at the depot or depots, to be covered by a fleet of vehicles which must serve a set of customers, satisfying all the operational and human constraints, while taking into account a given objective. The most common objective is to minimize the involved transportation cost. However, and as stressed in [142], other objectives can be considered as the minimization of the used vehicles, minimization of number of drivers, minimization of the penalties associated to each incomplete demand, or the maximization of the balance of the load or travel time of the different routes. Additionally, more than one objective may be considered, by combining the mentioned objectives or others. Some exhaustive surveys and annotated bibliographies in the context of VRP can be found in [89, 143, 106, 67]. In the next subsections, we review some variants of the VRP.

2.1.1 Capacity constraints

The most usual and basic constraints treated in the literature have been with no doubt related to the capacities of the vehicles. This capacity is usually expressed in terms of maximum weight, area, volume or units (pallets or stacks) of items which the vehicle can carry. The corresponding problem is known as Capacitated Vehicle Routing Problem (CVRP). In the CVRP, a single depot is considered as the source of a set of K identical vehicles and their returning point. One customer is visited exactly once and its demand is satisfied only from the depot where the items are loaded. Therefore, it is not possible to split the demand. The objective of the CVRP is to serve all customers within a single visit using exactly K vehicles without exceeding their capacity, while minimizing the involved transportation cost.

2.1.2 Type of service

In some problems, serving one customer can be performed in more than a single visit. As a consequence, the demand of one or more customers may exceed the capacity of the vehicle. Such problems are called Split Delivery Vehicle Routing Problem (SDVRP). The SDVRP is distinct from the problems in which customers are visited with a certain frequency. Those situations require to have a plan of routes to be performed within a given period. Such problems are known as Period Vehicle Routing Problem (PVRP). A review on PVRP and real applications can be found in [60]. Other well-known variant require to visit each customer within a specific time interval. These problems are known as Vehicle Routing Problems with Time Windows (VRPTW). Usually, this time interval is expressed in units for both the earliest and the latest time to serve a given customer, assuming that the vehicle leaves the depot at the instant zero. Additionally, each arc has an associated travel time. The amount of time required to serve a given customer is quantified and it relies on the customer to be served. This service may start within this interval, even though that earlier arrivals are allowed. In these cases the vehicle must wait the earliest time to serve such customer. Serving one customer may include delivering demands, collecting orders or both. This kind of variants belong to the so-called Pickup and Deliver Problems (PDP) which will be analyzed in depth in Section 2.2.

2.1.3 Operational constraints

In some real contexts, and in opposition to the CVRP, the available fleet is composed by a set of vehicles with different capacities. The corresponding problems are known as Heterogeneous Fleet VRP (HFVRP). The diversity of vehicles is sometimes necessary due to difficult access areas or restricting traffic rules to certain type of vehicles in the customer surrounding area. A review on HFVRP, its variants and on approaches to tackle this problem can be found in [7]. Another real and well-known variant combines the routing aspects with inventory management. These problems are known as Inventory Routing Problems (IRP) [34, 14]. The IRP consists in the stock management of both customers and supplier in order to remain the store-level under the maximum capacity avoiding stock failures, using capacitated vehicles to serve customers from the supplier. Therefore, two types of costs should be taken into account in the IRP context: the transportation/routing costs and the holding costs at customers and at the supplier.

2.1.4 Other assumptions

In some variants of the VRP, other assumptions can be taken into account due to the multiplicity of aspects in real situations. For instance, one may assume that one vehicle can perform more than one route (Multi-Trip VRP) or more than one depot is available to supply the set of customers (Multi-Depot VRP). Other problems consider the uncertainty given to the problem data, usually the demand of customers or travel times. The variants of VRP that deal with these stochastic information are known as Stochastic Vehicle Routing Problem (SVRP). Some efforts have been made to deal with VRP in which information may only be known and updated during routing operations in real time. Such problems are known as Dynamic Vehicle Routing Problems (DVRP) [123].

The variants defined above assume representations where customers are points where the service may occur, and the routing paths used by vehicles are edges (arcs) crossing two customers or one customer and the depot. In some real situations, such as kerbside collection or the street sweeping, it is suitable to assume that the service is performed along the edges instead of considering that such service is performed in nodes. Such problems are known as Capacitated Arc Routing Problems. In [152] the authors present a review with special emphasis on the decade 1998-2008.

2.2 Pickup and delivery problems

In the last decades, Pickup and Delivery Problems (PDP) deserved attention by the scientific community in the transportation field. The PDP can be briefly defined by the routing problems where goods or even people must be collected and distributed from an origin to a destination. Due to the extensive number of approaches and denominations in this field, Parragh *et al.* [120, 121] suggested a classification where the general Pickup and Delivery Problem is divided in two classes. The first class considers the problems in which delivered goods (or, respectively, the picked goods) are only loaded (or, respectively, unloaded) at one or more depots, meaning that the transport of goods is only performed from or to the depot. The second class considers the transportation of goods from pickup customers to delivery customers. The problems of the first class are usually known as Vehicle Routing Problems with Backhauls (VRPB) whereas the ones belonging to the second class are known as Vehicle Routing Problems with Pickups and Deliveries (VRPPD).

The first class (VRPB), is divided in four subclasses. In the first subclass, customers are assigned to one and only one of two disjoint clusters: the one for the delivery customers (also known as linehaul customers) and another one for the pickup customers (also known as backhaul customers). In this subclass, in each route of the solution backhaul customers can only be visited after visiting all linehaul customers assigned to that route. The problems belonging to this subclass are named as Vehicle Routing Problem with Clustered Backhauls (VRPCB). In the second subclass, this constraint does not apply, even though the division into disjoint sets of delivery and pickup customers is still mandatory. Since customers of both clusters can be indistinctly visited without that precedence constraint, the authors referred to the problems of this subclass as Vehicle Routing Problems with Mixed Linehauls and Backhauls (VRPMB). The last two subclasses consider the situations where customers require both a quantity to be delivered and a quantity to be picked up. However, in the third subclass, a given customer can be visited once or twice. If it is visited twice, the visits can be divided in one visit to deliver goods and another one to pick up goods. The problems of this subclass are known as Vehicle Routing Problems with Divisible Delivery and Pickup (VRPDDP). On the contrary, in the fourth subclass, both loading and unloading operations must be performed simultaneously in exactly one visit, and then the authors called these problems as Vehicle Routing Problems with Simultaneous Delivery and Pickup (VRPSDP).

The second class of this classification (VRPPD) can be divided in two subclasses based on the relation between pickup and delivery customers. The first VRPPD subclass refers to the situations in which collected goods from any pickup customer are available to satisfy the demand of any delivery customer, and so the linehaul customers and backhaul customers are unpaired. The problems of this subclass are denoted by Pickup and Delivery Vehicle Routing Problems. On the contrary, the second subclass refers to the cases in which the transported goods or people have a specific origin and a specific destination (paired origins and destinations vertices). This subclass includes the Pickup and Delivery Problem (PDP) and the Dial-A-Ride Problem (DARP). It is worth noting that each class of this classification includes the respective version for the single vehicle cases.

Another comprehensive survey and classification for the static pickup and delivery problems is presented in [12]. The static denomination is due to the fact that all information about the problem is deterministic and known before the construction of the solutions. In this classification, the scheme $[x|y|z]$ is used, where x defines the relation between the origins and the destinations:

***many-to-many* (M-M):** if any customer or the depot can be the origin or the destination of goods;

***one-to-many-to-one* (1-M-1):** if the source of delivered goods and the destination of picked goods is only the depot;

***one-to-one* (1-1):** if each good has to be transported from a single origin to a single destination.

The y field defines how the loading operations must be performed. If one customer is visited exactly once and loading and unloading operations must be performed simultaneously, the authors used the denomination "PD". On the contrary, if this requirement does not apply, these operations can be combined or not ("P-D"). If at each customer it is only possible to deliver or to collect, then the notation "P/D" is used instead. It is also in the y field that problems with transshipments are defined by associating the letter "T". The last field of this classification scheme, z, corresponds to the number of used vehicles. For the cases in which it is not possible to represent a field, the authors use the notation "-".

As in VRP, in some pickup and deliver contexts the information is dynamic meaning that it is not known in advance or it is updated when performing the routes. These problems are known as dynamic pickup and delivery problems. In [13], a comprehensive survey on these problems is presented.

2.3 Routing with loading constraints

In the context of the CVRP, the majority of the approaches in the literature assume that the capacity of the vehicle is a simple one-dimensional measure (typically a maximum weight or volume). In many real applications, this approach may not be adequate. That happens for example when the vehicles travel with their maximum load. In these cases, distributing the loads among the available space may be a real issue due to the loading constraints, which take into account the shape of the loads. Therefore, it can be possible to satisfy the weight capacity while it can be impossible to reach a feasible layout for all the loads.

In this sense, and in order to incorporate the loading specificities, some works were presented in the last decade, by integrating the multi-dimensional bin packing problem with the CVRP. The resulting problem is described in the literature as CVRP with loading constraints (L-CVRP). The L-CVRP is NP-hard since it integrates two NP-hard combinatorial problems, resulting in a very challenging optimization problem.

Indeed, the methods applied to this variant have to consider both the definition of the routes and the loading of the items in the vehicle in either two- (2L-CVRP) or three-dimensions (3L-CVRP).

The loading component of the L-CVRP is treated as a multi-dimensional packing problem such as Bin Packing Problem, Orthogonal Packing Problem or Strip Packing Problem. For the three-dimensional case, the loading component can be treated as a Container Loading Problem.

2.3.1 Definition of the 2L-CVRP

In the 2L-CVRP, there is a homogeneous fleet with a two-dimensional rectangular loading surface. The demand of each customer is composed by a finite number of two-dimensional rectangular items and by the weight of all items. The 2L-CVRP consists in finding a set of routes starting and ending at the depot which minimize the total travel cost, satisfying the following constraints:

(C1) The number of routes in the solution cannot be greater than the fleet size;

(C2) All customers must be served in a single visit, *i.e.*, the demand of each customer cannot be split;

(C3) A feasible orthogonal loading is required for each used vehicle, *i.e.*, the items must be completely within the surface, must not overlap, and the edges of the items must be parallel to the surface edges;

Some additional issues can be considered in the 2L-CVRP, such as the weight capacity constraints, demands composed by circular items, pickup and delivery scenarios, among others.

2.3.1.1 Sequential constraints

An important issue that must be taken into account in the L-CVRP is the relative position of the items in the vehicles, and its relation with the sequence of visits that the vehicle has to do. In practice, it is important that the delivery of an item to a customer does not require moving the other items for customers which are served later. Therefore, the items must be placed inside the vehicle according to the order by which the customers will be visited.

Formally, the sequential constraints impose that unloading an item in the customer must be performed in a straight movement, through a free passage towards the door, and whose width is greater than or equal to the width of the item. This means that no lateral movements are allowed for the items to be delivered to other customers to unblock the passage of the items of the mentioned customer. During the unloading operation, the item must preserve its edges parallel to the edges of the surface. Note that there are no sequential constraints between items for the same customer.

Figure 2.1 depicts this situation, where $I_{i,j}$ represents the item j of customer i: unloading items for customer 1 can be done by a straight move for each item. However, in (a), when unloading items for customer 2, item $I_{3,1}$ blocks the passage of item $I_{2,1}$. Therefore, in (a), the loading is sequential infeasible. In (b), the sequential constraints are satisfied, since items belonging to the same customer can be unloaded in a straight movement without rearranging items to be delivered.

The advantage of these constraints in the real context is clear. Unloading items is less time consuming since only a single movement is necessary, without rearranging items nor unloading and loading items from other customers.

The problems that consider explicitly these constraints are known as sequential 2L-CVRP. For these problems which do not include these constraints, the designation unrestricted 2L-CVRP is used instead. Some authors called these constraints as rear loading constraints or LIFO (last in, first out) constraints. This later denomination is due to the fact that if the items are loaded into the vehicle in the opposite way in which they are unloaded, the sequential constraints are always satisfied.

2.3.1.2 Orientation constraints

The items may have a fixed or variable orientation. Usually, these constraints are related to the way the vehicles are loaded. Indeed, they can be either rear-loaded, side-loaded or loaded in both ways. In this latter case, the maximum size of the door may only block an item for a given orientation.

If items have a fixed orientation, they must not be rotated in the surface. These

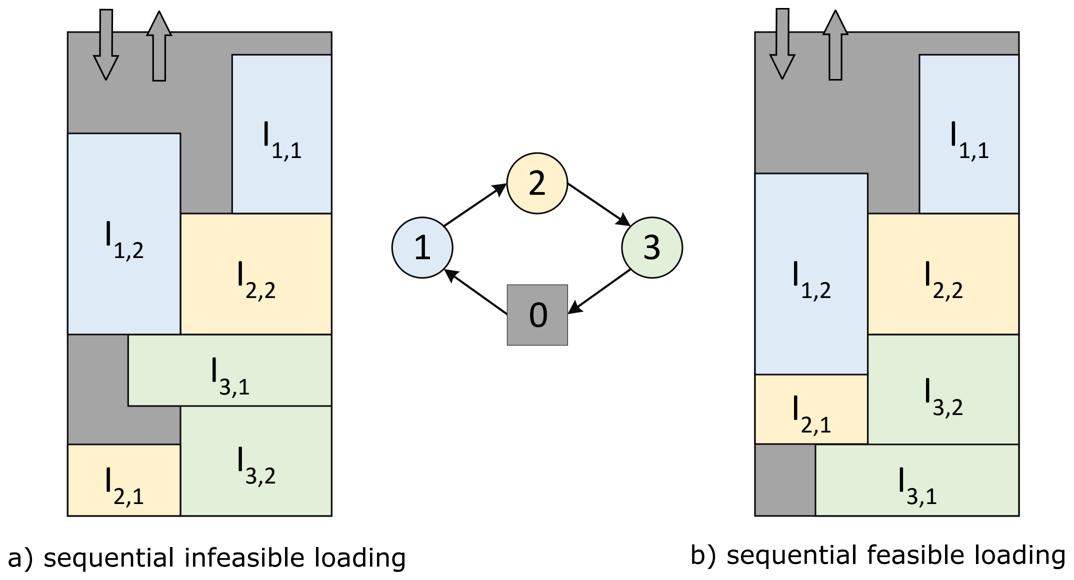
I1,1
I1,2
I2,2
I3,1
I3,2
I2,1
2
1
3
0
I1,1
I1,2
I2,2
I2,1
I3,2
I3,1
a) sequential infeasible loading
b) sequential feasible loading

Figure 2.1: Sequential constraints in the 2L-CVRP context

cases happen in some real situations, where the weight, the size or the nature of cargo operations may preclude the rotation of items.

The problems in which this constraint arises are known as oriented problems. In contrast, in the rotated problems, the items are allowed to be rotated in the surface, usually by 90 degrees. These constraints are known as horizontal constraints, since an item can only be rotated in the horizontal plane.

2.3.2 Definition of the 3L-CVRP

The 3L-CVRP is a natural extension of 2L-CVRP to the three-dimensional axis. The routing aspects of the problem are preserved. However, each vehicle of the fleet must be seen as a three-dimensional rectangular loading container. Additionally, the demand of each customer is composed by three-dimensional rectangular items (boxes). Therefore, the 3L-CVRP calls for the determination of a set of optimal routes starting and ending at the depot which minimize the total travel cost while satisfying the constraints (C1) and (C2) presented in Section 2.3.1, and:

(C3') A feasible orthogonal loading is required for each used vehicle, *i.e.*, the items must be completely within the container, must not overlap, and the edges of the items must be parallel to the container edges.

Some additional constraints can be applied due to the dimension of the problem. We explore these constraints in next subsections.

2.3.2.1 Multi-drop and sequential constraints

Some authors addressed the 3L-CVRP with multi-drop constraints. These constraints were defined by Bischoff and Ratcliff [17] for scenarios in which subsets of items have different destinations. In these cases, unloading operations can be favoured if items of the same subset are close together and if the arrangement of the subsets is related as much as possible with the sequence of delivery of such subsets.

Some works deal with the multi-drop constraints by defining a maximum distance which can be reached over the existing layout in order to rearrange items for a given customer served earlier [82, 83]. This strategy can take advantage of empty spaces left during loading operations.

The sequential constraints referred to above in Section 2.3.1.1 can be seen as a specific multi-drop condition [24]. We recall that sequential constraints ensure that unloading one item from a given customer does not require to move other items to be delivered to other customers. In the context of the 3L-CVRP, it is not sufficient to have a free passage between items and the rear side of the vehicle. It is also required that no items from other customers are on top of those being unloaded, even if they are not in contact. In Figure 2.2, two examples are presented, assuming the route *0-1-2-3-0*. The layout (a) is sequential infeasible since item $I_{2,1}$ is blocking item $I_{1,1}$, and in addition, item $I_{2,2}$ is on top of item $I_{1,1}$. A possible feasible solution is represented in (b).

2.3.2.2 Load stability

The load stability is a classical issue in the Container Loading Problem. This issue aims to prevent significant moves of the items during the transport or during the loading and unloading operations [17]. The importance of the stability is twofold since it avoids the damage in the items and prevents personnel injuries [24].

Bortfedlt and Wäscher [24] presented the difference between the vertical and horizontal stability. The former requires that the loading pattern is not modified by the

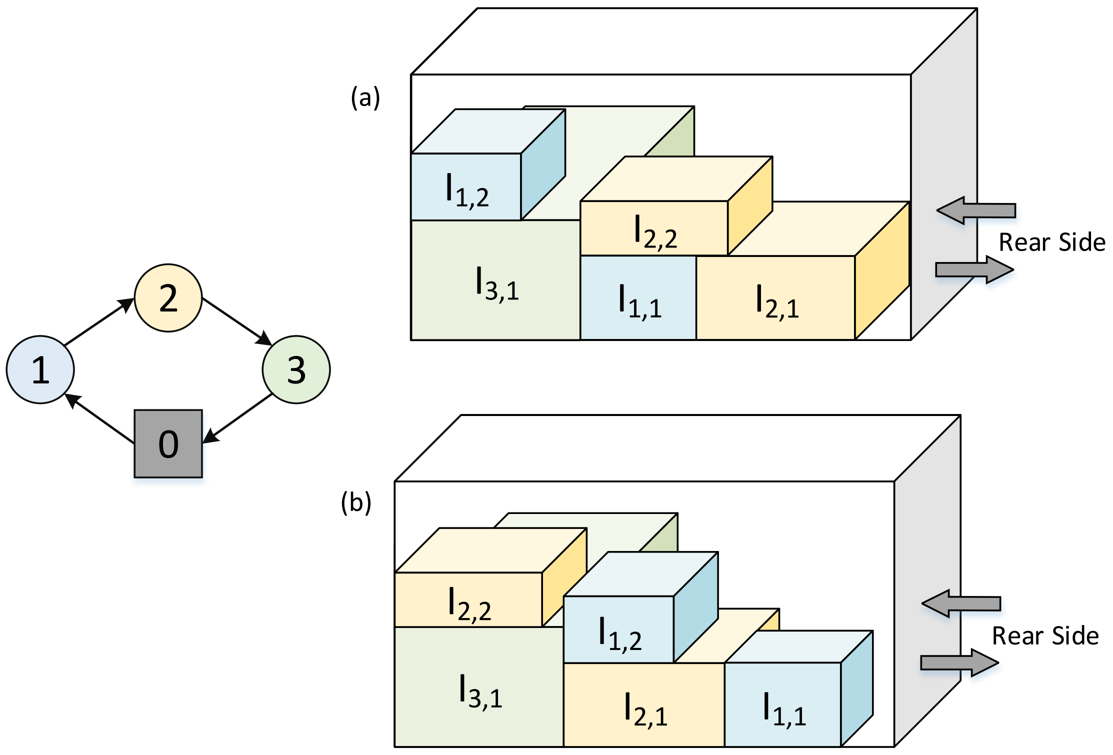

Figure 2.2: Sequential constraints in the 3L-CVRP context

fall of items through the action of the gravity force only, and thus, considering that the vehicle is not moving. Therefore, some authors called this issue as static stability. In contrast, the horizontal or dynamic stability considers that items may not be shifted in the horizontal plane when the vehicle is moving.

In the context of the 3L-CVRP, the vertical stability of the cargo inside the vehicle is always considered and it is ensured by the so-called supporting constraints. These constraints guarantee that the bottom edge of the box must be completely or partially supported by the other items or by the floor of the container. For the cases in which the full support is not mandatory but the supporting constraints are required, at least a given percentage of the area of the bottom face of the box must be supported by other items or by the floor of the container. For both full and partial support cases, if the item is placed on the floor, the supporting constraints are obviously satisfied.

The horizontal stability in the 3L-CVRP is not as common as the vertical stability. However, some works consider that a given percentage of the lateral sides of the item must be in contact with other items or with the walls of the container.

2.3.2.3 Orientation

In Section 2.3.1.2, horizontal orientation constraints were introduced in the context of the 2L-CVRP. In the 3L-CVRP, both horizontal and vertical orientation can be considered. The horizontal constraints refer to the rotation of items in the loading area of the vehicle.

In the vast majority of approaches, items have a fixed vertical orientation. This constraint arises in contexts in which items have to remain stand up straight and cannot be placed upside-down. In this sense, vertical orientation constraints are useful in order to preserve both the integrity of the items and the loading stability, since the strength of the box can be different when the item is turned upside down [24].

2.3.2.4 Load bearing and fragility

By definition, a three-dimensional loading pattern is usually composed of items on top of each other. However, stacking items can damage the items that are underneath, due to their physical strength. Indeed, a given box has a limit of pressure which can support, and from which the integrity of both the items and the layout is not guaranteed. In order to avoid this situation, load bearing constraints can be considered. These constraints are also inherited from the Container Loading Problem and can be defined as the maximum pressure or weight that an item can support. This weight or pressure is due to the item or items that are on top of the considered item, and it is measured in units of force per area unit. Usually, for a sake of simplicity, this parameter corresponds to the pressure that can be applied at any point of the top face of the item.

In the literature, the load bearing constraints are also known as stacking constraints. However, in many approaches, when one wants to place items on top of each other, it is usual to consider the so-called fragility constraints.

The fragility of items can be seen as a particular case of load bearing, in which the items are flagged as fragile or non-fragile. The non-fragile items can only be placed on top of items which are also non-fragile, while fragile items can be placed on top of fragile or non-fragile items.

It is important to note that in some scenarios of the 3L-CVRP, it may happen that all items of a given order cannot be placed over each other. These situations are usual when, for instance, the transported items are refrigerators. In this case, the problem reduces to a 2L-CVRP.

2.3.3 Classification

Fuellerer *et al.* [61] proposed a problem classification according to the loading aspects for the Capacitated Vehicle Routing Problem with loading constraints. This typology consists in the three-field classification scheme $x|yz|$L where x represents the dimension of loading constraints (two- or three-dimensional), y represents the denomination resulting from the consideration or not of sequential constraints (S for *Sequential* or U for *Unrestricted*, respectively) and z represents the orientation aspects of the problem (O for fixed orientation and R for rotated problems). Using this scheme, eight possible classification can be distinguished for the 2L-CVRP and 3l-CVRP. The resulting problems are presented in table 2.1.

Table 2.1: L-CVRP classification according to Fuellerer *et al.* [61]

Dimension	Sequential Constraints	Orientation Constraints	
		Oriented	*Rotated*
Two	*Sequential*	2|SO|L	2|SR|L
	Unrestricted	2|UO|L	2|UR|L
Three	*Sequential*	3|SO|L	3|SR|L
	Unrestricted	3|UO|L	3|UR|L

2.3.4 A modified version - M3L-CVRP

Some works consider a different definition of the sequential constraints. Consequently, such modification gives rise to a new version of the 3L-CVRP which is called M3L-CVRP.

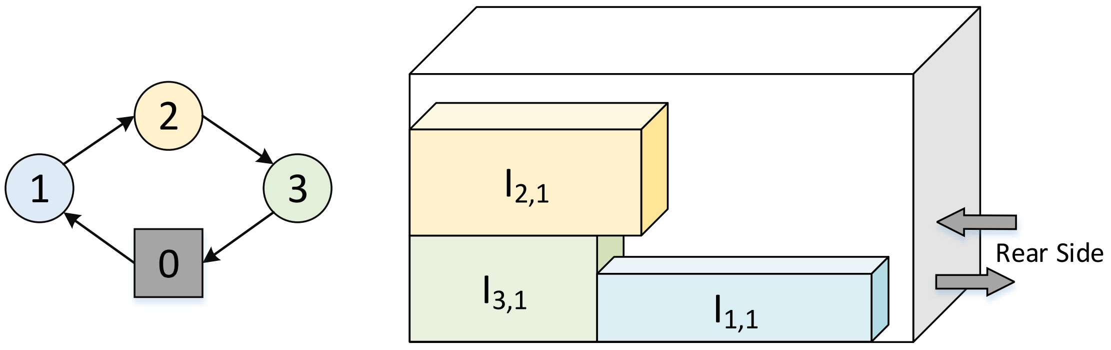

Figure 2.3: Side view of the vehicle (adapted from [141])

This version takes into account a less restrictive LIFO policy, in which items for customers to be served before can be under the items to be served later and have overlapping projections in the horizontal plane, but their edges cannot be in contact since, in detail, items to be served before may not support items to be served later. In contrast, this assumption is forbidden in the 3L-CVRP even if items to be served before are not supporting items to be served later. An example of a valid layout is presented in Figure 2.3: the layout is clearly infeasible for the 3L-CVRP, but it is feasible for the M3L-CVRP.

The first M3L-CVRP approach was suggested in [141] and it was also adopted in [158, 28]. The authors claim that the M3L-CVRP is suitable for manual loading operations in contrast with 3L-CVRP. Indeed, as can be seen in Figure 2.3, it is clear that item $I_{1,1}$ can be manually unloaded without rearranging items since this item is not supporting item $I_{2,1}$. However, if it is required to carry up items in a lifter, item $I_{1,1}$ must not be unloaded before item $I_{2,1}$.

2.3.5 Elementary shortest path problem with loading constraints

Iori and Martello [77] considered the use of column generation techniques as an open perspective to tackle routing problem with loading constraints.

Several works considered the application of column generation to the CVRP, and presented the application of the Dantzig-Wolfe decomposition to a CVRP. As a result, one can obtain a restricted master problem (RMP) and a pricing subproblem which is a shortest path problem with resource constraints. These resource constraints are directly related with the capacity of vehicle for the CVRP. However, in the classical CVRP, the customer must be visited once. Since the solution of the pricing subproblem must be a valid solution of the CVRP, the obtained shortest path must be elementary.

The Elementary Shortest Path Problem with Resource Constraints (ESPPRC) is NP-hard and it can be effective in the resolution of the pricing problem of a column generation algorithm for VRP [57, 131, 133]. In this sense, it is possible to apply a Dantzig-Wolfe decomposition to the 2L-CVRP and then obtain an Elementary Shortest Path Problem with two-dimensional Loading Constraints (2L-ESPP). To the best of our knowledge, there is no approaches for the 2L-ESPP.

2.3.6 2L-CVRP with mixed linehauls and backhauls

As referred to in Section 2.2, the Vehicle Routing Problem with Mixed Linehauls and Backhauls (VRPMB) is a subclass of problems belonging to the Vehicle Routing Problem with Backhauls (VRPB), in which the customers are divided in two different groups: linehaul and backhaul customers. Each linehaul customer has a given demand quantity that can only be provided by the depot, while each backhaul customer has a given supply quantity which the only destination is the depot. In the VRPMB, both the linehaul and backhaul customers can be indistinctly visited, which means that within the same route, it is not required to visit all linehauls customers before visiting backhauls customers.

The Capacitated Vehicle Routing Problem with Mixed Linehauls and Backhauls with two-dimensional Loading constraints (2L-CVRPMB) is an extension of the 2L-CVRP where each linehaul customer (or, respectively, each backhaul customer) has a demand (or, respectively, a supply) composed by two-dimensional rectangular items. The 2L-CVRPMB consists in finding the set of optimal routes, each one starting and ending at the depot and satisfying the following constraints:

(C1) The number of used vehicles cannot be greater than the fleet size;

(C2) Each customer is visited exactly once;

(C3) The weight of items in each vehicle and in each arc of the solution cannot exceed the weight capacity;

(C4) Items have a fixed orientation in the loading area, *i.e.*, items cannot be rotated;

(C5) In each vehicle, the items must be completely within the surface, must not overlap, and the edges of each items must be parallel to the loading area edges;

(C6) Unloading an item at a given customer must be performed in a straight movement, without moving items of other customers, and thus the loaded items from backhaul customers cannot block items to be delivered.

2.4 Cutting and packing problems

Cutting and packing problems are combinatorial optimization problems which arise in several real-world contexts. Examples of cutting and packing problems can be found in a wide number of situations in the industry like cutting of large rolls in smaller ones, packing boxes into a vehicle or, less intuitively, assigning personal shifts to an organizational schedule.

In the standard problem, two sets are considered: a set of small items and a set of large objects. The set of small items must be assigned to the set of large objects, without overlapping, in order to optimize a given criterion, while satisfying the measurements of the large objects.

Due to the huge number of works, extensions and variants of the standard problem using different classifications, Dyckhoff [55] proposed a typology of cutting and packing problems, in order to categorise these problems. This classification is based in four criteria:

1. Dimensionality

   (1) one-dimensional

- **(2)** two-dimensional
- **(3)** three-dimensional
- **(N)** N-dimensional ($N > 3$)

2. Kind of assignment
   - **(B)** all large objects and a selection of small items
   - **(V)** a selection of large objects and all small items
3. Assortment of large objects
   - **(O)** one large object
   - **(I)** many identical large objects
   - **(D)** several different large objects
4. Assortment of small items
   - **(F)** few small items of different figures
   - **(M)** many small items of many different figures
   - **(R)** many small items of relatively few different figures
   - **(C)** many identical small items

Later, Wäscher *et al.* [147] mentioned that despite the relevance of the Dyckhoff's typology, some limitations were found. The ambiguity arises when a problem may belong to different categories. Furthermore, these categories are not always homogeneous. Due to these considerations, Wäscher *et al.* proposed a so-called improved typology for cutting and packing problems [147] based on the Dyckhoff's typology. In order to ensure the quality of their classification, this improved typology was applied to 445 problems from 413 works published between 1995 and 2004.

The typology suggested by Wäscher *et al.* consists in 5 main criteria. The first one is the dimensionality, which is similar to the one proposed Dyckhoff's typology, but the problems with more than 3 dimensions are considered as variants. The kind of assignment is also similar to the first typology presented above, and it considers two different situations: the output maximization and the input minimization. The former considers the case where a subset of small items has to be assigned to a given set of large objects, while the latter considers that all small items are assigned to a subset of large objects. The assortment of small items considers three cases: identical small items, weakly heterogeneous assortment of small items or a strongly heterogeneous assortment of small items. The assortment of large objects considers one large object or several objects. Finally, the last criterion is the shape of the small items which can be regular or irregular. Clearly, this criterion is only considered for the two- or three-dimensional problems.

The application of two criteria, namely, the kind of assignment and the assortment of small items, results in the definition of six basic problem types. For each of those problems, the application of the criterion assortment of large objects results in fourteen intermediate problem types. Again, the application of the two criteria, namely, the dimensionality and the shape of small items, results in refined problem types.

Wäscher *et al.* [147] stressed that from the general definition of cutting and packing problems, there are several variants and extended problems. The variants are related to the assumptions of the problem which are different, while the extended problems consider additional features not related to the cutting or packing. For instance, a variant arise when a large object is composed of a heterogeneous material, whereas a problem which deals with sequencing of the patterns performs a extended problem.

2.4.1 Packing problems as a L-CVRP subproblem

The loading part of the L-CVRP is in fact a subproblem of the L-CVRP. To tackle this subproblem, some authors proposed approaches for different problems as the Bin Packing Problem (BPP), Strip Packing Problem (SPP), or the Orthogonal Packing Problem (OPP), in which unloading constraints must be considered. The BPP consists in packing a set of small and rectangular items into a set of large items (known as bins) so as to minimize the number of used bins. On the other hand, the SPP consists in packing a set of small and rectangular items into a strip of finite width and virtual infinite height. The OPP consists in finding if a set of small items fits on a large item.

Some works in the literature consider packing problems with sequential constraints, which can be seen as a problem that forms part of the L-CVRP. To tackle this type of problems, some contributions are proposed in the literature for the strip packing problem, the orthogonal packing problem or the knapsack problem, in which unloading constraints must be considered. In these problems and in contrast to the L-CVRP, the sequence in which items must be unloaded is known in advance. The first approximation approach for the strip packing problem with unloading constraints is presented in [44], the authors present a 5.745-approximation algorithm which starts to pack items into bins using a level bin packing method, according to the opposite order in which the items will be unloaded, defined as a class of the item. Afterwards each bin is concatenated forming a feasible packing solution. In [43], a similar approach to [44] was performed for the rotated version of the problem, in a 6.75-approximation algorithm. Additionally, the authors proposed an adaptation of the so-called first-fit decreasing height algorithm which is applied to the subset of items belonging to a same class, and in which no rotations are allowed. The authors proved that this approach is a 1.75 asymptotic approximation algorithm if the number of class is bounded by a constant. Finally, the authors proposed a greedy randomized adaptive search algorithm which is based on [1] with modifications in order to consider the unloading constraints.

Chapter 3

State of the art

Contents

3.1 Introduction

In this chapter a survey in routing problems with loading constraints is provided. Special focus is given to the capacitated vehicle routing problems with two- or three-dimensional loading constraints, denoted hereafter by 2L-CVRP and 3L-CVRP, respectively. Approaches for the 2L-CVRP and 3L-CVRP are reviewed in Section 3.2 and in Section 3.3, respectively. Both sections are divided by either the adopted methods, or the additional constraints arising in the L-CVRP context. The explicit consideration of loading constraints is not confined to the capacitated vehicle routing field and, in this sense, other routing problems with loading constraints are presented in Section 3.5.

3.2 Approaches for the 2L-CVRP

The Capacitated Vehicle Routing Problem with Two-dimensional Loading constraints (2L-CVRP) was formally presented in Chapter 2. Since it is a NP-hard problem, the vast majority of contributions presented in the literature are based in heuristics. Few works tackled the 2L-CVRP by using exact methods. In Section 3.2.1, these methods are reviewed and an integer programming formulation presented in the literature is described. The remaining works suggested methods that are based in heuristics either for the routing part, or for the packing component. Among these, meta-heuristics are more common, some with nature-inspired algorithms. The inclusion of additional constraints in the 2L-CVRP, such as partial conflicts (Section 3.2.9), stochastic demand (Section 3.2.10), and time windows (Section 3.2.11) is also analyzed.

3.2.1 Exact approaches

The first 2L-CVRP approach is due to Iori *et al.* [78]. The authors proposed an integer model formulation adapted from the vehicle flow model for the CVRP with decision variables with a single index [142]. They considered a set of nodes V (depot is node 0), a set of edges E and K vehicles. The cost of travelling along the edge $e \in E$ is denoted by c_e. The decision variables z_e are binary and take the value one if a vehicle uses edge e in its route, and value zero otherwise. Additionally, $\delta(S)$ represents the set of edges with one endpoint in S and the other in $V\backslash\{S\}$, σ represents the bijection which defines the order by which the customers are visited, and $\Sigma(S)$ represents the collection of sequences σ ($\sigma(i)$ is the order of visit of customer i) in which (S,σ) is a feasible route, and $E(S,\sigma)$ is the set of edges of that route. Finally, *r(S)* represents the minimum number of vehicles required to serve all the customers in S. The model of Iori *et al.* is presented next.

$$\min \quad \sum_{e\in E} c_e z_e \tag{3.1}$$

$$\text{s.t.} \quad \sum_{e\in\delta(i)} z_e = 2 \qquad \forall i \in V\backslash\{0\} \tag{3.2}$$

$$\sum_{e\in\delta(0)} z_e = 2K \tag{3.3}$$

$$\sum_{e\in\delta(S)} z_e \geq 2r(S) \qquad \forall S \subseteq\in V\backslash\{0\}, S \neq \emptyset \tag{3.4}$$

$$\sum_{e\in E(S,\sigma)} z_e \leq |S|-1 \quad \forall (S,\sigma) \quad s.t.\ \sigma \notin \Sigma(S) \tag{3.5}$$

$$z_e \in \{0,1\} \qquad \forall e \in E\backslash\delta(0) \tag{3.6}$$

$$z_e \in \{0,1,2\} \qquad \forall e \in \delta(0) \tag{3.7}$$

The set of constraints (3.2) impose one and exactly one visit to all customers since two edges are incident to each node, while constraint (3.3) refers to the exit and return of exactly K vehicles. The set of constraints (3.4) are denominated by capacity cut-constraints and impose connectivity and the satisfaction of capacity limits [142]. The set of constraints (3.5) impose the loading feasibility of each route. The constraints (3.6) and (3.7) define the decision variable values.

Firstly, a relaxed version of model (3.1)-(3.7) is solved by removing constraints (3.4) and (3.5). The linear programming (LP) relaxation is strengthened using multi-star inequalities. Then, a branching scheme starts by applying separation procedures for the removed constraints.

Concerning the constraint set (3.4), the separation procedures consist in heuristic algorithms adapted from [117]. The objective is to find violated inequalities using minimum-cuts and if they exist, they are added to the model. When no violated inequalities are found, and the current solution is integer, then an exact algorithm verifies the feasibility of each route.

Additionally, a heuristic procedure is applied in order to update the incumbent solution. However, while computing this heuristic, it is also possible to detect invalid inequalities of type (3.5), and then the associated constraints can be added to the model. In the sequel, we describe this heuristic algorithm and the exact procedure to check the loading feasibility.

The heuristic procedure is based in the insertion of customers one by one, starting from the customer which is farther from the depot, and then adding customers according to a modified insertion cost. This cost is a weighted function with three parts: routing cost, residual loading area and residual weight capacity. If a feasible solution is achieved, a first-improvement local search procedure seeks for an improved solution. This heuristic runs 10 times with different weights for the three parts referred to above, and the best one is selected. If its value is better than the current best solution, the loading feasibility is tested through a branch-and-bound algorithm. If any route is infeasible, the constraint associated to that violation is added to the model.

As stated, the procedure to check the loading feasibility of each route relies on a branch-and-bound approach. At the root, lower bounds for the two-dimensional bin packing problem are computed [108]. If the greatest lower bound is greater than one, clearly at least two vehicles are necessary for that route, and thus the route is infeasible. Otherwise, if the lower bound is less than or equal to one, items are sorted according to the customer they are demanded, in the reversed order of visit. The items to be delivered to the same customer are sorted according to non-increasing width, breaking ties by non-increasing height. A bottom-left heuristic is applied to this sequence taking into account all the loading constraints.

If it is not possible to derive a valid packing, a search tree is used. At the root node an empty loading area is considered. For each item a descendent node is created by placing the item at the bottom-most and leftmost position. For each of these nodes, a child node is created for each item to be placed in each feasible position.

The authors defined a set of instances for the 2L-CVRP [76, 78, 64] extended from CVRP instances [130, 142], using the same complete graph, the weight demand of each customer, and the weight capacity of each vehicle. The loading surface is rectangular with height H equal to 40 and width W equal to 20.

In order to establish the number of items required by each customer and their sizes, five classes were created for each CVRP instance:

- **Class 1:** Each customer demands a single item with both width and height size equal to one. This class corresponds to pure CRVP instances.
- **Class 2 to 5:** The number of items required by each customer is obtained by a uniform distribution in the interval [1,n] where n is the number of the class. For each item one of the three shape categories (vertical, homogeneous or horizontal) is selected with equal probability. For each class and for each shape, a range is defined in Table 3.1. In this table, m_i corresponds to the number of items of customer i, h_{il} is the height of item l of customer i, and w_{il} corresponds to the width of item l of customer i. The dimensions of each item are uniformly generated according to the given range.

		Vertical		Homogeneous		Horizontal	
Class	m_i	h_{il}	w_{il}	h_{il}	w_{il}	h_{il}	w_{il}
1	1	1	1	1	1	1	1
2	[1,2]	$[\frac{4H}{10}, \frac{9H}{10}]$	$[\frac{W}{10}, \frac{2W}{10}]$	$[\frac{2H}{10}, \frac{5H}{10}]$	$[\frac{2W}{10}, \frac{5W}{10}]$	$[\frac{H}{10}, \frac{2H}{10}]$	$[\frac{4W}{10}, \frac{9W}{10}]$
3	[1,3]	$[\frac{3H}{10}, \frac{8H}{10}]$	$[\frac{W}{10}, \frac{2W}{10}]$	$[\frac{2H}{10}, \frac{4H}{10}]$	$[\frac{2W}{10}, \frac{4W}{10}]$	$[\frac{H}{10}, \frac{2H}{10}]$	$[\frac{3W}{10}, \frac{8W}{10}]$
4	[1,4]	$[\frac{2H}{10}, \frac{7H}{10}]$	$[\frac{W}{10}, \frac{2W}{10}]$	$[\frac{H}{10}, \frac{4H}{10}]$	$[\frac{W}{10}, \frac{4W}{10}]$	$[\frac{H}{10}, \frac{2H}{10}]$	$[\frac{2W}{10}, \frac{7W}{10}]$
5	[1,5]	$[\frac{H}{10}, \frac{6H}{10}]$	$[\frac{W}{10}, \frac{2W}{10}]$	$[\frac{H}{10}, \frac{3H}{10}]$	$[\frac{W}{10}, \frac{3W}{10}]$	$[\frac{H}{10}, \frac{2H}{10}]$	$[\frac{W}{10}, \frac{6W}{10}]$

Table 3.1: Ranges for items sizes in 2L-CVRP instances [76, 78, 64]

From the 180 instances obtained (36 instances for each class), the authors tested their approach in 60 instances (12 CVRP extended instances for each class). It was able to solve instances with up to 25 customers in less than one hour. Furthermore, it was possible to achieve the optimal solution in 55 out of 60 instances. The instances proposed in this work will be the comparison point for the vast majority of the 2L-CVRP approaches.

Recently, based on this branch-and-cut approach, another exact method was presented in [74] for the 2L-CVRP and 3L-CVRP. The authors analyzed different routing and packing separation procedures. After deriving an integer solution provided by branch-and-bound, each route is verified concerning its feasibility. Furthermore, the authors suggested a strategy to find infeasible routes from a non-integer solution. To prove the feasibility of a given route, the authors resort to several heuristic methods. If these fail to prove its feasibility, exact methods can be used based in branch-and-bound [107] and in constraint programming [33].

Other exact approach for a variant was addressed by Pollaris *et al.* [126]. In this problem, pallets have to be sequenced in order to satisfy axle weight constraints. The pallets are alternately loaded into two horizontal stacks. The authors stressed that axle weight constraints are important since its violation causes not only an infringement of traffic regulations but also a risk to road safety.

3.2.2 Tabu search

Tabu search was successfully applied to the 2L-CVRP by Gendreau *et al.* [64] who addressed the unrestricted and sequential version of the problem (2L|UO|L and 2L|SO|L, respectively). The initial solution is obtained through two heuristics: one for the generality of instances and another which is also executed for Euclidean instances. Both heuristics were adapted in order to be able to deal with loading and sequential constraints. The former is adapted from the classical savings algorithm of Clarke and Wright [32], but when the routes are associated and the capacity constraint is satisfied, then a loading algorithm is executed to check the loading packing feasibility either for the sequential or unrestricted sequential case. If the number of used vehicles is greater than the fleet size, the process is repeated accepting infeasible loading packings. The latter algorithm is adapted from [37] and it consists in randomly selecting a ray and considering segments linking each customer to the depot. The customers are assigned to a vehicle as the angle between the ray and the corresponding customer segment increases. The capacity constraint has to be satisfied and a loading algorithm is executed to check the loading packing feasibility (sequential or unrestricted). It is only possible to accept a loading infeasible packing for the last assigned vehicle.

The loading check algorithm referred to above for both heuristics works as follows. The items are sorted according to both their sizes and the visit sequence of the customers. Thus, customers are sorted by the reverse order of visit and for each customer the items are sorted by decreasing order of width, breaking ties by decreasing order of height. The first item is placed at the origin (0, 0) and the following items are placed in the position that maximizes the touching perimeter [94] satisfying the loading constraints and not exceeding the height of the vehicle. If it is only possible to satisfy the loading constraints, the item is placed in the position which minimizes the required height. If no such position exists, the algorithm returns an infinite height.

The movements that characterize the neighborhood structure consist in exchanging a customer from a route to another route. After they have been applied, the movements are inserted into the tabu list. Infeasible routes are allowed in the sense that the weight or the height of the loading area can be exceeded. In these cases, the movements are penalized according to the dimension of this violation.

The results reported by the authors show that this approach is competitive with exact approaches, finding the optimal solution in roughly 50% of the number of optimal solutions obtained by the exact approach in [78].

3.2.2.1 Guided Tabu Search

In 2009, a guided tabu search algorithm was proposed by Zachariadis *et al.* [154] for the 2L-CVRP considering items with fixed orientation and both sequential and unrestricted versions of the problem.

To verify the loading feasibility of the problem the authors presented a heuristic bundle composed by five packing heuristics. All these heuristics start from a sequence of items. For the sequential version, this sequence is obtained by sorting customers by the reverse order of visit. For each customer, the items are sorted by decreasing order of area. For the unrestricted version, it is only necessary to sort the items by decreasing order of their values of area. Although these five packing heuristics share the same sequence, they differ in the way the loading position is chosen.

In this heuristic bundle, there are two heuristics based on the bottom-left, being each one applied to an axis: one axis parallel to the rear-side of the vehicle, and another perpendicular to it [31]. The third and fourth heuristics aim to maximize the touching perimeter: one of them considers the walls as part of the touching perimeter and the other excludes the walls [94]. Finally, the fifth proposed heuristic consists in computing a rectangular surface for each possible loading position coordinate, and selecting the position whose surface area is smaller.

The referred to above heuristic is successively applied until a feasible solution is found. If at the end of the last heuristic a valid solution is not achieved, the route is considered to be infeasible.

The initial solution is obtained by a constructive heuristic. An empty route is generated for each vehicle. The customers are sorted by decreasing order of the values of their demanded area, and by this order, the insertion of each customer in all positions of the routes is tested. The feasibility of the routes is verified through the bundle of packing heuristic referred to above. The customer is assigned to the feasible route that minimizes the free area of the vehicle after loading its items. Its position in the route is the one in which the additional cost is minimized.

After they have been applied, the movements are inserted into the tabu list. Infeasible routes are allowed in the sense that the weight or the height of the loading area can be exceeded. In these cases, the movements are penalized according to the dimension of this violation.

The guided tabu search algorithm starts from the initial solution referred to above. The *guided* denomination is due to the fact that the objective function incorporated in the tabu search algorithm is modified in order to increase diversification, guiding the search to unexplored solutions. The diversification is performed by penalizing long arcs in the solution. For this purpose, a utility function is used to find the arcs to be penalized in the objective function. In Figure 3.1, a scheme aims to represent the guided local search mechanism: each s_i represents a valid solution in iteration i. The penalization of the objective function is represented in dashes, and it guides the algorithm to optimal solution s^*.

The neighborhood structures are defined by three types of movements based on

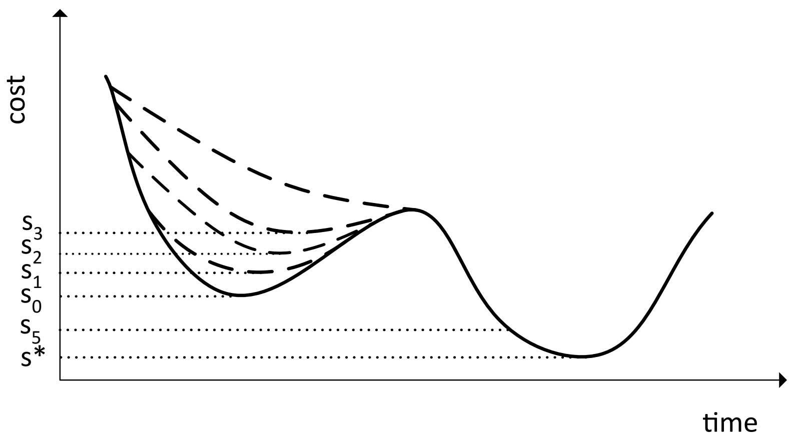

Figure 3.1: Guided local search (adapted from [47])

[41, 93]. The first consists in shifting the position of one customer within a route or shifting one customer from a route to another route [148]. The second type of movement consists in exchanging the position of a pair of customers within the same route or between two routes [148]. Finally, the last movement is based on the route interchanging, commonly denominated by *2-opt* [41, 93]. If it is applied within the same route, two non-consecutive arcs are removed and the sequence of customers between these arcs is reversed. Two new arcs are created to link this new sequence to the other customers. On the other hand, if this movement is applied between two routes, one arc is removed in each route. Then, the first part of one route is linked with the second part of the other route.

One of the three movements is performed to a feasible neighbour solution only if such move is not included in the tabu list or if it is the best solution ever achieved.

In order to accelerate the convergence of the algorithm, the authors proposed two procedures: the neighborhood is reduced in each iteration, and information relative to the feasibility of the routes is stored. Thus, it is not necessary to re-evaluate whether the solution is feasible or not.

The computational performance of this approach was tested with the instances proposed in [64]. The authors compared their results with the results presented in [64], concluding that there is an improvement of 1,54% of the solution values of instances of class 1. Considering instances of class 2 to class 5, for the unrestricted and sequential versions the enhancements were on average 1,62% and 1,62% respectively.

3.2.2.2 Extended guided tabu search

Based on the work of Zachariadis *et al.* [154], Leung *et al.* [92] proposed an Extended Guided Tabu Search (EGTS) for the 2L-CVRP. This approach aims to apply a guided local search extension suggested by Millis *et al.* [111]: if one move can lead to a new best solution, then the penalty associated to that move is removed. Millis *et al.* [111] denominated this process by aspiration move. They demonstrate that this procedure can enhance the performance of the algorithm.

The initial solution method relies on the guided tabu search starting solution of Zachariadis *et al.* [154], but with modifications for the case when there is no valid route for a given customer. In these cases, the customer is exchanged with another customer already assigned to one route, selected stochastically.

To check the loading feasibility, the authors applied the heuristic bundle presented by Zachariadis *et al.* [154], and one more packing algorithm named Lowest Reference Line Best-Fit Heuristic (LBFH). The LBFH can be described as follows. The lowest reference line is the one under which no items can be placed. In this line, two endpoints are considered to place the item. The item must be placed at the bottom corner of the envelope which maximizes the touching perimeter. The selection of an item relies on the quality of the fitness which is evaluated considering the corner points that fit the placement. Consequently, this loading heuristic aims to promote the minimization of the number of resulting corner points. An example of this process is presented in Figure 3.2. In (a) four corner points are represented. In (b) an item X is placed at the right corner of the lowest reference line while in (c) other item Y is placed at the same point. Therefore, layout (c) is more suitable since two corner points fit the placement of piece Y.

The quality of this approach was tested on benchmark instances [76, 78, 64]. The computational results considering the extended guided tabu search without LFBH (Classes 2 to 5), outperform the guided tabu search approach [154] by 0,58% and 0,26% on average, for the sequential an unrestricted versions, respectively. The com-

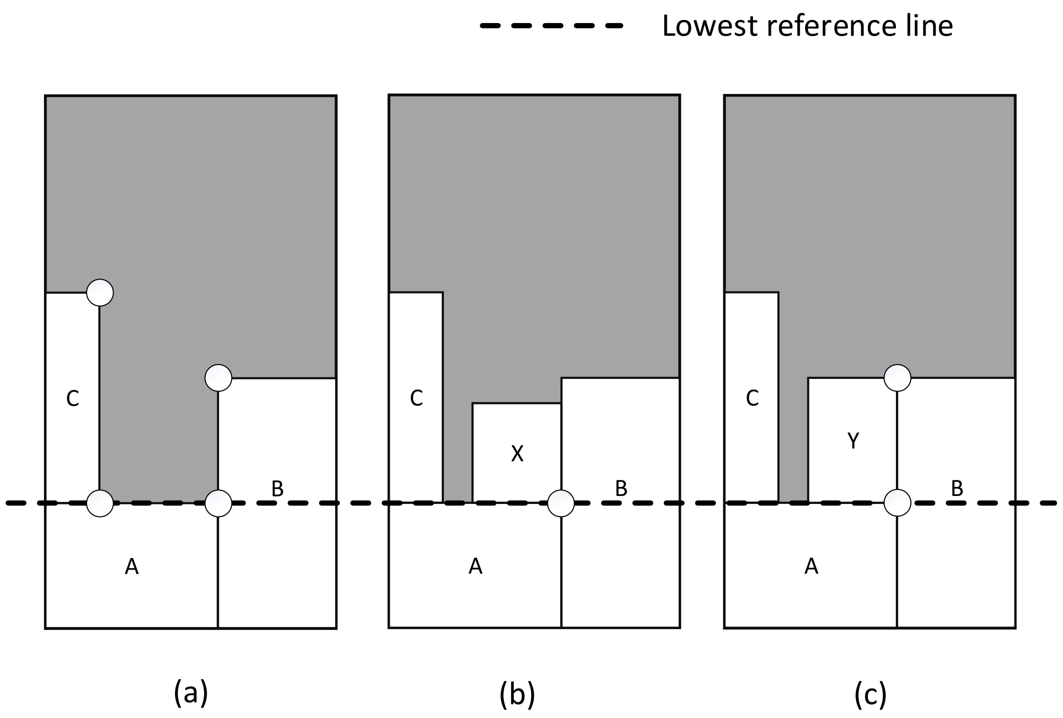

Figure 3.2: Lowest reference line best-fit heuristic (adapted from [92])

putational results obtained for EGTS with LFBH are more significant. For classes 2 to 5, the obtained results outperform the guided tabu search on average 1,28% and 1,06% for the sequential and unrestricted cases, respectively.

3.2.3 Ant colony optimization

The first work reporting on the 2L-CVRP whose items may have a variable orientation is due to Fuellerer *et al.* [61]. In this case, items are allowed to be rotated 90 degrees in the horizontal plane. The rotation version increases the complexity of the problem. However, a higher number of valid packings is achieved and consequently the routing costs can be reduced. An example of a possible advantage of the inclusion of rotation is shown in Figure 3.3. In (a), no rotation of items is allowed and the obtained packing is infeasible, since not all items are completely within the loading area. In (b), a possible improvement can be achieved if rotation is possible.

In the mentioned work, the authors describe an Ant Colony Optimization algo-

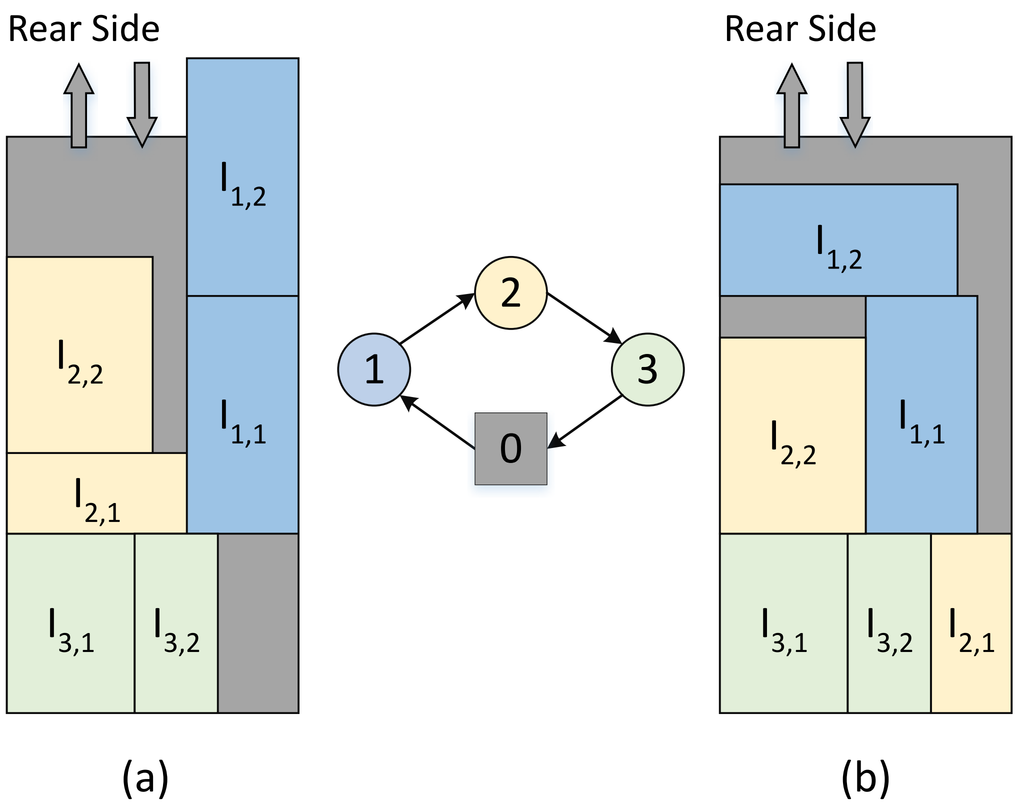

Figure 3.3: Possible improvement when considering rotated items

rithm (ACO) for the 2L-CVRP, which is an extension of the ACO for the CVRP [129].

The generation of solutions relies on the classical Savings Algorithm of Clarke and Wright [32]. From the perspective that combining pairs of customers in the same route produces a saving, a list of savings for all pairs of customers is built. However, whereas the classical Savings Algorithm has a full list of savings obtained by combining customers, in this approach, the authors suggested a restricted list of feasible and more profitable combinations of customers at each step.

Concerning the overall approach, each ant of the population stochastically selects a combination from this list. The probability of selecting a certain combination depends on a mathematical formula. For the CVRP, this formula takes into account the savings and the pheromone information. This formula was adapted to the 2L-CVRP in order to consider the items area, and thus favour a better filling of the bin.

After merging the routes, the restricted list is updated and the process is repeated until a heuristic solution is found. Then, a local search is applied to each ant through movements. During the local search procedure, if any solution requires more vehicles than those that compose the fleet, the objective function is penalized and the pheromone information will not be updated.

In order to provide the loading feasibility of one route, the authors resort to lower bounds based on the procedure described in [108], but extended to the case in which the items have a variable orientation. If it is not possible to prove the infeasibility of the route using the lower bounds, the items are sorted and then two heuristic already referred to above are applied: a bottom-left filling heuristic and a maximum touching perimeter algorithm. If both these heuristics fail, a local search algorithm is applied, switching items in the sequence of input, and executing the two heuristics again. If this alternative fails, a truncated branch-and-bound method is then applied.

The authors tested their approach in the benchmark instances referred to above [76, 78, 64]. For small size instances (less than 40 customers) and considering the sequential constraint and oriented items (2|SO|L), this approach was able to find the optimal solution in 46 out 58 optimal solutions obtained in [78]. In the overall set of instances, and comparing with the tabu search approach [64], there is an average improvement of 3,65%.

In order to evaluate the impact of the loading constraints in the overall approach, the authors relaxed two constraints. If the unrestricted version of the problem is considered (2|UO|L), an average improvement of 3,27% is achieved. If the rotation of items is allowed (2|SO|L), the average improvement is slightly better (3,38%). For the unrestricted and non-oriented problem (2|UR|L) the average improvement is 5,06%.

3.2.4 Simulated annealing

Leung *et al.* [91] proposed a new simulated annealing approach for the 2L-CVRP, considering both sequential and unrestricted versions and fixed orientation.

The initial solution is constructed using the same procedure as in [92], and the neighborhood structures are the ones suggested in [154]. To verify the loading feasibility of a given solution, the authors used the heuristic bundle presented in [92]. In order to accelerate the simulated annealing algorithm, the authors implemented an effective data structure to keep the information about the feasibility or infeasibility of a given route.

The average results considering the instances in classes 2 to 5 present an improvement of 3,19% and 1,38% compared to the tabu search approach [64] and to the guided tabu search approach [154], respectively. Similar improvements are reached in the sequential version.

The simulated annealing was also successfully applied to a realistic variant of the 2L-CVRP which considers a heterogeneous fleet [90]. Since the vehicles have different capacities and dimensions, they also have different fixed and variable costs. Other constraints remain the same, and both sequential and unrestricted versions are considered. The main innovative feature of this work is the inclusion of heuristic local search algorithms in the simulated annealing method. In this approach, one movement is performed to a feasible solution, by randomly selecting with equal probability one of three neighborhoods structures. These neighbour structures are based on those presented in [92], and as stated above. However, it is worth noting that in these neighborhoods structures exchanges are only allowed if they occur between vehicles of the same type, due to the infeasible solutions that could be obtained if this assumption was not considered.

If the obtained solution is better than the current solution, then the current solution is automatically updated. Otherwise, the obtained solution is only accepted based on the simulated annealing acceptance chance. Then, and with a probability of 5%, three local search methods are applied, each one based on a neighborhood structure defined above and using a first criterion improvement. Based on the benchmark instances for the 2L-CVRP [76, 78, 64], a new set of instances was created aiming to embed the new problem requirements, namely the heterogeneous fleet and costs. The authors provided a comparative analysis between the enhanced simulated annealing algorithm and without the local search heuristic. This analysis demonstrates both a marked improvement and a computational time decrease when such methods are embedded in the overall approach, with an improvement of more than 23% for all the classes and for both unrestricted and sequential versions. Furthermore, this approach was also tested on benchmark instances, and the obtained results are competitive with those obtained in other approaches.

3.2.5 Biased-randomized algorithm

Dominguez *et al.* [52] presented a biased-randomized algorithm for the unrestricted 2L-CVRP whose items are allowed to be rotated (2|UO|L and 2|UR|L).

The overall approach starts through one procedure based on the Clarke and Wright savings list, which is sorted by decreasing order, and at each iteration, an edge is chosen using a biased probability. This probability leads to different solutions at each call, but still aims to favour higher savings. Whenever an edge is selected, the merging of the two routes is only performed if the obtained route is valid concerning the capacity and the loading constraints. To verify the feasibility of the loading, the authors suggested a multi-start biased-randomized version of the best-fit heuristic with items rotation proposed in [26] which consists in applying the biased process to the items order. The process runs while the savings list is not empty.

For the rotated version of the problem, the authors provided a comparison with [61], obtaining improvements for each class that vary from 0,04% to 1,11% (average values of generated solutions in 10 runs). For the oriented problem, competitive results with those in other approaches presented so far were achieved.

3.2.6 Evolutionary local search

Duhamel *et al.* [54] proposed a multi-start evolutionary local search for the 2L-CVRP considering only the unrestricted version of the problem, and oriented items (2|UO|L). One of the innovative features of this approach is the handling of loading constraints, which are transformed in constraints of a Resource Constrained Project Scheduling Problem (RCPSP) with a single resource and with no precedence constraint. Indeed, and as stressed in [54], the RCPSP is less constrained than the two-dimensional orthogonal packing problem and, consequently, can be less difficult to solve. This process is shown in Figure 3.4: a valid layout (unrestricted oriented) presented in (a) is converted in a RCPSP instance.

The overall approach is based on a modified Greedy Randomized Adaptive Search

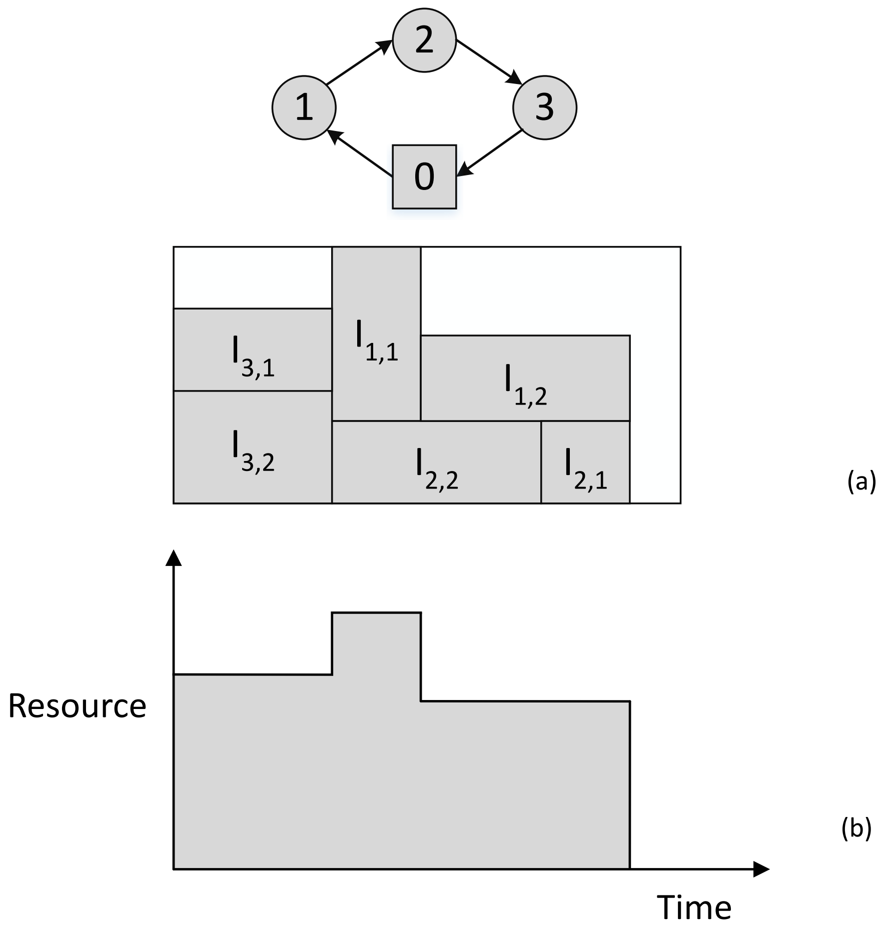

Figure 3.4: Conversion of a two-dimensional layout to RCPSP (adapted from [54])

Procedure (GRASP) whose local search procedure is performed with an Evolutionary Local Search (ELS) algorithm. During the execution of the denominated ”GRASP x ELS” algorithm, only the constraints related to RCSPSP and to the capacity of the vehicle (weight) are considered. To solve the RCPSP, the authors suggest a randomized activity selection.

The GRASP x ELS algorithm is applied to solutions that alternate between two different representations (inspired in [128]): a set of routes or a giant route resulting from the concatenation of this set. The initial solution is obtained through one heuristic procedure which iteratively applies four heuristics: a randomized version of the classical Clarke and Wright savings algorithm [32], an adaptation of the path scanning algorithm [66], an adaptation of the randomized version of path scanning and, finally, a random generation heuristic.

For both path scanning and its randomized version, the used criteria are based on the maximal or minimal values of the total weight to deliver, the total demanded area to deliver, or the distance to the depot (with or without considering if the vehicle area is half-loaded). Immediately after generating initial solutions, the set of routes is concatenated into a giant route. The ELS algorithm iteratively performs mutation and local search procedures during a given number of iterations which is also the number of generated solutions.

The mutation is performed on the giant route referred to above, which has a smaller solution space. Then, the giant route is divided into a set of routes and a local search is applied to this set. The neighborhood structures consist in 2-opt and swap moves which are both applied inside a single route and between two routes. Finally, the set is converted back to the concatenated version in order to proceed with the remaining iterations of ELS.

For each iteration of ELS, if an improvement is achieved, the solution is kept into a list of the best solutions. All the solutions of this list are transformed back into 2L-CVRP solutions, and the best one is selected.

The computational results of this approach presented new best known solutions in some instances in all classes (10 for the Class 1 and more than 20 for each of the other classes). The average values also outperformed the average values obtained in [64].

3.2.7 Compact metaheuristic

Zachariadis *et al.* [156] proposed a compact metaheuristic algorithm for the 2L-CVRP in both versions, sequential (2|SO|L) and unrestricted (2|SO|L).

The initial solution is performed through starting from one empty route for each vehicle. A utility function is defined aiming to maximize the area utilization. The pair route/customer, which maximizes the utility function, is selected and the customer is assigned to the route in the sequence point that minimizes the route cost.

The overall approach is based on local search, using a Static Move Descriptor (SMD) representation, originally proposed in [153]. From the perspective that only a subset of arcs of the solution will be removed and replaced, re-evaluating all the solution is not necessary since the other arcs of the solution remain unmodified. Therefore, each local search move can be encoded into a SMD instance, including the type of the move to be performed, the point where the move is applied and the inherent difference in the objective function (denominated by cost tag). This cost takes into account only the removed and generated arcs: the cost tag is equal to the total costs of new arcs minus the total cost of the removed arcs. Thus, it is not necessary to recalculate the overall cost of the solution at each local search movement.

Three possible moves are allowed, based on the (1) relocation or (2) swap moves of one customer and on the (3) route crossover and one Fibonacci heap is associated to each move type. All moves are applied and the corresponding SMD instances are kept in Fibonacci heaps, according to their cost. One move is randomly selected (all the moves have the same probability of being selected), and the SMD instances are verified regarding the diversification status and the feasibility of the 2L-CVRP constraints. The best move is then applied.

To ascertain the packing feasibility of the solution, the authors proposed a heuristic procedure using a list of available positions which is updated at each iteration based on [40]. These points are named as extreme points. The list of available positions is initialized with the origin, and the pair item/position which maximizes a utility function based on two terms (or three) for the unrestricted version (sequential version, respectively). For the unrestricted version, the utility function is composed by the touching perimeter [94] of the corresponding pair item/position (aiming a feasible loading) minus a large positive constant multiplied by the number of times in which the pair was examined (leading the search for unexplored loadings). The sequential version includes both terms referred to above and another one to give higher priority for items of customers served later. In order to reduce repeated operations, the authors implemented memory structures (based on hash tables), that keep the (partial) feasible loadings configurations.

The authors tested their approach in the Gendreau *et al.* instances [64] and compared the results with those proposed in [92, 61, 54]. The results concerning the unrestricted version show an average improvement in 21 of 36 benchmark instances, reaching the best solution in 9 cases. Considering the sequential version of the problem, an average improvement of 1,21% occurred in 30 out of 36 instances.

3.2.8 Variable neighborhood search

Another recent approach for the 2L-CVRP is proposed in [151] for sequential and unrestricted versions based on the Variable Neighborhood Search (VNS).

The initial solution is obtained by a process similar to the one proposed in [154, 92], presented above. Briefly, in this process the customers are sorted by decreasing order of the area of their demanded items, and successively inserted in the feasible route that minimizes the free area of the vehicle after loading their items. The insertion position for the customer is the one that minimizes the insertion cost. If this procedure is unable to insert one given customer, the customer insertion order is modified by exchanging positions between that customer and another one randomly chosen.

The authors introduced some modifications to the classical VNS procedure. Firstly, the shaking phase, i.e., the generation of random neighbour solutions, is repeated a certain number of times, providing different solutions for the local search procedure. At this shaking phase, the neighborhood structures are both based in exchanging sequences of two or three consecutive customers and in the six neighborhoods structures presented in [154] and already described in Section 3.2.2.1. It is worth noting that these six suggested neighborhoods structures will be used in the local search. In that phase, a first improvement strategy is performed, which means that all neighborhoods are explored in a random order until a better solution is found.

Additionally, the VNS is also enhanced with a diversification procedure which is applied each time it is not possible to achieve a better solution. From the incumbent solution, some customers are randomly selected to be removed. These customers are sorted according to the area of their demanded items, and then reinserted using a similar procedure of obtaining an initial solution, which is referred to above. The number of customers to be removed is the minimum between 50% of the total set of customers and 10% of these plus the number of iterations without improvements. Therefore, a larger the number of iterations without improvement will provide greater diversity by removing more customers. This diversity feature provides different initial solutions and it is suitable to avoid local optimal solutions.

In order to evaluate the packing feasibility of a given route, the authors proposed a tabu procedure adapted from [149], which can be summarized as follows. An initial sequence for packing the items is obtained by sorting the items of each customer using three rules: in decreasing order of area, width and length, respectively. For the sequential version, the customers are sorted in reversed order of visit, which will have a direct impact in the sequence of items. A skyline packing heuristic [149] tries to derive a valid packing for this sequence. If it is not possible to achieve a feasible packing, a set of sequences is generated from that sequence by swapping the position of two items. From this set, the sequence that maximizes the area utilization is selected to establish the order in which the items will be placed by the packing heuristic. If it is not possible to derive a feasible layout, the swapped items are stored in a tabu list.

As referred to above, the packing heuristic is based in [149]. In that work, the current layout is represented by a contour denominated by *skyline*. The *skyline* results from the junction of vertical strips, each two strips having the same x-coordinate but different y-coordinate. The candidate positions for placing items are a set of points in this skyline. An example of this representation is presented in Figure 3.5. The authors claim that this representation is more efficient. However some portions of the loading area are not considered to place items.

For a given sequence each item is selected to be placed in the loading area. All possible positions are evaluated using a fitness function that takes into account, among other components, the wasted area with the insertion of the related item. The position that maximizes this fitness function is selected to place the item and the skyline is modified in order to incorporate this modification.

The computational results for the unrestricted version could find better values for 65 out of 144 instances when compared with previous approaches presented in the literature.

3.2.9 2L-CVRP with partial conflicts

The Two-dimensional Bin Packing Problem with Conflicts (2BPPC) was firstly proposed by Khanafer et al. [85] aiming to minimize the number of bins used to pack a set of two-dimensional items taking into account incompatibilities between items that

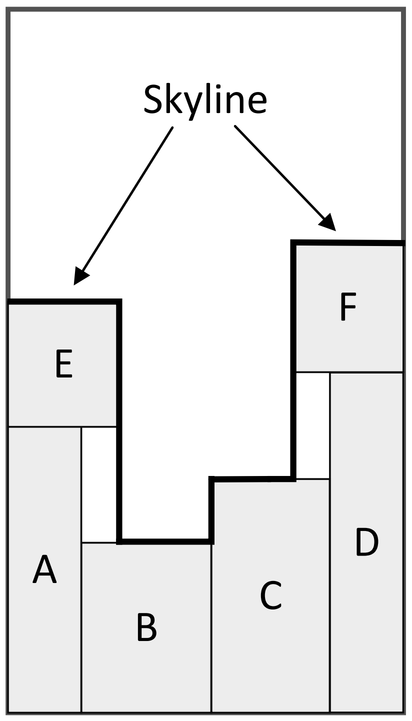

Figure 3.5: Skyline representation (adapted from [149])

could not be packed in the same bin.

Some authors addressed one similar problem where the conflict items may be packed into the same bin, however they must keep a minimum distance between them. This problem is known as Two-dimensional Bin Packing Problem with Partial Conflicts (2BPPC), as proposed in [68].

Hamdi-Dhaoui *et al.* [69] extended the 2L-CVRP by including the constraints of the 2BPPC. The resulting problem is a two-dimensional vehicle routing problem with partial conflicts (2LPC-CVRP). The authors proposed a bi-objective algorithm based on the Non-dominated Sorting Genetic Algorithm II (NGSA-II) [48]. Therefore, the objective is twofold aiming to minimize the total transportation costs and to minimize the loading difference between the most loaded and the least loaded route.

3.2.10 2L-CVRP with stochastic demand

As referred to in Chapter 2, one variant in the routing field is the one that deals with uncertainty. In the L-CVRP context, to the best of our knowledge, only one work deals with stochastic data [42], more precisely the weight and the size of items are only known when loading the vehicle. Therefore, if items could not be loaded due to the lack of space, then it is assumed that these items should be left on the dock. Consequently, there is a cost associated to this non-compliance situation, which relies on the number of customers whose demand was not completely satisfied. The objective is to minimize both routing and non-compliance costs.

Although information about the size and the weight of items is not completely known, there is a finite number of values for these parameters. For each of these values, a probability is associated.

The suggested problem is solved by the so-called L-Shaped method, which consists in relaxing the constraints related to the capacity and to the loading and in replacing the non-compliance costs by a lower bound. Then, a branch-and-cut method is applied, with new lower bounding functionals, which are in fact lower bounds for non-compliance costs.

In addition to the stochastic instances, this approach was tested in the set of (deterministic) benchmark instances [76, 78, 64]. The results pointed out that this algorithm solved instances with about 70 customers and 200 items in a small amount of time.

3.2.11 2L-CVRP with time windows

As referred to in Chapter 2, the Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the Capacitated Vehicle Routing Problem (CVRP) in which each customer must be served within a specific time interval. Some works of L-CVRP consider these extension in addition to loading constraints.

One of these approaches is due to Khebbache-Hadji *et al.* [86] who addressed the 2L-CVRP with time windows using heuristic and memetic algorithms. Memetic algorithms are hybrid methods that combine genetic algorithms with local search procedures, and were firstly proposed in [113]. In [86], no sequential constraints were considered, and items are not allowed to be rotated (2|UO|L).

Regarding the heuristic methods, the authors combine two routing heuristics and three packing procedures presented in the literature, resulting in six heuristic algorithms. The objective of the implementation of these algorithms is twofold. On the one hand, it is possible to compare the six approaches. On the other hand, these heuristics will provide initial solutions for the memetic algorithm.

Considering this memetic method, the local search is applied with a given probability to a child right after the crossover operations and before the mutation. The local search phase includes four neighborhood structures. Among these structures, two can be applied within the same route or between two routes by moving one customer or a sequence of two consecutive customers, and by swapping one or two customers. The other two movements include inverting the sequence of visits (2-Opt) and interchanging the last part of two routes.

The heuristic and memetic methods were tested in an adapted subset of benchmark instances proposed in [64]. The memetic algorithm improves the solutions provided by heuristics in 205 seconds on average.

3.3 Approaches for the 3L-CVRP

In Section 3.2, we reviewed the approaches for the 2L-CVRP presented in the literature. Some of those algorithms were extended to tackle the 3L-CVRP. The main features of original approaches were preserved. As a consequence, in Section 3.3.1 we present such extensions giving special emphasis for the specific aspects included when applying the same method to three-dimensional problem. For a better understanding, we refer to the original approach in each 3L-CVRP contribution. The following sections provide a survey on the methods of 3L-CVRP.

3.3.1 Generalization of 2L-CVRP approaches

The first work referring to the combination of CVRP with three-dimensional loading constraints was proposed in 2006 [63] by only considering the sequential version of the problem. Additional constraints are considered such as fragility constraints and supporting area constraints. The items can be rotated 90 degrees in the horizontal plane, but they have a fixed vertical orientation.

The authors proposed a tabu search algorithm that allows movements that produce infeasible solutions, generalizing the algorithm proposed for the 2L-CVRP [64]. Furthermore, the loading component of the problem is performed by an inner tabu search algorithm whose movements, which characterize the neighborhood structure, consist in exchanging the loading sequence of the items. From this new sequence, two heuristic procedures are applied. These heuristics are extended versions of two-dimensional heuristics: the bottom-left algorithm [6] and the touching perimeter algorithm [94].

Two sets of computational results are reported: one for instances provided in the literature, and another set of real-world instances provided from the furniture industry in Italy.

In order to compare the impact of each type of constraint, the authors presented the solution values relaxing one loading constraint type at a time. They concluded that relaxing sequential constraints or supporting area constraints lead to an average improvement (8,74% and 9,86%, respectively) that is greater than relaxing the fragility constraints (2,66%). Relaxing the three set of constraints leads to an average improvement of 15,87%.

Based on the guided tabu search for the 2L-CVRP [154], Tarantilis *et al.* [141] extended this approach to the 3L-CVRP. In this approach, fragility and supporting area constraints are considered, and items must have a fixed vertical orientation, but can be rotated 90 degrees in the horizontal plane.

Some modifications were applied to tackle the three-dimensionality of the problem. The initial solution is obtained in an analogous process for the 2L-CVRP [154]. However, in each insertion, the customer is assigned to the feasible route that minimizes the free volume of the vehicle after loading the customer items.

In order to test if the loading constraints are satisfied, six packing heuristics are suggested, each one having a different criterion to select the position to place the item. This bundle is composed by heuristics extended from the ones presented by the same authors for the 2L-CVRP [154]. The first and the second heuristics promote the selection of the position with minimum value at a certain axis (L-axis and W-axis, respectively) breaking ties with the minimum values of the other two axes. The third heuristic aims to maximize the touching area between the item and either the current layout or the walls of the container of the vehicle, excluding the bottom face. As in

[154], a version that excludes the walls of the container was also considered in the fourth heuristic. In these two last heuristics, if two or more positions have the same touching area, the position with the minimum W-axis value is selected. Hence, two heuristics were created based on the third and fourth heuristics referred to above, sharing almost the same methods, but using the minimum L-axis value to break the ties.

The suggested extended approach was applied to the instances suggested in [63] and to a new set composed of twelve instances. The obtained results point out an improvement of 21 out of the 27 best results in [63], with an average improvement of 3,54%. Improvement is larger for larger instances.

As in [63], it was observed that relaxing one loading constraints at a time leads to an improvement. For the instances proposed in [63], the average improvement of the solution is 2,85% if one fragility constraint is not considered, while sequential constraint and supporting area constraint have a more effective impact if they are not considered (4,61% and 5,61%, respectively).

In this work, the authors also considered the M3L-CVRP. This problem was defined in Section 2.3.4. As stated before, this version takes into account a less restrictive definition of sequential constraints.

Lacomme et al. [88] generalized the GRAPxELS approach for the unrestricted 3L-CVRP considering both the oriented and rotated versions. However, instead of solving the three-dimensional packing problem through the RCPSP, a new three-dimensional packing two-step heuristic procedure is suggested. In the first step, only two dimensions are considered, and thus the height is considered as a cost. The objective is to find a position to each item in the two-dimensional bin, taking into account that the packing has to be orthogonal and the cost resulting from the sum of the overlapping is limited to capacity of the bin (which is the height of the vehicle). The second step aims to find the third coordinate position such that the result is a feasible three-dimensional packing.

The authors report on results considering benchmark instances and real instances based on the travel distance among 96 French cities. For the 27 benchmark instances [63], and considering the unrestricted 3L-CVRP with rotations, this approach was able to achieve 16 best solutions. A slightly impact on the quality of the solutions is observed if the horizontal rotation of items is allowed. It is worth noting that evolutionary local search was also effective for a similar problem in which instead of minimizing the distance, it is required to minimize the fuel consumption [157].

Finally, Fuellerer *et al.* [62] extended their Ant Colony Optimization approach [61] to the 3L-CVRP with sequential, fragility and supporting constraints. The items can be rotated in the horizontal plane, but they are oriented concerning the vertical orientation.

As in [61], lower bounds are computed to avoid unnecessary calls to loading heuristics. The feasibility of loading is assessed by extending the two-dimensional heuristics, namely the bottom-left and the touching perimeter heuristics. The savings are computed taking into account both the cost savings and the density of the packing, using for the latter component the volume of items. The authors compared the obtained results with the tabu search algorithm presented in [63]. The ACO algorithm was able to produce an average improvement of 5.77% within a lower amount of time, in spite of using distinct machines for the two compared approaches. Additionally, by relaxing the fragility, sequential and supporting constraints, this approach achieves better results when compared with tabu search.

3.3.2 Exact approach

Junqueira *et al.* [83] proposed an integer linear programming model for the 3L-CVRP with additional loading constraints.

These constraints include vertical load stability, multi-drop situations and load bearing. It is worth noting that the authors considered not only load bearing by imposing a maximum number of boxes that can be stacked, but also considered fragility constraints by preventing any placement upon a fragile item.

As explained before, multi-drop constraints ensure that items belonging to the same customer are placed near to each other in order to both simplify unloading operations and avoid rearranging the items in the vehicle. Additionally, the order of visit must be taken into account when loading the vehicle. Taking into account a partial layout, when loading items for a customer to be served earlier, the authors proposed a threshold that corresponds to the maximum distance that can be performed in order to arrange those items. This assumption takes advantage of empty holes that have been created in the layout.

The authors extended the time-dependent formulation for the Travelling Salesman Problem [59] for the 3L-CVRP. The authors assessed the effectiveness of their approach in randomly generated instances. They concluded that such formulation is suitable for medium-size problems.

3.3.3 Tabu search

Zhu *et al.* [158] proposed a two-stage Tabu Search (TS) for the 3L-CVRP. To verify the feasibility of the loading a local search procedure is adopted. A loading sequence is created by sorting items firstly by the reverse order of the visit sequence. For items of the same customer, non-fragile items are before the fragile items, breaking ties using the volume of items (bulky items first). Then, an improved version of both Deepest-Bottom-Left-Fill (DBLF) and Maximum Touching Area (MTA) heuristic are applied to this sequence.

The initial solution is obtained through the savings algorithm [32]. Whenever two routes are merged, the feasibility of the resulting route (capacity and loading constraints) must be assessed (using heuristic to be presented below). If the number of obtained routes is greater than the fleet size, another iteration of the savings algorithm is performed but relaxing the capacity constraints and allowing loadings that require a length size greater than the length of the vehicle.

The movements that define the neighbour structure consist in *2-opt* and *2-swap* applied in one route, moving one customer from one route to another, crossing-over between two routes and finally splitting one route into two routes.

The two-stage denomination is due to the first phase of the algorithm that is applied if the starting solution is infeasible. In this case, the excessive weight and length are penalized in the objective function during this phase. The phase two is only applied to feasible solutions and only generates feasible solutions. This phase aims to minimize the total cost.

The authors compared their approach with [63, 141, 62], reaching better results in 20 out of the 27 instances with an average improvement of 0,57%. The impact of the constraints in the performance of the approach is, on average, higher for the sequential constraints (average improvement of 7,73% if these constraints are not considered) and it represents roughly 65% of the achieved improvement if fragility, support and sequential constraints are not considered. As in [154], a M3L-CVRP was presented. The obtained results reached better results in 8 out of the 12 instances of Tarantilis *et al.* [141] with an average improvement of 1,28%.

A tabu search algorithm was successfully applied in [110] for the 3L-CVRP with fragility and supporting constraints (3|SR|L). However, such application is only devoted to the loading component, while the routing part is solved by means of a genetic algorithm. The packing heuristic is applied for a given route by reversing the order of visit of all customers. For each customer, non-fragile items have priority over fragile items. Using this sequence, a front-left-bottom heuristic (based in two-dimensional heuristic proposed in [6]) and a touching-area heuristic (also based in a two-dimensional approach suggested in [94]) are used attempting to derive a feasible packing. If it is not possible to derive a feasible solution, the items are divided in two groups: the ones that completely fit in the vehicle and the ones that are placed outside the vehicle, even partially. Each pair of items belonging to different groups is evaluated to be exchanged using a score function taking into account the possible new routing cost and the number of times that items were already exchanged. The items belonging to the pair with the best score are exchanged. Additionally, these items are forbidden to be exchanged within a certain number of iterations.

The effectiveness of this approach was tested in the set of 27 instances presented in [63]. The obtained results improve the solutions presented in [63] for all instances (with an average improvement of 8.98%) and in [141] for 20 instances (with an average improvement of 4.84%). Furthermore, this approach was also able to outperform 16 solutions obtained in [62] with an average improvement of 1.84%. The authors stressed that computational time is reduced in more than 96% when compared with [63, 141, 62].

Concerning the analysis of each type of constraint, the authors presented average improvements that are quite similar when relaxing sequential or supporting constraints (4.24% and 4.08%, respectively). Less significant improvements are obtained when the fragility constraint is not considered. Relaxing all the constraints referred to above leads to an average improvement of 11.20%

In [140] a 3L-CVRP approach is addressed, considering fragility, supporting and sequential constraints. Tabu search is used in the routing part of the problem while the loading component is solved by two heuristics: the least waste heuristic and the touching area algorithm. The innovative feature of this work is a new mechanism to update the loading points to place the items. The authors claim that this mechanism can lead to a greater utilisation of the loading space.

The effectiveness of this approach was also assessed in benchmark instances [63, 141]. The authors presented lower average costs when comparing with many 3L-CVRP approaches. This approach was also able to achieve new best solutions.

3.3.4 Hybrid algorithm

Bortfeldt [20] proposed a hybrid approach for the 3L-CVRP (3|SR|L) considering as additional constraints the fragility and support constraints. The overall approach consists in a tabu search procedure for defining the routes and in a tree search algorithm for the loading problem.

The tabu search approach is initialized according to a multi-start randomized savings algorithm. Four movements are allowed if they lead to feasible solutions, including swap positions for two customers on the same rote (1) or from two different routes (2), and shift one customer to any position of the same route (3) or to any position in another route (4).

Starting from the initial solution referred to above, the author divides the overall approach in two phases. The first phase aims to reduce the number of used vehicles until a certain threshold, and for that reason, only movements performed between two routes are allowed (swap and shift moves involving two routes). At this stage, the evaluation/selection of the best move relies on the packed volume: one move is better than another if it leads to a smaller packed volume.

Once the referred to above threshold is reached, the second phase begins, aiming to reduce the total travel cost. At this point, all moves are allowed.

In order to avoid calculating the packing feasibility of all feasible moves, the author suggests generating all moves without evaluating the packing feasibility at the beginning. All moves are sorted by decreasing order of their quality, considering the phase in which they are performed. The feasibility of a route resulting from each move is verified, until a given number of feasible moves is reached.

As referred to above, to assess the packing feasibility of each route, a tree search algorithm using a depth-first search strategy is proposed. Each node provides information about the current packing, the items to place, and the list of potential placements. A list of the denominated current placements is derived from the list of potential placements taking into account filtering rules based both on the ranking of the items (which is greater if the item belongs to a customer to be visited later) and on the reference points of the placements. Each current placement is alternatively performed, and thus the set of items to place and the potential placements are updated.

This approach was tested in two sets: the 27 instances proposed in [63] and the set instances presented in [141]), reaching 35 new best solutions for both sets.

Concerning the first set of instances, the hybrid approach presents an average improvement of 8% compared to GTS, 3,9% compared to [141], and 0,8% compared to [62]. The second set of experiments shows an average improvement of 3,8%.

As in other works, the impact of the loading constraints was tested through the relaxation of one loading constraint type at a time for both sets of instances. The results for the first set of instances show that the average improvements obtained relaxing supporting constraints and sequence constraints are greater than relaxing the fragility constraints. Relaxing all these constraints leads to an average improvement of 9,7%. For the second set of instances, relaxing loading constraints has a higher impact.

3.3.5 3L-CVRP with time windows

Moura and Oliveira [115] developed two approaches for the 3L-CVRP with time windows: a sequential method and a hierarchical method. The sequential method consists in building a sequential candidate list (SCL) that is used by three distinct search methods. The first one is a Monte Carlo procedure that generates random solutions from the SCL. The second method is based on the local search heuristic through the 2-opt strategy whose neighborhood is composed by the SCL. Finally, the sequential approach is also composed by a GRASP procedure whose elite list is built according to a given ranking. The local search phase of this GRASP procedure is based on a 2-opt strategy.

The hierarchical approach is composed by three phases: constructive, post-constructive and local search. In the constructive phase, a GRASP algorithm is used only to build the routes. Then, a post-constructive phase tries to minimize the number of routes. In the local search phase, a 2-opt procedure is applied on the routes and between the routes.

The authors proposed a set of instances to test their approach by considering the variation of customers demand. Through the analysis of the results, the authors conclude that for a smaller number of boxes, sequential and hierarchical approaches are similar. However, if the number of boxes is higher, the GRASP algorithm from the sequential approach leads to better solutions.

Another approach for the 3L-CVRP with time windows is based in a multi-objective genetic algorithm [114]. The initial population is obtained through the application of the sequential approach referred to above [115]. The chromosome of this genetic algorithm is represented by a string of equal length of the sequential candidate list. Each gene of a given chromosome represents a candidate of this list.

To evaluate a given solution, two objectives are considered: the number of vehicles and the distance. These objectives are analyzed through a Pareto ranking procedure, which is used to build potential non-nominated solutions. From this list, two solutions will be randomly chosen to recombination.

The author reports on computational experiments using the instances described in [115]. The obtained results show that this multi-objective genetic approach is competitive with [115] by decreasing the number of used vehicles in some instances, especially in contrast with the hierarchical approach.

Bortfeldt and Homberger [23] tackled the same variant of the problem, by dividing the overall approach into three steps as follows. Firstly, only the loading packing is considered and thus, for each customer, a three-dimensional strip packing problem (3D-SPP) is solved, obtaining so many packing configurations as customers. Then, a vehicle routing problem with time windows (VRPTW) is solved considering that a route is only feasible if the sum of all loading lengths is not greater than the vehicle length. Lastly, a final packing is reached by sorting customers by the reverse order of visit and placing their loading configurations side by side.

In the first phase, each 3D-SPP is solved by a tabu search algorithm described in [21] whereas the VRPTW is solved by a hybrid heuristic including an evolutionary strategy to minimize the number of used vehicles, and a tabu search approach to minimize the total distance [75].

The computational experiments were performed with both the 46 instances proposed in [115] and a new set of 120 instances. By comparing with [115], this approach was able to save 51 vehicles in the total set of 46 instances, saving roughly 34% of the total distance.

3.3.6 Honey bee mating optimization

Recently, another nature-inspired search method for the resolution of the 3L-CVRP was proposed in [134], based on a Honey Bee Mating Optimization (HBMO) algorithm and motivated by the good results obtained by this algorithm for the CVRP [104, 105]. In this work, stability, fragility and sequential constraints were also considered.

The initial population of honey bees is obtained through a modified Greedy Randomized Adaptive Search Procedure (GRASP), denominated by Multiple Phase Neighborhood Search GRASP (MPNS-GRASP). As happens with the HBMO, the MPNS-GRASP was successfully applied to the CVRP [103]. The advantage of this modified version of the GRASP algorithm is that instead of using a single greedy function, the MPNS-GRASP uses different greedy functions in each iteration or even a combination of these functions. Furthermore, the Expanding Neighborhood Search (ENS) is used in order to provide more flexibility to the local search phase.

From the obtained initial population, the solution with the minimum cost is selected to represent the queen of the hive. All the other solutions are considered as drones. New broods, which are in fact new solutions, are obtained by crossing queen and drones genotypes. Then, local search is applied to each obtained solution, representing the workers of the hive. If a new solution improves the solution associated to the queen, then the queen is replaced by the new brood, which corresponds to the new best solution.

The overall approach consists in a hybrid algorithm which integrates the HBMO algorithm with six packing heuristics. While the HBMO intends to improve the solutions of the VRP in terms of distance and satisfying both the capacity and the volume constraints, the packing heuristics verify the loading feasibility of each route. These heuristics are based on those proposed in [141], and were already described above.

The authors used the 27 benchmark instances in order to compare the described approach with Tabu Search (TS) [63], Guided Tabu Search (GTS) [141] and Ant Colony Optimization (ACO) approaches [62]. The obtained results present an improvement when compared with those obtained with TS and GTS with an average improvement of 7,00% and 3,58%, respectively. When compared with the ACO, the hybrid approach improvement is less significant, with an average value of 0,68%.

3.3.7 3L-CVRP with heterogeneous fleet

A Variable Neighborhood Search (VNS) approach for the 2L-CVRP [151] was presented in Section 3.2.8. However, there is an earlier contribution that suggested a VNS algorithm for an extension of 3L-CVRP in which different types of vehicles were considered [150]. This is the first 3L-CVRP approach which explicitly considers this type of constraint.

Each type of vehicle can be described by a given weight capacity, by a certain fixed cost, and by values of height, length and width of the vehicle container. As a consequence, and since different vehicles represent distinct consumptions, the cost of travelling along a given edge relies on the vehicle traversing it. Therefore, the cost of a given route is represented as the sum of fixed cost with the cost of edges covered by a given vehicle.

The overall approach is called adaptive VNS and is similar to the one presented in [151]. We recall that such approach is composed by a VNS algorithm enhanced with diversification procedures when no better solutions can be found. The diversification is greater for larger number of iterations without improvements.

The loading feasibility of a route is assessed by one of two heuristics relying on the total number of items in the route. If this number is less than a given threshold defined by the user, then all permutations of items are used as input for an extreme point based heuristic [40]. Otherwise, a random local search procedure is used by swapping the position of two items in the sequence. Additionally, the authors implemented a tree data structure, denoted as trie, in order to keep the loading feasibility information as well as the number of times that such route is evaluated.

This adaptive VNS was tested in 27 instances presented in [63] and in 12 instances proposed in [141]. For the first set, the obtained results show an average improvement when compared with other works [63, 141, 62, 158, 20, 134]. From those works, only in [141, 158, 20] was the second set of instances tested. The adaptive VNS was able to produce better solutions in 10 instances, since two instances could not be solved because no solution was found complying with the allowed number of vehicles.

3.4 Applications of L-CVRP

Ceschia *et al.* [28] addressed a real-world problem from an industrial company that considers split deliveries. The sizes of the demanded items are not quite different and a heterogeneous fleet is considered. Load bearing constraints are also considered. The sequence constraint is adapted in order to achieve a problem closer to the real context. A so-called reachability constraint is added to the problem imposing a maximum distance between any item and the loading operator (or forklift), taking into account the current layout. To exemplify this situation, an example is described in Figure 3.6. If the maximum distance is greater than or equal d_1, but less than d_2, it is possible to unload item $I_{1,1}$ but it is impossible to unload item $I_{1,2}$. Consequently, this layout violates the reachability constraint.

The so-called robust stability constraints are also considered. The main difference between these constraints and the stability constraints is that each item must have a minimum supporting area, considering not only the item (or items) below and in

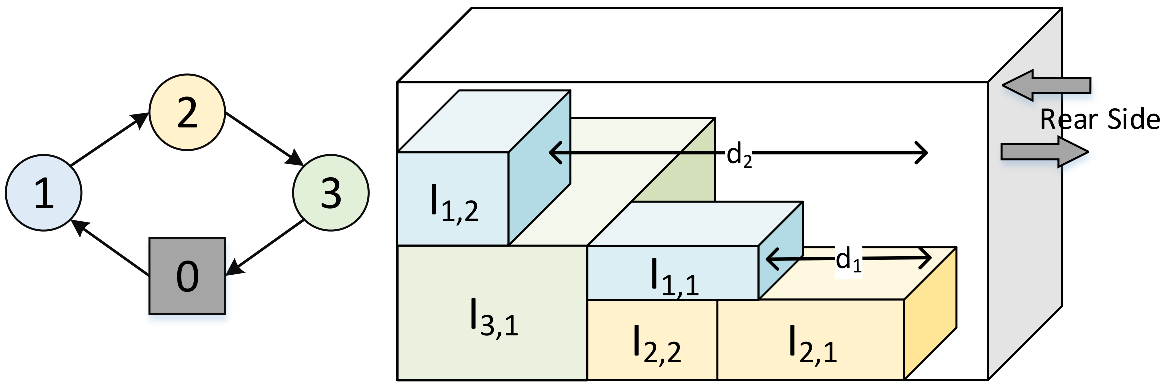

Figure 3.6: Reachability constraint (adapted from [28])

contact with it, but also all items in the stack below its surface. These types of constraints prevent leaning stacks of items.

The overall approach is based in alternating between a simulated annealing algorithm and a large neighborhood search. As in other approaches, movements for infeasible solutions are accepted (only for the loadings whose weight or volume are excessive) and penalized accordingly.

The approach was tested in a new set of real instances provided by the company referred to above. In order to establish comparisons with other approaches, the set of instances proposed in [63] was also used in the computational experiments, obtaining results competitive with those obtained in [63, 141].

Another application of the 3L-CVRP can be found in [3]. In this work, Aprile *et al.* addressed the 3L-CVRP arising in the production of sofas. Besides the three-dimensional packing constraints, no additional constraints were required, meaning that no sequential, fragility or supporting constraints were taken into account. The authors tackled this real problem in three steps. Firstly, a loading pre-computation phase is performed by packing the items of each customer separately at the vehicle. In practice, this phase is equivalent to solving a three-dimensional strip packing problem for each customer. At the second step, all pairs of combinations of customers are evaluated. The best matching configurations were used in the third phase, which is in fact a heuristic based in simulated annealing. The proposed algorithm was able to solve instances with up to 100 customers in roundly 90 seconds.

Attanasio *et al.* [5] addressed a case study of a company which produces semi-finished films in Italy. Sequential constraints, time windows and due dates are also considered in a multi-day scheduling. The authors considered two-dimensional items with the same size and can be rotated in the loading area. The authors suggested a simplified model in which additional cuts are added when a heuristic fails to prove the feasibility of a route.

Martínez and Amaya [109] addressed one problem provided by a company in their food delivery service. The items are circular since they correspond to Spanish dishes which are delivered in specific and circular pans with different sizes. Time windows and multiple trips were also considered. The authors developed two approaches. The first approach consists in a linear model. The second relies on a sequential insertion heuristic to find an initial solution and in a tabu search approach to improve such solution.

Junqueira and Morabito [81] proposed heuristic methods for a 3L-CVRP arising in a Brazilian carrier. The suggested methods were adapted from the combination of classical heuristics, and can tackle large-size instances.

3.5 Other routing problems with loading constraints

Other routing problems deal with different type of loading constraints when compared with those presented for the L-CVRP. In some of those problems, routing aspects are similar but the vehicle is replaced by a ship or the cargo is palletized and, consequently, the pallets are items to be loaded in the vehicles. In other situations, items may not be in contact and thus they cannot be transported in the same compartment or even in the same vehicle. However, other problems must combine loading constraints with pickup and delivery. In the following subsections, we review some of these routing problems with loading constraints.

3.5.1 Palletized items

Palletized items arise in many situations, such as the distribution of chip-boards. It is considered that demand of each customer is composed by different types of chip-boards. The chip-boards of the same type for the same customer are jointly palletized, performing an item. The items are delivered to customers using a fleet usually composed by the same type of vehicles, each one divided in a given number of piles (stacks). However, long chip-boards may occupy more than one pile, and thus different chip-board types may require a different number of occupied piles. Additionally, sequential constraints are imposed. Consequently, it is clear that some holes may arise in the layout, and this fact may cause load instability. In these cases, such holes may be filled by bulk material and thus supporting constraints can be ignored.

The described problem is denominated by One Vehicle Loading Problem (1-VLP). The first work addressing the 1-VLP is due to Doerner *et al.* [51]. The authors proposed a tabu search and an ant colony optimization algorithm. Tricoire *et al.* [144] proposed a Variable Neighborhood Approach (VNS) that provided an upper bound and possible valid cuts for a branch-and-cut algorithm. Both works referred to above were tested in real instances provided by an Austrian timber distribution company.

Another problem considering palletized items is presented in [155]. The demand of each customer is composed by three-dimensional items that are previously assigned to a given pallet. It is considered that one pallet is sufficient to carry the total demand of a given customer. The pallets are then loaded into the vehicle. As stated by the authors, this problem corresponds to solving, for each customer, an instance of 3L-CVRP with supporting constraints and variable orientation of items. Sequential constraints are not considered since the vehicle driver is able to reach the pallets.

3.5.2 Heterogeneous and conflicting items

Some routing problems deal with situations in which it is common to transport goods that are incompatible in the sense that they may not share the same space in the vehicle. In such cases, items may be transported in the same vehicle but in different compartments. The corresponding problem is known as the Multi-Compartment Vehicle Routing Problem (MC-VRP).

The main difference between MC-VRP and the 1-VLP defined in Section 3.5.1

is that items belonging to different compartments are not in contact, while in the MP-VRP one item may occupy more than a single pile. This feature may be suitable for transporting conflicting or strongly heterogeneous items within the same vehicle.

Some real applications for this problem can be found in the gasoline distribution, when a single vehicle transports different types of petrol products [38, 39, 25]. Additionally, legal requirements must impose that items must be transported in different compartments. One example of this situation is the transportation of provisions to farms where, despite the separation in different compartments, the food for a given specie must be loaded in the same compartment in future transportations, as referred to in [56].

Other situations arise when items need different transport conditions, as it happens when serving convenience stores with dry, refrigerated and frozen items within the same vehicle [30]. A similar problem is proposed in [11]. However, items may not be transported within the same vehicle and thus multiple trips must be performed. The authors denominated this problem as minimum multiple trip vehicle routing problem. Time windows for different type of items are also considered.

3.5.3 Loading and ship routing

Routing and loading scenarios are not limited to the road transportation. In fact, some loading constraints can be found in the maritime logistics. The Container Stowage Problem (CSP) consists in determining the way a set of containers must be placed in a set of possible positions of a container ship while minimizing the transportation cost. Additional constraints may be considered as the stability of the container ship, due dates and sequential constraints. On the other hand, the Ship Routing Problem (SRP) consists in a particular Capacitated Vehicle Routing Problem in which vehicles are replaced by ships.

In [116], the CSP was integrated with SRP resulting in the denominated Container Stowage and Ship Routing Problem (CSSRP). The authors considered the cases in which due dates are applied to the containers. They stated that, since the costs strongly rely on handling operations, incorporating sequential constraints (LIFO policy) naturally reduces the amount of time in unloading operations.

3.5.4 Auto-carrier transportation

The Auto-carrier transportation problem arises in the car industry when it is necessary to transport a set of vehicles or trucks from a depot to a set of customers (more precisely, car dealers). Usually, the fleet is heterogeneous and each auto-carrier is composed by upper and lower platforms to increase the number of transported cars. Trailers may be also considered. Additional constraints are considered in the literature, such as due dates and loading constraints.

Due to the features of both cars and auto-carriers, it is clear that sequential constraints play an important role: lateral shifts are not possible and cars can only be loaded or unloaded in the back side of the vehicle. Furthermore, translation and rotation operations may be performed to unload vehicles from upper platforms.

The first approach to the auto-carrier problem was presented in 2002 and it is due to Tadei *et al.* [139] who addressed a real problem which arises in an Italian company. This model is decomposed by assigning auto-carriers to geographical regions. For each one, a constructive heuristic computes a valid solution which is improved with first improvement local search.

There also recent works that tackled the auto-carrier transportation problem [49, 36]. It is noteworthy that the contributions in the literature for the auto-carrier problem have a strong focus on practical applications since the instances are provided by the vehicle production industry.

3.5.5 Pickup and delivery with loading

As stated in Section 2.2, PDP are divided in two classes: one in which the transportation is performed from or to the depot, and another one in which the transportation is performed between vertices. This section is devoted to the second class. The pickup and delivery problems with loading constraints in the field of the first class will be analyzed later.

For instance, the Pickup and Delivery Travelling Salesman Problem with Loading constraints (PDTSPL) considers the transportation from pickup vertices to delivery vertices and the assumption that pickups and deliveries are performed according to a LIFO or a FIFO policy. Since only one dimension is considered, the vehicle can be seen as a single stack where the items are placed on its top. Consequently, LIFO constraints are only satisfied if when serving one customer, the item for this customer is on top of the stack, *i.e.*, no items are between the item to be unloaded and the rear side of the vehicle. On the other hand, FIFO constraints impose that items are unloaded by the sequence in which they were loaded.

Some works extended the PDTSPL with LIFO constraints to the case in which two routes are considered: one to collect items from pickup customers and another one to deliver them to delivery customers. Additionally items are placed on multiple stacks. These problems are known as Double Travelling Salesman with Multiple Stacks (DTSMS). A survey for PDTSL and DTSMS can be found in [127]. Few contributions addressed pickup and delivery problems with two- or three-dimensional items. These problems will be analyzed in depth in Chapter 7.

3.6 Conclusion

In this chapter, a survey in routing problems with loading constraints was presented. Considering the L-CVRP, three aspects are worth noting. Firstly, and regardless of the complexity of the problem, the number of contributions for the L-CVRP along the recent years is increasing. Together with the growth of publications in this field, the number of new optimal solutions for benchmark instances is also growing. Secondly, the L-CVRP has diverse practical application. In this chapter, some real-world situations arising in the context of L-CVRP were presented. These applications gave rise to methods which were applied in real instances. Finally, and due to inherent complexity of this problem, the vast majority of approaches are based on heuristic methods. Solving the L-CVRP to optimality for large size instances remains a true challenge.

Chapter 4

Variable neighborhood search for the elementary shortest path problem with loading constraints[1]

Contents

[1]The contribution presented in this chapter has been published in [125]: T. Pinto, C. Alves, and J. Valério de Carvalho. Variable Neighborhood Search for the Elementary Shortest Path Problem with Loading Constraints, *Lecture Notes in Computer Science*, volume 9156, book section 34, pages 474-489. Springer International Publishing, 2015. ISBN 978-3-319-21406-1.

4.1 Introduction

In this chapter, we address the elementary shortest path problem with 2-dimensional loading constraints. The aim is to find the path with the smallest cost on a graph where the nodes represent clients whose items may have different heights and widths. Beyond its practical relevance, this problem appears as a subproblem in vehicle routing problems with loading constraints where feasible routes have to be generated dynamically. To the best of our knowledge, there are no results reported in the literature related to this problem. Here, we explore a variable neighborhood search approach for this problem. The method relies on constructive heuristics to generate feasible paths, while improved incumbents are sought in different neighborhoods of a given solution through a variable neighborhood search procedure. The resulting variants of the algorithm were tested extensively on benchmark instances from the literature. The results are reported and discussed at the end of the chapter.

4.2 Elementary shortest path problem with 2-dimensional loading constraints

The elementary shortest path problem with loading constraints arises in particular in the context of vehicle routing problems where the items have at least two dimensions, and the capacity of the vehicles is a strong constraint. The problem occurs typically when feasible routes have to be generated dynamically as happens for example in column generation based approaches.

Some attempts in solving the 2L-CVRP exactly through column generation are reported in [124], while this approach is seen by Iori and Martello in [77] as worthwhile of investigation in order to determine effectively exact solutions for this problem. The Dantzig-Wolfe decomposition principle on which column generation approaches are based has been extensively applied to vehicle routing problems. The standard reformulation that results from this decomposition is a set partitioning problem which is solved typically by dynamic column generation. The related pricing subproblem is an elementary shortest path problem with additional resource constraints. This problem has received much attention in the literature [58, 132, 133]. Applying the Dantzig-Wolfe decomposition to the L-CVRP yields also a set partitioning problem which can be solved through column generation. The pricing subproblem remains an elementary shortest path problem, but now the resource constraints become 2- or 3-dimensional packing constraints which are much more difficult to handle than other standard constraints as the capacity constraints, for example.

To the best of our knowledge, the elementary shortest path problem with loading constraints has never been addressed in the literature. In this chapter, we describe and analyze a solution approach based on variable neighborhood search for the problem with 2-dimensional items and sequential constraints. To generate feasible routes, we use different constructive methods that handle the packing part of the problem through alternative strategies based on bottom-left and level packing placement rules. Local search is supported on several neighborhoods defined from both the routing and packing definition of the solutions. The combination of the different strategies described in this chapter leads to different variants of the variable neighborhood search algorithm. The performance of these variants is evaluated and compared through extensive computational experiments on benchmark instances from the literature for the 2L-CVRP.

The elementary shortest path problem with 2-dimensional loading constraints (2L-ESPP) is defined on a graph $G = (V, E)$ with the set of nodes V representing the n clients of the problem plus the depot 0 from which the vehicle leaves initially and to which it should come back at the end of the visits, and E representing the set of edges of the graph. The travelling cost associated to the edges $(i, j) \in E$ will be denoted by c_{ij}. The loading area of the vehicle has a total width denoted by W and a maximum height of H units. Each client $i \in V \setminus \{0\}$ has b_i 2-dimensional items of width and height respectively equal to w_i and h_i. The visit of a client implies that all its items are loaded on the vehicle. Hence, we assume that the load associated to any client $i \in V \setminus \{0\}$ fits in the vehicle. The 2L-ESPP consists in finding the minimum cost route for the vehicle that starts and ends at the depot 0 and that visits at most once the clients in V. Note that here we assume that there are negative costs for some edges of the graph, which happens typically when the problem is defined as a pricing subproblem of a column generation model for the 2L-CVRP.

In this chapter, we address the case where the items of the clients have a fixed rotation, and where the sequential or LIFO constraint applies. The latter implies that, during the loading and unloading of the items of a given client, the items of all the other clients that are already in the vehicle cannot be moved. Furthermore, lateral movements of the items inside the vehicle are forbidden. Hence, the loading and unloading operations can only be done in a direction that is parallel to the left and right sides of the vehicle. Figure 4.2 illustrates the case of a feasible and an infeasible loading pattern for the instance of Example 1 according to this sequential constraint. In the example represented in this figure, the sequence of visits is $(0, 2, 1, 0)$. However, in the pattern (b) of Figure 4.2, one of the items of client 2 cannot be unloaded without moving first the items of client 1.

A solution for the 2L-ESPP is defined as a sequence of clients that starts and ends at the depot 0, together with a placement position for each item of the clients that are visited. Alternatively, the latter can be replaced by defining the sequence by which the items of each client should be loaded on the vehicle, and by defining the placement rule that is used. Let S denote the sequence of clients visited in a solution of the 2L-ESPP. We have

$$S = (s_1, s_2, \ldots, s_{|S|}),$$

with $s_1 = s_{|S|} = 0$, while $s_2, \ldots, s_{|S|-1}$ represent the clients visited by the corresponding route. The cost of the solution associated to S will be denoted by $z(S)$, *i.e.* $z(S) = \sum_{i=1}^{|S|-1} c_{s_i s_{i+1}}$. Moreover, let P define the order by which the items of the clients of S are placed in the vehicles, such that

$$P = (p_2, \ldots, p_{|S|-1}),$$

with $p_i = (p_i^1, p_i^2, \ldots, p_i^{b_{s_i}})$, $i = 2, \ldots, |S| - 1$, and p_i^j representing the index of the j^{th} item of client s_i to be placed in the vehicle.

The following example illustrates through a small instance the details related to the definition of the 2L-ESPP and its solutions.

Example 1 *Consider the instance of the 2L-ESPP represented in Figure 4.1. The set of nodes V is composed by $n = 4$ clients and the depot 0, i.e. $V = \{0, 1, 2, 3, 4\}$. The costs c_{ij}, $(i, j) \in E$, are shown beside the edges of the graph. The items of each client are identified through the tuple (i, k), with i representing the item and k the index of the item ($k = 1, \ldots, b_i$). A feasible solution for this instance is the following:*

$$S = (0, 2, 1, 0) \qquad \text{and} \qquad \mathrm{P} = ((1, 3, 2), (1, 2)).$$

assuming that a standard bottom-left rule is used to place the items. The corresponding loading pattern is illustrated in Figure 4.2-(a). The cost of this solution is $z(S) = -5$. It is easy to see that all the items of the clients can be unloaded from the vehicle without lateral movements or moving the items of the other clients. Figure 4.2-(b) shows an alternative loading pattern that violates this sequential constraint.

4.3 Feasible solutions with constructive heuristics

To build an initial solution for the problem, we adopted a constructive approach in which the clients are added one by one to the route. The next client to be evaluated is inserted in the current route only if all its items can be loaded on the vehicle according to the constraints that apply to the loading patterns. The clients are evaluated following a nearest neighbor approach. Starting from the depot, two different strategies were considered to select the first client:

(FN) the first client to be evaluated is the one that is nearest to the depot 0 (*i.e.* $argmin\{c_{0i} : i = 1, \ldots, n\}$);

(FR) the first client to be evaluated is selected randomly.

The acronyms (FN) and (FR) used above will be used later to distinguish between the variants that we obtain by using one of these two strategies. After selecting the first client, the remaining ones are evaluated by non-decreasing order of the cost of the edge that connects them to the last client in the current route. The client that is evaluated is inserted in the route if all its items can be loaded on the vehicle. If no

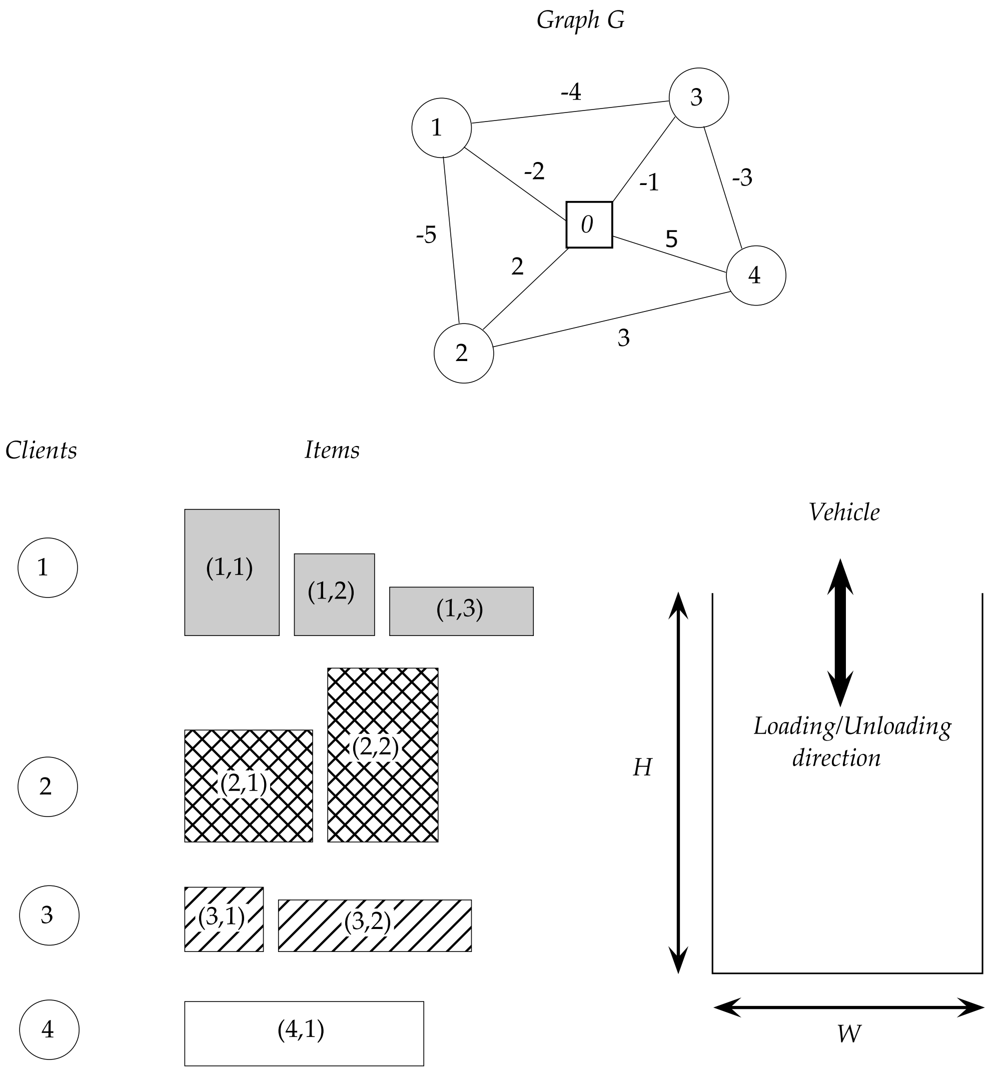

Figure 4.1: Instance of the 2L-ESPP (Example 1)

more clients can be inserted, the route is closed by connecting the last client to the depot.

The hardest part when building a solution for the 2L-ESPP is to find (if it exists) a feasible arrangement of the items on the vehicle such that there are no overlaps, all the items are put inside the boundaries of the vehicle without rotation, and the sequential constraint is satisfied. To address this issue, we considered three different strategies:

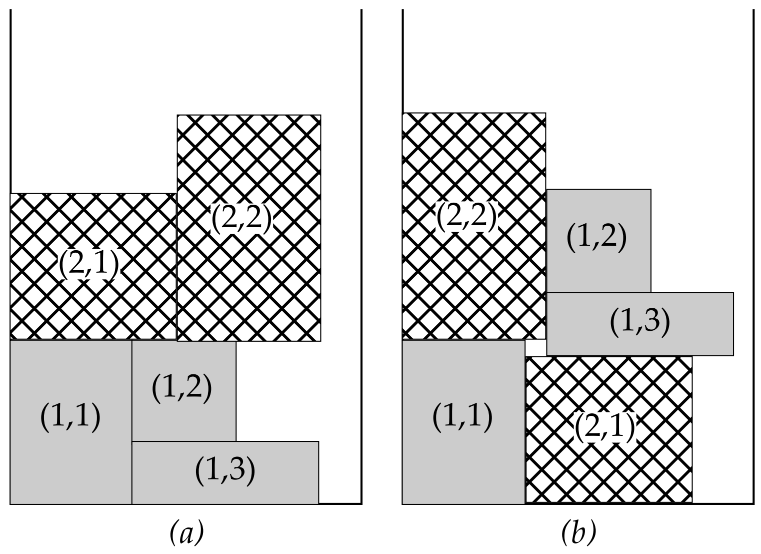

Figure 4.2: Feasible (a) and infeasible loading pattern (b) (Example 1)

(BL) a standard bottom-left placement rule;

(RBL) a revised bottom-left procedure;

(LP) a level packing approach.

The strategy (BL) consists in placing the next item in the bottom and leftmost free position of the vehicle that ensures that all the loading constraints that apply are satisfied. Each time an item is placed on the vehicle, at most four orthogonal free spaces are generated identifying the different areas of the vehicle where the next items can be placed. In turn, after placing an item in a free space, this free space (and all the others that are intersected by the item) is removed, and replaced by an updated set of free spaces. Placing an item according to the strategy (BL) is equivalent to finding the free space whose width and height are equal to or larger than the size of the item, and whose bottom and leftmost position is the smallest among all the free spaces provided that it leads to a feasible placement. After finding this free space, the item is placed in its bottom and leftmost position. Figure 4.3 illustrates the concept of free space. Free space 1 and 2 (black dotted lines) are the free spaces respectively at the left and right of the item generated after its placement on the vehicle. Free spaces 3 and 4 (grey dashed lines) are the free spaces respectively above and below the item.

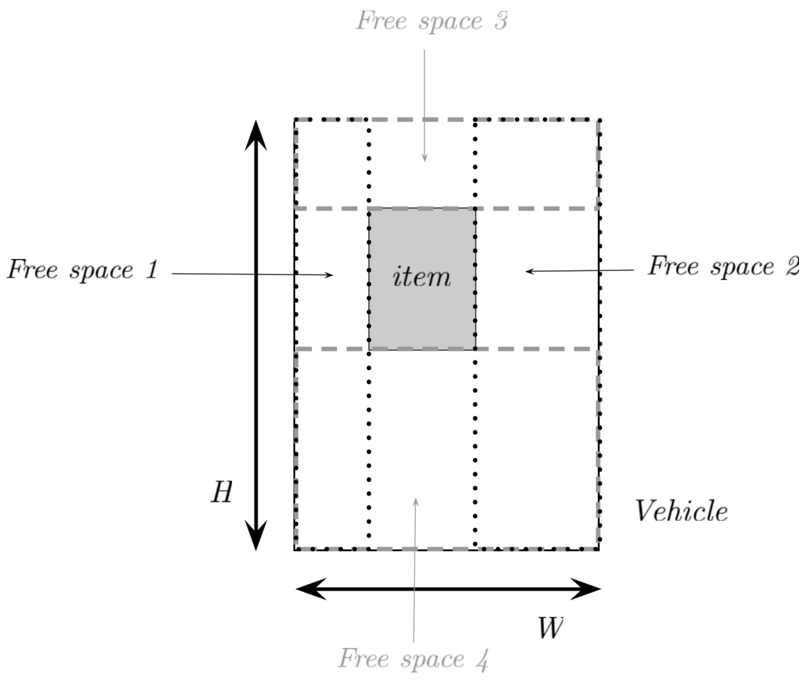

Figure 4.3: Free spaces

The revised bottom-left procedure (RBL) consists in selecting the free space where the item produces the best fit, and then in placing the item in the bottom and leftmost position of this free space. The free space with the best fit is the one whose area is the nearest to the area of the item. The idea is to place the items in the areas of the vehicle where the packing generates the least possible waste, thus favouring the filling of holes. Again, only the free spaces whose bottom and leftmost position yields a feasible placement are considered for selection.

The last strategy (LP) that we explored consists in placing the items of the selected client in horizontal levels, and then in placing the levels on the vehicle one above another. The first item to be placed in a level determines the maximum height of the items that can be placed after it in that level. To select the level where the next item will be placed, we considered the following two strategies:

($LP.FF$) the next item is placed in the first open level where it fits;

($LP.BF$) the next item is placed in the open level where it best fits.

The level that was placed in the upmost position of the vehicle for the previous client is considered as an open level when placing the items of the next client. After placing all the items on the levels, the levels are placed on the vehicle so that the one with the largest remaining space is placed in the upmost position. One of the advantages of level packing approaches is that they ensure that the patterns satisfy the sequential constraint, thus avoiding the necessity of checking the placement positions before placing the items. The loading patterns resulting from this level packing approach are similar to the cutting patterns that arise in 2-dimensional guillotine cutting stock problems.

After choosing a client to insert in the route, its items are selected one by one, and placed in the vehicle according to the strategies described above. The next item to be placed is selected according to two different orderings based on the following criteria:

(OH) height of the items;

(OA) area of the items.

In both cases, the items are ordered in non-increasing order of their height and area, respectively.

4.4 Variable neighborhood search algorithm

The strategies described in the previous section yield different variants of a constructive method for building feasible solutions for the 2L-ESPP. In order to improve the solutions generated by these algorithms, we developed a local search approach embedded into a variable neighborhood search (VNS) algorithm. The VNS metaheuristic was described first by Mladenovic and Hansen in [112], and since then, it has been applied with success to solve different combinatorial optimization problems [72]. The aim of VNS is to explore systematically different neighborhoods of the solutions to diversify the search and escape from local optima. Here, VNS is used to drive the search into 7 alternative neighborhoods of the solutions generated through the constructive algorithms arising from the combination of the different strategies described above. In the following section, we define the neighborhood structures that we used to support the local search procedures. The details of our VNS algorithm are given in Section 4.4.2.

4.4.1 Neighborhood structures

The representation of the solutions of the 2L-ESPP defined in Section 4.2 relies on two main elements: the sequence by which the clients are visited in a given route, and the characterization of the loading pattern used to arrange the corresponding items in the vehicle so that all the loading constraints that apply (no overlaps, fixed orientation and the sequential constraint) are satisfied. To explore the search spaces defined through these two aspects of the solutions, we defined 7 neighborhood structures that can be divided into routing neighborhoods and packing neighborhoods. The definition of the neighborhood structures relies on the constructive algorithms defined above. Let CH denote the constructive heuristic used to build the initial solution for the instance, and which is obtained by combining the strategies described in Section 4.3. In our implementation, we assumed that the constructive heuristic CH used to generate the initial solution is also the one that is used to define the neighbors of a given solution. When generating the neighbors, the difference is that part of the sequences of clients (and the corresponding sequence by which the items are placed in the vehicle) is fixed. The neighborhood structures are defined in the sequel.

Routing neighborhoods

NS_1 **Swapping two clients in the route**

Given a solution whose sequence of visited clients is S (as defined in Section 4.2), the neighbors of this solution consist in all the feasible solutions obtained by swapping two clients in this route, by applying CH with the resulting sequence of clients (keeping the sequence of items for each client), and by adding more clients from the last one forward in the route using CH. Let S_c denote the sequence of clients in the route associated to the current solution, such that:

$$S_c = (s_1, s_2, s_3, \ldots, s_{|S_c|-1}, s_{|S_c|}),$$

One of the neighbors of this solution obtained by swapping s_2 and $s_{|S_c|-1}$ is the following:

$$S'_c = (s_1, s_{|S_c|-1}, s_3, \ldots, s_2, s_{|S_c|}),$$

provided that the items of the clients in S'_c can be put in the vehicle using CH, and no more clients can be added at the end of S'_c. Moreover, if P_c defines the sequences by which the clients of S_c are placed in the vehicle (again as defined in Section 4.2), *i.e.*

$$P_c = (p_2, p_3, \ldots, p_{|S_c|-1}),$$

then the corresponding sequences of items associated to S'_c will be

$$P'_c = (p_{|S_c|-1}, p_3, \ldots, p_2).$$

NS_2 **Shifting a client in the route**

The neighbors of a solution whose sequence of clients is S are obtained by choosing an item and placing it in a different position of the sequence, by applying CH with the resulting sequence of clients (keeping the sequence of items for each client), and by adding more clients from the last one forward in the route using CH. The following solution is a neighbor of the solution S_c defined above obtained by selecting the client s_2 and by placing it in the third position of the sequence, *i.e.*

$$S'_c = (s_1, s_3, s_2, \ldots, s_{|S_c|-1}, s_{|S_c|}).$$

Again, in this case, we are assuming that no more clients can be added at the end of the route by applying CH.

NS_3 **Removing a client from the route**

The neighbors of a solution are obtained by removing a client from the route, by applying CH with the resulting sequence of clients in the same conditions as in the previous neighborhood structures, and by inserting clients at the end of the sequence (before the depot and if they fit in the vehicle) using again CH. Considering the solution defined above S_c, a neighbor can be obtained by removing s_2, and by adding a new client s_{new} at the end of the sequence using CH, assuming that s_{new} is not in S_c, *i.e.*

$$S'_c = (s_1, s_3, \ldots, s_{|S_c|-1}, s_{new}, s_{|S_c|}).$$

NS_4 **Removing a client and all its successors from the route**

The neighbors of a solution are obtained by removing all the clients from a given position of the sequence up to the end, by adding a selected client at the end of the sequence and by placing its items using CH. The vehicle is filled by applying strictly the heuristic CH starting from the last client that was inserted. Considering again the solution S_c, a neighbor is obtained by removing all the clients from s_3 up to the end, and by adding a new client s_{new} using CH, assuming that this new client is not in S_c, *i.e.*

$$S'_c = (s_1, s_2, s_{new}, s_{|S_c|}).$$

NS_5 **Exchanging a client by another in the route**

The neighbors of a solution are obtained by exchanging a client by another that is not in the sequence, by placing the items of the clients in the resulting sequence using CH, and by adding other clients at the end of the sequence (before the depot) using again CH. Note that the sequence by which the items of the clients are placed in the vehicle remains unchanged for all the clients that were already in the route. A possible neighbor of the solution S_c defined above is obtained by exchanging s_3 by a new client s_{new} that is not in S_c, as follows:

$$S'_c = (s_1, s_2, s_{new}, \ldots, s_{|S_c|-1}, s_{|S_c|}).$$

In this case, it is also assumed that no more clients can be added at the end of sequence using CH.

Packing neighborhoods

NS_6 **Swapping two items**

The neighbors of a solution are obtained by selecting a client in the route and by swapping two items in the sequence that defines the order by which its items are placed in the vehicle. Then, the heuristic CH is used to build the solution that corresponds to these sequences of clients and items (if possible), and to add other clients at the end of the sequence (before the depot) if they fit in the vehicle. Let S^c be the sequence of visited clients in the current solution, and let P^c denote the corresponding sequences by which the items are placed in the vehicle. As an example, let S^c and P^c be defined respectively as follows:

$$S_c = (s_1, s_2, s_3, s_4, s_5),$$

with $s_1 = s_5 = 0$, and

$$P_c = ((p_2^1, p_2^2), (p_3^1, p_3^2, p_3^3), (p_4^1)).$$

A possible neighbor S'_c of this solution is defined as follows:

$$S'_c = S_c \qquad \texttt{and} \qquad P'_c = ((p_2^1, p_2^2), (p_3^2, p_3^1, p_3^3), (p_4^1)).$$

It is obtained by swapping the first and second item of s_3 in P_c, provided that all the items can be placed according to P'_c in the vehicle by applying CH, and that no more clients can be added at the end of the sequence.

NS_7 **Shifting an item**

The neighbors of a solution are obtained by selecting a client in the route and by placing one of its items in a different position in the sequence that defines the order by which the items of this client were inserted in the vehicle. As in the previous neighborhood structure, the heuristic CH is used to build the solution associated to these sequences of clients and items. The same heuristic is used to add other clients at the end of the sequence and before the depot, if possible. The following solution S'_c is a neighbor of the solution S_c defined above for NS_6, and it is obtained by placing the third item of s_3 in the first position, *i.e.*

$$S'_c = S_c \qquad \texttt{and} \qquad P'_c = ((p_2^1, p_2^2), (p_3^3, p_3^1, p_3^2), (p_4^1)).$$

4.4.2 Variable neighborhood search

To explore the search spaces defined through the neighborhood structures described in the previous section, we developed a variable neighborhood search algorithm that applies local search on these 7 neighborhoods. The initial solution is generated by applying one of the constructive heuristics that result from the combination of the different strategies described in Section 4.3, namely $\{(FN), (FR)\}$, $\{(BL), (RBL), (LP)\}$, $\{(LP.FF), (LP.BF)\}$ (if (LP) has been selected), and $\{(OH), (OA)\}$. Then, the 7 neighborhoods are explored in cycle until a maximum computing time limit is reached. A solution is generated in a shaking phase from the current incumbent solution in the neighborhood that is being explored, and a local search procedure is applied right after in the same neighborhood to determine an improved solution. In our implementation, we resorted to a first improvement local search procedure that stops when it finds a solution that is better than the solution generated in the shaking phase, or if no better solution exists in this neighborhood. Note that all the solutions that are explored are necessarily feasible solutions for the problem. Our variable neighborhood search algorithm is described in Algorithm 1. The constructive heuristic is denoted by `findInitialSolution()`, while the shaking and local search procedures are represented respectively by `shaking`$((S, P), NS_k)$ and `firstImprovement`$((S', P'), NS_k)$, with (S, P) denoting the current incumbent solution, (S', P') the solution generated in the shaking phase, and NS_k the neighborhood that is being explored.

4.5 Computational experiments

To evaluate and compare the performance of the different variants of our variable neighborhood search algorithm, we conducted an extensive set of computational experiments on 180 benchmark instances of the 2L-CVRP used by Iori *et al.* in [78] and by Gendreau *et al.* in [64]. Note that, in the former, the largest instances were not used due to their complexity. The number n of clients of these instances ranges from 15 up to 255, while the total number of items varies between 15 and 786. A complete description of the instances can be found in [64]. For our experiments, we multiplied all the costs (distances) associated to the edges by -1. The algorithms were coded

Algorithm 1: Variable neighborhood search algorithm

```
Input:
    $I$: instance of the 2L-ESPP;
    $CH$: constructive heuristic defined from the combination of the different
    strategies $\{(FN),(FR)\}$, $\{(BL),(RBL),(LP)\}$, $\{(LP.FF),(LP.BF)\}$
    (if $(LP)$ has been selected), and $\{(OH),(OA)\}$;
    Set of neighborhood structures $NS = \{NS_1, NS_2, \dots, NS_7\}$;
    Limit $t_{max}$ on the total computing time;
Output:
    Feasible solution $(S,P)$ of value $z(S)$;
$(S,P)$:=findInitialSolution();
repeat
    $k := 1$;
    while $k \leq 7$ do
        $(S',P')$ := shaking($(S,P), NS_k$);
        $(S'',P'')$ := firstImprovement($(S',P'), NS_k$);
        if $z(S'') \leq z(S)$ then
            $(S,P) := (S'',P'')$;
            $k := 1$;
        end
        else
            k:=k+1;
        end
    end
until cpuTime() $> t_{max}$;
return $(S,P)$ ;
```

in C++, and the tests were run on a PC with an i7 CPU with 2.9 GHz and 8 GB of RAM.

We tested the 16 variants of our algorithm resulting from the combination of the strategies described in Section 4.3, namely $\{(FN),(FR)\}$, $\{(BL),(RBL),(LP)\}$, $\{(LP.FF),(LP.BF)\}$ (if (LP) has been selected), and $\{(OH),(OA)\}$. The instances were divided in 22 groups according to the number of clients. The average results for each group are reported in Tables 4.1-4.4. In Table 4.1, we report on the results obtained with the standard bottom-left placement rule (BL) for all the possible combinations of strategies involving (FN), (FR), (OH) and (OA). Table 4.2 provides the results achieved with the revised bottom-left placement rule (RBL), while Table 4.3 gives the results when the level packing procedure (LP) is used with $(LP.FF)$. In table 4.4, we report on the results of the combination of the level packing procedure (LP) with the best-fit rule $(LP.BF)$. All the tests were run with a maximum computing time limit of 3 seconds. The meaning of the columns in these tables is the following:

n: number of clients;

M: average number of items;

$inst$: number of instances in the corresponding group;

ord: criterion used to order the items of a client ((OH) or (OA));

z_{CH}: average value of the initial solution generated using the constructive heuristic resulting from the combination of the strategies described in Section 4.3;

$\%_{fill}^{CH}$: average percentage of space used in the vehicle by the initial solution generated using the constructive heuristic;

z_{VNS}: average value of the best solution obtained with the variable neighborhood search algorithm described in Algorithm 1;

$\%_{fill}^{VNS}$: average percentage of space used in the vehicle by the best solution obtained with the variable neighborhood search algorithm;

imp: percentage of improvement achieved with the variable neighborhood search algorithm, *i.e.* $imp = (z_{VNS} - z_{CH})/z_{CH}$.

Additionally, in Tables 4.1-4.3, the average results for all the instances are reported in the line *avg*.

The constructive heuristic runs typically in a very few milliseconds, and hence most of the computing time is spent in the local search phase of the algorithm. Despite the small value used in our experiments for the maximum computing time, the results show that the variable neighborhood search algorithm can improve significantly the

				(FN)					(FR)				
n	M	*inst*	*ord*	z_{CH}	$\%^{CH}_{fill}$	z_{VNS}	$\%^{VNS}_{fill}$	*imp*	z_{CH}	$\%^{CH}_{fill}$	z_{VNS}	$\%^{VNS}_{fill}$	*imp*
15	31,1	10	(OH)	-335,30	61,26	-349,30	65,07	4,18	-318,90	61,21	-341,30	63,87	7,02
20	39,5	10		-423,80	60,82	-483,00	67,33	13,97	-435,30	63,02	-475,70	66,56	9,28
21	39,4	10		-561,90	63,93	-621,70	69,91	10,64	-542,70	65,63	-594,00	66,81	9,45
22	39,4	10		-940,80	62,23	-1076,70	68,42	14,45	-966,50	64,32	-1049,40	68,26	8,58
25	56,0	5		-532,20	60,12	-561,80	66,56	5,56	-452,00	61,12	-550,60	67,96	21,81
29	57,8	10		-984,70	63,54	-1137,00	69,39	15,47	-969,90	62,10	-1149,50	65,53	18,52
30	63,8	5		-512,40	62,72	-561,00	69,22	9,48	-497,60	65,70	-556,20	69,92	11,78
32	62,5	15		-1718,87	61,84	-1831,33	69,02	6,54	-1619,07	60,55	-1858,00	69,52	14,76
35	74,4	5		-576,20	60,74	-648,80	68,46	12,60	-554,40	62,68	-639,80	63,40	15,40
40	79,2	5		-611,60	57,78	-754,20	71,58	23,32	-643,60	63,26	-766,80	69,46	19,14
44	86,2	5		-1250,40	58,76	-1432,00	70,46	14,52	-1262,80	63,70	-1431,00	70,26	13,32
50	105,2	5		-770,80	62,14	-853,60	70,30	10,74	-717,20	63,42	-826,20	71,18	15,20
71	146,0	5		-542,00	61,40	-607,00	70,20	11,99	-548,60	68,76	-602,20	72,96	9,77
75	150,3	20		-1068,45	64,47	-1215,80	70,75	13,79	-1038,85	63,16	-1201,70	70,69	15,68
100	204,3	15		-1399,80	65,99	-1561,00	72,73	11,52	-1376,00	65,90	-1554,40	72,47	12,97
120	245,6	5		-2552,20	70,72	-2659,00	73,26	4,18	-2483,40	68,16	-2711,20	73,06	9,17
134	271,4	5		-2545,80	70,42	-2805,00	73,52	10,18	-2545,80	69,58	-2748,20	76,90	7,95
150	294,4	5		-1880,40	70,14	-2026,40	74,20	7,76	-1838,20	65,48	-2025,80	71,90	10,21
199	399,6	15		-2308,40	67,25	-2490,47	74,90	7,89	-2225,07	68,77	-2442,20	72,68	9,76
240	484,8	5		-1133,00	64,64	-1247,80	74,98	10,13	-1138,20	73,66	-1228,40	76,54	7,92
252	504,4	5		-1439,60	65,14	-1521,00	77,68	5,65	-1416,60	62,96	-1516,80	77,80	7,07
255	509,0	5		-1022,00	67,26	-1124,80	77,54	10,06	-1030,80	71,20	-1092,00	77,42	5,94
			avg.	**-1141,39**	**63,79**	**-1253,12**	**71,16**	**10,67**	**-1119,16**	**65,20**	**-1243,70**	**70,69**	**11,85**
15	31,1	10	(OA)	-331,40	60,03	-351,40	62,74	6,04	-288,30	57,59	-348,00	65,21	20,71
20	39,5	10		-412,10	60,50	-477,10	66,77	15,77	-398,90	63,40	-468,90	66,70	17,55
21	39,4	10		-546,50	61,69	-612,10	67,42	12,00	-554,40	62,60	-619,30	69,18	11,71
22	39,4	10		-958,20	64,97	-1042,20	68,73	8,77	-914,90	60,91	-1052,10	67,20	15,00
25	56,0	5		-539,80	64,60	-576,20	67,60	6,74	-498,20	63,44	-581,40	70,00	16,70
29	57,8	10		-947,70	62,79	-1146,50	69,05	20,98	-980,80	62,76	-1145,10	67,06	16,75
30	63,8	5		-519,00	64,88	-541,60	66,46	4,35	-505,40	62,72	-542,20	67,98	7,28
32	62,5	15		-1718,93	60,99	-1852,87	67,44	7,79	-1617,80	61,50	-1891,47	69,35	16,92
35	74,4	5		-614,40	61,28	-651,00	66,64	5,96	-622,00	65,72	-666,80	69,14	7,20
40	79,2	5		-669,40	64,90	-727,60	70,74	8,69	-690,20	63,78	-740,20	69,06	7,24
44	86,2	5		-1250,40	58,76	-1468,00	68,66	17,40	-1226,40	64,44	-1411,40	67,84	15,08
50	105,2	5		-777,60	66,76	-826,60	66,28	6,30	-763,60	67,10	-865,40	70,86	13,33
71	146,0	5		-542,80	59,98	-587,60	72,30	8,25	-550,40	61,26	-614,60	71,64	11,66
75	150,3	20		-1098,30	67,02	-1211,10	73,04	10,27	-1032,45	63,61	-1207,85	70,57	16,99
100	204,3	15		-1411,00	67,97	-1575,00	72,86	11,62	-1414,87	70,09	-1573,73	72,59	11,23
120	245,6	5		-2548,60	71,56	-2706,60	73,10	6,20	-2389,20	69,76	-2680,60	72,44	12,20
134	271,4	5		-2562,40	71,24	-2724,00	71,46	6,31	-2365,40	71,04	-2710,40	72,84	14,59
150	294,4	5		-1884,80	70,00	-2003,80	74,18	6,31	-1788,00	66,22	-1965,20	75,50	9,91
199	399,6	15		-2286,27	69,96	-2448,40	76,00	7,09	-2252,60	70,74	-2477,67	74,36	9,99
240	484,8	5		-1157,00	71,94	-1227,00	77,50	6,05	-1153,20	73,22	-1249,60	76,54	8,36
252	504,4	5		-1427,60	60,56	-1529,60	76,60	7,14	-1443,40	73,22	-1551,00	77,56	7,45
255	509,0	5		-1044,40	75,38	-1093,60	78,94	4,71	-1018,20	73,76	-1084,20	76,78	6,48
			avg.	**-1147,66**	**65,35**	**-1244,54**	**70,66**	**8,85**	**-1112,21**	**65,86**	**-1247,60**	**70,93**	**12,47**

Table 4.1: Computational results with (BL)

				(FN)					(FR)				
n	M	$inst$	ord	z_{CH}	$\%_{fill}^{CH}$	z_{VNS}	$\%_{fill}^{VNS}$	imp	z_{CH}	$\%_{fill}^{CH}$	z_{VNS}	$\%_{fill}^{VNS}$	imp
15	31,1	10	(OH)	-269,80	48,18	-339,30	61,98	25,76	-278,90	52,02	-339,60	61,92	21,76
20	39,5	10		-385,90	50,06	-460,50	61,63	19,33	-392,00	53,38	-452,70	60,74	15,48
21	39,4	10		-538,00	57,77	-595,70	64,20	10,72	-519,50	53,51	-607,70	67,10	16,98
22	39,4	10		-934,50	57,98	-1049,00	66,27	12,25	-890,50	60,76	-1034,90	63,75	16,22
25	56,0	5		-519,60	56,20	-571,80	67,92	10,05	-386,00	42,70	-552,20	68,46	43,06
29	57,8	10		-880,70	53,33	-1115,20	65,33	26,63	-893,10	56,23	-1087,90	66,37	21,81
30	63,8	5		-472,20	57,54	-552,80	71,52	17,07	-467,80	50,16	-553,00	64,96	18,21
32	62,5	15		-1564,73	54,04	-1828,80	65,40	16,88	-1561,93	55,82	-1881,33	67,51	20,45
35	74,4	5		-570,40	51,24	-648,20	65,74	13,64	-518,20	58,04	-653,40	65,90	26,09
40	79,2	5		-603,20	49,22	-703,00	68,38	16,55	-642,20	53,62	-712,80	67,58	10,99
44	86,2	5		-1243,00	53,34	-1409,40	67,56	13,39	-1119,20	50,90	-1378,20	67,04	23,14
50	105,2	5		-752,80	59,20	-790,80	66,22	5,05	-688,00	56,20	-797,20	67,94	15,87
71	146,0	5		-516,20	49,46	-609,20	68,18	18,02	-545,40	60,78	-622,20	73,04	14,08
75	150,3	20		-1026,05	59,16	-1175,25	67,61	14,54	-986,55	57,35	-1161,60	68,79	17,74
100	204,3	15		-1359,07	56,15	-1553,07	70,36	14,27	-1306,93	57,04	-1540,87	71,47	17,90
120	245,6	5		-2390,20	64,56	-2683,60	71,18	12,28	-2354,00	65,14	-2736,60	69,92	16,25
134	271,4	5		-2449,20	64,56	-2809,20	72,54	14,70	-2310,40	55,76	-2745,00	73,86	18,81
150	294,4	5		-1791,60	60,40	-1965,20	72,54	9,69	-1771,00	57,00	-1955,60	72,36	10,42
199	399,6	15		-2240,87	62,76	-2425,13	72,35	8,22	-2181,33	59,96	-2434,13	73,62	11,59
240	484,8	5		-1124,40	56,52	-1196,40	71,02	6,40	-1135,80	66,26	-1232,00	76,32	8,47
252	504,4	5		-1400,60	56,58	-1497,40	74,96	6,91	-1380,60	64,90	-1507,20	73,92	9,17
255	509,0	5		-1012,20	58,48	-1069,00	76,54	5,61	-974,20	55,86	-1098,40	74,52	12,75
			avg.	**-1092,96**	**56,21**	**-1229,45**	**68,61**	**13,54**	**-1059,25**	**56,52**	**-1231,12**	**68,96**	**17,60**
15	31,1	10	(OA)	-277,20	47,84	-335,20	64,64	20,92	-302,30	56,79	-341,90	64,97	13,10
20	39,5	10		-419,90	57,21	-460,60	66,86	9,69	-405,90	62,27	-452,80	63,39	11,55
21	39,4	10		-530,50	56,60	-601,90	65,96	13,46	-478,90	55,61	-603,90	65,52	26,10
22	39,4	10		-920,00	57,42	-1026,50	64,77	11,58	-805,80	50,90	-1003,20	64,28	24,50
25	56,0	5		-542,40	59,90	-553,20	65,62	1,99	-520,00	60,04	-588,00	67,96	13,08
29	57,8	10		-949,80	61,93	-1136,20	68,34	19,63	-883,30	56,59	-1134,20	66,34	28,40
30	63,8	5		-501,40	61,24	-555,40	68,74	10,77	-428,20	52,26	-546,40	65,58	27,60
32	62,5	15		-1623,93	55,68	-1873,07	67,30	15,34	-1545,73	57,83	-1821,27	67,06	17,83
35	74,4	5		-582,60	55,40	-625,00	69,32	7,28	-518,40	52,38	-646,80	67,54	24,77
40	79,2	5		-576,20	46,54	-712,80	68,70	23,71	-646,40	57,82	-715,20	67,90	10,64
44	86,2	5		-1272,20	58,42	-1452,60	65,58	14,18	-1149,80	51,56	-1451,80	69,58	26,27
50	105,2	5		-727,80	53,86	-806,80	69,18	10,85	-723,80	60,16	-797,80	66,12	10,22
71	146,0	5		-544,80	57,90	-601,40	72,34	10,39	-536,00	60,50	-604,40	70,12	12,76
75	150,3	20		-1038,00	59,69	-1206,15	69,42	16,20	-1042,10	61,34	-1185,75	69,85	13,78
100	204,3	15		-1380,80	60,09	-1561,20	71,57	13,06	-1366,20	61,53	-1470,87	69,83	7,66
120	245,6	5		-2351,40	59,96	-2648,40	72,50	12,63	-2339,00	54,56	-2750,60	72,30	17,60
134	271,4	5		-2582,00	67,64	-2723,40	72,74	5,48	-2280,00	66,50	-2669,20	73,94	17,07
150	294,4	5		-1867,20	64,46	-1990,40	71,82	6,60	-1785,60	68,32	-1951,40	74,38	9,29
199	399,6	15		-2244,80	62,98	-2419,67	74,59	7,79	-2196,80	61,68	-2468,33	74,16	12,36
240	484,8	5		-1161,40	70,04	-1233,80	74,10	6,23	-1102,80	58,10	-1232,40	79,26	11,75
252	504,4	5		-1403,60	58,66	-1500,20	72,10	6,88	-1411,00	62,82	-1531,00	72,40	8,50
255	509,0	5		-1033,20	64,10	-1082,00	76,36	4,72	-994,60	64,56	-1078,20	70,84	8,41
			avg.	**-1115,05**	**58,98**	**-1232,09**	**69,66**	**11,34**	**-1066,48**	**58,82**	**-1229,34**	**69,24**	**16,06**

Table 4.2: Computational results with (RBL)

				(FN)					(FR)				
n	M	$inst$	ord	z_{CH}	$\%_{fill}^{CH}$	z_{VNS}	$\%_{fill}^{VNS}$	imp	z_{CH}	$\%_{fill}^{CH}$	z_{VNS}	$\%_{fill}^{VNS}$	imp
15	31,1	10	(OH)	-246,40	39,03	-293,80	51,74	19,24	-256,10	39,35	-293,30	51,00	14,53
20	39,5	10		-332,40	38,31	-394,40	54,39	18,65	-331,50	42,05	-399,70	50,76	20,57
21	39,4	10		-487,50	48,08	-526,70	51,37	8,04	-445,30	42,42	-539,90	54,67	21,24
22	39,4	10		-772,70	42,13	-901,80	50,89	16,71	-773,70	42,51	-897,70	52,67	16,03
25	56,0	5		-437,80	45,26	-475,00	51,22	8,50	-400,60	47,74	-477,20	49,16	19,12
29	57,8	10		-840,70	45,14	-1018,40	52,99	21,14	-816,40	40,43	-973,30	50,08	19,22
30	63,8	5		-431,40	46,90	-501,00	56,42	16,13	-419,00	46,62	-464,40	54,48	10,84
32	62,5	15		-1462,47	46,86	-1554,40	51,45	6,29	-1452,93	46,20	-1631,53	51,47	12,29
35	74,4	5		-532,40	43,68	-587,00	55,40	10,26	-468,20	37,52	-563,60	53,64	20,38
40	79,2	5		-556,60	39,88	-619,40	50,78	11,28	-541,60	46,42	-645,40	54,32	19,17
44	86,2	5		-1113,80	40,56	-1217,60	52,68	9,32	-1018,60	46,16	-1311,20	56,12	28,73
50	105,2	5		-666,60	41,30	-740,40	52,68	11,07	-667,40	41,84	-759,80	54,34	13,84
71	146,0	5		-519,20	49,80	-563,40	58,78	8,51	-477,40	43,30	-555,80	53,96	16,42
75	150,3	20		-984,20	46,16	-1095,85	54,78	11,34	-927,70	43,20	-1087,70	57,22	17,25
100	204,3	15		-1313,40	49,90	-1410,00	58,85	7,35	-1260,87	48,48	-1384,33	55,43	9,79
120	245,6	5		-2302,40	50,54	-2635,80	62,06	14,48	-2223,00	48,96	-2558,60	59,52	15,10
134	271,4	5		-2258,20	55,56	-2510,40	61,70	11,17	-2194,60	52,60	-2392,20	62,02	9,00
150	294,4	5		-1724,00	49,64	-1935,40	62,02	12,26	-1743,60	51,02	-1806,20	64,92	3,59
199	399,6	15		-2180,60	48,18	-2374,33	62,55	8,88	-2179,53	53,97	-2371,60	66,35	8,81
240	484,8	5		-1116,20	55,44	-1192,80	68,30	6,86	-1074,80	48,44	-1188,00	62,98	10,53
252	504,4	5		-1365,80	46,74	-1468,60	70,78	7,53	-1335,60	51,02	-1464,40	63,90	9,64
255	509,0	5		-986,20	50,70	-1039,80	65,70	5,44	-981,40	54,02	-1038,20	63,24	5,79
			avg.	**-1028,68**	**46,35**	**-1138,92**	**57,16**	**11,38**	**-999,54**	**46,10**	**-1127,46**	**56,47**	**14,63**
15	31,1	10	(OA)	-250,22	36,92	-295,78	50,67	18,21	-252,11	36,67	-287,78	46,26	14,15
20	39,5	10		-346,50	37,75	-395,10	50,59	14,03	-339,20	42,76	-412,80	51,44	21,70
21	39,4	10		-486,60	46,81	-538,10	51,53	10,58	-450,80	45,31	-549,70	51,89	21,94
22	39,4	10		-756,20	40,47	-895,30	51,72	18,39	-769,20	43,04	-879,10	48,35	14,29
25	56,0	5		-493,75	41,73	-525,75	51,18	6,48	-423,20	44,20	-476,40	54,86	12,57
29	57,8	10		-965,00	40,43	-1054,75	45,96	9,30	-815,00	45,36	-1026,20	50,45	25,91
30	63,8	5		-430,00	45,14	-490,80	55,38	14,14	-410,40	40,04	-483,20	55,86	17,74
32	62,5	15		-1427,80	42,59	-1581,13	51,43	10,74	-1412,64	43,27	-1683,79	50,77	19,19
35	74,4	5		-532,40	43,68	-591,00	51,36	11,01	-492,20	41,90	-563,00	51,52	14,38
40	79,2	5		-638,75	37,30	-692,25	43,70	8,38	-556,80	42,84	-642,40	55,06	15,37
44	86,2	5		-1121,80	41,66	-1253,60	57,18	11,75	-1067,20	38,30	-1312,00	58,38	22,94
50	105,2	5		-658,80	40,36	-754,20	54,00	14,48	-653,40	47,52	-748,40	55,02	14,54
71	146,0	5		-502,20	46,98	-554,40	54,72	10,39	-497,00	41,76	-555,00	49,10	11,67
75	150,3	20		-1032,58	46,18	-1119,79	53,54	8,45	-924,65	40,84	-1090,60	57,05	17,95
100	204,3	15		-1291,27	48,62	-1463,00	60,69	13,30	-1282,93	47,11	-1422,40	58,85	10,87
120	245,6	5		-2288,00	47,74	-2524,40	61,88	10,33	-2236,20	45,16	-2555,00	61,74	14,26
134	271,4	5		-2309,40	49,20	-2579,20	58,62	11,68	-2555,75	43,00	-2847,50	56,80	11,42
150	294,4	5		-1708,80	49,54	-1871,40	59,36	9,52	-1673,00	44,42	-1898,20	62,82	13,46
199	399,6	15		-2178,00	47,87	-2355,53	61,90	8,15	-2149,33	49,75	-2353,27	62,86	9,49
240	484,8	5		-1378,25	59,38	-1423,75	64,83	3,30	-1086,40	45,42	-1194,40	62,14	9,94
252	504,4	5		-1373,40	48,76	-1442,80	61,60	5,05	-1359,40	51,14	-1477,80	63,08	8,71
255	509,0	5		-974,00	43,44	-1031,00	57,68	5,85	-1194,75	53,03	-1262,75	66,70	5,69
			avg.	**-1051,99**	**44,66**	**-1156,05**	**54,98**	**10,61**	**-1027,34**	**44,22**	**-1169,17**	**55,95**	**14,92**

Table 4.3: Computational results with (LP) and $(LP.FF)$

Table 4.4: Computational results with (LP) and $(LP.BF)$

				(FN)					(FR)				
n	M	$inst$	ord	z_{CH}	$\%^{CH}_{fill}$	z_{VNS}	$\%^{VNS}_{fill}$	imp	z_{CH}	$\%^{CH}_{fill}$	z_{VNS}	$\%^{VNS}_{fill}$	imp
15	31,1	10	(OH)	-246,40	39,03	-289,20	48,70	17,37	-238,20	40,59	-295,20	49,58	23,93
20	39,5	10		-332,40	38,31	-390,80	51,70	17,57	-373,40	46,24	-410,80	53,67	10,02
21	39,4	10		-487,50	48,08	-530,50	51,34	8,82	-417,80	41,47	-533,60	50,04	27,72
22	39,4	10		-772,70	42,13	-888,00	51,79	14,92	-710,50	41,95	-876,80	50,98	23,41
25	56,0	5		-437,80	45,26	-496,20	51,62	13,34	-376,20	42,16	-478,00	51,62	27,06
29	57,8	10		-840,70	45,14	-982,90	49,60	16,91	-825,60	41,02	-965,00	50,52	16,88
30	63,8	5		-431,40	46,90	-487,00	54,94	12,89	-384,20	43,20	-473,80	49,80	23,32
32	62,5	15		-1462,47	46,86	-1564,13	49,47	6,95	-1347,07	44,41	-1642,47	53,89	21,93
35	74,4	5		-532,40	43,68	-588,80	55,14	10,59	-456,20	39,70	-581,60	51,08	27,49
40	79,2	5		-556,60	39,88	-625,80	49,04	12,43	-560,20	40,22	-653,20	54,16	16,60
44	86,2	5		-1113,80	40,56	-1209,20	50,16	8,57	-1040,40	42,04	-1264,00	58,70	21,49
50	105,2	5		-666,60	41,30	-766,80	57,52	15,03	-694,20	49,08	-758,40	56,98	9,25
71	146,0	5		-519,20	49,80	-565,20	56,90	8,86	-480,60	40,70	-545,00	56,68	13,40
75	150,3	20		-984,20	46,16	-1088,00	55,01	10,55	-938,10	47,56	-1094,95	57,15	16,72
100	204,3	15		-1313,40	49,90	-1425,53	59,68	8,54	-1279,47	48,59	-1443,93	59,88	12,85
120	245,6	5		-2302,40	50,54	-2560,80	63,20	11,22	-2261,20	46,62	-2427,20	55,82	7,34
134	271,4	5		-2258,20	55,56	-2465,80	57,24	9,19	-2246,20	49,18	-2516,20	56,64	12,02
150	294,4	5		-1724,00	49,64	-1892,60	62,48	9,78	-1665,60	49,88	-1847,00	60,68	10,89
199	399,6	15		-2180,60	48,18	-2353,00	62,36	7,91	-2126,47	51,90	-2362,60	63,29	11,10
240	484,8	5		-1116,20	55,44	-1191,60	67,18	6,76	-1097,20	49,68	-1204,00	64,76	9,73
252	504,4	5		-1365,80	46,74	-1465,00	62,62	7,26	-1364,60	47,48	-1437,60	67,00	5,35
255	509,0	5		-986,20	50,70	-1033,40	59,18	4,79	-991,00	51,52	-1058,00	66,22	6,76
			avg.	**-1028,68**	**46,35**	**-1130,01**	**55,77**	**10,92**	**-994,29**	**45,24**	**-1130,43**	**56,32**	**16,15**
15	31,1	10	(OA)	-250,22	36,92	-284,67	46,58	13,77	-237,80	37,08	-270,60	41,82	13,79
20	39,5	10		-346,50	37,75	-408,40	53,49	17,86	-352,70	40,67	-411,30	49,06	16,61
21	39,4	10		-486,60	46,81	-530,40	49,97	9,00	-435,80	42,03	-542,80	53,04	24,55
22	39,4	10		-756,20	40,47	-872,60	49,36	15,39	-671,10	43,02	-878,60	49,67	30,92
25	56,0	5		-493,75	41,73	-540,50	44,15	9,47	-446,20	44,02	-470,00	51,58	5,33
29	57,8	10		-965,00	40,43	-1097,38	46,89	13,72	-807,00	41,90	-989,50	50,19	22,61
30	63,8	5		-430,00	45,14	-479,20	57,02	11,44	-495,50	46,40	-540,75	50,13	9,13
32	62,5	15		-1427,80	42,59	-1558,13	51,59	9,13	-1413,47	43,84	-1614,80	51,61	14,24
35	74,4	5		-532,40	43,68	-580,20	53,52	8,98	-548,00	41,28	-663,25	56,28	21,03
40	79,2	5		-638,75	37,30	-707,00	44,30	10,68	-502,80	39,50	-642,60	57,78	27,80
44	86,2	5		-1121,80	41,66	-1212,20	51,34	8,06	-1046,80	41,94	-1260,60	58,62	20,42
50	105,2	5		-658,80	40,36	-737,80	54,38	11,99	-656,40	47,82	-716,20	52,84	9,11
71	146,0	5		-502,20	46,98	-553,60	55,46	10,23	-492,00	43,90	-537,40	55,44	9,23
75	150,3	20		-1032,58	46,18	-1138,95	55,53	10,30	-948,15	44,34	-1096,90	57,55	15,69
100	204,3	15		-1291,27	48,62	-1467,33	61,37	13,64	-1264,40	49,39	-1463,87	61,06	15,78
120	245,6	5		-2288,00	47,74	-2530,40	59,92	10,59	-2270,60	49,00	-2490,00	59,86	9,66
134	271,4	5		-2309,40	49,20	-2497,40	56,40	8,14	-2107,40	45,84	-2545,40	60,08	20,78
150	294,4	5		-1708,80	49,54	-1936,60	62,60	13,33	-1686,60	42,34	-1879,20	59,70	11,42
199	399,6	15		-2178,00	47,87	-2354,87	62,77	8,12	-2167,80	53,61	-2381,07	64,03	9,84
240	484,8	5		-1378,25	59,38	-1428,75	63,70	3,66	-1324,25	54,55	-1397,75	66,53	5,55
252	504,4	5		-1373,40	48,76	-1493,60	70,96	8,75	-1383,60	53,32	-1493,40	66,48	7,94
255	509,0	5		-974,00	43,44	-1036,00	58,62	6,37	-982,80	56,64	-1048,20	67,96	6,65
			avg.	**-1051,99**	**44,66**	**-1156,64**	**55,00**	**10,57**	**-1010,96**	**45,56**	**-1151,55**	**56,42**	**14,91**

value of the solution. Depending on the quality of the initial solution the percentage of improvement goes up to nearly 43%. This percentage tends to be larger when the level packing procedure is used to build the loading patterns.

The best average results are obtained using the strategies (BL), (FN) and (OH). The strategy (FN) that consists in inserting first the client that is nearest to the depot generates usually the best initial solutions when compared to (FR). In some cases, choosing randomly the first client to insert in the route yields better initial solutions, but even in these cases, the local search procedure tends to reach better solutions at the end of the computing time with the strategy (FN) than it does with the strategy (FR). Ordering the items by height (OH) or by area (OA) has a more significant impact when the level packing procedure is used to place the items in the vehicle. When the bottom-left based strategies (BL) and (RBL) are used, these two orderings yield results that are not significantly different for these instances. The results concerning the level packing procedure with the best-fit rule $(LP.BF)$ are very near from those obtained with the level packing approach with the first-fit rule $(LP.FF)$ for these instances.

The variants of the algorithm based on the bottom-left placement procedures find solutions with a high percentage of used space in the vehicles. This percentage is typically higher for the largest instances. It goes up to 78.94% when the strategies (BL), (FN) and (OA) are used on the instances with 255 clients and an average of 509 items per instance. The percentage of used space tends to decrease with the level packing procedures. This trend was expectable given that the loading patterns generated through the level packing procedure are more constrained (guillotinable patterns) than those generated with the approaches relying on the bottom-left rules. In general, the results obtained with level packing procedure are outperformed by those achieved with the bottom-left based approaches.

4.6 Conclusion

In this chapter, we explored the first solution algorithm for the 2L-ESPP. The approach is based on constructive heuristics to generate initial feasible solutions for the problem, and on variable neighborhood search to look for improved incumbents. We described different alternative neighborhood structures based on the routing and packing characteristics of the solution. We provided also the first results concerning the resolution of this problem for a large set of benchmark instances of the 2L-CVRP. The results illustrate the effectiveness of the variable neighborhood search procedure in improving the solutions of the constructive heuristics. These results allowed the comparison between the different strategies described in this chapter. Besides the practical relevance of the problem, these results may contribute for the resolution of the 2L-CVRP through column generation algorithms since the 2L-ESPP is the pricing subproblem that results from the corresponding Dantzig-Wolfe decomposition of this problem. This decomposition will be analyzed in the next two chapters, and in this sense, the VNS algorithm will be used for solving the subproblem of a column generation algorithm.

Chapter 5

Column generation based approaches for the vehicle routing problem with two-dimensional loading constraints

Contents

5.1 Introduction

In this chapter, we present a column generation approach for the Capacitated Vehicle Routing Problem with Two-dimensional Loading constraints (2L-CVRP), defined in Section 2.3.1, and assuming a virtually unlimited number of vehicles. The subproblem is solved heuristically through variable neighborhood search. Branch-and-price is used when it is not possible to add more attractive columns and the solution remains fractional. In order to accelerate the convergence of the algorithm, a family of valid dual inequalities is proposed.

5.1.1 Exact methods to the L-CVRP

The first exact method for the 2L-CVRP is due to Iori *et al.* [78]. This work was analyzed in Section 3.2.1. The authors proposed a branch-and-cut approach for the 2L-CVRP. Firstly, the integer model formulation is solved without considering loading or capacity-cut constraints. In what concerns the capacity-cut constraints, separation procedures adapted from [117] are then applied. These procedures consist in heuristics methods to the minimum-cut problem in order to find violated inequalities. When an integer solution is obtained, the feasibility of the packing is verified through a branch-and-bound algorithm. Additionally, a heuristic method is used to verify if it is possible to achieve a better solution. If the heuristic achieves a better solution, the loading feasibility of each route is verified with the branch-and-bound algorithm. If all the routes are feasible, then the best solution is updated. Otherwise, a new constraint for each invalid route is added to the model.

To the best of our knowledge, the first column generation formulation for the 2L-CVRP was presented in [35]. The authors proposed a branch-and-cut-and-price algorithm whose subproblem only deals with area and capacity constraints. For this purpose, an elementary shortest path problem is solved by only considering the area and the capacity as resources. Then, valid inequalities are added to the master problem in order to ensure the loading feasibility of routes.

Other authors also considered a simpler subproblem, but in contrast to the work presented in [35], no constraints are added to the master problem. It is the case of the contributions presented in [137] for the 2L-CVRP or the work suggested in [101] for the 3L-CRVRP.

Concerning the approach presented in [137], the restricted master problem is initialized with a set of valid solutions obtained through both constructive heuristics and local search methods. These methods consist in iterated local search algorithms [95] applied to the solutions provided by constructive methods. Two general approaches were developed. In the first approach, each inserted column corresponds to a feasible solution concerning the loading constraints. The feasibility of a route is verified by a heuristic bundle proposed in [154]. When no more columns can be added to the restricted master problem, a branch-and-bound is used in order to obtain an integer solution. On the other hand, in the second approach, the packing feasibility of each route is only verified after the column generation method and after the branch-and-bound method. If the solution includes routes that are infeasible concerning loading constraints, the corresponding columns are removed from the restricted master problem. To verify the feasibility the authors resort to the bundle referred to above and to a branch-and-bound algorithm adapted from [78]. For both approaches, and in order to solve the subproblem, four heuristics are successively applied in order to derive one solution with negative reduced cost. If they fail, the label correcting algorithm [57] is used instead.

As referred to above, a column generation approach for the 3L-CVRP is proposed in [101]. The restricted master problem is initialized with each customer assigned to one route. The pricing subproblem is solved with one of two methods. The first method consists in relaxing loading constraints and in solving an instance of the elementary shortest path problem with resource constraints considering the volume and the capacity as resources. For this purpose, the authors resort to the label correcting algorithm [57]. The feasibility of the obtained route is verified by the extreme point based heuristic [40]. If it is not feasible, then customers are successively removed until a valid solution is found or until the reduced cost turns positive. The second method relies on a heuristic method, which aim to generate a list of feasible routes corresponding to a negative reduced cost, even though there is not any route corresponding to the minimum reduced cost. The obtained results are competitive with those obtained with tabu search approaches.

Recently, and based on the work branch-and-cut approach of Iori *et al.* [78], another exact method was presented in [74] for both the 2L-CVRP and 3L-CVRP. Again, a relaxed version of the integer programming formulation is solved. Concerning the routing component, multiple separation procedures are applied. Moreover, when no more cuts are found, separation procedures concerning the packing are applied. These procedures are divided in two main strategies. The first strategy is applied right after reaching an integer solution provided by the branch-and-bound, by verifying the feasibility of each route included in the solution. On the contrary, the second strategy does not require the execution of the branch-and-bound since it relies on a procedure to find infeasible routes of a non-integer solution. The feasibility of a given route is firstly verified by several heuristics procedures. Additionally, the authors suggested two exact approaches that can be applied if the heuristics referred to above fail to prove the feasibility of a given route. These approaches are based on the branch-and-bound algorithm for packing problems [107] and in constraint programming for the orthogonal packing [33].

Finally, Junqueira *et al.* [83] proposed the first mathematical formulation for the 3L-CVRP. The authors extended the time-dependent formulation for the Travelling Salesman Problem [59] for this problem. Additional constraints are considered such as vertical load stability, multi-drop situations and load bearing. These constraints arise frequently in real-world situations. The authors assessed the effectiveness of their approach in randomly generated instances. They concluded that their approach is able to deal with medium-size problems, but presents obvious limitations for large-size instances.

5.1.2 Column generation and routing

As in a wide variety of optimization problems, column generation approaches were successfully applied in the routing field. The subproblem of a column generation algorithm for a vehicle routing problem with additional constraints corresponds to an Elementary Shortest Path Problem with Resource Constraints (ESPPRC). This problem intends to find the elementary shortest path, beginning and ending at the depot and satisfying the capacity constraints. Additionally, as an elementary path, each customer belonging to the path can only be visited once. Exact approaches for the ESPPRC can be found in [57, 131, 133, 8]. Since ESPPRC is NP-hard [53], in some approaches the elementary constraint is relaxed. In this case, the solution generated by the subproblem may include cycles due to the negative reduced costs in the edges. In [79], the authors presented some strategies to eliminate cycles of length greater than 3.

Techniques based on column generation tackled several variants and extensions of the vehicle routing problem, and they were effective in dealing with a multiplicity of constraints. One good example is the inclusion of specific constraints in order to adapt the problem formulation to a given real-world scenario. In the literature, some authors refer to these problems as rich routing problems. In the following, we review some works in which the column generation framework was effective in complex routing problems with a great diversity of assumptions.

Ceselli *et al.* [29] addressed the multi-depot vehicle routing with heterogeneous fleet and time windows. Additional considerations were taken into account such as splitting the orders, incompatibilities between items and between these and locations, maximum route length, among others. The authors suggested a heuristic column generation operating sequentially in three phases: in the first phase no orders can be split, in the second phase some orders can be split, and in the third phase all orders are allowed to be split. The subproblem is solved through a heuristic method consisting in an adaptation of the dominance criteria within the bidirectional dynamic programming [131].

In [15], a branch-and-cut-and-price algorithm is proposed for a multi-depot vehicle routing problem considering heterogeneous fleet and time windows. The authors claim that this approach is the first exact method for this problem. The restricted master problem is initialized with one column for each customer, a set of columns provided by a greedy heuristic and a dummy column. The subproblem is solved by three sequential methods. The first one is a greedy heuristic. If this heuristic fails to achieve one column with negative reduced cost, then a heuristic dynamic programming procedure is used, and if it is not sufficient to find such column, the authors resort to an exact algorithm proposed in [131] enhanced with decremental state space relaxation [133]. The authors used two branching strategies, namely branching on the number of vehicles and branching on arcs.

Another rich routing problem solved with a branch-and-price algorithm was addressed in [119]. The problem consists in a livestock collection problem, which incorporates inventory constraints in a capacitated vehicle routing problem. The fleet is heterogeneous and each vehicle may perform multiple trips. An interesting feature of this contribution is the loading problem that arises when transporting animals from different types. Each vehicle has different compartments and may have an upper and lower levels. Since each animal type needs specific height and floor space, and different types of animal may not share the same compartment, different configurations are needed in the vehicle. The authors stated that the order that animals are loaded into the vehicle will have impact in the capacity of the vehicle. Furthermore, some precedence constraints are enforced.

As it can be seen, column generation frameworks were applied in a vast diversity of routing problems. In many of them, specific constraints arising from real-world applications are incorporated. Despite the inherent complexity resulting from the inclusion of real assumptions, the column generation approaches proved to be effective in dealing with such complexity. This success strongly motivated the contribution presented in this chapter.

5.2 Column generation and branch-and-price frameworks

5.2.1 Dantzig-Wolfe decomposition

Decomposition methods may rely on the reformulation of hard optimization problems into smaller ones, which can be easier to solve, and then on the combination of their solutions within the scope of the original problem. Among the different decomposition methods, the Dantzig-Wolfe decomposition [46] is one of the most well-known. In their renowned work [46], Dantzig and Wolfe formally presented a decomposition principle to be applied to a linear programming with the following form:

$$\min \quad cx \tag{5.1}$$

$$\text{subject to} \quad Ax \geq b, \tag{5.2}$$

$$x \in X, \tag{5.3}$$

$$x \in \mathbb{R}^n_+, \tag{5.4}$$

According to the theorem of Minkowski, any polyhedron X can be expressed as the convex combination of the extreme points of X $(P_1, P_2, \ldots, P_n)$ plus a non-negative combination of its extreme rays $(R_1, R_2, \ldots, R_p)$, as follows:

$$X = \left\{ x \in \mathbb{R}^n_+ : \sum_{i=1}^{n} \lambda_i P_i + \sum_{j=1}^{p} \mu_j R_j, \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \geq 0, \forall i, \mu_j \geq 0, \forall j \right\} \tag{5.5}$$

If the polyhedron is bounded, it can be defined solely by the convex combination of its extreme points:

$$X = \left\{ x \in \mathbb{R}^n_+ : \sum_{i=1}^{n} \lambda_i P_i, \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \geq 0, \forall i \right\}. \tag{5.6}$$

By replacing this definition in the original formulation, we obtain a reformulated model with decision variables λ_i, each one corresponding to an extreme point. This reformulated model has less constraints than the original formulation, but in contrast, it may have an exponential number of variables due to the exponential number of extreme points. A clear advantage of the Dantzig-Wolfe decomposition is that if the polyhedron X has not the integrality property, the Linear Programming (LP) relaxation of the reformulated method may be better than LP-relaxation of the original model. For this reason, it is usual to state that the formulations provided by Dantzig-Wolfe decomposition are stronger than the ones derived by linear programming. If X has the integrality property, the reformulation is as stronger as the LP relaxation.

It is important to note that we may have more than one set of constraints of type (5.3). Such cases correspond to constraints that do not include the same decision variables, and thus they can be seen as independent blocks with different polyhedra X_i $(i = 1, 2, \ldots, s)$ defining their domains. In contrast, constraints (5.2) cover all the decision variables. This structure is denominated by an angular block structure.

5.2.2 Column generation

As referred to above, applying Dantzig-Wolfe decomposition to a bounded model may result in a reformulated model, the master problem, with an exponential number of decision variables (extreme points). Therefore, column generation is inextricably linked to the resulting formulations since it only deals with a subset of decision variables, avoiding its explicitly enumeration.

The column generation algorithm is based on two main elements: a master problem and a set of subproblems. Constraints of type (5.2) are usually tackled in the master problem, while each subproblem is associated to one polyhedron X_i $(i = 1, 2, \ldots, s)$.

The master problem is initialized with a restricted set of decision variables. For this reason, in the literature it is usual to denominate the master problem as Restricted Master Problem (RMP).

In each iteration, solving the RMP will provide the information (more precisely the dual variables) in order to verify if any variable (extreme point) within the polyhedron or polyhedra is attractive. This can be formulated as one or more subproblems (one for each defined polyhedron), whose aim is to find attractive variables within their domains. The most attractive variable among all subproblems, or more than one attractive variable, may be added to the RMP.

The RMP is re-optimized, providing again the dual variables to be used by subproblems. The process is repeated until no attractive variables are found by subproblems. In this case, the solution associated to the RMP is optimal. The Figure 5.1 attempts to represent the interaction between problem and subproblems.

Concerning the RMP, some strategies are used to build the initial set of columns in an early stage as the use of heuristics or random solutions. Alternatively, if there are no columns at the beginning, an artificial variable can be used. This variable corresponds to a column with a high cost, which ensures the feasibility of the RMP, even if it corresponds to an infeasible solution in the scope of the original problem. The use of an artificial column also guarantees feasibility of the problem within the branch-and-bound method: the set of columns at a given node can be insufficient to derive a valid solution when considering the branching constraint. Since the cost of this variable is very high, it will only be used if no other columns are able to derive

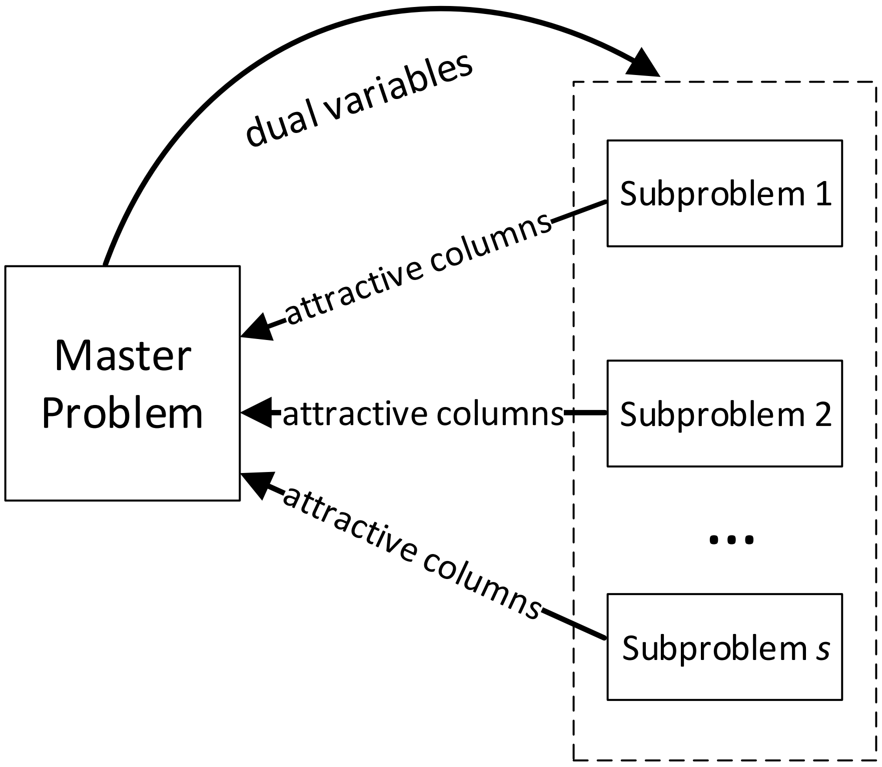

Figure 5.1: Column generation algorithm

a valid solution.

On the other hand, for complex subproblems it can be difficult to achieve the optimal solution. It is possible to use heuristics to solve the subproblem, but there is no guarantee that this method returns a solution with negative reduced cost. Other strategies include the relaxation of some constraints in the subproblem. Clearly, this may lead to an infeasible solution provided by the column generation algorithm. As referred to in Section, some works tackled such situations by verifying the feasibility of the final solution and by removing the columns that are associated to that infeasible solution [137].

5.2.3 Branch-and-price

At the end of the column generation algorithm, there is no guarantee that the optimal solution of the reformulated model is an integer solution. Indeed, the obtained solution corresponds to the linear programming relaxation, and thus the set of columns at the

RMP may not be sufficient to derive an integer solution.

In order to derive an integer solution, it is necessary to combine the branch-and-bound approach with the column generation. At the root of the branching tree, the LP of the reformulated model is provided with a specific number of columns. Thus, branching constraints are introduced. At lower levels of the tree, new columns may be needed, as many as required to solve the LP relaxation of that node. This integrated method is denominated by branch-and-price.

An important issue of branch-and-price is the branching scheme. Branching schemes based on the variables of the reformulated model may provide the regeneration of some columns: if a given decision variable is turned to zero, the corresponding column will be set to zero and it may happen that the subproblem returns that variable as the most attractive. Therefore, regeneration leads to deadlock situations. To overcome this situation, the vast majority of branch-and-price approaches suggest that the branching partitions must be applied to the original variables of the original model.

5.3 Column generation model for the 2L-CVRP

5.3.1 Original formulation

In Section 2.3.1 the 2L-CVRP was described. The first mathematical formulation for this problem is due to Iori *et al.* [78]. We adapted their formulation in order to consider the two-indexed binary variables.

Let $G = (V, A)$ be a complete directed graph where V represents a set of $n + 1$ nodes and A represents the set of arcs. The set V includes the depot (denoted by 0) and a set of n customers (N). We will consider a homogeneous fleet with a virtually unlimited number of vehicles. Additionally, let σ represent the bijection which defines the order by which the customers are visited, and $\Sigma(S)$ represents the collection of sequences σ in which (S,σ) is a feasible route. The set of edges of such route is defined by $A(S, \sigma)$. Each binary decision variable x_{ij} takes value 1 if the arc (i, j) is traversed by one vehicle, and it takes value 0 otherwise. Travelling through this arc has a cost c_{ij}. Additionally, let $\delta^+(i)$ be the set of customers linked to i in which i is their origin. Analogously, let $\delta^-(i)$ be the set of customers linked to i in which i is their destination.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.7}$$

$$\text{subject to} \sum_{j\in\delta^+(i)} x_{ij} = 1, \forall i \in N \tag{5.8}$$

$$\sum_{i\in\delta^-(j)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = 0, \forall j \in V \tag{5.9}$$

$$\sum_{(i,j)\in A(S,\sigma)} x_{ij} \leq |S| - 1 \quad \forall (S,\sigma) \quad \text{such that } \sigma \notin \Sigma(S) \tag{5.10}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{5.11}$$

The set of constraints (5.8) imposes that each customer must be visited once. The set of constraints (5.9) imposes flow conservation while the set of constraints (5.10) imposes the loading feasibility of each route. The constraints (5.11) define the decision variable values.

5.3.2 Master problem

After applying the Dantzig-Wolfe decomposition [46] to the formulation presented in (5.7)-(5.11), the master problem is an integer programming model composed by partitioning constraints, while remaining constraints are considered in the subproblem. Note that convexity constraint is omitted since the fleet is homogeneous and virtually infinite.

Let Ω be the set of all valid routes, *i.e.*, the set of all extreme points. Each decision variable of the master problem corresponds to a column and it is defined by λ_r ($r \in \Omega$). Therefore, the each decision variable takes value 1 if it is included in the solution, and 0 otherwise. Each column represents a feasible route, and since the subproblem is limited, a given column corresponds to an extreme point of the valid space. Each route r has a cost c_r and it can be described by a vector $(a_{1r}, a_{2r}, \ldots, a_{nr})^T$ where each coefficient a_{ir} takes value 1 if customer i ($i \in N$) is visited in route r ($r \in \Omega$), and it takes value 0 otherwise. In Table 6.1 we present an example of the simplex representation of the master problem.

$$\min \sum_{r \in \Omega} c_r \lambda_r \tag{5.12}$$

$$\text{subject to} \sum_{r \in \Omega} a_{ir} \lambda_r = 1, \forall i \in N \tag{5.13}$$

$$y_r \in \{0,1\}, \forall r \in \Omega \tag{5.14}$$

Table 5.1: Simplex representation of the master problem

customers	λ_1	λ_2	λ_3	$\dots$	$\lambda_{\lvert\Omega\rvert-1}$	$\lambda_{\lvert\Omega\rvert}$		
1	a_{11}	a_{12}	a_{13}	$\dots$	$a_{1,\lvert\Omega\rvert-1}$	$a_{1,\lvert\Omega\rvert}$	$=$	1
2	a_{21}	a_{22}	a_{23}	$\dots$	$a_{2,\lvert\Omega\rvert-1}$	$a_{2,\lvert\Omega\rvert}$	$=$	1
3	a_{31}	a_{32}	a_{33}	$\dots$	$a_{3,\lvert\Omega\rvert-1}$	$a_{3,\lvert\Omega\rvert}$	$=$	1
4	a_{41}	a_{42}	a_{43}	$\dots$	$a_{4,\lvert\Omega\rvert-1}$	$a_{4,\lvert\Omega\rvert}$	$=$	1
5	a_{51}	a_{52}	a_{53}	$\dots$	$a_{5,\lvert\Omega\rvert-1}$	$a_{5,\lvert\Omega\rvert}$	$=$	1
6	a_{61}	a_{62}	a_{63}	$\dots$	$a_{6,\lvert\Omega\rvert-1}$	$a_{6,\lvert\Omega\rvert}$	$=$	1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$		$\dots$
$m-1$	$a_{m-1,1}$	$a_{m-1,2}$	$a_{m-1,3}$	$\dots$	$a_{m-1,\lvert\Omega\rvert-1}$	$a_{m-1,\lvert\Omega\rvert}$	$=$	1
m	$a_{m,1}$	$a_{m,2}$	$a_{m,3}$	$\dots$	$a_{m,\lvert\Omega\rvert-1}$	$a_{m,\lvert\Omega\rvert}$	$=$	1
	c_1	c_2	c_3	$\dots$	$c_{\lvert\Omega\rvert-1}$	$c_{\lvert\Omega\rvert}$		

5.3.3 Initialization

The column generation approach for the 2L-CVRP begins with a subset of columns, which corresponds to a subset of valid routes. In order to build this subset, we used three strategies. The first strategy consists in the generation of single customer routes. One route is associated to one and only one customer. With this strategy we generate as many columns as the number of customers in the instance. No feasibility tests are required since it is assumed that the demand of each customer fits in one vehicle. The third second relies on the dual valid inequalities to be presented in Section 5.4. Following this strategy it is possible to derive at most n^2 columns, where

n is the number of customers. Finally, the third strategy consists in the use of a metaheuristic to derive routes with a high percentage of used space in the vehicle. In this sense, we resort to the Variable Neighborhood Search algorithm for the elementary shortest path problem with two-dimensional loading constraints presented in Chapter 4. Briefly, this algorithm relies on constructive procedures to generate initial solutions, and in different packing and routing neighborhood structures to improve the former solutions. In order to generate routes with compact layout, the costs of all arcs are set at -1. The solution provided by the VNS algorithm will consist in a route with a desirable compact layout and the corresponding column will be added to the RMP. Then, the arcs of the inserted route will take a very high cost, and the process is repeated according to a parameter n_{init}. We also used an artificial variable. As stated in Section 5.2.2, this variable ensures the feasibility of the restricted master problem.

5.3.4 Subproblem

The subproblem corresponds to an Elementary Shortest Path Problem with Resource and Sequential Constraints (ESPPRSC). The ESPPRSC intends to find the column with most negative reduced cost. The solution consists in an elementary shortest path, beginning and ending at the depot, satisfying the loading constraints, and visiting each customer at most once.

After solving the LP-relaxation of the RMP of formulation (5.12)-(5.14), one can obtain the dual variables π_i $(i \in N)$ associated to each constraint of type (5.13). The expression of the reduced cost c'_r for a given route $r \in \Omega$ is given by:

$$c'_r = c_r - \sum_{i \in N} \pi_i a_{ir}.$$

Considering P_r the sequence of traversed arcs in route r $(r \in \Omega)$ such that $P_r = \{(0, i_1), (i_1, i_2), \ldots, (i_{|P_r|-1}, 0)\}$, the expression of the reduced cost can be reformulated as follows:

$$c'_r = \sum_{(i,j) \in P_r, i \neq 0} (c_{ij} - \pi_i) + c_{0,i_1}.$$

Let x_{ij} be the decision variable, which takes value 1 if arc $(i,j) \in A$ is part of the solution, and it takes value 0 otherwise. Therefore, the subproblem can be formulated as follows.

$$\min \sum_{(i,j)\in A} c'_{ij} x_{ij} \tag{5.15}$$

$$\text{subject to} \sum_{j\in\delta^+(0)} x_{0j} = 1 \tag{5.16}$$

$$\sum_{i\in\delta^-(0)} x_{i0} = 1 \tag{5.17}$$

$$\sum_{i\in\delta^-(j)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = 0, \forall j \in N \tag{5.18}$$

$$\sum_{(i,j)\in A(S,\sigma)} x_{ij} \leq |S| - 1 \quad \forall (S,\sigma) \quad \text{such that } \sigma \notin \Sigma(S) \tag{5.19}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{5.20}$$

where

$$c'_{ij} = \begin{cases} c_{ij} - \pi_i, & \forall (i,j) \in A, i \neq 0 \\ c_{ij}, & \forall (0,j) \in A \end{cases}$$

The constraints (5.16)-(5.17) impose that the vehicle exits and returns to the depot. The set of constraints (5.18) require flow conservation while the set of constraints (5.19) impose the loading feasibility of routes. The constraints (5.20) define the decision variable values.

The subproblem is solved using the variable neighborhood search algorithm for the elementary shortest path problem with two-dimensional loading constraints presented in Chapter 4.

5.4 Stabilization strategies

Despite the effectiveness of column generation algorithms, they may present a slow convergence. Some strategies that are commonly used to improve the efficiency of column generation algorithms can be found in [97, 96]. Among these strategies, restricting the dual space proved to be effective in accelerating the convergence of the algorithm.

During the column generation algorithm, the subproblem iteratively suggests one or more attractive columns which are added to the RMP. Each column that is added to the RMP corresponds to a constraint in the dual space of the problem. Therefore, adding columns to the RMP can restrict the dual space of the problem. Consequently, the subset of columns that the subproblem can find can also be limited. In this sense, it is possible to derive valid dual cuts which may be inserted in the RMP before its LP-relaxation. Thus, the dual space is restricted right at the initialization stage.

In [145], a family of dual cuts for the cutting stock problem was presented. The author stated that from a primal point of view, each dual cut corresponds to the use of one item in order to cut other items, only if the sum of their widths is less than or equal to the width of the former item. In the following, we describe the procedure in order to derive valid cuts in the scope of the column generation approach for the 2L-CVRP. Since one route corresponds to a sequence of visit, it is possible to replace one customer $i \in N$ by other customer $j \in N$ $(i \neq j)$ in the position of the route provided that:

D1 the number of items of customer j is less than or equal to the number of items of customer i;

D2 the area of one item of customer i can be used to place at most one item of customer j, without exceeding the height and width of the former item;

D3 all the items of customer j can be placed in the vehicle satisfying the previous condition.

The idea besides these three conditions is that the loading area of the vehicle, which is occupied with items of customer i, can be used to place the items of customer j. We apply this procedure to all the customers in order to verify all pairs of customers i and j $(i, j \in N, i \neq j)$ that satisfy these three conditions.

At this step, it is required to compute the cost of replacing customer i by customer j in the sequence of the route. Since this route is not known, it is not possible to determine this cost effectively. In this sense, we compute $\bar{c}_{ij}$ as the cost difference between the two less costly arcs that are incident to i and the two higher costly arcs that are incident to j. Let i_1 and i_2 be, respectively, the first and the second closer customers to customer i. Let j_1 and j_2 be, respectively, the first and second customer which are more far from j. Thus,

$$\bar{c}_{ij} = c_{j_1,j} + c_{j_2,j} - c_{i_1,i} - c_{i_2,i} \tag{5.21}$$

Let π_i and π_j be the dual variables of constraints (5.13) of the RMP associated to customers i and j, respectively.

Proposition 5.1 *A family of valid dual cuts for the 2L-CVRP can be expressed as:*

$$-\pi_i + \pi_j \leq \bar{c}_{ij}, \quad \forall i, j \in N, i \neq j, \tag{5.22}$$

*if customers i and j satisfy the conditions (**D1**)-(**D3**), referred to above.*

Proof. The proof is based on the one for a family of dual inequalities, in the scope of the vehicle routing problem with different service constraints, presented in [98]. The proof starts by establishing valid conditions in the dual space and it demonstrates, by contradiction, that, in the scope of an invalid cut in the dual space, the validity conditions are not obeyed.

Let Ω be the set of columns corresponding to routes. Then,

$$\sum_{m \in N} a_{mr} \leq c_r, \forall r \in \Omega. \tag{5.23}$$

In the optimal solution, customer i is associated to exactly one route. Let $A_r = (a_{1r}, \ldots, 1, \ldots, 0, \ldots, a_{mr})^T$ with an unit coefficient in row i. This route has a reduced cost equal to zero. Therefore, according to the expression referred in (5.15), $c_r = \sum_{m \in N} \pi_m a_{mr}$. Let s be another route such that $A_s = (a_{1s}, \ldots, 0, \ldots, 1, \ldots, a_{ms})^T$, *i.e.*, $a_{is} = 0$ and $a_{js} = 1$, and all the other coefficients equal to the ones in A_r. Then,

$$c_s - c_r \leq \bar{c}_{ij}. \tag{5.24}$$

If there is a cut which is not valid, then, $-\pi_i + \pi_j > \bar{c}_{ij}$. Thus, $-\pi_i + \pi_j > c_s - c_r$. Consequently, $-\pi_i + \pi_j > c_s - \sum_{m \in N} \pi_m a_{mr}$. Since $-\pi_i + \pi_j = -\sum_{m \in N} \pi_m a_{mr} + \sum_{m \in N} \pi_m a_{ms}$, then, $\sum_{m \in N} a_{ms} > c_s$, not satisfying (5.23). ■

From the primal standpoint, each dual inequality of type (5.22) corresponds to the use of the space left by items of customer i in order to dispose items of customer j. It is worth noting that applying this procedure gives rise to a route which is necessarily feasible: all the packing constraints are obeyed since the placed items fit within the space of the removed items. Consequently, the unloading sequence of other items remains unchanged and sequential constraints are satisfied.

Example 5.1 An example is provided in Figure 5.2: customer 2 can be replaced by customer 4, satisfying the conditions referred to above. Therefore, $-\pi_2 + \pi_4 \leq \bar{c}_{24}$ is a valid dual.

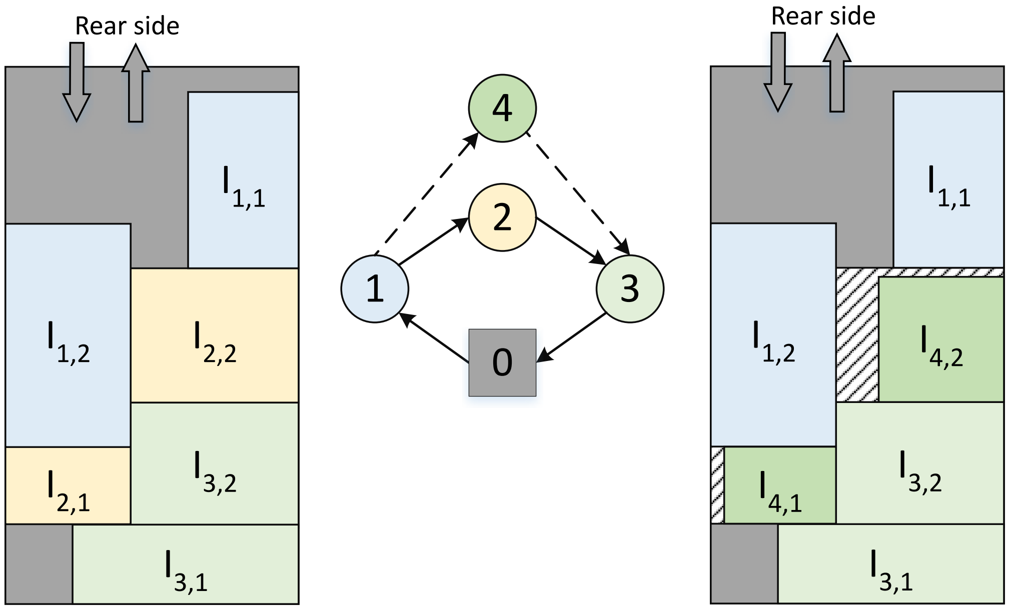

Figure 5.2: An example of primal standpoint of a valid dual cut

For an instance with n customers, it is possible to add at most n^2 valid dual cuts. These dual cuts correspond to columns of type $(a_{1r}, a_{2r}, \ldots, a_{ir}, \ldots, a_{jr}, \ldots, a_{nr})^T = (0, 0, \ldots, -1, \ldots, 1, \ldots, 0)^T$ with cost $\bar{c}_{ij}$. These columns are added to the RMP in the initialization phase, *i.e.*, before solving its LP-relaxation.

5.5 Branching schemes

5.5.1 Basics

As referred to in Section 5.2.3, it is important to select good branching schemes, which do not damage the structure of the subproblem. A branching scheme for the vehicle routing problem with time windows is presented in [50]. The authors suggested branching on a single arc. Each arc of the subproblem network is fixed. In one branch, a given arc (i, j) is fixed and takes value 1. Other arcs, whose destination is the node j, are removed as well as the arcs whose origin is the node i. The cost of these removed arcs is penalized. Therefore, the cost of the columns that correspond to solutions using such arcs is updated, imposing the use of arc (i, j) instead of using the alternative arcs due to their higher cost. In the other branch, the same arc (i, j) is removed. Similarly, the cost of this arc is penalized, and consequently, the cost of columns corresponding to solutions where arc (i, j) is used is updated. Therefore, columns corresponding to solutions including arc (i, j) may not be part of the optimal solution.

In order to deal with cycles, arcs associated to customers visited more than once are scored according to their flow. The highest scored arc is selected to partition. If there are no cycles, columns with fractional values are analyzed only if there is any arc with a flow different of the value 1. The columns are scored according to the variables value and each arc is scored according to the flow value.

It is important to note that in the context of the vehicle routing problem with time windows, if a given arc (i, j) takes the value 1, the total demand of customers i and j must not exceed the capacity of the vehicle and, additionally, it must be possible to serve both customers within their time window. These conditions are also applied to other customers already linked to i or j: at upper levels of the branching tree, it is possible to have branching rules that impose the use of an arc whose destination is i or whose origin is j. In this case, the conditions referred to above are applied to the set of customers that are being linked. In [84], a review of branching schemes is presented, namely branching on the number of vehicles, branching on flow variables and branching on resource windows.

5.5.2 Branching rules

The advantage of branching on the variables of the original problem is twofold. On the one hand, the structure of the subproblem is not modified. On the other hand, convergence is ensured. Our branching strategies rely in both single arc branching and in the flow branching. In the former case, the branching is performed in one variable of the original problem which corresponds to an arc. In the latter case, more than one original variable may be selected. In the following we present the partition rules for both cases.

5.5.2.1 Partition rules based on a single variable

BB1.1 Among all the arcs of the solution, the one that has the highest fractional flow is selected to branch, *i.e.*,

$$(i',j') = \text{arg max}_{(i,j)\in A}\{x_{ij} - \lfloor x_{ij}\rfloor \mid \lfloor x_{ij}\rfloor \neq 0, \lfloor x_{ij}\rfloor \neq 1\}. \tag{5.25}$$

This rule aims to explore the branching tree guided by the solution provided by the linear relaxation of the RMP.

BB1.2 Among all customers, the one that is visited in more routes of the LP solution is selected; among all incident arcs to the selected customer, the one that has the highest fractional flow is selected to branch.

Let $\Omega' \subseteq \Omega$ be the set of columns at the current node. We are interested in finding the customer k' such that:

$$k' = \text{arg max}_{k\in N}\left\{\sum_{r\in\Omega'} a_{kr} \middle| \lambda_r > 0, r \in \Omega'\right\}. \tag{5.26}$$

This rule aims to explore the difficulty that the model may have in assigning a given customer to a single route. On the one hand, by fixing the arc with the highest flow incident to that customer to take the value 1, we are fixing the arc to be part of all forthcoming solutions on that branch. If the selected customer is the starting point of the selected arc (or, respectively, its end point), then all the arcs whose starting point (or, respectively, whose end point) correspond to such customer, are excluded from the optimal solution. On the other hand, fixing the selected arc to take value 0 will exclude it from the forthcoming solutions on the other branch. This will force a subset of arcs incident to the customer to take different values since the fractional flow of the excluded arc must be redistributed.

BB1.3 Among all customers, the one that is visited in less routes of the LP solution is selected; among all arcs that are incident to selected customer, the one that has the highest fractional flow is selected. Again, we consider $\Omega' \subseteq \Omega$ as the set of columns at the current node. We are interested in finding the customer k' such that:

$$k' = \arg\ \min_{k \in N} \left\{ \sum_{r \in \Omega'} a_{kr} \middle| \lambda_r > 0, r \in \Omega' \right\}. \tag{5.27}$$

BB1.4 Among all arcs with fractional flow, the one that has the lowest cost is selected, *i.e.*,

$$(i', j') = \arg\ \min_{(i,j) \in A} \{c_{ij} \mid \lfloor x_{ij} \rfloor \neq 0, \lfloor x_{ij} \rfloor \neq 1\}. \tag{5.28}$$

This rule aims to fix the arcs by the increasing order of cost.

BB1.5 Among all routes of the LP solution, the one that visits more customers is selected; from the set of arcs traversed by this route, the arc that has the highest fractional flow is selected. Let r' be selected route, and $\Omega' \subseteq \Omega$ be the set of columns at the current node. Thus,

$$r' = \arg\ \max_{r \in \Omega'} \left\{ \sum_{i \in N} a_{ir} \middle| \lambda_r > 0 \right\}. \tag{5.29}$$

It is expected that the selected route has a good layout since it combines the maximum number of customers. Since the loading component is the most critical in the context of the problem, this rule aims to promote the fixation of arcs belonging to these type of routes. In this sense, it could be advantageous to fix the arcs belonging to such routes.

BB1.6 Among all arcs that are incident to the depot, the one that has the highest fractional flow is selected.

$$(i', j') = \arg \max_{(i,j) \in A, i=0 \vee j=0} \{x_{ij} - \lfloor x_{ij} \rfloor \mid \lfloor x_{ij} \rfloor \neq 0, \lfloor x_{ij} \rfloor \neq 1\}. \tag{5.30}$$

If there are no arcs satisfying this condition, then the rule $BB1.1$ is applied. This rule aims to fix the values of the arcs from the depot, which are in fact the origin of the flow.

After selecting the arc, a binary branching is performed, creating two nodes. Let (i, j) be the selected arc, and x_{ij} the current fractional flow in that arc. In one branch, the following constraint is enforced:

$$x_{ij} = 1 \tag{5.31}$$

while in the other branch it is provided that

$$x_{ij} = 0. \tag{5.32}$$

On the one hand, with constraint (5.31), the arc must be included in the solution, and possibly it can be added to an existing sub-route. Consequently, customer j will be visited right after customer i. On the other hand, in opposition, constraint (5.32) imposes that the arc is excluded from any solution in the corresponding branch.

The insertion of each branching constraint consists in adding one row in the RMP. Let Ω' be the set of columns at the current node (with $\Omega' \subset \Omega$) and let (i', j') the selected arc. The left hand side of the inserted row corresponds to a vector $a_{new} = (a_{new,1}, a_{new,2}, a_{new,3}, \ldots, a_{new,|\Omega'|-1}, a_{new,|\Omega'|})$ where each element $a_{new,r}$ $(r = 1, 2, ..., |\Omega'|)$ takes value 1 if the arc (i', j') is traversed by route r, and it takes value 0 otherwise. The right hand side of the row takes value 1 if it imposed to traverse such arc, and it takes value zero otherwise.

5.5.2.2 Partition rules based on sets of variables

Using the rules with a single variable may lead to a branching tree with several levels. As referred to above, one may branch on the flow of a subset of arcs instead of a single variable. In these cases a set of original variables is selected. The sum of their flows will be selected to branch and then it is possible to have a greater impact in the solution. In one branch, the flow between these arcs must be increased, while in the other branch the opposite is imposed. In the following, we present the rules used to select those arcs.

BB2.1 Among the set of routes of the solution, the one that has the highest fractional value is selected. The sum of the flow in the arcs traversed by this route is computed. Let r be the selected route and P_r be the sequence of arcs traversed in route r. In one branch it is imposed that:

$$\sum_{(i,j)\in P_r} x_{ij} \geq \left\lceil \sum_{(i,j)\in P_r} x_{ij} \right\rceil, \tag{5.33}$$

while in the other branch the following constraint is applied:

$$\sum_{(i,j)\in P_r} x_{ij} \leq \left\lfloor \sum_{(i,j)\in P_r} x_{ij} \right\rfloor. \tag{5.34}$$

Guided by the solution provided by the linear relaxation of the RMP, this rule aims to explore the branching tree by fixing the sum of the flow of the arcs belonging to routes with highest fractional value.

BB2.2 Among the set of arcs of the solution, a subset of arcs with the highest flow is selected. Let $A' \subset A$ be that subset, whose cardinality is a parameter ($m_{arcs} = |A'|$). Then a binary branching is performed by imposing in one branch that

$$\sum_{(i,j)\in A'} x_{ij} \geq \left\lceil \sum_{(i,j)\in A'} x_{ij} \right\rceil, \tag{5.35}$$

while in the other branch it is provided that

$$\sum_{(i,j)\in A'} x_{ij} \leq \left\lfloor \sum_{(i,j)\in A'} x_{ij} \right\rfloor. \tag{5.36}$$

BB2.3 Among the set of arcs of the solution, a subset of arcs with the lowest flow is selected. Then, the branching is performed with the same branching constraints presented in rule $BB2.2$.

BB2.4 Among the set of routes of the solution, the one that has more visited customers is selected. The branching is performed with the same branching constraints presented in rule $BB2.1$.

The insertion of branching constraints resulting from partition rules with more than one original variable relies on the same method as the one used in single arc branching.

5.6 Partial enumeration algorithms

Taking into account the set of partition rules, it is necessary to define how to apply and combine them, and, in this sense, which strategies are used to explore the nodes of the branching tree. All of them rely on a depth-first search strategy for choosing the node of the branching tree to explore. Whenever it is not possible to dive down a given node, the search continues at the upper level of the branching tree. This happens when the LP relaxation of the RMP associated to the node is infeasible. It is worth noting that it is not possible to leave unexplored a node when the value of the LP relaxation of the RMP is greater than the value of the incumbent solution. This is due to the fact that the solution provided by the RMP relies on column generation based heuristic. Consequently, there is no guarantee that the value provided by the RMP is in fact a lower bound. In the following, we describe the three strategies implemented.

(Single rule) The branch-and-price is performed using only one partition rule. This means that in a given node, whenever there are no attractive columns and the solutions remains fractional, the search space is divided into two distinct search spaces according to the selected partition rule. The partition rule is always the same within all the branching tree. With this strategy, a different branching tree is derived for each branching rule.

(Random strategy) According to this strategy, a different partition rule is used in each branching operation: whenever is necessary to branch, a partition rule is randomly selected. This strategy provides greater diversity of the divided search spaces in each branch.

(Sequential strategy) In this strategy, the ten partition rules are sequenced according to the order they are presented in Section 5.5.2.1, *i.e.*, from the rule $BB1.1$

to the rule $BB2.4$. Then, the first partition rule of this sequence is selected to perform branching. The partition rule remains the same within the next n_{max} branching executions without improvement of the incumbent solution. When n_{max} is reached, the second partition rule is adopted from the current node for the next n_{max} iterations. The process is repeated by sequentially crossing all elements of the sequence. If all partition rules are used, the sequence is crossed again starting from the first rule.

5.7 Computational results

To evaluate and compare the performance of the different enumeration algorithms presented in Section 5.6, we conducted a set of preliminary computational experiments on 25 instances of the 2L-CVRP proposed in [76, 78, 64]. This set is described in Table 5.2, where m represents the number of customers while it represents the number of items. Additionally, the height and the width of the vehicle are presented in the last two columns.

The algorithms were coded in C++, and the tests were run on an Intel Xeon Processor E5-1620 v3 with 3.50 GHz and 64 GB of RAM.

5.7.1 Preliminary computational experiments

In the preliminary tests, we imposed a time limit of 300 seconds in the LP relaxation of the RMP, and a time limit of 1800 seconds in the branch-and-price. The parameters n_{init}, m_{arcs} and n_{max} are set at 50, 10 and 10, respectively.

In Table 5.3, we report on the average results for the 10 strategies described in Section 5.6, namely the Single Rule (SR), the Random Strategy (RS) and the Sequential Strategy (SS). It is important to note that strategy SR consists in 10 different runs, each one considering one partition rule. There is always an integer incumbent solution, which is initialized with a single customer route for each customer. Strategies SR $BB1.2$ and SR $BB1.3$ are omitted, since they not lead to improved results. The meaning of the columns in Table 5.3 is the following:

- $Inst$: number of the instance according to Table 5.2;

Table 5.2: Set of instances

Instance	*Name*	*Class*	*m*	*it*	*height*	*width*
1	E016-03m	1	15	15	40	20
2	E016-03m	2	15	24	40	20
3	E016-03m	3	15	31	40	20
4	E016-03m	4	15	37	40	20
5	E016-03m	5	15	45	40	20
6	E026-08m	1	25	25	40	20
7	E026-08m	2	25	40	40	20
8	E026-08m	3	25	61	40	20
9	E026-08m	4	25	63	40	20
10	E026-08m	5	25	91	40	20
11	E051-05e	1	50	50	40	20
12	E051-05e	2	50	82	40	20
13	E051-05e	3	50	103	40	20
14	E051-05e	4	50	134	40	20
15	E051-05e	5	50	157	40	20
16	E076-08s	1	75	75	40	20
17	E076-08s	2	75	112	40	20
18	E076-08s	3	75	154	40	20
19	E076-08s	4	75	198	40	20
20	E076-08s	5	75	236	40	20
21	E151-12b	1	150	150	40	20
22	E151-12b	2	150	225	40	20
23	E151-12b	3	150	298	40	20
24	E151-12b	4	150	366	40	20
25	E151-12b	5	150	433	40	20

- I_{OPT}: number of the instances (according to column *Inst*) in which the algorithm was able to achieve solutions that are better than the initial incumbent;
- sp_{LP}: number of subproblems solved before branching;
- $cols_{LP}$: number of generated columns during the LP relaxation of the RMP;
- sp_{BB}: number of subproblems solved in the branch-and-price;
- $cols_{BB}$: number of generated columns in the branch-and-price;
- nod_{BB}: number of branching nodes generated during branch-and-price, excluding the root;
- z_{LP} cost of the LP solution;
- z_{OPT} value of the best solution achieved.

The obtained results show that strategies based only on one rule (SR) lead to values of cost of the LP solution similar to the ones obtained with random and se-

Table 5.3: Computational results for the preliminary set of tests

	$Inst$	I_{OPT}	sp_{LP}	$cols_{LP}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{OPT}
SR (BB1.1)	1 to 5	1; 2; 4; 5	46	44	74	64	10	281,09	343
	5 to 10	1;3	63	61	111	101	10	530,64	976
	11 to 15	1;5	71	69	171	164	6	1016,76	2000
	16 to 20	1	82	80	243	240	3	1523,21	3015
	21 to 25	1	81	79	422	421	1	3737,07	5998
	avg.		*69*	*67*	*204*	*198*	*6*	*1417,75*	*2466*
SR (BB1.4)	1 to 5	1; 2; 3; 4; 5	40	38	101	91	11	282,47	298
	5 to 10	1; 2; 3; 4; 5	62	60	80	74	6	535,88	549
	11 to 15	1	75	73	187	184	3	977,48	2001
	16 to 20	1	82	80	250	248	2	1511,51	3012
	21 to 25	1	81	79	424	423	1	3705,37	5997
	avg.		*68*	*66*	*208*	*204*	*5*	*1402,54*	*2371*
SR (BB1.5)	1 to 5	1; 2	45	43	31	9	22	277,21	461
	5 to 10	1	63	61	63	42	20	530,04	1129
	11 to 15	1	75	73	150	142	8	986,43	2006
	16 to 20	1	82	80	235	233	2	1547,96	3008
	21 to 25	1	82	80	418	417	1	3722,53	5997
	avg.		*69*	*67*	*179*	*169*	*11*	*1412,83*	*2520*
SR (BB1.6)	1 to 5	1; 2; 3; 4; 5	44	42	23	19	4	278,87	278
	5 to 10	1; 3; 4; 5	62	60	99	91	8	541,77	663
	11 to 15	1	73	71	166	160	6	1023,62	2001
	16 to 20	1	82	80	237	234	3	1501,04	3011
	21 to 25	1	81	79	421	420	1	3749,97	5997
	avg.		*69*	*67*	*189*	*185*	*4*	*1419,05*	*2390*
SR (BB2.1)	1 to 5	1; 2; 3; 4; 5	47	45	49	43	6	277,67	282
	5 to 10	1; 2; 3; 5	65	63	118	110	8	533,56	695
	11 to 15	1	77	75	196	191	5	999,07	2006
	16 to 20	1	83	81	232	228	3	1529,20	3010
	21 to 25	1	82	80	405	404	1	3773,60	5998
	avg.		*71*	*69*	*200*	*195*	*5*	*1422,62*	*2398*
SR (BB2.2)	1 to 5	1; 2; 4; 5	44	42	70	62	7	279,99	343
	5 to 10	1	59	57	126	121	5	530,78	1127
	11 to 15	1	74	72	199	197	3	1000,01	2007
	16 to 20	1	82	80	269	267	2	1523,10	3012
	21 to 25	1	82	80	417	417	1	3722,49	5996
	avg.		*68*	*66*	*216*	*213*	*4*	*1411,27*	*2497*
SR (BB2.3)	1 to 5	1; 2; 4	46	44	72	68	4	280,50	404
	5 to 10	1	59	57	145	141	4	544,71	1132
	11 to 15	1	75	73	208	205	2	987,18	2001
	16 to 20	1	83	81	263	262	1	1518,59	3014
	21 to 25	1	81	79	425	424	1	3729,47	5997
	avg.		*69*	*67*	*222*	*220*	*2*	*1412,09*	*2510*
SR (BB2.4)	1 to 5	1; 2; 3; 4; 5	46	44	83	74	10	283,91	299
	5 to 10	1; 2; 3; 4; 5	64	62	116	106	10	531,53	549
	11 to 15	1	77	75	178	175	3	999,46	2001
	16 to 20	1	82	80	255	253	2	1534,11	3012
	21 to 25	1	81	79	425	425	1	3737,81	5998
	avg.		*70*	*68*	*211*	*206*	*5*	*1417,36*	*2372*
RS	1 to 5	1; 2; 4	47	45	68	63	4	277,82	403
	5 to 10	1; 4	62	60	133	128	6	524,79	972
	11 to 15	1	74	72	189	185	3	985,05	2008
	16 to 20	1	81	79	250	248	2	1513,53	3012
	21 to 25	1	81	79	430	429	1	3742,99	5996
	avg.		*69*	*67*	*214*	*211*	*3*	*1408,83*	*2478*
SS	1 to 5	1; 2; 3; 4; 5	45	43	76	67	9	283,12	285
	5 to 10	1	62	60	119	109	9	541,35	1130
	11 to 15	1	75	73	167	160	7	977,43	2004
	16 to 20	1	82	80	243	241	3	1528,39	3012
	21 to 25	1	82	80	419	418	1	3798,95	5998
	avg.		*69*	*67*	*205*	*1969*	*6*	*1425,81*	*2486*

quential schemes. Considering the values obtained with branch-and-price, the strategies SR ($BB1.4$) and SR ($BB2.4$) provide better average values. Not surprisingly, these strategies are also the ones providing a higher number of instances where the best solution is better than the initial incumbent (instances 1-11, 16, and 21). The number of generated columns tends to be greater for instances with a higher number of customers. In these instances, it is more difficult to achieve a solution better than the incumbent. Therefore, the branching continues giving rise to larger branching trees. However, within strategy SR ($BB1.1$), it was possible to update the incumbent for instance 15, which has 50 customers and more than 150 items. The strategy SR ($BB1.5$) leads to the worst average cost values. Indeed, with the exception of the pure CVRP instances, the algorithm was able to update the initial incumbent only for instance 2. Similar results were found by strategy SR ($BB2.3$). Finally, results concerning the random strategy and sequential strategy (RS and SS) lead to similar results.

5.7.2 Second set of computational experiments

Considering the preliminary results, we conducted a second set of computational experiments. For this purpose, we considered three subset of instances. According to Table 5.2, we selected the instances with 15, 50 and 150 customers, in order to build a set of instances with different sizes. Additionally, the time limit for the LP relaxation was increased to 450 seconds, as well as the branch-and-price duration, set to a maximum of 3600 seconds. The values of the remaining parameters were not modified.

In Tables 5.4 - 5.11, we present the second set of computational experiments for the single rule strategy, according to the different partition rules. Strategies SR $BB1.2$ and SR $BB1.3$ are omitted, since they not lead to improved results. Table 5.12 corresponds to the random strategy, while Table 5.13 presents the obtained results for the sequential strategy. In all the following tables, we use the same columns defined in Section 5.7.1, and the additional notation:

- t_{PP}: computing time for the initialization of the RMP (in seconds);

- t_{LP}: computing time for the LP relaxation (in seconds);
- t_{BB}: computing time for the branch-and-price phase (in seconds);
- t_{tot}: total computing time (in seconds).

Note that there is always an incumbent solution that is integer since its value is initialized with the cost corresponding to a single customer route for each customer. In this sense, all instances where an integer solution better than this initial value was found, were highlighted in bold (column z_{opt}). It must be observed that for the vast majority of the cases in which it was not possible to update the initial incumbent, the branch-and-price phase generates hundreds of columns while a small number of nodes is analyzed.

The subproblem is solved through heuristic methods, and then the LP-relaxation value cannot be used as a lower bound. Indeed, it is possible to achieve an integer solution better than the LP relaxation, as it can be seen, for example, in Table 5.4 for instance 2.

In some cases an integer solution is found without having to resort to branch-and-price. It is the case of all instances belonging to Class 1 (pure CVRP instances). Additionally, it is also the case of instance 2 (for strategies SR ($BB1.5$), SR ($BB1.6$), SR ($BB2.2$), SR ($BB2.3$), and SS) and the case of instance 4 (for the strategies SR ($BB1.1$), SR ($BB1.4$) and SR ($BB2.2$)).

Although all the cases reported to above are related to instances of 15 customers, an integer solution without having to resort to branch-and-price was obtained for instance 14, which has 50 customers and more than 130 items (considering the sequential strategy in Table 5.13).

The obtained results for strategy SR ($BB1.1$) show an average improvement in the cost of the best solution achieved for instances 1 to 5 (roughly 17%), when compared with the preliminary tests. For the remaining instances, the values are very close to those obtained in the preliminary tests for the same strategy. Similar improvements were found within strategy SR ($BB1.4$) for instances 11 to 15, reaching an average improvement of roughly 12,69%.

The values for strategy SR ($BB1.5$) are very similar to those obtained in the preliminary phase. The strategy SR ($BB2.1$) presents the worst average values concerning instances with 15 customers, but presents important improvements for the instances with 50 customers (roughly 14%). The strategy SR ($BB2.3$) leads to an average improvement of 12,17% considering 15 customers. For instances with a greater number of customers, the average improvement is less significant.

Table 5.4: Computational results for the single rule strategy with partition rule BB1.1

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	5	3	0	0	0	237,00	**237**	3,00	117,21	0,00	120,21
2	48	46	39	37	2	291,00	**281**	18,01	345,18	456,27	819,46
3	66	64	235	214	21	282,96	**314**	30,01	474,26	3604,86	4109,13
4	64	62	0	0	0	301,00	**301**	21,02	459,21	0,00	480,23
5	60	58	267	236	31	279,82	**282**	36,03	408,24	3611,06	4055,32
avg.	*49*	*47*	*108*	*97*	*11*	*278,36*	*283*	*21,61*	*360,82*	*1534,44*	*1916,87*
11	9	7	0	0	0	541,00	**541**	3,03	211,77	0,00	214,80
12	136	134	261	241	20	925,99	2366	3,00	456,30	3605,24	4064,55
13	143	141	263	237	26	929,77	2366	3,00	465,35	3614,31	4082,66
14	119	117	300	279	21	1142,25	2366	3,00	462,44	3608,71	4074,15
15	147	145	373	352	21	932,08	2366	3,01	453,81	3626,37	4083,20
avg.	*111*	*109*	*239*	*222*	*18*	*894,22*	*2001*	*3,01*	*409,93*	*2890,93*	*3303,87*
21	10	8	0	0	0	1002,00	**1002**	3,31	191,77	0,00	195,08
22	151	149	689	685	4	3358,14	7244	3,01	455,69	3600,68	4059,38
23	150	148	676	674	2	3810,80	7244	3,06	454,58	3636,97	4094,61
24	150	148	756	752	4	4154,27	7244	3,02	453,70	3607,88	4064,59
25	147	145	834	833	1	4029,55	7244	3,24	455,27	3613,75	4072,26
avg.	*122*	*120*	*591*	*589*	*2*	*3270,95*	*5996*	*3,13*	*402,20*	*2891,86*	*3297,18*

Table 5.5: Computational results for the single rule strategy with partition rule BB1.4

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	6	4	0	0	0	231,00	**231**	3,02	129,19	0,00	132,21
2	60	58	232	203	29	289,75	**324**	33,01	465,22	3610,62	4108,85
3	74	72	212	178	34	282,80	**315**	27,01	501,22	3601,90	4130,13
4	48	46	0	0	0	306,00	**306**	18,01	390,16	0,00	408,17
5	64	62	306	283	23	272,25	**341**	48,03	417,26	3614,09	4079,38
avg.	*50*	*48*	*150*	*133*	*17*	*276,36*	*303*	*25,82*	*380,61*	*2165,32*	*2571,75*
11	5	3	0	0	0	565,00	**565**	3,07	151,12	0,00	154,19
12	138	136	384	372	12	914,30	2366	3,00	453,26	3602,19	4058,46
13	128	126	373	356	17	964,53	2366	3,01	453,35	3605,53	4061,89
14	113	111	288	271	17	1230,99	**1074**	3,01	456,32	3602,65	4061,98
15	143	141	469	460	9	980,54	2366	3,01	453,69	3605,07	4061,76
avg.	*105*	*103*	*303*	*292*	*11*	*931,07*	*1747*	*3,02*	*393,55*	*2883,09*	*3279,66*
21	10	8	0	0	0	1011,00	**1011**	3,35	240,29	0,00	243,65
22	150	148	724	722	2	3418,31	7244	3,01	453,05	3604,88	4060,94
23	150	148	670	667	3	3907,73	7244	3,03	454,68	3618,04	4075,75
24	150	148	761	759	2	4310,32	7244	3,02	453,40	3607,51	4063,93
25	145	143	856	855	1	4101,26	7244	3,48	455,74	3600,24	4059,46
avg.	*121*	*119*	*602*	*601*	*2*	*3349,72*	*5997*	*3,18*	*411,43*	*2886,13*	*3300,74*

Table 5.6: Computational results for the single rule strategy with partition rule BB1.5

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	5	3	0	0	0	241,00	**241**	3,02	105,12	0,00	108,14
2	54	52	0	0	0	281,00	**281**	30,02	336,19	0,00	366,21
3	65	63	88	17	71	283,08	596	15,01	492,22	3601,79	4109,02
4	52	50	103	30	73	299,08	596	21,01	456,22	3631,54	4108,76
5	70	68	91	24	67	281,89	596	36,02	489,27	3614,03	4139,32
avg.	*49*	*47*	*56*	*14*	*42*	*277,21*	*462*	*21,02*	*375,80*	*2169,47*	*2566,29*
11	8	6	0	0	0	554,00	**554**	3,03	217,57	0,00	220,59
12	133	131	149	100	49	898,71	2366	3,01	456,31	3611,25	4070,57
13	142	140	219	175	44	974,18	2366	3,01	459,39	3623,24	4085,63
14	92	90	253	223	30	1304,55	2366	3,01	459,35	3611,38	4073,73
15	145	143	270	236	34	964,77	2366	3,01	453,79	3607,14	4063,93
avg.	*104*	*102*	*178*	*147*	*31*	*939,24*	*2004*	*3,01*	*409,28*	*2890,60*	*3302,89*
21	9	7	0	0	0	1007,00	**1007**	3,23	234,20	0,00	237,43
22	150	148	702	699	3	3421,42	7244	3,02	453,11	3609,84	4065,97
23	150	148	663	660	3	3970,45	7244	3,02	455,67	3601,60	4060,28
24	148	146	686	683	3	4111,08	7244	3,03	453,19	3603,75	4059,97
25	147	145	852	851	1	4078,67	7244	3,90	453,91	3617,98	4075,78
avg.	*121*	*119*	*581*	*579*	*2*	*3317,72*	*5997*	*3,24*	*410,02*	*2886,63*	*3299,89*

Table 5.7: Computational results for the single rule strategy with partition rule BB1.6

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	5	3	0	0	0	234,00	**234**	3,03	135,24	0,00	138,27
2	73	71	0	0	0	281,00	**281**	24,01	477,20	0,00	501,22
3	67	65	148	120	28	277,67	**286**	39,02	420,23	2590,37	3049,61
4	55	53	64	54	10	303,67	**291**	21,01	456,25	1077,53	1554,79
5	69	67	277	250	27	279,60	**287**	30,01	483,28	3608,12	4121,41
avg.	*54*	*52*	*98*	*85*	*13*	*275,19*	*276*	*23,42*	*394,44*	*1455,20*	*1873,06*
11	7	5	0	0	0	539,00	**539**	3,01	184,34	0,00	187,35
12	137	135	263	243	20	920,34	2366	3,01	456,32	3611,21	4070,54
13	138	136	338	314	24	991,39	2366	3,01	453,40	3620,61	4077,02
14	109	107	265	243	22	1209,19	2366	3,00	459,37	3620,74	4083,11
15	146	144	372	353	19	935,54	2366	3,00	468,65	3638,63	4110,28
avg.	*107*	*105*	*248*	*231*	*17*	*919,09*	*2001*	*3,01*	*404,42*	*2898,24*	*3305,66*
21	10	8	0	0	0	1002,00	**1002**	3,34	245,57	0,00	248,91
22	151	149	668	664	4	3310,02	7244	3,01	456,01	3617,66	4076,68
23	150	148	669	666	3	3944,82	7244	3,40	453,74	3606,99	4064,12
24	149	147	681	676	5	4185,00	7244	3,01	454,05	3616,07	4073,14
25	146	144	810	808	2	4096,23	7244	3,05	454,22	3614,74	4072,01
avg.	*121*	*119*	*566*	*563*	*3*	*3307,61*	*5996*	*3,16*	*412,72*	*2891,09*	*3306,97*

Table 5.8: Computational results for the single rule strategy (partition rule BB2.1)

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	7	5	0	0	0	231,00	**231**	3,03	141,20	0,00	144,22
2	52	50	38	34	4	291,75	**281**	30,02	342,15	468,24	840,41
3	75	73	158	109	49	287,09	**379**	39,02	462,24	3607,77	4109,03
4	57	55	170	133	37	300,67	**304**	21,02	459,20	3637,76	4117,98
5	79	77	242	211	31	276,33	**287**	48,04	543,33	3647,06	4238,42
avg.	*54*	*52*	*122*	*97*	*24*	*277,37*	*296*	*28,22*	*389,62*	*2272,17*	*2690,01*
11	9	7	0	0	0	542,00	**542**	3,04	163,37	0,00	166,41
12	134	132	359	349	10	918,25	2366	3,00	459,35	3614,25	4076,60
13	144	142	424	410	14	955,76	**987**	3,01	459,34	3623,74	4086,09
14	97	95	324	312	12	1277,18	2366	3,01	453,33	3603,17	4059,51
15	142	140	490	481	9	956,15	2366	3,01	459,81	3632,39	4095,20
avg.	*105*	*103*	*319*	*310*	*9*	*929,87*	*1725*	*3,01*	*399,04*	*2894,71*	*3296,76*
21	10	8	0	0	0	1008,00	**1008**	3,38	215,35	0,00	218,73
22	150	148	679	676	3	3269,48	7244	3,04	453,13	3604,17	4060,34
23	150	148	681	678	3	3950,66	7244	3,03	455,85	3611,00	4069,88
24	150	148	684	680	4	4198,37	7244	3,34	455,58	3624,29	4083,21
25	148	146	834	833	1	4129,13	7244	3,04	454,10	3607,23	4064,37
avg.	*122*	*120*	*576*	*573*	*2*	*3311,13*	*5997*	*3,17*	*406,80*	*2889,34*	*3299,31*

Table 5.9: Computational results for the single rule strategy (partition rule BB2.2)

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	7	5	0	0	0	229,00	**229**	3,01	138,58	0,00	141,59
2	51	49	0	0	0	293,00	**293**	18,07	427,53	0,00	445,60
3	75	73	193	158	35	280,00	**287**	33,12	520,89	3619,07	4173,07
4	64	62	0	0	0	293,00	**293**	18,07	484,76	0,00	502,82
5	56	54	318	295	23	276,91	596	42,15	397,46	3622,09	4061,70
avg.	*51*	*49*	*102*	*91*	*12*	*274,38*	*340*	*22,88*	*393,84*	*1448,23*	*1864,96*
11	8	6	0	0	0	555,00	**555**	3,14	157,67	0,00	160,81
12	131	129	461	453	8	930,87	2366	3,03	454,65	3601,10	4058,77
13	141	139	393	381	12	972,83	2366	3,01	460,68	3610,27	4073,96
14	95	93	319	306	13	1333,76	2366	3,01	454,63	3604,04	4061,69
15	141	139	511	505	6	979,88	2366	3,01	454,85	3602,41	4060,27
avg.	*103*	*101*	*337*	*329*	*8*	*954,47*	*2004*	*3,04*	*396,50*	*2883,56*	*3283,10*
21	6	4	0	0	0	997,00	**997**	3,32	174,25	0,00	177,58
22	150	148	703	701	2	3400,55	7244	3,03	455,94	3604,00	4062,97
23	150	148	655	651	4	4030,92	7244	3,32	455,38	3603,15	4061,86
24	149	147	747	743	4	4151,91	7244	3,60	454,98	3603,17	4061,75
25	147	145	818	817	1	3979,01	7244	3,11	455,07	3605,65	4063,82
avg.	*120*	*118*	*585*	*582*	*2*	*3311,88*	*5995*	*3,28*	*399,12*	*2883,19*	*3285,59*

Table 5.10: Computational results for the single rule strategy (partition rule BB2.3)

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	6	4	0	0	0	224,00	**224**	3,01	150,57	0,00	153,58
2	63	61	0	0	0	287,00	**287**	33,12	460,67	0,00	493,79
3	66	64	266	241	25	283,33	596	15,05	505,85	3607,05	4127,95
4	67	65	162	115	47	301,50	596	21,08	472,70	3619,05	4112,82
5	69	67	323	304	19	284,50	596	48,17	460,65	3604,09	4112,92
avg.	*54*	*52*	*150*	*132*	*18*	*276,07*	*460*	*24,09*	*410,09*	*2166,04*	*2600,21*
11	6	4	0	0	0	576,00	**576**	3,07	178,92	0,00	181,99
12	130	128	580	577	3	947,96	2366	3,01	472,71	3607,12	4082,84
13	139	137	494	489	5	948,00	2366	3,01	457,71	3601,02	4061,73
14	104	102	284	275	9	1243,83	2366	3,03	454,65	3604,17	4061,84
15	130	128	363	361	2	1025,44	2366	3,03	454,76	3601,67	4059,45
avg.	*102*	*100*	*344*	*340*	*4*	*948,25*	*2008*	*3,03*	*403,75*	*2882,79*	*3289,57*
21	11	9	0	0	0	1007,00	**1007**	3,56	258,17	0,00	261,72
22	149	147	1013	1012	1	3320,85	7244	3,03	453,23	3602,93	4059,19
23	150	148	769	768	1	3809,33	7244	3,46	455,52	3600,09	4059,07
24	149	147	803	801	2	4210,57	7244	3,01	454,87	3602,47	4060,36
25	146	144	875	874	1	4093,24	7244	3,65	453,35	3606,43	4063,43
avg.	*121*	*119*	*692*	*691*	*1*	*3288,20*	*5997*	*3,34*	*415,03*	*2882,39*	*3300,75*

Table 5.11: Computational results for the single rule strategy (partition rule BB2.4)

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	6	4	0	0	0	229,00	**229**	3,01	195,81	0,00	198,82
2	60	58	74	66	8	288,19	**281**	30,11	460,67	1050,82	1541,60
3	67	65	131	73	58	277,00	**297**	24,09	493,77	3622,20	4140,06
4	59	57	157	107	50	307,00	**342**	18,07	475,72	3622,11	4115,90
5	68	66	248	222	26	278,67	**291**	48,17	406,46	3628,13	4082,76
avg.	*52*	*50*	*122*	*94*	*28*	*275,97*	*288*	*24,69*	*406,49*	*2384,65*	*2815,83*
11	5	3	0	0	0	542,00	**542**	3,01	114,88	0,00	117,89
12	135	133	375	366	9	919,32	**1104**	3,01	460,67	3604,17	4067,85
13	146	144	359	340	19	949,63	**1102**	3,03	457,69	3601,06	4061,78
14	95	93	295	273	22	1315,12	**1098**	3,01	478,73	3619,21	4100,95
15	144	142	307	299	8	931,42	**1027**	3,01	455,04	3605,14	4063,18
avg.	*105*	*103*	*267*	*256*	*12*	*931,50*	*975*	*3,01*	*393,40*	*2885,91*	*3282,33*
21	6	4	0	0	0	998,00	**998**	3,34	148,56	0,00	151,90
22	150	148	764	760	4	3467,62	7244	3,03	455,08	3600,19	4058,30
23	150	148	737	734	3	3965,66	7244	3,24	455,83	3600,71	4059,78
24	149	147	784	782	2	4147,36	7244	3,71	453,82	3609,44	4066,97
25	147	145	867	866	1	4019,73	7244	3,23	455,69	3603,31	4062,23
avg.	*120*	*118*	*630*	*628*	*2*	*3319,67*	*5995*	*3,31*	*393,80*	*2882,73*	*3279,84*

Table 5.12: Computational results for the random strategy

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	7	5	0	0	0	235,00	**235**	3,01	156,70	0,00	159,71
2	59	57	243	215	28	288,83	**308**	12,04	478,75	3610,08	4100,87
3	63	61	230	203	27	279,71	**315**	33,024	522,26	3620,18	4175,45
4	56	54	40	34	6	291,00	**291**	21,08	409,49	680,47	1111,03
5	86	84	324	301	23	281,20	**295**	21,08	535,94	3613,03	4170,04
avg.	*52*	*50*	*152*	*138*	*14*	*274,01*	*282*	*14,30*	*395,22*	*1975,90*	*2385,42*
11	10	8	0	0	0	539,00	**539**	3,03	251,19	0,00	254,22
12	139	137	409	402	7	916,75	2366	3,03	457,66	3601,00	4061,69
13	143	141	409	399	10	942,41	**990**	3,03	454,63	3613,06	4070,72
14	84	82	315	300	15	1351,32	**1227**	3,01	469,70	3625,31	4098,02
15	144	142	466	456	10	942,17	2366	3,01	454,96	3602,09	4060,06
avg.	*104*	*102*	*320*	*311*	*8*	*938,33*	*1498*	*3,02*	*417,63*	*2888,29*	*3308,94*
21	6	4	0	0	0	1016,00	**1016**	3,29	147,58	0,00	150,87
22	150	148	706	705	1	3483,78	7244	3,03	454,45	3601,31	4058,78
23	149	147	664	661	3	3899,39	7244	3,01	453,63	3636,90	4093,54
24	149	147	748	744	4	4622,28	7244	3,23	453,20	3604,66	4061,09
25	146	144	840	839	1	4055,50	7244	3,25	453,13	3619,83	4076,21
avg.	*120*	*118*	*592*	*590*	*2*	*3415,39*	*5998*	*3,16*	*392,40*	*2892,54*	*3288,10*

Table 5.13: Computational results for the sequential strategy

Inst	sp_{LP}	$cols_{lp}$	sp_{BB}	$cols_{BB}$	nod_{BB}	z_{LP}	z_{opt}	t_{PP}	t_{LP}	t_{BB}	t_{tot}
1	9	7	0	0	0	238,00	**238**	3,01	189,23	0,00	192,24
2	55	53	0	0	0	287,00	**287**	39,02	420,39	0,00	459,41
3	75	73	150	130	20	277,33	**286**	24,02	507,55	2335,31	2866,87
4	60	58	0	0	0	293,00	**293**	21,08	532,91	0,00	553,99
5	74	72	255	231	24	278,27	**287**	18,07	460,64	3600,14	4078,85
avg.	*55*	*53*	*81*	*72*	*9*	*274,72*	*278*	*21,04*	*422,14*	*1187,09*	*1630,27*
11	8	6	0	0	0	542,00	**542**	3,03	221,16	0,00	224,19
12	139	137	343	322	21	933,66	2366	3,01	454,66	3634,09	4091,76
13	134	132	365	349	16	989,20	2366	3,01	454,65	3607,13	4064,79
14	96	94	0	0	0	1238,00	**1238**	3,03	457,64	0,00	460,67
15	140	138	384	369	15	935,56	2366	3,04	455,01	3605,04	4063,09
avg.	*103*	*101*	*218*	*208*	*10*	*927,68*	*1776*	*3,02*	*408,62*	*2169,25*	*2580,90*
21	4	2	0	0	0	998,00	**998**	3,32	116,31	0,00	119,64
22	150	148	660	657	3	3295,23	7244	3,21	454,16	3603,89	4061,26
23	150	148	659	657	2	3946,71	7244	3,04	454,63	3606,37	4064,04
24	150	148	699	697	2	4133,27	7244	3,01	454,82	3601,69	4059,52
25	147	145	871	870	1	3990,57	7244	3,18	454,27	3608,32	4065,77
avg.	*120*	*118*	*578*	*576*	*2*	*3272,76*	*5995*	*3,15*	*386,84*	*2884,05*	*3274,05*

5.8 Conclusions

In this chapter, we presented a column generation based heuristic algorithm for the Capacitated Vehicle Routing Problem with Two-dimensional Loading constraints (2L-CVRP). The master problem relies on a set partitioning formulation, while the subproblem corresponds to an elementary shortest path problem with loading constraints. In order to solve the subproblem, we use the variable neighborhood search algorithm described in Chapter 4. The overall approach includes the generation of valid dual inequalities in order to accelerate the convergence. A branch-and-price approach was also implemented. We conducted an extensive computational results using benchmark instances from literature. The obtained results provided good solutions for small size instances. However, for instances with a greater number of customers, the algorithm tends to have more difficulty in finding integer solutions with acceptable values.

Chapter 6

Column generation based heuristics for the vehicle routing problem with loading constraints

Contents

6.1 Introduction

In the previous chapter, it became clear that column generation algorithm for the 2L-CVRP has some limitations, particularly when dealing with a large number of customers. To overcome this limitation, in this chapter, we explore a set of new heuristic strategies integrated within the column generation algorithm, to solve the problem described in Section 2.3.1. These heuristics rely on constructive procedures that iteratively build a solution using the solution of LP relaxation or the solutions provided by a mixed integer programming model. In the latter case, some variables are selected, and enforced to be integer. The pricing subproblem is also heuristically solved, using the variable neighborhood search algorithm described in Chapter 4.

6.2 Heuristics based on mathematical programming

Due to the complexity of many problems, heuristics based on mathematical programming can be effective in achieving good solutions in acceptable time. Ball [9] defined four classes of heuristics based on mathematical programming. The first class includes the decomposition methods, which are applied to a original problem, dividing it into a set of subproblems that are solved up to optimality. It is desirable that this procedure turns the original problem in a set of smaller size subproblems. The second class refers to the improvement heuristics. These heuristics consist in methods that make use of an initial solution in order to derive a better one, using a mathematical programming model. The third class corresponds to the methods that use mathematical programming algorithms in order to derive approximate solutions. Some of the methods included in this class can be used for column generation approaches, such as diving heuristics, that we will describe in next section. Finally, the last class refers to the relaxation of the original problem. Some relaxation methods are mentioned by the author as the Lagrangian relaxation based heuristics or rounding the solution to a liner programming solution, among others.

6.2.1 Column generation based heuristics

Column generation algorithms may have very slow convergence for hard optimization problems. This limitation motivated several column generation based heuristics approaches. In some cases, the pricing subproblem is solved through heuristic methods, as presented in Chapter 5. In other cases, the heuristics are applied in the space of the master problem, ensuring that the subproblem remains tractable. In this chapter, we will give special emphasis to the latter type of heuristic methods.

Joncour *et al.* [80] analyzed two different methods. The first one is the so-called restricted master heuristic which consists in confining the RMP to a subset of columns and solving it as an integer programming formulation. The authors stated that this subset can be heuristically created or it can rely on the columns generated within the LP relaxation. Additionally, this confined subset can result from the combination of both procedures. However, the simple selection of a subset of columns can lead to infeasible solutions, and in this sense, some additional procedures may be needed. The use of this method was recently proposed for the routing problem with profits [4] or for the distance constrained multiple vehicle traveling purchaser problem [16].

In contrast to the first method, the second one starts from an empty solution. It relies on the definition of heuristic procedures to select columns to be added to the RMP until a feasible solution is reached. As an example of this second method, the authors presented the rounding heuristics, which consists in iteratively fixing a column with fractional value to an integer value. Rounding heuristics are commonly used in a wide range of problems. Concerning the recent approaches for the routing field, this rounding method was applied in the overall approach proposed by Spliet and Desaulniers [138] for the discrete time window assignment vehicle routing problem, or in the algorithm proposed by Macedo *et al.* for the multi-trip location routing [100]. However, and as stressed in [80], it can be difficult to achieve a feasible solution, and to overcome this limitation, several works make use of diving heuristics. These heuristics consist in fixing the variables to take a given value and solving again the LP relaxation of the RMP, which corresponds to a depth-first heuristic in the branching tree.

6.3 Column generation based heuristics for the 2L-CVRP

In this section, six different column generation heuristics are presented. With the exception of the first one, all the other approaches have the same principle: the solution is iteratively built, based on the selection of a given column provided by the solution of the set-partitioning problem (5.12)-(5.14) formulated in Chapter 5. In some approaches, the column is selected among those generated during the solution of the LP relaxation. In others, the column is selected only after fixing some variables to be integer, and solving the restricted master problem as a static Mixed Integer Programming (MIP) model.

Again, with the exception for the first strategy, different criteria were used to select one column to be added to a partial solution. After selecting a given column, which corresponds to a route, all the customers visited by the route associated to that column are removed from the formulation, and the LP relaxation is solved for the new formulation. This procedure will iteratively tackle a problem smaller than the previous one, and can help the convergence of the algorithm.

6.3.1 Approach I

In a first step, the LP-relaxation is solved using the column generation approach provided in Chapter 5, without branch-and-price, *i.e.*, the LP-relaxation is solved only at the root of the branching tree. Generally speaking, the approach defined in that chapter consists in a restricted master problem with a set-partitioning structure and a subproblem modelled as an elementary shortest path problem with two-dimensional loading constraints. The subproblem is solved with a variable neighborhood search algorithm, and some valid dual inequalities are used aiming to speed up the convergence of the algorithm.

After obtaining the optimal LP solution, and using all the columns of the restricted master problem, all the variables are enforced to take an integer value, *i.e.*, the restricted master problem is solved as a static integer programming (IP) model. This procedure corresponds to the restricted master heuristic but using all the columns generated during the LP-relaxation.

With this procedure, we aim to obtain a fast solution. However, enforcing integrality right after the LP relaxation of the restricted master problem can lead to very poor quality solutions. In the following approaches, other methods are described, which may expectably overcome this issue.

6.3.2 Approach II

Packing is the key feature of the 2L-CVRP, and it is known that good quality solutions for the 2L-CVRP are related with higher usage of the loading area of vehicles. Since the solution provided by *Approach I* (Section 6.3.1) is feasible, it can be analyzed in terms of usage of the vehicle loading area. In this sense, the column corresponding to the route with the highest usage is added to the final solution.

After adding a route to the final solution, the size of the model formulation is reduced. Whenever a route is selected as a part of the final solution, all the constraints of type (5.13) associated to the customers visited in the route are removed from the master problem formulation. Then, the LP relaxation of the new restricted master problem is solved and the procedure is repeated. This approach is summarized in Algorithm 2.

The procedures referred to above aim to iteratively add to the final solution the best combinations of customers, *i.e.*, the customers which provide the best usage of the loading area. The best usage route of the LP relaxation corresponds to a route combining the optimization of area usage and the distance cost. From a constructive standpoint, in each iteration, the best route will be added to the partial solution. Then, the model will recombine the remaining customers.

6.3.3 Approach III

It is clear that the solution provided by LP relaxation is at least as good as the solution provided by MIP when fixing all the variables to be integer. However, the former solution may contain variables with fractional values, and therefore, the solution is not feasible in the context of the 2L-CVRP. Nevertheless, each variable corresponds

Algorithm 2: Approach II

```
Input: $N$: set of customers;
Output: $S$: Feasible solution of the $2L-CVRP$;

$S = \emptyset$
repeat
    .Solve the LP relaxation;
    .Solve RMP as a static IP;
    .Select the best usage route $r$;
    .Add $r$ to partial solution $S$;
    .Remove customers visited in $r$ from $N$;
    .Resize the master problem formulation;
until $N = \emptyset$;
return $S$;
```

to a feasible route.

It is possible to select routes to be added to a partial solution right after obtaining the LP relaxation of the RMP. At this step, it is necessary to define a rule to select the routes to be added. The iterative inclusion of routes associated to decision variables whose values are close to one may provide good approximations to the optimal solution. This is due to the fact that if those values are close to one, then a solution where they are in fact one may not be too far from the LP solution.

In this approach, not all variables whose values are close to one are added to the partial solution. It can be profitable to add just a subset of routes and to solve again the LP-relaxation. This is due to the fact that an integer solution can be obtained with the insertion of that subset. Furthermore, it can happen that variables that are not close to one in the beginning of the algorithm, can achieve that value in following iterations.

Let Ω be the set of all variables $(\lambda_1, \lambda_2, \ldots, \lambda_p)$ generated in the LP relaxation. Each variable corresponds to one route. Then, n_{fix} variables are selected among those which comply to the following constraint:

$$\lambda_i \geq 1 - \epsilon_3, \lambda_i \in \Omega. \tag{6.1}$$

Typical values for ϵ_3 are between 0.1 and 0.25. The n_{fix} routes are fixed to one to be part of the final solution. As happens in Section 6.3.2, after adding this subset of routes to the final solution, the size of the model is reduced by removing the constraints associated with each customer belonging to the set of n_{fix} routes. The LP relaxation of the new restricted master problem is solved and the procedure is repeated until all customers are part of the final solution or until there are no routes satisfying condition (6.1). In the former case, a feasible solution is obtained. In the latter case, the last RMP is solved as a static IP model. This strategy is outlined in Algorithm 3.

Algorithm 3: Approach III

```
Input: $N$: set of customers;
       $n_{fix}$: maximum number of routes added in each iteration;
       $\epsilon_3$: parameter for selecting variables;
Output: $S$: Feasible solution of the $2L-CVRP$;

$S = \emptyset$
repeat
    .Solve the LP relaxation;
    .Select up to $n_{fix}$ routes satisfying condition (6.1);
    .Add the selected routes to partial solution $S$;
    .Remove customers visited in selected routes from $N$;
    .Resize the master problem formulation;
until $N = \emptyset \bigvee \nexists \lambda_i \in \Omega$ satisfying condition (6.1);
if $N \neq \emptyset$ then
    .Solve RMP as a static IP;
end
return $S$;
```

An example is presented in Figure 6.1. For the sake of simplicity, only a partial solution is presented. Table 6.1 corresponds to the partial LP relaxation solution. In this example, it is assumed $\epsilon_3 = 0.1$ and n_{fix} is equal to one.

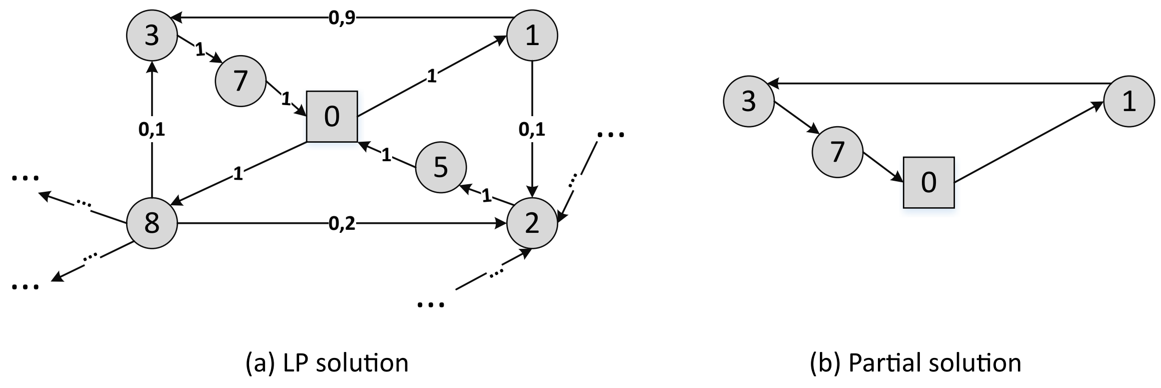

Figure 6.1: An example of *Approach III*

Table 6.1: Partial Solution of LP relaxation

customer	λ_1	...	λ_5	...	λ_8	λ_9	...		
1	1	...	1	...			...	=	1
2	1	...		...	1		...	=	1
3		...	1	...		1	...	=	1
4		...		...			...	=	1
5	1	...		...	1		...	=	1
6		...		...			...	=	1
7		...	1	...		1	...	=	1
8		...		...	1	1	...	=	1
9		...		...			...	=	1
	0,1	...	0,9	...	0,2	0,1	...		

6.3.4 Approach IV

In the fourth approach, the solution of the LP relaxation is also analyzed. In this solution, variables taking values close to an integer value (0 or 1) suggest that one may have a good approximation to the optimal solution, when those variables are in fact integer. Therefore, forcing those variables to be integer and solving the RMP as a static MIP can provide a good quality solution that is not far from the LP solution.

In other words, this procedure will require the decision of including or not the routes corresponding to those variables in the solution. If they are in fact included, it can mean that a good quality solution includes this route. Thus, it is possible to build a heuristic procedure which iteratively adds routes with this MIP formulation.

All variables whose values are close to zero or close to one are enforced to be integer. More precisely, and assuming the notation above, we enforce integrality to all variables satisfying the condition

$$\lambda_i \geq 1 - \epsilon_4, \forall \lambda_i \in \Omega, \tag{6.2}$$

or the condition

$$\lambda_i \leq \epsilon_4, \forall \lambda_i \in \Omega. \tag{6.3}$$

Typical values for ϵ_4 are between 0.1 and 0.25. After solving the MIP model, the decision variables which were enforced to be integer and take now value one correspond to routes which will be added to the partial solution. Therefore, the customers of these routes are removed from the model. The size of the model is reduced since all the constraints associated to these customers are removed from the master problem formulation.

At this step, we solve the LP relaxation of the new master problem, and the process is repeated from the beginning, until all customers are assigned to the partial solution or until there are no columns satisfying the condition 6.2 or 6.3. In the former case, a final heuristic solution is obtained. In the latter case, the last RMP is solved resorting to the *Approach I.* This strategy is outlined in Algorithm 4.

6.3.5 Approach V

In the previous approach, it can happen that there are no variables satisfying the conditions (6.2) or (6.3). Consequently, the overall method is confined to solve the RMP as an IP model, hence reducing to the *Approach I.* These situations happen when the value of the variables are not close to zero or one. Thus, it is possible to adapt *Approach IV*, but, instead of selecting variables with values close to zero or one,

Algorithm 4: Approach IV

```
Input:
    $N$: set of customers;
    $\epsilon_4$: parameter for selecting variables; typically, $\epsilon_4 \in [0.1, 0.25]$;
Output:
    $S$: Feasible solution of the $2L - CVRP$;

repeat
    .Solve the LP relaxation;
    .Enforce integrality for all variables satisfying condition
    (6.2) or (6.3);
    .Solve RMP as a static MIP;
    .Add routes for the selected variables taking value 1 to $S$
    .Remove customers assigned to $S$ from $N$;
    .Resize the master problem formulation;
until $N = \emptyset \bigvee \nexists \lambda_i \in \Omega$ satisfying condition (6.2) or (6.3);
if $N \neq \emptyset$ then
    .Solve RMP as a static IP;
end
return $S$;
```

selecting those that have values more distant from integer values. This procedure will enforce the model to select or not these variables to be part of the solution.

More precisely, and taking into account the notation on previous approach, after solving the LP relaxation we enforce integrality to all variables which satisfy the following condition:

$$\epsilon_5 \leq \lambda_i \leq 1 - \epsilon_5, \forall \lambda_i \in \Omega. \tag{6.4}$$

Typical values for ϵ_5 are between 0.35 and 0.4. As in the previous approach, the master problem is solved as a static MIP model, and the decision variables (*i.e.*, the routes) which take value one are added to the partial solution. The model is resized and the LP relaxation of the new model is solved. Again, the process is repeated until all customers are assigned to the partial solution or until there are no columns satisfying the condition 6.4. In this last case, the restricted master problem is solved as a static integer programming model.

6.3.6 Approach VI

If the solution obtained with LP relaxation includes more than one route visiting the same customer, then the variables of the columns corresponding to those routes have fractional values. If all these values were integer, then there would be only one route for assigning each customer. Based on this idea, it is possible to build a heuristic solution by selecting one and only one route for customers belonging to several routes. Indeed, this approach aims to iteratively enforce the model to select only one route for each customer assigned to more than one route.

More precisely, after solving the LP relaxation, we select the customer that is assigned to more fractional routes. A possible interpretation is that this these customers are the ones which the model has more difficult to assign. The decision variables that are associated to routes visiting this customer are enforced to be integer and the restricted master problem is solved as a static MIP model.

From the obtained solution, among the variables which were enforced to be integer, we select the one which takes value one. This variable corresponds to a route which is added to the partial solution. Therefore, the model is reformulated by removing all constraints related to the customers visited in the selected route. Then, the LP relaxation is solved and the process is repeated until all customers are assigned to the partial solution.

With this procedure, the MIP model will provide only one route for a given selected customer. Consequently, all the other combinations are forbidden, which can simplify the remaining problem in next iterations. This approach is summarized in Algorithm 5.

Algorithm 5: Approach VI

```
Input:
    $N$: set of customers;
Output:
    $S$: Feasible solution of the $2L-CVRP$;

repeat
    .Solve the LP relaxation;
    .Select the customer visited in more routes;
    .Enforce integrality to all the variables associated to that
    customer;
    .Solve RMP as a static MIP;
    .Add routes for the selected variables taking value 1 to $S$
    .Remove customers from $N$
    .Resize the master problem formulation
until $N = \emptyset$;
return $S$;
```

6.3.7 Approach VII

It is clear that selecting the customer which is more divided among the routes, and imposing integrality for all the decision variables associated to that routes will enforce the model to select one and only one variable to take value 1. All the other variables will take value 0.

This can lead to a MIP solution which is significantly different of the LP solution: a possibly great number of columns may be excluded from the solution, while the value of one variable may be increased. In this last case, if the value of this variable in the LP solution is lower than 0.5, the cost of the MIP solution is increased more than the cost of such variable. In order to avoid a significant difference between the LP and MIP solutions, the last approach starts by selecting the customer assigned to less routes in the LP solution. Thus, the solution of the MIP model will iteratively selects the best route for the less divided customer by enforcing all the variables corresponding to routes visiting that customer to be integer. After solving the resulting MIP problem, among these integer variables, the one (*i.e.*, route) selected to take value one will be added to the partial solution. The model is resized and the procedure is repeated until all customers are assigned to the partial solution.

6.4 Computational experiments

6.4.1 Set of instances

In order to assess the quality of the six approaches we use a subset of instances proposed for the 2L-CVRP [76, 78, 64], which were described in detail in Section 3.2.1. In Table 6.2, we present the 90 instances used in our tests, using the following notation:

- *Instance*: original name of the instance;
- *Class*: class of the instance;
- m: number of customers;
- $\#it$: number of items of all customers;
- H: height of the loading area of the vehicle;
- W: width of the loading area of the vehicle.

The algorithms were coded in C++, and the tests were run on an Intel Xeon Processor E5-1620 v3 with 3.50 GHz and 64 GB of RAM.

6.4.2 Performance of the approaches

We conducted two set of computational experiments. In the first set, we ran all instances for all strategies with a time limit of 60 seconds for the LP relaxation. All the parameters were fixed to the values as follows: $n_{fix} = 5$, $\epsilon_3 = 0.1$, $\epsilon_4 = 0.1$, $\epsilon_5 = 0.35$. In Table 6.3 we present the results for the first set of experiments. We present the results obtained in Approach I, II, VI and VII. The strategies not covered in this table produce results that are outperformed by the three strategies II, VI and

Table 6.2: Set of instances used in computational experiments

Instance	*Class*	*m*	*#it*	*H*	*W*	*Instance*	*Class*	*m*	*#it*	*H*	*W*
E016-03m	1	15	15	40	20	E045-04f	1	44	44	40	20
	2	15	24	40	20		2	44	66	40	20
	3	15	31	40	20		3	44	87	40	20
	4	15	37	40	20		4	44	112	40	20
	5	15	45	40	20		5	44	122	40	20
E021-04m	1	20	20	40	20	E051-05e	1	50	50	40	20
	2	20	29	40	20		2	50	82	40	20
	3	20	46	40	20		3	50	103	40	20
	4	20	44	40	20		4	50	134	40	20
	5	20	49	40	20		5	50	157	40	20
E022-04g	1	21	21	40	20	E072-04f	1	71	71	40	20
	2	21	31	40	20		2	71	104	40	20
	3	21	37	40	20		3	71	151	40	20
	4	21	41	40	20		4	71	178	40	20
	5	21	57	40	20		5	71	226	40	20
E023-03g	1	22	22	40	20	E076-07s	1	75	75	40	20
	2	22	32	40	20		2	75	114	40	20
	3	22	41	40	20		3	75	164	40	20
	4	22	51	40	20		4	75	168	40	20
	5	22	55	40	20		5	75	202	40	20
E026-08m	1	25	25	40	20	E101-08e	1	100	100	40	20
	2	25	40	40	20		2	100	157	40	20
	3	25	61	40	20		3	100	212	40	20
	4	25	63	40	20		4	100	254	40	20
	5	25	91	40	20		5	100	311	40	20
E031-09h	1	30	30	40	20	E121-07c	1	120	120	40	20
	2	30	50	40	20		2	120	183	40	20
	3	30	56	40	20		3	120	242	40	20
	4	30	82	40	20		4	120	299	40	20
	5	30	101	40	20		5	120	384	40	20
E033-03n	1	32	32	40	20	E135-07f	1	134	134	40	20
	2	32	44	40	20		2	134	197	40	20
	3	32	56	40	20		3	134	262	40	20
	4	32	78	40	20		4	134	342	40	20
	5	32	102	40	20		5	134	422	40	20
E036-11h	1	35	35	40	20	E151-12b	1	150	150	40	20
	2	35	56	40	20		2	150	225	40	20
	3	35	74	40	20		3	150	298	40	20
	4	35	93	40	20		4	150	366	40	20
	5	35	114	40	20		5	150	433	40	20
E041-14h	1	40	40	40	20	E200-16b	1	199	199	40	20
	2	40	60	40	20		2	199	307	40	20
	3	40	73	40	20		3	199	402	40	20
	4	40	96	40	20		4	199	513	40	20
	5	40	127	40	20		5	199	602	40	20

VII. The results are grouped according to the number of customers of the instances. In this sense, we present the average values for each 5 instances with the same number of customers. In Table 6.3 we use the following notation:

- m: number of customers;
- sp_{LP}: average number of call of the subproblem;
- $cols_{LP}$: average number of generated columns during the LP relaxation of the RMP;
- z_{LP} cost of the LP solution;
- z_{OPT} value of the best solution achieved;
- t_{PP}: computing time for the initialization of the RMP (in seconds);
- t_{LP}: computing time for the LP relaxation (in seconds);
- t_{TOT}: total computing time (in seconds);
- *avg.*: average of each column.

The computational results presented in Table 6.3 show that the LP relaxation provides similar solutions in all approaches. This is not surprising since all the approaches have the same maximum computing time. However, these values are not equal due to the application of heuristic methods (including random procedures) applied to the subproblem. Strategies II, VI and VII tend to present best average values of cost. Among the best strategies, approach II seems to be more effective for instances containing a moderate number of customers, providing the best average results for the groups of 15 to 32, 40 and 44 customers. On the contrary, approach VI leads to the best average results for the groups of 35, 50, 71, and 100 to 199 customers. Only one group (75 customers) achieves the best average result using strategy VI.

The number of columns generated in the LP relaxation tends to be greater for the best strategies when compared to Approach I. Consequently, the number of calls to the subproblems also shows the same trend. Note that *Approach I* relies on the strict resolution of RMP as a static IP. The Approach VII tends to present a greater number of generated columns when compared to Approach II, with few exceptions. One of these exceptions is the group of 134 customers: approach VII presents a solution that

Table 6.3: Computational results for the first set of tests (approaches I, II, VI and VII)

	Approach I							*Approach II*						
m	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}
15	17	15	321,17	359	22,28	63,84	63,84	45	40	317,01	316	20,48	76,51	254,15
20	17	15	481,67	524	3,01	80,12	80,13	55	47	488,19	490	3,01	65,66	415,59
21	18	16	516,35	623	3,01	68,09	68,10	72	64	499,86	562	3,02	69,90	473,40
22	18	16	913,88	1078	3,02	69,86	69,87	76	69	903,75	940	3,02	78,29	426,97
25	17	15	682,63	746	3,01	70,48	70,48	99	91	694,17	652	3,02	68,62	489,31
30	18	16	702,15	744	3,01	72,91	72,92	105	95	687,40	608	3,01	78,73	570,50
32	18	16	4290,78	4769	3,02	72,32	72,33	101	90	4215,88	3904	3,01	70,31	631,17
35	18	16	842,75	924	3,01	63,26	63,27	121	107	833,66	806	3,02	64,30	806,60
40	18	16	949,57	1028	3,03	71,83	71,84	132	117	983,59	879	3,02	75,19	862,69
44	18	16	1816,52	2033	3,03	71,78	71,79	170	154	1859,76	1521	3,01	71,56	924,57
50	17	15	1404,45	1485	3,03	66,97	66,98	137	124	1387,97	1201	3,01	81,33	836,17
71	18	16	1164,09	1270	3,03	72,77	72,81	251	219	1211,75	1020	3,11	70,55	2039,36
75	18	16	2194,40	2363	3,10	75,98	76,01	307	274	2210,74	1951	3,08	85,10	2065,52
100	18	16	3189,54	3358	3,21	68,48	68,51	409	365	3222,93	2601	3,04	66,47	2753,74
120	18	16	7199,96	7884	3,18	75,55	75,57	804	743	7109,46	5730	3,14	71,61	3731,95
134	17	15	5866,38	6204	3,17	66,13	66,17	1195	1113	5779,80	5256	3,31	65,22	5063,94
150	18	16	4933,87	5219	3,29	73,72	73,77	1077	1003	4926,47	4065	3,17	66,54	4583,58
199	17	15	6669,47	6852	3,33	66,27	66,29	1348	1240	6737,48	5475	3,26	73,29	6727,62
avg.	18	16	2452,20	2637	4,16	70,57	70,59	361	331	2448,33	2110	4,04	72,18	1869,83

	Approach VI							*Approach VII*						
m	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}
15	37	33	313,71	353	19,82	66,64	212,54	46	42	309,88	327	22,22	77,47	245,62
20	64	56	477,22	677	3,01	80,47	492,50	56	50	488,96	644	3,01	70,27	306,26
21	71	63	517,92	595	3,01	70,90	410,85	68	62	502,30	615	3,01	69,67	305,70
22	73	65	923,39	1233	3,00	75,07	477,44	83	76	929,38	1480	3,01	71,48	399,97
25	90	81	695,73	732	3,02	72,68	499,71	109	99	680,13	674	3,01	72,69	529,61
30	73	67	686,32	713	3,01	72,71	348,39	104	96	680,78	689	3,01	72,12	494,37
32	137	124	4239,58	4503	3,02	79,33	779,66	138	126	4219,03	4326	3,01	76,34	719,06
35	118	105	831,88	865	3,01	65,48	860,26	121	110	813,34	796	3,02	67,35	657,94
40	128	116	961,57	1009	3,02	67,36	690,49	177	162	971,45	939	3,05	59,59	847,47
44	168	153	1889,62	2093	3,02	83,04	922,84	201	182	1889,66	1598	3,12	74,59	1173,14
50	142	128	1374,74	1345	3,03	77,63	860,39	214	197	1377,84	1129	3,03	71,01	1022,46
71	194	170	1180,70	1115	3,08	77,86	1449,91	368	341	1208,82	995	3,05	67,54	1595,80
75	305	275	2209,33	1844	3,10	80,26	1919,15	449	416	2210,53	2014	3,06	70,05	2010,07
100	345	302	3263,95	2812	3,21	73,26	2754,58	570	530	3265,44	2372	3,11	76,65	2437,03
120	559	497	7121,63	6584	3,18	85,91	4033,85	821	764	7153,81	5677	3,12	78,01	3510,57
134	869	787	5784,90	5945	3,20	71,89	5154,94	864	802	5791,41	4607	3,12	71,17	3878,26
150	686	612	4888,39	4312	3,24	73,95	4771,62	1122	1057	4909,62	3679	3,18	63,20	3953,57
199	1145	1031	6677,52	5754	3,49	82,60	7281,25	1561	1468	6726,44	4951	3,25	66,23	5708,50
avg.	289	259	2446,56	2360	4,03	75,39	1884,47	393	365	2451,60	2084	4,13	70,86	1655,30

outperforms the one obtained in Approach II in roughly 12,35%, while the number of generated columns is less than 300 when compared to the second approach.

As it can be seen, the Approach I presents similar computing time for each group. These value are clearly outperformed in the best strategies (more than 1800 on average for Approach II and more than 1600 seconds on average for Approach VII). Generally, the amount of computational time tends to increase with the number of customers, with few exceptions.

All groups where Approach VII finds the best average cost present average computing time lower than the ones provided by Approach II, with an exception for the group of 50 customers.

In the second set of computational experiments, we ran all the instances for all strategies with an increased time limit for the LP-relaxation (120 seconds). All the parameters referred to above are still the same. Table 6.4 presents the average results for this set of experiments

When comparing the results in Table 6.3 and Table 6.4, it can be seen that the total average computing time increases roughly 1000 seconds for approaches II, VI and VII. Approach II leads to the best average results. Indeed, this approach was able to find the best solution in the groups of 15 to 120 customers. Approach VII was only able to outperform the solutions of Approach II for three groups (134, 150 and 199 customers). When comparing these two approaches, the best solution for each group is also the one that is obtained in less computing time, with few exceptions. When compared with Approach I, Approach II achieves an average improvement of 25,7% in terms of costs, while Approach VI achieves an average improvement of 23,38%. Approach VI does not lead to any best solution among all groups. However, this approach yields better results than Approach VII for 3 groups (22, 25 and 50). Among all strategies, Approach VII leads often to longer computing time.

6.5 Conclusions

In this chapter, we presented a set of column generation heuristics. All these heuristics rely on decisions that are taken considering the set partitioning problem formulation

Table 6.4: Computational results for the second set of tests (approaches I, II, VI and VII)

	Approach I							*Approach II*						
m	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}
15	31	29	294,38	314	22,22	130,34	130,35	59	55	290,29	286	22,82	112,90	392,67
20	32	30	435,28	476	3,01	127,35	127,36	73	67	433,37	456	3,02	103,36	589,71
21	33	31	456,35	523	3,01	126,16	126,18	90	83	452,56	493	3,01	79,92	652,22
22	33	31	831,57	958	3,01	129,14	129,17	110	104	822,15	851	3,00	109,31	584,31
25	34	32	596,46	665	3,02	126,14	126,16	123	115	584,13	573	3,01	99,11	826,78
30	34	32	606,14	675	3,02	129,85	129,87	141	134	605,98	569	3,02	112,37	813,75
32	33	31	3584,84	4171	3,01	119,54	119,57	129	120	3526,05	3682	3,03	99,77	1007,97
35	33	31	744,57	832	3,02	123,83	123,85	144	134	751,41	718	3,02	99,19	1125,48
40	34	32	859,25	978	3,02	123,91	123,95	177	166	811,29	783	3,04	84,96	1118,23
44	34	32	1587,42	1850	3,02	131,85	131,90	196	183	1586,20	1431	3,02	97,43	1408,80
50	33	31	1215,57	1319	3,01	132,42	132,46	220	206	1232,22	1044	3,03	98,70	1635,28
71	34	32	1022,43	1172	3,03	129,90	129,95	271	249	1021,00	810	3,04	96,23	2162,89
75	34	32	1940,19	2166	3,09	128,73	128,78	291	270	1925,00	1516	3,06	101,41	2204,67
100	34	32	2884,20	3089	3,20	123,33	123,38	379	351	2870,73	2043	3,16	79,61	2914,82
120	34	32	6122,04	7027	3,11	127,96	128,02	736	691	6114,27	4331	3,31	109,44	5288,78
134	34	32	5071,79	5776	3,34	125,47	125,54	1487	1421	4974,82	4357	3,30	129,42	8090,24
150	34	32	4494,79	4795	3,19	125,03	125,09	972	915	4506,92	3377	3,15	121,22	6904,29
199	34	32	6224,09	6557	3,41	129,01	129,07	2039	1943	6211,37	4890	3,18	130,52	11650,70
avg.	33	31	2165,08	2408	4,15	127,22	127,26	424	400	2151,10	1789	4,18	103,60	2742,87

	Approach VI							*Approach VII*						
m	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}	sp_{LP}	$cols_{LP}$	z_{LP}	z_{OPT}	t_{PP}	t_{LP}	t_{TOT}
15	61	57	292,84	355	25,22	117,69	374,65	61	57	288,82	307	21,02	122,49	422,07
20	84	79	433,62	652	3,01	132,13	596,91	81	75	439,71	517	3,01	117,15	623,93
21	97	92	461,98	643	3,01	129,19	537,05	92	87	456,62	612	3,01	132,76	504,53
22	100	94	823,73	1051	3,01	122,52	640,75	113	106	827,87	1213	3,01	100,91	648,59
25	112	104	579,84	608	3,01	96,14	791,43	112	104	587,68	642	3,00	100,89	743,92
30	171	161	602,50	707	3,01	137,58	1112,17	154	145	593,96	614	3,01	114,80	953,65
32	167	158	3554,57	4551	3,01	129,78	1015,78	167	157	3563,73	3825	3,01	135,23	1116,67
35	156	144	732,32	859	3,01	94,99	1198,15	183	172	748,45	797	3,02	108,18	1312,84
40	168	157	838,43	1061	3,02	81,81	1072,28	222	211	834,63	853	3,05	111,98	1289,51
44	181	167	1578,11	1767	3,11	100,54	1351,72	234	218	1553,89	1633	3,04	106,53	1702,56
50	268	251	1196,61	1212	3,02	120,94	2018,79	295	279	1233,31	1230	3,01	116,14	1915,24
71	361	336	1022,41	993	3,05	108,87	2602,38	485	463	997,48	931	3,10	128,30	2672,91
75	418	391	1976,58	1988	3,04	87,43	2842,66	550	525	1961,94	1679	3,04	87,94	2934,80
100	565	527	2851,18	2629	3,10	128,08	4175,26	693	663	2918,53	2192	3,06	119,20	3628,36
120	818	760	6109,69	6140	3,21	67,32	6238,98	1003	956	6146,09	4703	3,08	133,71	5688,23
134	1027	960	5060,90	5144	3,02	100,56	7918,90	1206	1153	5088,19	4079	3,15	128,20	6445,16
150	1178	1117	4535,65	4025	3,13	104,90	7097,51	1577	1524	4508,18	3167	3,17	127,20	6413,79
199	1623	1532	6122,94	5113	3,30	141,70	11189,22	2152	2080	6172,22	4217	3,64	125,72	8781,53
avg.	420	394	2154,11	2194	4,29	111,23	2931,92	521	499	2162,29	1845	4,08	117,63	2655,46

of the master problem. The basic idea in each one of them was to build a feasible solution from the one obtained from the LP-relaxation. This was performed by fixing some fractional variables to become part of the solution, or by solving a MIP model with some of these variables enforced to be integer. Iteratively, the model is resized and it becomes smaller, which can make it less difficult to solve.

Chapter 7

A metaheuristic algorithm for pickup and delivery problems with loading constraints

Contents

7.1 Introduction

In this chapter, a capacitated vehicle routing problem with loading constraints and mixed linehauls and backhauls is presented. The addressed problem belongs to the subclass of pickup and delivery problems. Two-dimensional loading constraints are also considered.

These constraints arise in many real-world situations, and can improve efficiency since backhaul customers do not need to be delayed in a route when it is possible to load their items earlier and without rearrangements of the items.

To tackle this problem, we present a computational study on variants of the variable neighborhood search. The initial solution relies on an insertion heuristic. Both the shaking and local search phases resort to ten neighborhood structures. Some of those structures were specially developed for this problem. The validation of routes is heuristically obtained with a classical bottom-left method enhanced to tackle the explicit consideration of loading constraints.

All the strategies that are presented were tested. An exhaustive computational study is performed on instances adapted from benchmark instances of integrated routing and loading problems.

This chapter is organized as follows. In Section 7.2, we describe the Capacitated Vehicle Routing Problem with 2-dimensional Loading constraints and Mixed linehauls and Backhauls (2L-CVRPMB). In Section 7.3 a state of the art is provided. In Section 7.8.1, we represent the solution while in Section 7.6 the method to find an initial solution is presented. The neighborhood structures and the variants of the variable neighborhood search algorithm are presented in Section 7.7 and in Section 7.8, respectively. In Section 7.9 the computational results of this approach are presented and discussed. Finally, some conclusions are drawn in Section 7.10.

7.2 Problem definition

The Capacitated Vehicle Routing Problem with 2-dimensional Loading constraints and Mixed Linehauls and Backhauls (2L-CVRPMB) can be described as follows. Let $G = (V, E)$ be a complete graph, where $V = \{0, 1, \ldots, l, l+1, l+2, l+b\}$ is the set of nodes and E is the set of edges. The set V includes $n+1$ vertices corresponding to the depot (vertex 0) and to the n customers including l linehaul customers and b backhaul customers. The set E includes all pair of vertices such that $E = \{(i,j) : i,j \in V, i \neq j\}$. Let c_{ij} be the non-negative cost of traversing the edge (i,j). There is a set of K identical vehicles, each one having a weight capacity C and a two-dimensional rectangular loading area with height H and width W. Each linehaul customer i (or, respectively, each backhaul customer i) has a demand (or, respectively, a supply) composed by m_i two-dimensional rectangular items. Each item j ($j = 1, 2, \dots, m_i$) of customer i ($i = 1, 2, \dots, l+1, l+2, l+b$) is denoted by $I_{i,j}$.

The 2L-CVRPMB consists in finding the set of optimal routes, each one starting and ending at the depot and satisfying the following constraints:

(C1) the number of used vehicles cannot be greater than K;

(C2) each customer is visited exactly once;

(C3) the total weight associated to each arc of each route cannot exceeded the weight capacity of the vehicle;

(C4) items have a fixed orientation in the loading area, *i.e.*, items cannot be rotated;

(C5) in each vehicle, the items must be completely within the surface, must not overlap, and the edges of each items must be parallel to the loading area edges;

(C6) unloading an item at a given customer must be performed in a straight movement, without moving items of other customers, and the loaded items from backhaul customers cannot block items to be delivered nor be moved after being positioned in the loading area (sequential constraints).

An example of the 2L-CVRPMB is presented in Figure 7.1. Only one route is considered.

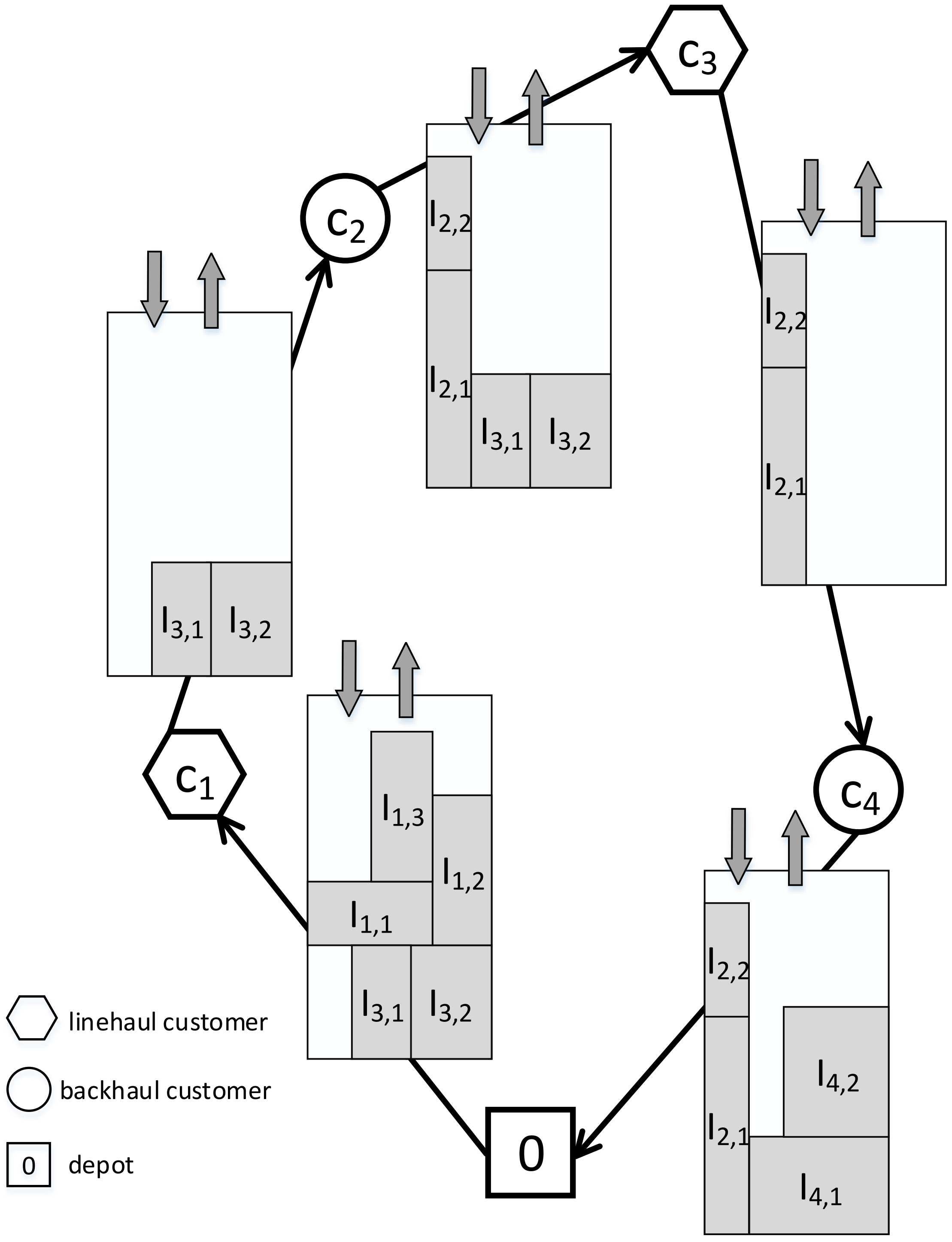

Figure 7.1: An instance of the 2L-CVRPMB

7.3 State of the art

7.3.1 The vehicle routing problem with mixed linehauls and backhauls

The Vehicle Routing Problem with Mixed Linehauls and Backhauls (VRPMB) is a subclass of problems belonging to the Vehicle Routing Problem with Backhauls (VRPB), in which the customers are divided in two different groups: linehaul and backhaul customers. Each linehaul customer has a given demand that can only be provided by the depot, while each backhaul customer provides a given supply whose only destination is the depot. In the VRPMB, both the linehaul and backhaul customers can be indistinctly visited, which means that within the same route, it is not required to visit all linehaul customers before visiting backhaul customers.

In [65], the Clarke and Wright method is applied to assign all linehaul customers to the routes. Then, an insertion-based heuristic evaluates the insertion of each backhaul customer in all route points where that customer can be feasibly inserted. This insertion cost is obtained by the additional routing cost when inserting that customer in a given arc, and by adding the product of a penalty multiplier with the number of delivers after that insertion. The minimum cost defines the backhaul customer and its insertion point. The process is repeated until all backhaul customers are visited.

The insertion cost was revised in [27] by subtracting a value which is proportional to the distance between the backhaul customer to be inserted and the depot, and by adding the product of a penalty multiplier with the quantity still to be delivered after the insertion point.

Based on this insertion cost suggested in [27], a more concise formula is proposed in [135]. The multiplier associated to the remaining load that has to be delivered is calculated in distance units. The authors proposed four heuristics procedures. Each heuristic starts from a solution where all linehaul customers are already assigned to the routes. The first heuristic is based on [65] and in [27] by computing the insertion cost of each customer/arc and inserting the customer with the least cost. The second heuristic computes both the insertion cost for each customer/arc as in the first heuristic and the insertion cost of each pair of backhaul customers. The least cost defines the insertion of one or two customers. This procedure allows small improvements when two backhaul customers are near each other. Since this situation can happen with more than two customers, the authors suggested two more heuristics in which clusters of backhaul customers can be inserted. In these heuristics, the insertion costs for each customer, for each pair of customers, and for each cluster are computed in each iteration, and the best cost defines the customer (or group of customers) and its position in a route. The analysis of the insertion costs will allow to have the perception, at each iteration, if each customer with a smaller insertion cost is inserted alone or associated with another customer. In order to define the clusters, the authors defined a graph of all backhaul customers, where all customers that are better inserted in pairs are linked by an edge. The difference between the two last heuristics is the strategy to build clusters. For the third heuristic, only customers which form a complete graph can be grouped in a cluster, while in the fourth heuristic, it is sufficient for customers to form a connected graph to be grouped in a cluster.

An improved insertion heuristic is presented in [146]. It is also possible to insert one or two customers at each iteration, depending on the least insertion cost. This approach is improved with a so-called restriction percentage. The restriction percentage is defined by the user and it is based on his/her experience to define how backhaul and linehaul customers can appear mixed in a route. This percentage value is used when evaluating the insertion of a backhaul customer in a given arc. The percentage of linehaul goods already delivered must be greater than or equal to the restriction percentage, in order to enable the insertion of a backhaul customer in that arc. Otherwise, the insertion on that arc is not be evaluated again.

A heuristic for the VRPMB and for the Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) is suggested in [118]. As referred to in Section 2.2, the VRPSDP is a pickup and delivery problem in which customers require both a quantity to be delivered and a quantity to be picked up, and in which customers must be served in a single visit. The heuristic starts by obtaining a so-called weakly feasible solution. One solution is weakly feasible if it satisfies the maximum route length, and if the capacity of the vehicle is not exceeded by the total load to be delivered or by the total load to be picked up. In contrast, a strongly feasible solution satisfies the maximum route length and, additionally, the vehicle capacity is not exceeded in each arc of the route. Some improvements are then performed by applying local search on the initial solution, and turning the solution in a strongly feasible solution. Additional local search procedures are applied, improving the quality of the solution while keeping it as strongly feasible.

7.3.2 Pickup and delivery with loading

The interest of Pickup and Delivery Problems (PDP) in which loading constraints are considered has been increasing. Some examples include the Pickup and Delivery Travelling Salesman Problem with loading constraints and the Double Travelling Salesman with Multiple Stacks, as defined in Section 3.5.5.

Other approaches addressed PDP considering more than a single dimension of the goods, generalizing the capacitated vehicle routing problem with loading constraints defined in Section 2.3. A constraint programming model based on the scheduling field for the two-dimensional loading with sequential constraints is presented in [102].

The PDP with three-dimensional loading constraints was addressed in [10]. The approach is based on a local search method defined by swapping the vertices if that leads to a lower route length. After deriving a valid packing solution, another local search procedure is performed by assigning customers to other routes.

It is important to note that for both approaches referred to above, the transportation of goods is performed between the customers. In contrast, transportation from and to the depot is considered in [22]. These authors addressed in particular the 3L-CVRP with clustered backhauls. As referred to before, backhaul customers should be visited after the linehaul customers within the same route. Sequential constraints are taken into account, and these constraints should be satisfied also when visiting backhaul customers. In this work, two procedures were suggested and, in both of them, routing and packing were solved by different procedures, and then integrated to compose a solution. The routing component is solved by two algorithms. The first algorithm is based on the adaptive large neighborhood search, while the second relies on a variable neighbourhood search. The packing component of both problems is tackled through a tree search heuristic presented in [20].

7.4 Solution representation

In the context of the 2L-CVRPMB, a feasible solution S with value $z(S)$ can be represented by a sequence of routes, as follows:

$$S = (S_1, S_2, \dots, S_{|S|}),$$

with $S_i = (c_1, c_2, c_3, \dots, c_{|S_i|-1}, c_{|S_i|})$ corresponding to the order in which customers $c_2, c_3, \dots, c_{|S_i|-1}$ are visited in route S_i, with $c_1 = c_{|S_i|} = 0$, since each route starts and ends at the depot. Additionally, P_i defines the order by which the items of customers of S_i are placed in the vehicle using the heuristic proposed in Section 7.5, *i.e.*,

$$P_i = (p_2, \dots, p_{|S_i|-1}),$$

with $p_j = (p_j^1, p_j^2, \dots, p_j^{m_{c_j}})$, $j = 2, \dots, |S_i| - 1$, and p_j^k representing the index of the k^{th} item of customer c_j to be placed in the vehicle. The relative position of each item in the vehicle is obtained by the strict application of the mentioned heuristic to the order P_i. We will

7.5 Feasibility of routes

The capacity constraints in terms of weight are ensured not only by imposing a weight limit to the weight of items to deliver, but also by limiting the load to the weight capacity of the vehicles in each arc of the route. The same idea applies to the total area of loaded items. On the other hand, the feasibility of each route in terms of packing and sequential constraints is required not only to build an initial solution, but also in each iteration of the variable neighborhood search algorithm.

In order to assess the feasibility of one route in the context of 2L-CVRPMB, we adapted the classical bottom-left heuristic by incorporating procedures to deal with sequential constraints for different types of customers.

Some authors addressing loading constraints in problems with clustered backhauls [22] impose LIFO constraints for both linehaul customers and backhaul customers. Imposing LIFO constraints for backhaul customers comes from the assumption that an item is loaded in a given position of the loading area, and hence it cannot be rearranged. Note that one layout with items to be delivered to linehaul customer can be seen as the layout of the vehicle when it leaves the depot. Similarly, the layout of the vehicle when it returns to the depot only has items provided by backhaul customers.

In contrast to the case of clustered backhauls, the feasibility of one route with mixed customers results from ensuring a feasible layout for linehaul customers and also from ensuring that every collected items of backhaul customers does not block items to be delivered. As a consequence, the collected items cannot be placed between the rear-side of the vehicle and the items to deliver.

In order to manage the relative position of the items in the loading area, we resort to the concept of free spaces, as presented in Chapter 4. The loading area is considered to be the first free space. When an item is placed in the loading area, at most four free spaces are created. These free spaces identify the places where the next items can be placed. The original free space where the item was placed is removed from the list of free spaces. Taking into account all the free spaces, we select the bottom-most and leftmost position where the item fits, such that the capacity and sequencing constraints are satisfied. The item is placed in such a way that its bottom-left corner matches the bottom-left corner of the free space.

The modified bottom-left heuristic starts by attempting to derive valid packing layouts only for the linehaul customers. If it is not possible to achieve a valid packing for the linehaul customers of the route, then clearly the original route is infeasible.

If a feasible layout for linehaul customers is achieved, the obtained layout will be analyzed seeking one or more free spaces with height H, since it means that one item from a backhaul customer can be placed without blocking any item for a linehaul one. This procedure allows to know if the vehicle has a free space available for backhaul customers right after leaving the depot. If one or more free spaces with this height are found, they are added to a list of free spaces devoted to collected items from backhaul customers.

Thereafter, the customers of the route are successively analyzed by the order in which they are visited. If the analyzed customer is a linehaul customer, then their items will be removed from the layout and the original free spaces of that items are recovered. Otherwise, if it is a backhaul customer, then we try to place their items in a free space for collected items, ensuring that all the constraints are obeyed. The process runs until all backhaul items are loaded on the vehicle, or until it is impossible to place a given item. In this last case the route is infeasible.

7.6 Initial solution

In this section, we describe how to build an initial solution heuristic for the 2L-CVRPMB. This approach is based on the insertion heuristic presented in [135].

First, a set of routes is built only for linehaul customers. For this purpose, we resort to the method using feasible solutions of the 2L-CVRP suggested in [154]. In this process, an empty route is generated for each vehicle. The set of linehaul customers is sorted according to the area of their demands. Each customer is successively inserted in one route by this order.

From the set of routes in which the customer can be inserted in a feasible way, the one which leads to the minimum remaining free area after that insertion is selected. The feasibility of this insertion is evaluated by the heuristic presented in Section 7.5. The route point, in which the insertion is valid, and it minimizes the insertion cost, is selected to insert the customer.

After deriving routes for all linehaul customers, the backhaul customers are then inserted using insertion heuristics based on [135]. However, the cost expression suggested in this work was modified. In [135], the cost of inserting a given backhaul customer in a given arc is based on the additional routing cost, on the distance between the customer and the depot, and on the remaining quantity (in terms of weight) to deliver. This last part is multiplied by the total length of the route which contains the arc. Let L_b be the quantity to deliver after customer b, T_{ab} be the route length of route which contains arc (a, b), and MQ be the capacity of the vehicle. The cost of inserting a backhaul customer c in the arc (a, b) is expressed as C_{abc}, and is given by:

$$C_{abc} = d_{ac} + d_{cb} - d_{ab} - \alpha d_{0c} + \beta L_b \left(\frac{T_{ab}}{MQ} \right), \tag{7.1}$$

where d_{ij} corresponds to the distance between node i and j. The parameters α and β are penalty parameters. In [135], the default values for the penalty parameters α and β are 1.5 and 1, respectively. The role of the penalty parameter β is to delay the insertion of backhaul customers in the context of the VRPMB [27, 135, 146]. This procedure is to prevent situations where the vehicles are full or a rearrangement of items may be needed. In our approach, for the 2L-CVRPMB, the loading constraints are incorporated and therefore such rearrangements are prevented. In this sense, the last part of the cost expression is removed, *i.e.*,

$$C_{abc} = d_{ac} + d_{cb} - d_{ab} - \alpha d_{0c}. \tag{7.2}$$

In [135], the authors extended the single insertion to the insertion of a pair of customers. Additionally, and as suggested in their work, we compute the cost of inserting two backhaul customers. Again, no penalties are associated to delay the insertion. Therefore the insertion cost for a pair of backhaul customers c and d in each arc (a, b) is computed, using the expression

$$C_{abcd} = \min\left(d_{ad} + d_{cb}; d_{ac} + d_{db}\right) - d_{ab} + d_{cd} - \frac{\alpha\left(d_{0c} + d_{0d}\right)}{\sqrt{2}}. \tag{7.3}$$

From the list of all backhaul customers, and using the cost expression (7.2), we find the customer and the arc leading to the best insertion cost. Similarly, using cost expression (7.3) we find the best cost for the insertion of a pair of customers and the corresponding arc. Between these two costs, the minimum cost defines both the insertion point and, if a single customer or a pair of customers is inserted. The process is repeated until all backhaul customers are assigned to the routes. One special case arises when there is no feasible position in any route for a given customer. In such case, the customer is assigned to a new route.

7.7 Neighborhood structures

In this section, we present the neighborhood structures that will be used by the VNS algorithms to explore the search space. We defined 10 neighborhood structures: nine of them are only based on routing structures while one can be seen as a packing and routing neighborhood structure.

As referred to in Section 7.8.1, the representation of the position of items in one layout relies on the order of placement using a packing heuristic. In this sense, a packing neighborhood structure must rely on this order.

Note that movements based exclusively on the order of placement have no impact on the costs of the 2L-CVRPMB, since the objective function relies only on routing costs. All the movements lead to infeasible or feasible solutions sharing the same objective function value since the sequence of visit remains unchanged. Consequently, we suggest a packing and routing neighborhood in which, besides swapping the position of two items in the order of placement within the same customer, another customer of the same type is inserted.

Routing neighborhoods

NS_1 **Swapping two linehaul customers within the route**

The neighbors of a solution S in the neighborhood $NS_1(S)$ consist in all the feasible solutions obtained by swapping the position of two linehaul within the same route, while keeping the sequence of items of each customer in S. Figure 7.2 presents a movement within NS_1, defined by swapping customer c_2 and c_4.

NS_2 **Swapping two backhaul customers within the route**

The neighbors of a solution S in the neighborhood $NS_2(S)$ consist in all the feasible solutions obtained by swapping the position of two backhaul customers within the same route, while keeping the sequence of items of each customer in S. Figure 7.2 presents a movement within NS_2, by swapping customer c_3 and c_5.

NS_3 **Shifting one customer within the route**

The neighbors of a solution S in the neighborhood $NS_3(S)$ consist in all the feasible solutions obtained by shifting the position of one customer of any type within the same route, while keeping the sequence of items of each customer in S. Figure 7.2 presents a movement within NS_3, by shifting customer c_4 to the first position of the route.

NS_4 **Shifting one customer for another route**

The neighbors of a solution S in the neighborhood $NS_4(S)$ consist in all the feasi-

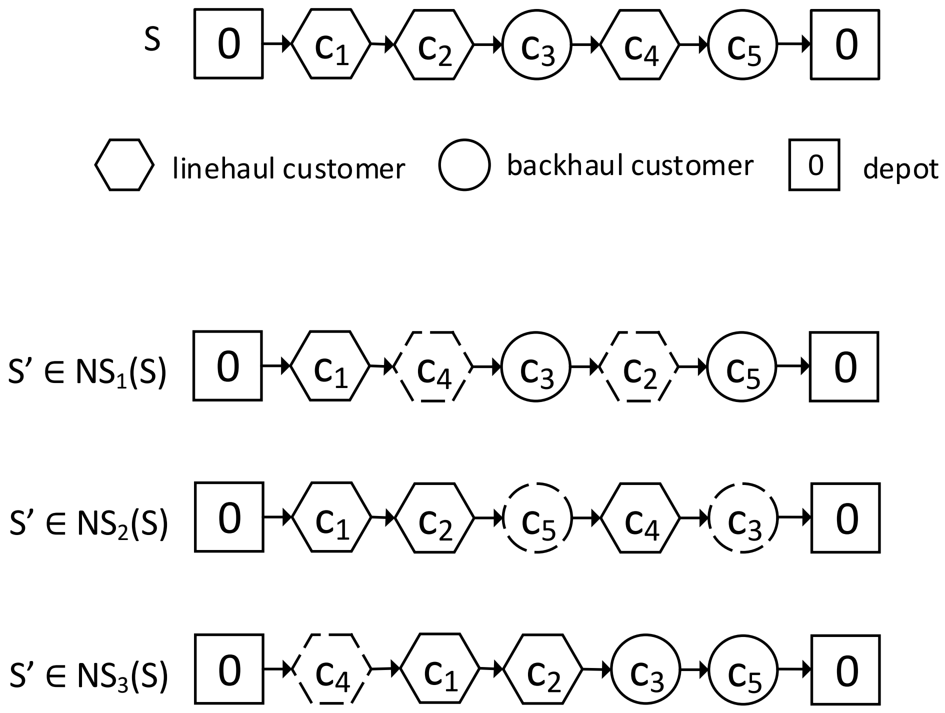

Figure 7.2: Example of movements in neighborhood structures NS_1, NS_2 and NS_3

ble solutions obtained by removing one customer from its position and inserting it in one position of another route. The sequence of items of each customer is not modified. Figure 7.3 presents a movement within NS_4, by shifting customer c_3 from route S_1 to the first position of route S_2.

NS_5 **Swapping two consecutive customers of different type**

The neighbors of a solution S in the neighborhood $NS_5(S)$ consist in all the feasible solutions obtained by swapping a sequence of two consecutive customers of one route (being the first a backhaul customer and the second a linehaul customer), with a sequence of two consecutive customers of any type of another route. The sequence of items of each customer is not modified. Figure 7.3 presents a movement within NS_5, defined by swapping customers c_3 and c_4 from route S_1 with customers c_8 and c_9 from route S_2.

NS_6 **Swapping three consecutive customers of different type**

The neighbors of a solution S in the neighborhood $NS_6(S)$ consist in all the feasible solutions obtained by swapping a sequence of three consecutive customers

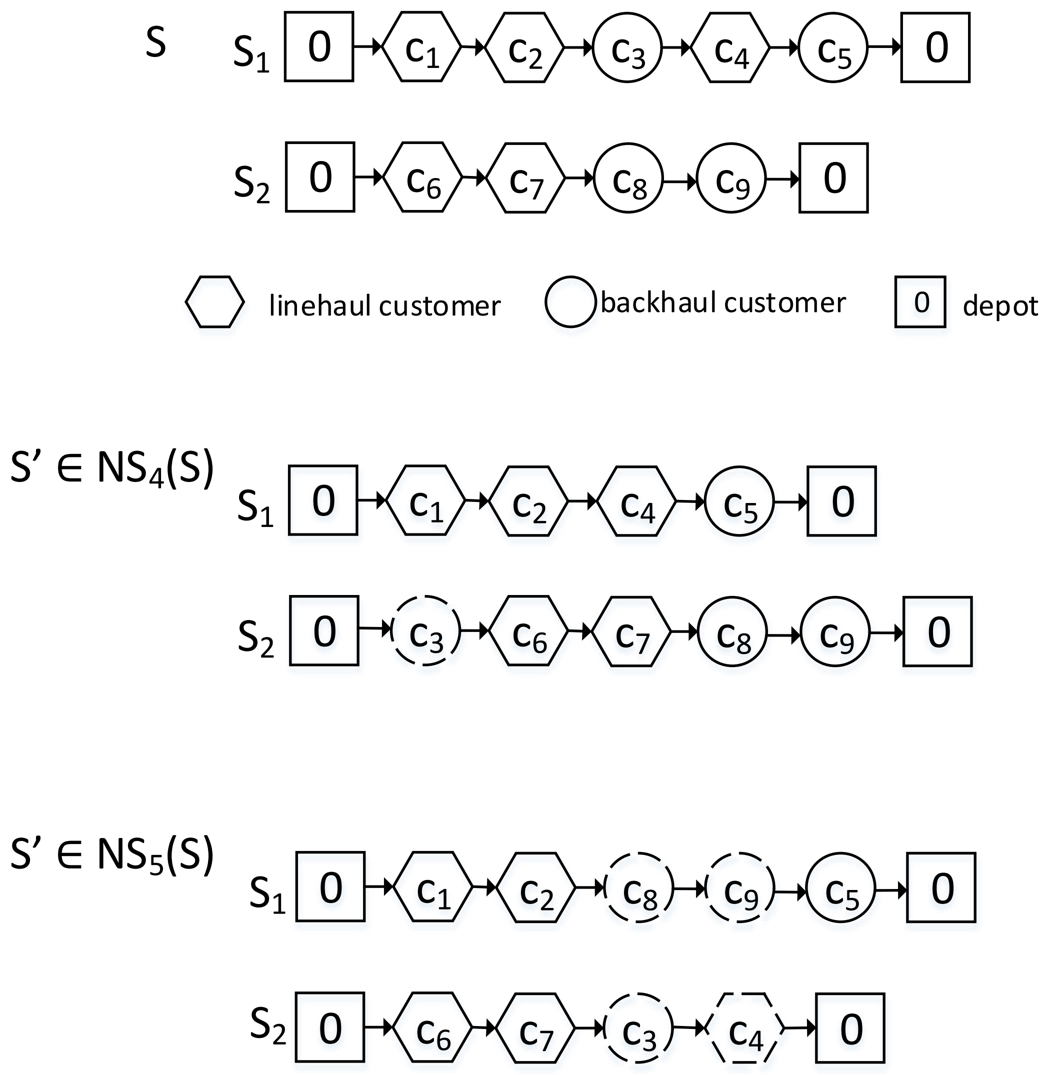

Figure 7.3: Example of movements in neighborhood structures NS_4 and NS_5

of one route (being the first a backhaul customer and the third a linehaul customer), with a sequence of three consecutive customers of any type of another route. The sequence of items of each customer is not modified. Figure 7.4 presents a movement within NS_6, by swapping customers c_3, c_4 and c_5 from route S_1 with customers c_7, c_8 and c_9 from route S_2.

NS_7 **Swapping two consecutive customers of any type**

The neighbors of a solution S in the neighborhood $NS_7(S)$ consist in all the feasible solutions obtained by swapping a sequence of two consecutive customers of one route with a sequence of two consecutive customers of another route. The sequences may be composed of customers of any type. The sequence of items of each customer is not modified. Figure 7.4 presents a movement within NS_7, obtained by swapping customers c_4 and c_5 from route S_1 with customers c_6 and c_7 from route S_2.

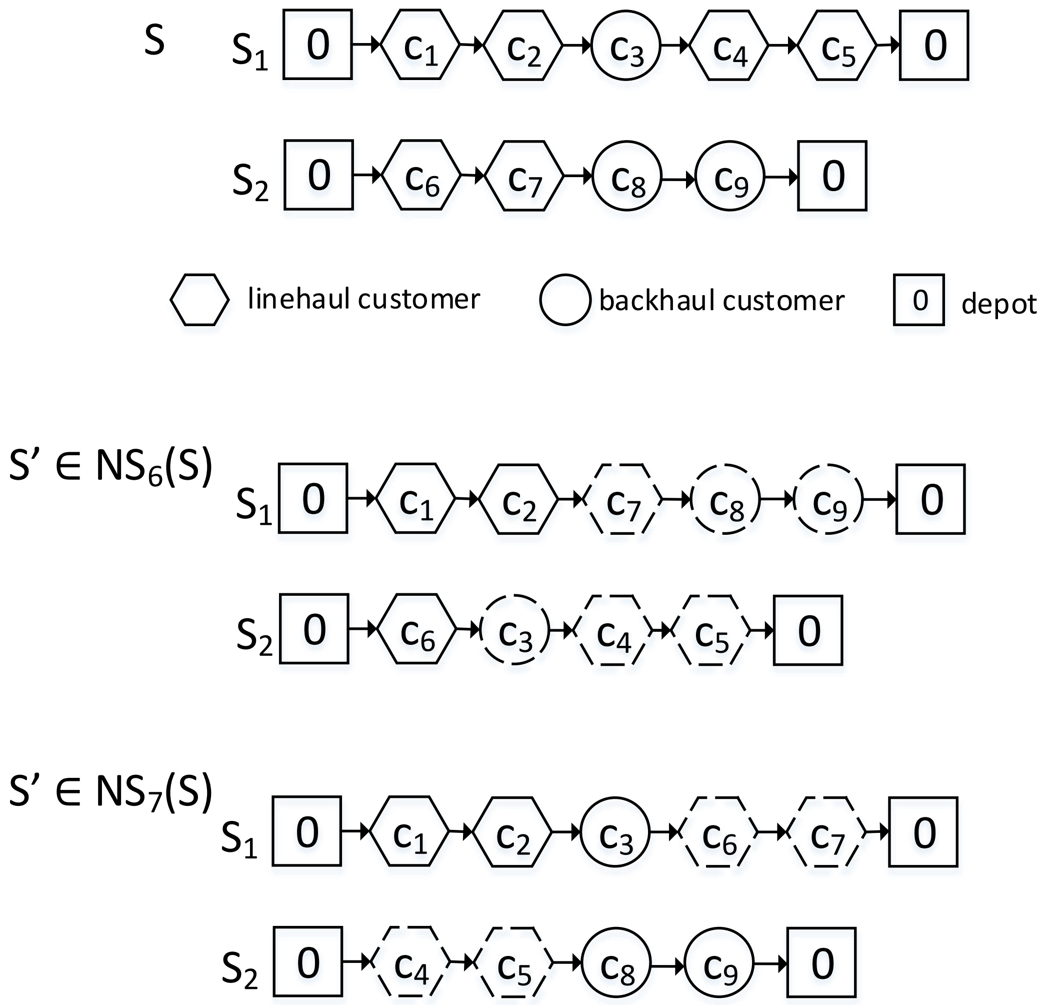
S
S_1 0 c_1 c_2 c_3 c_4 c_5 0
S_2 0 c_6 c_7 c_8 c_9 0
linehaul customer
backhaul customer
0 depot
S' ∈ NS_6(S)
S_1 0 c_1 c_2 c_7 c_8 c_9 0
S_2 0 c_6 c_3 c_4 c_5 0
S' ∈ NS_7(S)
S_1 0 c_1 c_2 c_3 c_6 c_7 0
S_2 0 c_4 c_5 c_8 c_9 0

Figure 7.4: Example of movements in neighborhood structures NS_6 and NS_7

NS_8 **Shifting two consecutive customers**

The neighbors of a solution S in the neighborhood $NS_8(S)$ consist in all the feasible solutions obtained by removing a sequence of two consecutive customers from its position and inserting it in a different position within the same route, while keeping the sequence of items of each customer in S. Figure 7.5 presents a movement within NS_8, obtained by shifting customer c_1 and c_2 to the third position and fourth position of the route S.

NS_9 **Shifting three consecutive customers**

The neighbors of a solution S in the neighborhood $NS_9(S)$ consist in all the feasible solutions obtained by removing a sequence of three consecutive customers from its position and inserting it in a different position within the same route, while keeping the sequence of items of each customer in S. Figure 7.5 presents a movement within NS_9, by shifting customer c_2, c_3 and c_4 to the third, fourth and fifth position of the route.

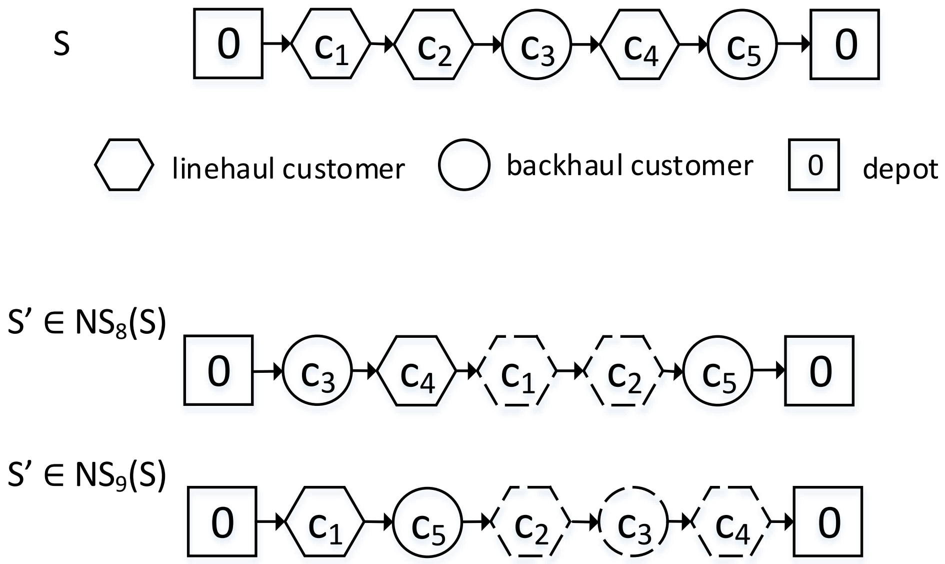

Figure 7.5: Example of movements in neighborhood structures NS_8 and NS_9

Packing and routing neighborhood

NS_{10} **Swapping two items and inserting one customer**

The neighbors of a solution S in the neighborhood $NS_{10}(S)$ consist in all the feasible solutions obtained by swapping the order of insertion of items of one customer in one route, and by inserting another customer of the same type in the same route. If the type of both customers is linehaul, then the customer is inserted at the beginning of the route, *i.e.*, between the depot and the first visited customer. Otherwise, the customer is inserted at the end of the route. Figure 7.6 presents a movement within NS_{10}, by swapping the insertion position of item $I_{3,1}$ with $I_{3,2}$ of customer c_3 and by shifting customer c_7 from route S_2 to the last position of route S_1.

7.8 Variable neighborhood search approaches for the 2L-CVRPMB

7.8.1 Variable neighborhood Search

The Variable Neighborhood Search (VNS) is a meta-heuristic initially suggested in [112], which consists in a systematic exploration of neighborhood structures. Generally speaking, the VNS can be divided in two main phases: the shaking phase and the local search phase.

In order to improve the solutions generated by the insertion heuristic (Section 7.6), in this section we propose a VNS algorithm for the 2L-CVRPMB. From the initial solution (S, P), as defined in Section , the VNS drives the search using the ten neighborhoods structures (NS_k with $k = 1, 2, \ldots, 10$) presented in Section 7.7. These structures are sequentially explored by the order they were presented. At each iteration, a neighbor solution (S', P') is randomly generated from the k^{th} neighborhood of (S, P). A local search method is applied to (S', P') in the same neighborhood structure, obtaining (S'', P''). We adopt a first improvement strategy (also known as first descent heuristic), *i.e.*, when the value of (S'', P'') is better than the value of (S, P), then (S, P) is updated and the search restarts from the first neighborhood $(k = 1)$. Otherwise, the process is iterated using the next neighborhood $(k = k + 1)$.

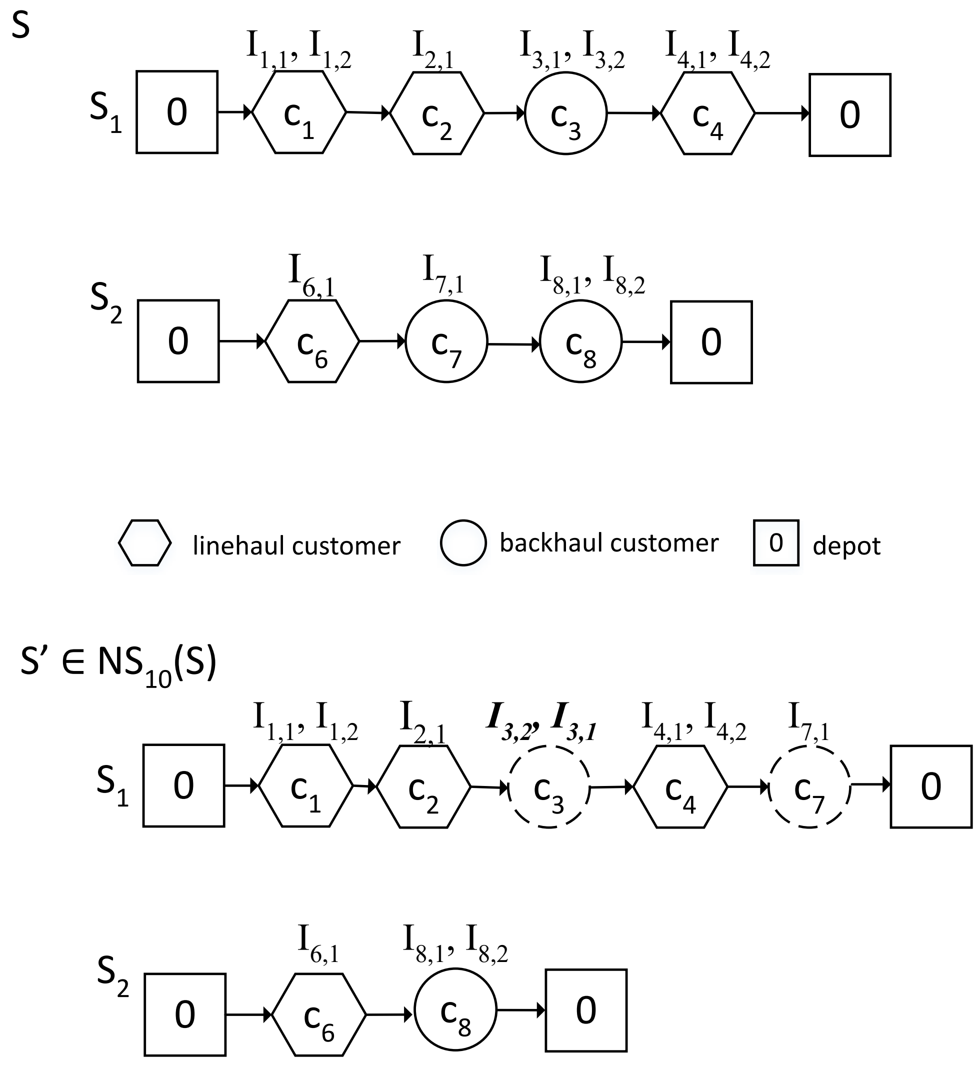

Figure 7.6: Example of a movement in neighborhood structure NS_{10}

The described process is repeated until a stopping condition is met. In this approach we impose a time limit t_{max}. It is worth noting that every neighbor solutions must be a feasible solution for the 2L-CVRPMB, and consequently we validate all the obtained solutions with the heuristic presented in Section 7.5.

The VNS algorithm for the 2L-CVRPMB is summarized in Algorithm 6. The insertion heuristic (Section 7.6) is denoted by `findInitialSolution()`. The shaking phase is represented by `shaking`$((S, P), NS_k)$, while the local search phase is denoted by `firstImprovement`$((S', P'), NS_k)$.

Algorithm 6: VNS algortihm for the 2L-CVRPMB

```
Input:
    $I$: instance of the 2L-CVRPMB;
    Set of neighborhood structures $NS = \{NS_1, NS_2, \ldots, NS_{10}\}$;
    Limit $t_{max}$ on the total computing time;
Output:
    Feasible solution $(S, P)$ for the 2L-CVRPMB of value $z(S)$;
$(S, P)$:=findInitialSolution();
repeat
    $k := 1$;
    while $k \leq 10$ do
        $(S', P')$ := shaking($(S, P), NS_k$);
        $(S'', P'')$ := firstImprovement($(S', P'), NS_k$);
        if $z(S'') \leq z(S)$ then
            $(S, P) := (S'', P'')$;
            $k := 1$;
        end
        else
            k:=k+1;
        end
    end
until cpuTime() $> t_{max}$;
return $(S, P)$ ;
```

7.8.2 General VNS

Additionally, we explored an approach based on a General Variable Neighborhood Search algorithm (GVNS) for the 2L-CVRPMB. In contrast to the VNS (Section 7.8.1) that uses a first improvement heuristic, the local search phase of the GVNS is performed by a Variable Neighborhood Descent (VND) method. The successful application of this algorithm in many works [73] motivated its use in the context of the 2L-CVRPMB.

The VND method consists in the sequential exploration of neighborhoods, selecting in each one the direction of steepest descent within the neighborhood structure [71]. In other words, the VND explores each neighborhood l ($l = 1, 2, \ldots, 10$) finding the best neighbor (S'', P'') within such structure. If the obtained solution (S'', P'') is better than the solution provided by the shaking phase (S', P'), then this solution is updated and the VND restarts from the first neighborhood structure ($l = 1$). Otherwise, the process is iterated using the next neighborhood ($l = l + 1$).

Apart from the local search phase, the rest of the algorithm relies on the algorithm described in Section 7.8.1. As happens with the VNS, only feasible solutions are explored. The GVNS algorithm for the 2L-CVRPMB is summarized in Algorithm 7. The insertion heuristic (Section 7.6) is denoted by `findInitialSolution()`. The shaking phase is represented by `shaking`$((S, P), NS_k)$ while the process to find the best neighbor is denoted by `bestImprovement`$((S', P'), NS_k)$.

7.8.3 Skewed general VNS

Hansen et al. [70] suggested an extension of the basic VNS that explores regions that are quite far from the incumbent solution. For that purpose, the incumbent solution can be updated with slightly worse solutions if they are relatively different from the incumbent, while keeping the best solution obtained so far.

In this section, we describe a Skewed General Variable Neighborhood Search (SGVNS) for the 2L-CVRPMB.

In order to evaluate the difference between two solutions, we propose a distance function $\rho(S, S'')$ that expresses the difference between the incumbent solution S and the local optimum S''. Since the 2L-CVRPMB considers two type of customers, we define a difference function relying on the variation of customers of the same type within one route. More precisely, if $\sigma(S_i)$ and $\gamma(S_i)$ corresponds, respectively, to the number of linehaul and backhaul customers visited in route S_i, the difference function can be defined as follows:

$$\rho\left(S, S''\right) = \frac{\sum_{i=1}^{|S|} |\sigma(S_i) - \sigma(S''_i)| + \sum_{i=1}^{|S|} |\gamma(S_i) - \gamma(S''_i)|}{l + b}, \tag{7.4}$$

Algorithm 7: GVNS algortihm for the 2L-CVRPMB

```
Input:
    $I$: instance of the 2L-CVRPMB;
    Set of neighborhood structures $NS = \{NS_1, NS_2, \dots, NS_{10}\}$;
    Limit $t_{max}$ on the total computing time;
Output:
    Feasible solution $(S, P)$ for the 2L-CVRPMB of value $z(S)$;
$(S, P)$:=findInitialSolution();
repeat
    $k := 1$;
    while $k \leq 10$ do
        $(S', P')$ := shaking($(S, P), NS_k$);
        while $l \leq 10$ do
            $(S'', P'')$ := bestImprovement($(S', P'), NS_l$);
            if $z(S'') \leq z(S')$ then
                $(S', P') := (S'', P'')$;
                $l := 1$;
            end
            else
                l:=l+1;
            end
        end
        if $z(S'') \leq z(S)$ then
            $(S, P) := (S'', P'')$;
            $k := 1$;
        end
        else
            k:=k+1;
        end
    end
until cpuTime() $> t_{max}$;
return $(S, P)$ ;
```

where l and b correspond, respectively, to the number of linehaul and backhaul customers of the solution. Generally, worse solutions are accepted if

$$z(S'') < z(S) + \alpha\rho(S, S''), \tag{7.5}$$

where $z(s)$ corresponds to the value of solution S. However, and motivated by the work presented in [99], we accept worse solutions if

$$z(S'') < (1 + \alpha\rho(S, S''))\ z(S). \tag{7.6}$$

With this procedure, we accept to update the incumbent with a worse value if the incumbent solution is sufficiently different from the local optimum solution. The greater the difference between them, the worse the quality of the accepted solution. As happens with the approaches defined in Section 7.8.1 and Section 7.8.2, only feasible solutions are explored. Algorithm 8 summarizes the SGVNS.

7.9 Computational experiments

7.9.1 Instances

The VNS approaches presented in Sections 7.8.1, 7.8.2, 7.8.3 were tested using instances derived from benchmark instances for the capacitated vehicle routing problem with two-dimensional loading constraints (2L-CVRP) [76, 78, 64]. These instances consider five classes. The first class has a single item width and height equal to one. For the other classes (2 to 5), the number of items ranges from 1 to the number of the class. The number of customers varies between 15 and 255, while the number of items ranges from 15 to 786. The vehicle has a loading area with a height equal to 40 units and width equal to 20 units.

Note that the 2L-CVRP does not include backhaul customers, motivating its adaptation to the 2L-CVRPMB. Based on the generation of instances for the VRPMB proposed in [135], we adapted the 2L-CVRP benchmark instances by replicating each one three times in order to obtain three instances. Each one has 50%, 25% and 10% of backhaul customers compared to the total number of customers. Hereafter, we will

Algorithm 8: Skewed GVNS algortihm for the 2L-CVRPMB

```
Input:
    $I$: instance of the 2L-CVRPMB;
    Set of neighborhood structures $NS = \{NS_1, NS_2, \ldots, NS_{10}\}$;
    Limit $t_{max}$ on the total computing time;
Output:
    Feasible solution $(S, P)$ for the 2L-CVRPMB of value $z(S)$;
$(S, P)$:=findInitialSolution();
$(S_{opt}, P_{opt})$:=$(S, P)$;
repeat
    $k := 1$;
    while $k \leq 10$ do
        $(S', P')$ := shaking($(S, P), NS_k$);
        while $l \leq 10$ do
            $(S'', P'')$ := bestImprovement($(S', P'), NS_l$);
            if $z(S'') \leq z(S')$ then
                $(S', P') := (S'', P'')$;
                $l := 1$;
            end
            else
                $l := l + 1$;
            end
        end
        if $z(S'') \leq z(S_{opt})$ then
            $(S_{opt}, P_{opt}) := (S'', P'')$;
        end
        if $z(S'') < (1 + \alpha\rho(S, S''))z(S)$ then
            $(S, P) := (S'', P'')$;
            $k := 1$;
        end
        else
            $k := k + 1$
        end
    end
until cpuTime() $> t_{max}$;
return $(S, P)$ ;
```

denote by Group A, B and C the instances which consider respectively 50%, 25%, and 10% of backhaul customers.

In order to build the Group C, and taking into account the order in which the customers appear in the instance, we assign as backhaul customers the ones whose ordinal number is multiple of 10. Similarly, for Group A, we assign as backhaul customers the ones whose ordinal number is multiple of 10 or 4. Finally, for Group A, we assign as backhaul customers the ones whose ordinal number is multiple of 10, 4 or 2. The demand of a given customer that is assigned to be a backhaul customer is converted into the supply of the same customer. For each instance, the number of vehicles is used to generate empty routes, when building the initial solution. However, and as stated in Section 7.6, it is possible to increase the number of vehicles when there is no available routes to a given customer.

As referred to in Section 2.3.6, sequential constraints apply to this problem. Applying these constraints with items in original instances may reduce how the set of customers that appear mixed in the route. These happens because each loaded item has a direct impact in loading and unloading operations. One the one hand, each loaded item cannot block items to be delivered. On the other hand, all the loaded items block the space between themselves and the bottom side of the vehicle. Since the width of items ranges between 2 and 18 units and the vehicle has 20 units of width, it is clear that mixing different types of customers is more difficult. In this sense, we adapted the width of each item, and considered only items with 30% of their original width.

With the mentioned procedure, we obtained 540 instances (180 instances for each group). The value of α in the difference function is set at 0.1. The algorithms were coded in C++, and the tests were run on a PC with an i7 CPU with 3.50 GHz and 32 GB of RAM. All the tests were run with a maximum computing time limit of 120 seconds. Partial results were reported during this limit. The instances were divided in 22 sets according to the number of customers. The average results for each set are reported in Tables 7.1-7.3. In Table 7.1, we report on the results obtained with VNS, GVNS and SGVNS for instances of Group A. The results for the same strategies applied to Group B and C are reported in Table 7.2 and Table 7.3. The meaning of the columns in these tables is the following:

n: number of customers;

b: number of backhaul customers;

$\#I$: number of instances in the corresponding set;

$\#it$: average number of item in the corresponding set;

z_{INS}: average value of the initial solution generated using the insertion heuristic described in Section 7.6;

z_{V5}: average value obtained by VNS algorithm within a time limit of 5 seconds; similarly, z_{V30}, z_{V60}, z_{V120} corresponds to the same value within 30, 60 and 120 seconds;

z_{G5}: average value obtained by GVNS algorithm within a time limit of 5 seconds; similarly, z_{G30}, z_{G60}, z_{G120} corresponds to the same value within 30, 60 and 120 seconds;

z_{S5}: average value obtained by SVNS algorithm within a time limit of 5 seconds; similarly, z_{S30}, z_{S60}, z_{S120} corresponds to the same value within 30, 60 and 120 seconds;

imp_V: percentage of improvement achieved with the VNS algorithm in 120 seconds, *i.e.*, $imp = |z_{V120} - z_{INS}|/z_{INS} * 100$; similarly, imp_G and imp_S corresponds to the same percentage for the GVNS and SGVNS, respectively.

Table 7.1: Computational results for Group A

					VNS					GVNS					SGVNS				
n	b	$\#I$	$\#it$	z_{INS}	z_{V5}	z_{V30}	z_{V60}	z_{V120}	imp_V	z_{G5}	z_{G30}	z_{G60}	z_{G120}	imp_G	z_{S5}	z_{S30}	z_{S60}	z_{S120}	imp_S
15	7	10	31	295	279	271	269	269	8,7	253	250	250	250	15,2	253	250	250	250	15,2
20	10	10	40	413	393	380	372	368	11,1	335	333	333	333	19,5	337	333	332	328	20,6
21	10	10	39	484	446	402	398	391	19,2	359	353	353	352	27,4	362	352	352	352	27,3
22	11	10	39	592	550	530	528	528	10,8	522	518	518	518	12,5	519	519	519	519	12,4
25	12	5	56	590	544	514	500	493	16,5	469	456	456	456	22,8	451	449	449	449	23,9
29	14	10	58	666	604	564	543	531	20,3	432	419	419	418	37,2	429	426	426	426	36,1
30	15	5	64	632	582	526	519	508	19,6	456	448	446	446	29,4	456	447	447	446	29,5
32	16	15	62	1470	1359	1266	1216	1187	19,2	1107	1094	1092	1091	25,8	1093	1080	1078	1073	27,0
35	17	5	74	719	698	633	613	569	20,8	536	530	523	519	27,8	538	519	506	506	29,5
40	20	5	79	850	787	703	691	665	21,7	627	618	612	609	28,3	624	611	610	608	28,5
44	22	5	86	987	954	918	866	820	16,9	723	699	699	697	29,3	714	690	686	686	30,5
50	25	5	105	715	704	668	655	609	14,8	504	488	480	480	32,9	491	478	474	470	34,3
71	35	5	146	486	437	424	401	381	21,6	301	278	274	269	44,8	314	279	276	276	43,3
75	37	20	150	1124	1097	1038	1005	967	14,0	722	689	680	672	40,3	737	693	673	660	41,3
100	50	15	204	1415	1378	1307	1249	1192	15,8	963	864	843	828	41,4	981	880	841	817	42,3
120	60	5	246	2071	2020	1689	1583	1486	28,2	1250	1138	1120	1113	46,2	1252	1200	1167	1115	46,1
134	67	5	271	2340	2238	2105	2038	1997	14,7	1599	1261	1190	1121	52,1	1620	1363	1304	1233	47,3
150	75	5	294	1886	1840	1709	1658	1579	16,3	1199	1055	1015	988	47,6	1225	1078	1046	997	47,1
199	99	15	400	2415	2322	2175	2109	2000	17,2	1853	1387	1311	1237	48,8	2255	1459	1370	1280	47,0
240	120	5	485	1116	1094	1030	976	912	18,3	992	618	568	526	52,8	1077	639	593	558	50,0
252	126	5	504	1345	1322	1262	1191	1159	13,8	1316	843	790	763	43,3	1318	891	850	798	40,6
255	127	5	509	1064	1048	955	913	854	19,7	1052	663	634	623	41,5	1046	680	635	611	42,5
Average				1076	1032	958	922	885	17,2	799	682	664	650	34,9	822	696	677	657	34,7

Table 7.2: Computational results for Group B

					VNS					GVNS					SGVNS				
n	b	$\#I$	$\#it$	z_{INS}	z_{V5}	z_{V30}	z_{V60}	z_{V120}	imp_V	z_{G5}	z_{G30}	z_{G60}	z_{G120}	imp_G	z_{S5}	z_{S30}	z_{S60}	z_{S120}	imp_S
15	3	10	31	343	316	301	301	300	12,5	284	280	272	272	20,6	282	276	273	271	20,9
20	5	10	40	473	421	393	384	378	20,1	338	338	338	338	28,6	341	340	339	339	28,4
21	5	10	39	514	485	441	432	430	16,4	397	387	387	385	25,1	381	377	377	377	26,5
22	5	10	39	729	676	641	625	625	14,4	560	554	553	550	24,6	556	547	547	543	25,6
25	6	5	56	736	621	592	581	580	21,1	549	539	535	533	27,5	547	541	540	539	26,7
29	7	10	58	829	717	637	619	604	27,1	478	471	471	470	43,3	480	473	473	473	43,0
30	7	5	64	718	645	591	579	563	21,6	517	496	494	490	31,7	516	506	498	485	32,4
32	8	15	62	1659	1534	1429	1308	1254	24,4	1136	1129	1124	1123	32,3	1151	1137	1126	1120	32,5
35	8	5	74	836	777	724	693	686	17,9	640	626	609	594	29,0	623	601	597	596	28,8
40	10	5	79	1059	910	817	785	769	27,4	744	703	692	692	34,7	725	704	699	695	34,4
44	11	5	86	1291	1137	976	916	893	30,8	718	704	704	704	45,4	721	700	699	695	46,1
50	12	5	105	870	818	768	732	701	19,5	511	503	500	499	42,7	535	502	496	493	43,4
71	17	5	146	634	506	451	416	402	36,6	316	289	277	273	57,0	316	294	274	271	57,3
75	18	20	150	1430	1358	1201	1112	1044	27,0	780	750	743	734	48,7	779	743	734	726	49,2
100	25	15	204	1825	1690	1505	1408	1324	27,5	933	866	852	834	54,3	952	861	848	837	54,2
120	30	5	246	2699	2578	2164	2000	1775	34,2	1763	1311	1261	1244	53,9	1826	1372	1307	1256	53,4
134	33	5	271	3166	2904	2294	2062	1962	38,0	1912	1557	1372	1309	58,6	2413	1498	1403	1271	59,9
150	37	5	294	2498	2427	2194	2029	1903	23,8	1189	1104	1056	1021	59,1	1213	1085	1019	1000	60,0
199	49	15	400	3268	3192	2957	2723	2501	23,5	2838	1429	1332	1260	61,5	3074	1467	1370	1304	60,1
240	60	5	485	1462	1447	1337	1270	1161	20,6	1396	841	617	579	60,4	1399	852	669	614	58,0
252	63	5	504	1895	1842	1745	1668	1518	19,9	1864	912	865	824	56,5	1863	983	932	872	54,0
255	63	5	509	1430	1367	1200	1113	1090	23,8	1405	1260	1060	898	37,2	1405	1270	1079	916	35,9
Average				1380	1289	1153	1080	1021	24,0	967	775	733	710	42,4	1004	779	741	713	42,3

Table 7.3: Computational results for Group C

					VNS					GVNS					SGVNS				
n	b	$\#I$	$\#it$	z_{INS}	z_{V5}	z_{V30}	z_{V60}	z_{V120}	imp_V	z_{G5}	z_{G30}	z_{G60}	z_{G120}	imp_G	z_{S5}	z_{S30}	z_{S60}	z_{S120}	imp_S
15	1	10	31	380	360	345	345	340	10,5	312	310	310	310	18,3	314	310	310	310	18,3
20	2	10	40	516	433	407	402	396	23,4	370	367	367	367	28,9	372	369	367	367	28,9
21	2	10	39	606	515	463	446	429	29,3	417	414	414	413	31,9	413	409	408	407	32,8
22	2	10	39	803	626	579	571	563	29,9	532	513	503	500	37,7	518	506	506	502	37,4
25	2	5	56	809	680	656	652	651	19,5	596	587	587	587	27,4	612	588	581	581	28,1
29	2	10	58	868	763	672	626	611	29,5	534	516	516	516	40,5	532	518	517	514	40,7
30	3	5	64	822	731	638	635	621	24,4	569	552	552	552	32,8	586	564	560	557	32,3
32	3	15	62	1976	1779	1498	1436	1375	30,4	1261	1227	1215	1215	38,5	1223	1204	1202	1202	39,2
35	3	5	74	1019	826	755	745	704	30,9	707	690	674	671	34,2	702	673	667	667	34,6
40	4	5	79	1138	976	918	883	865	23,9	824	815	804	797	29,9	823	793	788	780	31,4
44	4	5	86	1598	1298	995	955	909	43,1	736	733	729	729	54,4	758	731	724	717	55,1
50	5	5	105	1014	886	744	700	681	32,9	545	515	513	513	49,4	529	519	517	515	49,3
71	7	5	146	706	629	512	464	423	40,1	327	315	299	295	58,3	358	332	312	298	57,8
75	7	20	150	1611	1464	1259	1180	1100	31,7	862	818	805	796	50,6	858	813	800	790	51,0
100	10	15	204	2121	1993	1720	1555	1408	33,6	989	939	924	909	57,1	1039	938	912	899	57,6
120	12	5	246	3278	2915	2366	2120	1995	39,1	1793	1456	1413	1374	58,1	2018	1554	1431	1396	57,4
134	13	5	271	3441	3168	2571	2171	2026	41,1	1775	1500	1373	1307	62,0	2589	1464	1349	1300	62,2
150	15	5	294	2919	2748	2465	2244	2064	29,3	1653	1137	1078	1057	63,8	2315	1195	1105	1065	63,5
199	19	15	400	3866	3656	3250	3103	2806	27,4	3564	1472	1412	1362	64,8	3582	1541	1449	1374	64,5
240	24	5	485	1831	1791	1662	1495	1304	28,8	1747	697	655	632	65,5	1748	725	679	646	64,7
252	25	5	504	2239	2223	2109	2012	1779	20,6	2210	993	950	888	60,4	2212	1027	963	930	58,5
255	25	5	509	1711	1667	1468	1335	1232	28,0	1685	1293	1103	898	47,5	1683	1509	1127	938	45,2
Average				1603	1460	1275	1185	1104	29,4	1091	812	782	758	46,0	1172	831	785	762	45,9

7.9.2 Computational results and discussion

The results show that three variable neighborhood search algorithms improve significantly the value of the initial solution in all sets of all groups. Important improvements were achieved within a computing time of 5 seconds.

The achieved results for the all the sets of Group A show an average improvement of 17.2% using VNS algorithm. The lowest average improvement is observed for the set of 15 customers (8.7%), while the best average improvement is observed for the set of 120 customers (28.2%). This set includes 60 backhaul customers and 246 items on average. These results are clearly outperformed by the GVNS algorithm which presents an average improvement of 34.9% compared to the initial solution. Within this algorithm, the highest improvement is 52.8%, and it is obtained for the set of 240 customers (including 120 backhaul customers and 485 items on average). The lowest improvement is 12.5%, and it occurs for the set of 15 customers which includes 7 backhaul customers and 39 items. Similar results are found for the SGVNS algorithm, with an average improvement of 34.7%. For sets with a greater number of customers, both GVNS and SGVNS algorithms tend to present higher improvements. For these two strategies, the improvement is on average greater than 40% for all sets with more than 71 customers. It is worth noting that within only 5 seconds, both algorithms led to an average improvement greater than 20%.

Concerning the Group B (25% of backhaul customers in each instance), all the algorithms led to better improvement values when compared with Group A. Indeed, the VNS presents an average improvement of 24% while the GVNS achieves more than 42%. This latter strategy led to very similar values of those obtained by SGVNS. The SGVNS was able to produce slightly better values in 11 of 22 sets when comparing to the GVNS algorithm. These two strategies present significant improvements within only 5 seconds (more than 27%), and the best values are achieved in the set of 199 customers with an average improvement greater than 60%.

The instances belonging to Group C present the greatest average improvements. For the VNS algorithm, the best average improvement is observed for the set of 44 customers (43.1%). For the GVNS and SGVNS, the best average improvements are achieved for the set of 240 customers (average improvement of roughly 65%). For these two methods, the average improvement is greater than 45% for sets with 44 or more customers. Considering all sets, it is possible to achieve an average improvement of 8.9%, 31.9% and 26.9% for the VNS, GVNS and SGVNS, respectively. Concerning all groups, improvements tend to be more significant when the number of backhaul customer is reduced.

As referred to in Section 7.6, the number of used vehicles can be greater than the fleet size proposed in benchmark instances. Such situations arises when building the initial solution, and it is not possible to insert a given customer in the set of routes. It was observed that these situations happen only in two instances out of the 180 of Group C. However, the VNS and GVNS solutions present a number of vehicles that is not above the fleet size proposed in benchmark instances for both instances. The same behaviour is presented by the SGVNS algorithm for one of the two cases. For the other case, the number of used vehicles remains greater by one unit. The reduction of routes is due to the neighborhood structures including movements of customers to other routes, which could lead to the elimination of a given route.

One important feature in problems with mixed types of customers may be the position of the first backhaul customer within one route. If one vehicle is able to visit backhaul customers earlier, then it is possible to reduce routing costs. Indeed, savings may be lost if it is profitable to visit a given backhaul customer, but this visit is forbidden due to the packing layout configuration of linehaul items. Figure 7.7 presents the position variation of the first backhaul customer in the routes. Only instances of Group A were considered (50% of the number of customers are backhaul). Clearly, the GVNS and SGVNS algorithms present solutions in which backhaul customers start to be inserted earlier in one route. This reduction tends to stabilize after 30 seconds. The VNS presents a moderate reduction of the average position. Thus, the methods that provided higher improvements led also to solutions in which backhaul customers are visited earlier in the routes.

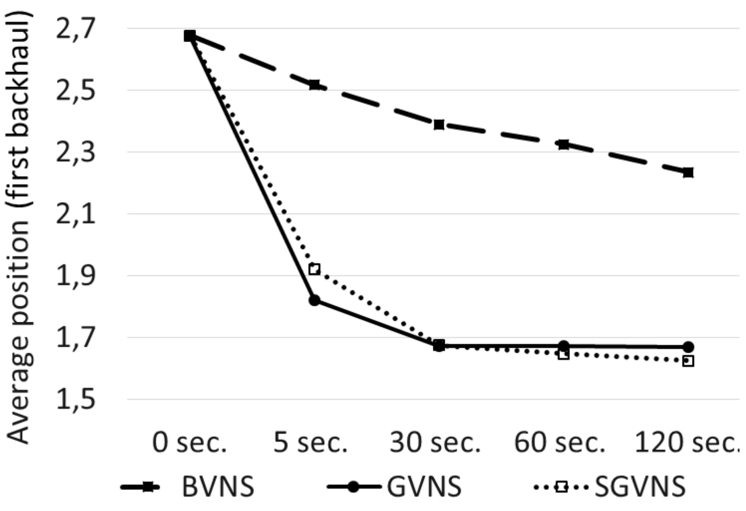

Figure 7.7: Average position of the first backhaul customer

7.10 Conclusion

In this chapter, a capacitated vehicle routing problem with loading constraints and mixed linehauls and backhauls was presented. Both the initial solution and the feasibility of the routes are provided by heuristics adapted from classical approaches described in the literature. We presented three variants of the well-known variable neighborhood search algorithm. In all of these algorithms, we proposed 10 neighborhood structures, including structures that were specifically developed for this problem. The computational experiments were conducted on several instances derived from benchmark instances from the literature. The obtained results show significant improvements for instances with many customers.

Chapter 8

Conclusions

Contents

8.1 Contributions

In this thesis, we developed a set of optimization tools for the vehicle routing problem with loading constraints. We also explored different variants of the problem that are characterized by their set of constraints on the loading and routing part of the problem. The developed methods are based on column generation models and on metaheuristics.

With this thesis, we are contributing with new algorithms to increase the efficiency in transportation and supply chain management, since it could improve the competitiveness companies which intend to differentiate its service level, taking into account complex constraints that can be found in real-world situations, while minimizing costs. In this sense, the contributions of this thesis are expected to have both scientific and practical relevance.

In Chapter 4, and to the best of our knowledge, we contributed with the first set of results for the elementary shortest path with two-dimensional loading constraints. We suggested different constructive strategies in order to build initial solutions. We also suggested various neighborhood structures, some of them based on packing features of the problem. We presented a computational study on a variant of the variable neighborhood search, using a large set of benchmark instances. Furthermore, the contribution of this approach is not confined to the problem itself. Indeed, the addressed problem may correspond to the subproblem of the vehicle routing problem with two-dimensional constraints, when Dantzig-Wolfe decomposition is applied. In this sense, the proposed methods were also used throughout the thesis.

In Chapter 5, we contributed with a branch-and-price algorithm for the vehicle routing problem with two-dimensional loading constraints. Column generation approaches for this problem are not frequently used. Therefore, we intended to contribute with new features such as different partition strategies of the branching tree. We also suggested a family of valid cuts that may accelerate the convergence of the algorithm.

We also contributed with a new set of column generation based heuristic methods for the vehicle routing with two-dimensional constraints, as presented in Chapter

6. These heuristics operate in the Dantzig-Wolfe reformulated model, and can be seen as constructive heuristics, which iteratively select one route to the solution, while reducing the complexity of the problem. We conducted on an exhaustive set of computational experiments, using for that purpose a large set of computational experiments.

In the final part of the thesis, in Chapter 7, we contributed with three algorithms for a pickup and delivery variant of the routing problem with loading constraints. These algorithms consist in variants of the variable neighborhood search, and resort to different neighborhood structures. Some of these structures were particularly developed for the suggested problem. We adapted a large set of benchmark instances, in order to consider the explicit consideration of integrated routing and loading features.

8.2 Future research

During the execution of this thesis, many ideas emerged that will be analyzed in a near future. Regarding the presented column generation approach, it is out purpose to analyze other acceleration strategies, in order to further enhance the convergence of the algorithm. Additionally, other heuristics in the space of the reformulated model will be explored such as the feasibility pump heuristics.

Concerning the subproblem, we aim to explore strong and exact models to the elementary shortest path problem with loading constraints. Despite both the complexity of the problem, and the promising results obtained, the use of metaheuristics leads to a solution not necessary optimal. This feature is important in branch-and-price: it is not possible to derive lower bounds from the LP relaxation, and consequently, we may dive deeply in the branching tree, and we still analyze not promising nodes. Additionally, we aim to explore new lower bounds and valid inequalities for routing problems with loading constraints, using for this purpose promising tools, namely the dual feasible functions.

In what concerns the proposed problems, we aim to explore further variants of the standard vehicle routing problem with two dimensional loading constraints, with additional restrictions and features such as multiple trips and time windows. For this purpose, we may extend some of the approaches developed in this thesis. Finally, we intend to apply the developed models to real instances arising in real applications of the problem, so that we can rigorously assess their adequacy.

Bibliography

[1] R. Alvarez-Valdés, F. Parreño, and J. Tamarit. Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35(4):1065–1083, 2008.

[2] C. Alves. Planeamento de rotas em sistemas de recolha/distribuio. Master's thesis, University of Minho, 2000.

[3] D. Aprile, J. Egeblad, A. Garavelli, S. Lisi, and D. Pisinger. Logistics optimization: vehicle routing with loading constraints. In *ICPR-19th International Conference on Production Research*, Valparaíso, Chile, 2007.

[4] C. Archetti, N. Bianchessi, and M. Speranza. Optimal solutions for routing problems with profits. *Discrete Applied Mathematics*, 161(4):547–557, 2013.

[5] A. Attanasio, A. Fuduli, G. Ghiani, and C. Triki. Integrated shipment dispatching and packing problems: a case study. *Journal of Mathematical Modelling and Algorithms*, 6(1):77–85, 2007.

[6] B. Baker, E. Coffman, Jr, and R. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.

[7] R. Baldacci, M. Battarra, and D. Vigo. *Routing a Heterogeneous Fleet of Vehicles*, volume 43 of *Operations Research/Computer Science Interfaces*, book section 1, pages 3–27. Springer US, 2008. ISBN 978-0-387-77777-1.

[8] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

[9] M. Ball. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16(1):21–38, 2011.

[10] T. Bartók and C. Imreh. Pickup and delivery vehicle routing with multidimensional loading constraints. *Acta Cybernetica*, 20(1):17–33, 2011.

[11] M. Battarra, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11):3041–3050, 2009.

[12] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: aclassification scheme and survey. *TOP*, 15(1):1–31, 2007.

[13] G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[14] L. Bertazzi and M. Speranza. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1(4):307–326, 2012.

[15] A. Bettinelli, A. Ceselli, and G. Righini. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 19(5):723–740, 2011.

[16] N. Bianchessi, R. Mansini, and M. Speranza. The distance constrained multiple vehicle traveling purchaser problem. *European Journal of Operational Research*, 235(1):73–87, 2014.

[17] E. Bischoff and M. Ratcliff. Issues in the development of approaches to container loading. *Omega*, 23(4):377–390, 1995.

[18] C. Bochtis, D.and Srensen. The vehicle routing problem in field logistics: Part II. *Biosystems Engineering*, 105(2):180–188, 2010.

[19] D. Bochtis and C. Srensen. The vehicle routing problem in field logistics part I. *Biosystems Engineering*, 104(4):447–457, 2009.

[20] A. Bortfeldt. A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research*, 39(9):2248–2257, Sept. 2012.

[21] A. Bortfeldt and H. Gehring. Two meta heuristics for strip packing problems. In D. Despotis and C. Zoponoudis, editors, *Proceedings of the Fifth International Conference of the Decision Sciences Institute*, number 2, pages 1153–1156, Athens, Greece, 1999.

[22] A. Bortfeldt, T. Hahn, D. Männel, and L. Mönch. Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. *European Journal of Operational Research*, 243(1):82–96, 2015.

[23] A. Bortfeldt and J. Homberger. Packing first, routing second- a heuristic for the vehicle routing and loading problem. *Computers & Operations Research*, 40 (3):873–885, 2013.

[24] A. Bortfeldt and G. Wäscher. Constraints in container loading – A state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013.

[25] G. Brown and G. Graves. Real-time dispatch of petroleum tank trucks. *Management Science*, 27(1):19–32, 1981.

[26] E. Burke, G. Kendall, and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671, 2004.

[27] D. Casco, B. Golden, and E. Wasil. Vehicle routing with backhauls: models, algorithms and case studies. In B. Golden and A. Assad, editors, *Vehicle routing: Methods and studies*, pages 127–147. Elsevier: Amsterdam, 1988.

[28] S. Ceschia, A. Schaerf, and T. Stützle. Local search techniques for a routing-packing problem. *Computers & Industrial Engineering*, 66(4):1138–1149, 2013.

[29] A. Ceselli, G. Righini, and M. Salani. A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, 43(1):56–69, 2009.

[30] E. Chajakis and M. Guignard. Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, 26(1):43–78, 2003.

[31] B. Chazelle. The bottom-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, 100(8):697–707, 1983.

[32] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[33] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, 35(3):944–959, 2008.

[34] L. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2013.

[35] J. Cordeau, M. Iori, S. Ropke, and D. Vigo. Branch-and-cut-and-price for the capacitated vehicle routing problem with two-dimensional loading constraints. In *ROUTE 2007*, Jekyll Island, U.S.A., 2007.

[36] J.-F. Cordeau, M. Dell'Amico, S. Falavigna, and M. Iori. A rolling horizon algorithm for auto-carrier transportation. *Transportation Research Part B: Methodological*, 76:68–80, 2015.

[37] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[38] F. Cornillier, F. Boctor, and J. Renaud. Heuristics for the multi-depot petrol station replenishment problem with time windows. *European Journal of Operational Research*, 220(2):361–369, 2012.

[39] F. Cornillier, F. F. Boctor, G. Laporte, and J. Renaud. A heuristic for the multi-period petrol station replenishment problem. *European Journal of Operational Research*, 191(2):295–305, 2008.

[40] T. G. Crainic, G. Perboli, and R. Tadei. Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on computing*, 20(3):368–384, 2008.

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

[42] J.-F. Ct, M. Gendreau, and J.-Y. Potvin. The vehicle routing problem with stochastic two-dimensional items. Technical report, CIRRELT-2013-84, 2013.

[43] J. L. M. da Silveira, F. K. Miyazawa, and E. C. Xavier. Heuristics for the strip packing problem with unloading constraints. *Computers & Operations Research*, 40(4):991–1003, 2013.

[44] J. L. M. da Silveira, E. C. Xavier, and F. K. Miyazawa. Two dimensional strip packing with unloading constraints. *Electronic Notes in Discrete Mathematics*, 37:99–104, 2011.

[45] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[46] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[47] M. Daum and W. Menzel. Parsing natural language using guided local search. In *ECAI 2002: 15th European Conference on Artificial Intelligence*, pages 435–439, 2002.

[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[49] M. DellAmico, S. Falavigna, and M. Iori. Optimization of a real-world auto-carrier transportation problem. *Transportation Science*, 49(2):402–419, 2014.

[50] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2): 342–354, 1992.

[51] K. F. Doerner, G. Fuellerer, R. F. Hartl, M. Gronalt, and M. Iori. Metaheuristics for the vehicle routing problem with loading constraints. *Networks*, 49(4):294–307, July 2007.

[52] O. Dominguez, A. Juan, and J. Faulin. A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research*, 21(3):375–398, 2014.

[53] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–978, 1994.

[54] C. Duhamel, P. Lacomme, A. Quilliot, and H. Toussaint. A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. *Computers and Operations Research*, 38(3):617–640, 2011.

[55] H. Dyckhoff. Cutting and packinga typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[56] A. Fallahi, C. Prins, and R. Wolfler Calvo. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 35(5):1725–1741, 2008.

[57] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[58] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[59] K. Fox, B. Gavish, and S. Graves. An n-constraint formulation of the (time dependent) traveling salesman problem. *Operations Research*, 28(4):1018, 1980.

[60] P. Francis, K. Smilowitz, and M. Tzur. *The Period Vehicle Routing Problem and its Extensions*, volume 43 of *Operations Research/Computer Science Interfaces*, book section 4, pages 73–102. Springer US, 2008. ISBN 978-0-387-77777-1.

[61] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 36(3):655–673, 2009.

[62] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori. Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3):751–759, Mar. 2010.

[63] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A Tabu Search Algorithm for a Routing and Container Loading Problem. *Transportation Science*, 40(3): 342–350, Aug. 2006.

[64] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51(1):4–18, Jan. 2008.

[65] B. Golden, E. Baker, J. Alfaro, and J. Schaffer. The vehicle routing problem with backhauling: two approaches. In *Proceedings of the twenty-first annual meeting of the SE TIMS*, Myrtle Beach, SC, USA, 1985.

[66] B. Golden, J. Dearmon, and E. Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10 (1):47–59, 1983.

[67] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, volume 43. Springer, 2008.

[68] K. Hamdi-Dhaoui, N. Labadie, and A. Yalaoui. Algorithms for the two dimensional bin packing problem with partial conflicts. *RAIRO-Operations Research*, 46(01):41–62, 2012.

[69] K. Hamdi-Dhaoui, N. Labadie, and A. Yalaoui. The bi-objective two-dimensional loading vehicle routing problem with partial conflicts. *International Journal of Production Research*, 52(19):5565–5582, 2014.

[70] P. Hansen, B. Jaumard, N. Mladenović, and A. Parreira. Variable neighborhood search for weighted maximum satisfiability problem. In *Les Cahiers du GERAD G-2000-62*, Montral, Canada, 2000.

[71] P. Hansen and N. Mladenović. *Variable Neighborhood Search*, pages 211–238. Springer US, Boston, MA, 2005. ISBN 978-0-387-28356-2.

[72] P. Hansen, N. Mladenovic, and J. Pérez. Variable neighborhood search: methods and applications. *Annals of Operations Research*, 175:367–407, 2010.

[73] P. Hansen, N. Mladenović, and J. Pérez. Variable neighbourhood search: methods andapplications. *Annals of Operations Research*, 175(1):367–407, 2010.

[74] P. Hokama, F. Miyazawa, and E. Xavier. A branch-and-cut approach for the vehicle routing problem with loading constraints. *Expert Systems with Applications*, 47:1–13, 2016.

[75] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

[76] M. Iori. *Metaheuristic algorithms for combinatorial optimization problems.* PhD thesis, DEIS, University of Bologna, Italy, 2004.

[77] M. Iori and S. Martello. Routing problems with loading constraints. *TOP*, 18 (1):4–27, 2010.

[78] M. Iori, J.-J. Salazar-Gonzalez, and D. Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation science*, 41(2):253–264, 2007.

[79] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-Cycle elimination for k $\geq$ 3. *INFORMS Journal on Computing*, 18(3): 391–406, 2006.

[80] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, and F. Vanderbeck. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

[81] L. Junqueira and R. Morabito. Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier. *Computers & Industrial Engineering*, 88:110–130, 2015.

[82] L. Junqueira, R. Morabito, and D. Yamashita. MIP-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, 199(1):51–75, 2012.

[83] L. Junqueira, J. Oliveira, Carravilla, and R. Morabito. An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. *International Transactions in Operational Research*, 20(5):645–666, 2013.

[84] B. Kallehauge, J. Larsen, O. Madsen, and M. Solomon. *Vehicle Routing Problem with Time Windows*, pages 67–98. Springer US, Boston, MA, 2005. ISBN 978-0-387-25486-9.

[85] A. Khanafer, F. Clautiaux, and E.-G. Talbi. New lower bounds for bin packing problems with conflicts. *European journal of operational research*, 206(2):281–288, 2010.

[86] S. Khebbache-Hadji, C. Prins, A. Yalaoui, and M. Reghioui. Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *Central European Journal of Operations Research*, 21(2):307–336, 2013.

[87] G. King and T. Mast. Excess travel: causes, extent, and consequences. *Transportation Research Record*, 1111:126–134, 1987.

[88] P. Lacomme, H. Toussaint, and C. Duhamel. A GRASPxELS for the vehicle routing problem with basic three-dimensional loading constraints. *Engineering Applications of Artificial Intelligence*, 26(8):1795–1810, Sept. 2013.

[89] G. Laporte and I. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262, 1995.

[90] S. Leung, Z. Zhang, D. Zhang, X. Hua, and M. Lim. A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research*, 225(2):199–210, 2013.

[91] S. Leung, J. Zheng, D. Zhang, and X. Zhou. Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. *Flexible services and manufacturing journal*, 22(1-2):61–82, 2010.

[92] S. Leung, X. Zhou, D. Zhang, and J. Zheng. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 38(1):205–215, 2011.

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

[94] A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357, 1999.

[95] H. Lourenço, O. Martin, and T. Stützle. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pages 363–397. Springer, 2010.

[96] M. Lübbecke. Column generation. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[97] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[98] R. Macedo, C. Alves, and J. M. V. de Carvalho. Exact algorithms for vehicle routing problems with different service constraints. In *CTW*, pages 215–218, 2009.

[99] R. Macedo, C. Alves, S. Hanafi, B. Jarboui, N. Mladenović, B. Ramos, and J. Valério de Carvalho. Skewed general variable neighborhood search for the location routing scheduling problem. *Computers & Operations Research*, 61: 143–152, 2015.

[100] R. Macedo, B. Ramos, C. Alves, J. Valério de Carvalho, S. Hanafi, and N. Mladenović. Integer programming based approaches for multi-trip location routing. In *Computational Management Science*, pages 79–90. Springer, 2016.

[101] B. Mahvash, A. Awasthi, and S. Chauhan. A column generation based heuristic for the capacitated vehicle routing problem with three-dimensional loading constraints. 48(3):448 – 453, 2015. 15th IFAC Symposium on Information Control Problems in Manufacturing.

[102] A. Malapert, C. Guéret, N. Jussien, A. Langevin, and L.-M. Rousseau. Two-dimensional pickup and delivery routing problem with loading constraints. Technical report, CIRRELT-2008-37, 2008.

[103] Y. Marinakis. Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807–6815, 2012.

[104] Y. Marinakis, M. Marinaki, and G. Dounias. *Honey Bees Mating Optimization Algorithm for the Vehicle Routing Problem*, volume 129 of *Studies in Computational Intelligence*, book section 13, pages 139–148. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78986-4.

[105] Y. Marinakis, M. Marinaki, and G. Dounias. Honey bees mating optimization algorithm for large scale vehicle routing problems. *Natural Computing*, 9(1): 5–27, 2009.

[106] Y. Marinakis and A. Migdalas. Annotated bibliography in vehicle routing. *Operational Research*, 7(1):27–46, 2007.

[107] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.

[108] S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399, 1998.

[109] L. Martinez and C. Amaya. A vehicle routing problem with multi-trips and time windows for circular items. *Journal of the Operational Research Society*, 64(11):1630–1643, 2013.

[110] L. Miao, Q. Ruan, K. Woghiren, and Q. Ruo. A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. *RAIRO - Operations Research*, 46(01):63–82, 2012.

[111] P. Mills, E. Tsang, and J. Ford. Applying an extended guided local search to the quadratic assignment problem. *Annals of Operations Research*, 118(1-4): 121–135, 2003.

[112] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[113] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[114] A. Moura. A multi-objective genetic algorithm for the vehicle routing with time windows and loading problem. In *Intelligent Decision Support*, pages 187–201. Springer, 2008.

[115] A. Moura and J. Oliveira. An integrated approach to the vehicle routing and container loading problems. *OR spectrum*, 31(4):775–800, 2009.

[116] A. Moura, J. Oliveira, and C. Pimentel. A mathematical model for the container stowage and ship routing problem. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(3):217–231, 2013.

[117] D. Naddef. *Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP*, volume 12 of *Combinatorial Optimization*, book section 2, pages 29–116. Springer US, 2007. ISBN 978-0-387-44459-8.

[118] G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141, 2005.

[119] J. Oppen, A. Lkketangen, and J. Desrosiers. Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, 37(7):1308–1317, 2010.

[120] S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems. *Journal fr Betriebswirtschaft*, 58(1):21–51, 2008.

[121] S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems. *Journal fr Betriebswirtschaft*, 58(2):81–117, 2008.

[122] M. Pavone. *Dynamic vehicle routing for robotic networks.* PhD thesis, Massachusetts Institute of Technology, 2010.

[123] V. Pillac, M. Gendreau, C. Guéret, and A. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1): 1–11, 2013.

[124] T. Pinto, C. Alves, and J. Valério de Carvalho. Column generation based heuristic for a vehicle routing problem with 2-dimensional loading constraints: a prototype. In *XI Congresso Galego de Estatística e Investigación de Operacións*, Spain, 2013.

[125] T. Pinto, C. Alves, and J. Valério de Carvalho. *Variable Neighborhood Search for the Elementary Shortest Path Problem with Loading Constraints*, volume

9156 of *Lecture Notes in Computer Science*, book section 34, pages 474–489. Springer International Publishing, 2015. ISBN 978-3-319-21406-1.

[126] H. Pollaris, K. Braekers, A. Caris, G. Janssens, and S. Limbourg. Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints. *EURO Journal on Transportation and Logistics*, pages 1–25, 2014.

[127] H. Pollaris, K. Braekers, A. Caris, G. Janssens, and S. Limbourg. Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, 37(2):297–330, 2014.

[128] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[129] M. Reimann, K. Doerner, and R. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4): 563–591, 2004.

[130] G. Reinelt. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

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

[132] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

[133] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

[134] Q. Ruan, Z. Zhang, L. Miao, and H. Shen. A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research*, 40(6):1579 – 1589, 2013.

[135] S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *The Journal of the Operational Research Society*, 50(10):1034–1042, 1999.

[136] M. Savelsbergh and M. Sol. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490, 1998.

[137] V. Souza. Algoritmos para o problema de roteamento de veículos capacitado com restrições de carregamento bidimensional. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, 2013.

[138] R. Spliet and G. Desaulniers. The discrete time window assignment vehicle routing problem. *European Journal of Operational Research*, 244(2):379 – 391, 2015.

[139] R. Tadei, G. Perboli, and F. Della Croce. A heuristic algorithm for the auto-carrier transportation problem. *Transportation Science*, 36(1):55–62, 2002.

[140] Y. Tao and F. Wang. An effective tabu search approach with improved loading algorithms for the 3L-CVRP. *Computers & Operations Research*, 55:127–140, 2015.

[141] C. Tarantilis, E. Zachariadis, and C. Kiranoudis. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):255–271, 2009.

[142] P. Toth and D. Vigo. An overview of vehicle routing problems. In *The vehicle routing problem*, pages 1–26. Society for Industrial and Applied Mathematics, 2002.

[143] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[144] F. Tricoire, K. Doerner, R. Hartl, and M. Iori. Heuristic and exact algorithms for the multi-pile vehicle routing problem. *OR Spectrum*, 33(4):931–959, 2011.

[145] J. Valério de Carvalho. Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005.

[146] A. Wade and S. Salhi. An investigation into a new class of vehicle routing problem with backhauls. *Omega*, 30(6):479–487, 2002.

[147] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[148] C. Waters. A solution procedure for the vehicle-scheduling problem based on iterative route improvement. *Journal of the Operational Research Society*, 38 (9):833–839, 1987.

[149] L. Wei, W.-C. Oon, W. Zhu, and A. Lim. A skyline heuristic for the 2D rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337–346, 2011.

[150] L. Wei, Z. Zhang, and A. Lim. An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. *Computational Intelligence Magazine, IEEE*, 9(4):18–30, 2014.

[151] L. Wei, Z. Zhang, D. Zhang, and A. Lim. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 243(3):798–814, 2015.

[152] S. Wøhlk. *A Decade of Capacitated Arc Routing*, volume 43 of *Operations Research/Computer Science Interfaces*, book section 2, pages 29–48. Springer US, 2008. ISBN 978-0-387-77777-1.

[153] E. Zachariadis and C. Kiranoudis. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, 37(12):2089–2105, 2010.

[154] E. Zachariadis, C. Tarantilis, and C. Kiranoudis. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 195(3):729–743, 2009.

[155] E. Zachariadis, C. Tarantilis, and C. Kiranoudis. The pallet-packing vehicle routing problem. *Transportation Science*, 46(3):341–358, 2011.

[156] E. Zachariadis, C. Tarantilis, and C. Kiranoudis. Integrated distribution and loading planning via a compact metaheuristic algorithm. *European Journal of Operational Research*, 228(1):56–71, 2013.

[157] Z. Zhang, L. Wei, and A. Lim. An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. *Transportation Research Part B: Methodological*, 82:20–35, 2015.

[158] W. Zhu, H. Qin, A. Lim, and L. Wang. A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. *Computers & Operations Research*, 39(9):2178–2195, 2012.