Università degli Studi di Udine

Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica

Dottorato di Ricerca in Ingegneria Industriale e dell'Informazione

Ph.D. Thesis

Local Search Algorithms for Integrated Logistics

Candidate:
Sara Ceschia

Supervisor:
Andrea Schaerf

February 28, 2012

Author's e-mail: sara.ceschia@uniud.it

Author's address:

Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine
Italia

Abstract

In different industrial fields, the competitiveness on lead times has pushed companies to equip themselves with advanced computer systems for designing and scheduling integrated logistics. This trend has increased the attention of the academic community to optimization problems that arise from real-world logistic applications. However, the gap between problem models presented in literature and those that occur in the real-world situations is still large.

With this thesis we aim to develop models and metaheuristic search methods to solve complex real-world problems in the domain of computational logistics. In particular our interest has been focused to routing and loading problems, which have several practical applications in transportation, cutting, packing, and scheduling.

As a first step, we present a rich *vehicle routing problem* which includes a heterogeneous fleet, a multi-day planning horizon, a complex carrier-dependent cost function for vehicles, and the possibility of leaving orders unscheduled.

We then analyze and extend the *container loading problem* by including some practical constraints such as box rotations, bearing weights of boxes, multiple containers of different types and the possibility of associating a destination to each item, such that a loading order has to be respected.

Lastly, we address a problem which can be considered the combination of the two previous ones. Starting from the *capacitated vehicle routing problem with three dimensional loading constraints*, we redefine the notions of stability of the cargo, fragility of items, loading and unloading policies. We also consider the case of split the demand of a customer among multiple vehicles.

Our solution approaches are based on local search which works on the search space explored using multiple neighborhoods. The actual loading is obtained by means of specific heuristics that load boxes on containers following different packing strategies, taking care of all the loading constraints.

The outcome is that our solvers have been able to solve complex problems and also to compete fairly well with state-of-the-art solvers developed *ad hoc* for the simpler problem.

In addition, we made available to the community a set of challenging real-world instances provided by industrial partners, that could potentially become a benchmark set for future researches.

Acknowledgments

The sincerest thank goes to my supervisor Andrea Schaerf. Thanks for everything he taught me, for his enthusiasm, for his constant support, for his generosity, and for his sense of humor that has contributed to make the working atmosphere such pleasant and lively.

Many thanks to Luca Di Gaspero for providing me ideas and feedback, for his invaluable help and patience, and for agreeing to share the office with me. I wish also to thank my friends (and colleagues) Tommaso and Amos, with whom I had the pleasure to share also this adventure. Special thanks to Pier Luca Montessoro for the kindness and the helpfulness he has always reserved to me, and to the coordinator of the Ph.D. school Roberto Rinaldo.

I am very grateful to Thomas Stützle for his inputs and suggestions on my work, and for enabling my research visit at the IRIDIA in Bruxelles. I want to thank Karl Doerner and Amedeo Cesta for their constructive comments and remarks, and the help in improving this thesis.

Thanks also to our industrial partners beanTech e EasyStaff for bringing to our attention the problems and providing the data.

I am greatly grateful to my family. Thanks to my mum, my dad, my aunt Laura and my uncle Luigi, for their love and their unconditional support.

Last but not least, Daniele: your candid sweetness, your caring attentions, your unwavering confidence and your contagious optimism make you very special.

Contents

1 Introduction **1**
- 1.1 Scientific and industrial context . . . 1
- 1.2 Objectives and contributions . . . 2
- 1.3 Scientific publications connected with the thesis . . . 2
- 1.4 Organization of the thesis . . . 3

2 Background **5**
- 2.1 Combinatorial optimization problems . . . 5
- 2.2 Local search algorithms . . . 6
  - 2.2.1 Basic concepts of local search . . . 6
  - 2.2.2 Simulated annealing . . . 7
  - 2.2.3 Tabu search . . . 8
- 2.3 Software tools and environment . . . 10

I State of the art **15**

3 Vehicle Routing Problems: a review **17**
- 3.1 Introduction . . . 17
- 3.2 Problems' definition and notation . . . 17
  - 3.2.1 Capacitated vehicle routing problem . . . 17
  - 3.2.2 Vehicle routing problem with time windows . . . 19
  - 3.2.3 Heterogeneous vehicle routing problem . . . 19
  - 3.2.4 Vehicle routing problem with private fleet and common carrier . . . 21
  - 3.2.5 Period vehicle routing problem . . . 21
  - 3.2.6 Split delivery vehicle routing problem . . . 22
  - 3.2.7 Three-dimensional loading capacitated vehicle routing problem . . . 23
- 3.3 Solution techniques . . . 23
  - 3.3.1 Exact methods . . . 24
  - 3.3.2 Heuristics . . . 24
  - 3.3.3 Metaheuristics . . . 25

4 Three dimensional packing problems: a review **27**
- 4.1 Introduction . . . 27
- 4.2 Problems classification . . . 27
  - 4.2.1 Additional issues . . . 28
- 4.3 Solution techniques . . . 29
  - 4.3.1 Exact algorithms . . . 30
  - 4.3.2 Heuristics . . . 30
  - 4.3.3 Metaheuristics . . . 31

II Real-world applications **33**

5 The vehicle routing problems with time windows and carrier-dependent costs **35**
- 5.1 Introduction . . . 35
- 5.2 Problem formulation . . . 36
  - 5.2.1 Features of the problem . . . 36

5.2.2 Vehicle cost functions . . . 38
5.2.3 Constraints and objective function . . . 39
5.3 Application of tabu search . . . 40
5.3.1 Search space, cost function, and initial solution . . . 40
5.3.2 Neighborhood relations . . . 42
5.3.3 Prohibition rules . . . 42
5.3.4 Search techniques . . . 43
5.4 Experimental analysis . . . 44
5.4.1 Benchmark instances . . . 44
5.4.2 General settings and implementation . . . 45
5.4.3 Results on prohibition rules for TS(Ins) . . . 46
5.4.4 Results on inter-route swap and intra-route swap neighborhoods . . . 46
5.4.5 Results on composite solvers . . . 46
5.4.6 Comparison with benchmarks of the VRPPC . . . 51
5.5 Summary . . . 51

6 The multiple container loading problem with bearing weights and multi-drop 53
6.1 Introduction . . . 53
6.2 Problem description . . . 53
6.2.1 Basic problem . . . 54
6.2.2 Complete problem . . . 55
6.3 Solution for the single destination problem . . . 56
6.3.1 Preprocessing . . . 56
6.3.2 Search space . . . 56
6.3.3 Loader . . . 57
6.3.4 Cost function . . . 58
6.3.5 Neighborhood relation . . . 58
6.3.6 Initial solution . . . 59
6.3.7 Metaheuristics . . . 60
6.4 Solution technique for the multi-drop problem . . . 60
6.5 Experimental analysis . . . 61
6.5.1 Instances . . . 61
6.5.2 Parameter setting . . . 61
6.5.3 Experimental results . . . 62
6.5.4 Comparison with related work . . . 63
6.6 Summary . . . 65

7 A routing and packing problem 67
7.1 Introduction . . . 67
7.2 Problem description . . . 67
7.2.1 3L-CVRP formulation . . . 67
7.2.2 Complete problem . . . 68
7.3 Solution technique . . . 72
7.3.1 Search space . . . 72
7.3.2 Cost function . . . 73
7.3.3 Initial solution . . . 73
7.3.4 Neighborhood relations . . . 73
7.3.5 Loading heuristics . . . 75
7.3.6 Metaheuristics . . . 76
7.4 Experimental analysis . . . 78
7.4.1 Parameter settings . . . 78
7.4.2 New instances . . . 79
7.4.3 Experimental results . . . 80
7.4.4 Comparison with related work . . . 80

7.5 Summary 81

8 Other Problems **83**
8.1 Timetabling problems 83
8.2 Healthcare problems 84

Conclusions **85**

A Notation **87**
A.1 Combinatorial optimization problems (Section 2.1) 87
A.2 Local search algorithms (Section 2.2) 87
A.2.1 Simulated annealing (Section 2.2.2) 87
A.2.2 Tabu search (Section 2.2.3) 87
A.3 Vehicle routing problems 88
A.3.1 Vehicle routing problem with time windows (Section 3.2.2) 88
A.3.2 Heterogenous vehicle routing problem (Section 3.2.3) 88
A.3.3 Vehicle routing problem with private fleet and common carrier (Section 3.2.4) 88
A.3.4 Period vehicle routing problem (Section 3.2.5) 88
A.3.5 Split delivery vehicle routing problem (Section 3.2.6) 89
A.4 Three dimensional packing problems 89
A.5 The vehicle routing problems with time windows and carrier-dependent costs . . . 89
A.5.1 Problem formulation (Section 5.2) 89
A.5.2 Application of tabu search (Section 5.3) 90
A.6 The multiple container loading problem with bearing weights and multi-drop . . . 90
A.7 A routing and packing problem 90
A.7.1 3L-CVRP formulation (Section 7.2.1) 90
A.7.2 Complete problem (Section 7.2.2) 91
A.7.3 Solution technique (Section 7.3) 91

List of Abbreviations **93**

Bibliography **95**

List of Figures

1.1 Outline of the thesis. 4

2.1 A general local search procedure 7
2.2 The pseudocode of the SA procedure 8
2.3 Flow chart of the SA procedure 9
2.4 The pseudocode of the TS procedure 10
2.5 Flow chart of the TS procedure 11
2.6 The pseudocode of the *BestMove* procedure of TS 12
2.7 Flow chart of the procedure for the selection of the best move in the TS algorithm. 13

3.1 A classification of some vehicle routing problems and their interconnections 17
3.2 A solution of an instance of CVRP. 18

4.1 A classification of Three Dimensional Packing Problems and their interconnections 27

5.1 An example of a state composed of 4 routes for 2 days (r_{jk} identifies the route j on day k). 41
5.2 Results for different tabu list lengths of TS(Ins) for case1-A with different prohibition rules. 47
5.3 Results for different tabu list lengths of TS(IntraSw) and TS(InterSw) for case2-A. . 48
5.4 Results of the F-Race selection procedure on the composite solvers. 49

6.1 Multi-drop container loading. 56
6.2 Loader with rigid and flexible layers. 57
6.3 Stack construction with aside boxes. 57
6.4 An example of move. 59
6.5 An example of duplicated move 60
6.6 Example of initial solution. 60

7.1 Violation of the Fragility constraint. 69
7.2 Skewed item stack 70
7.3 The Robust Stability constraint 70
7.4 Manual 3L-CVRP: a human operator can simply slide out the box 1 without touching box 2. 71
7.5 Situation impracticable for unloading box 1 71
7.6 The Reachability constraint 72
7.7 The search space consist of sequences of blocks of boxes. 73
7.8 An example of MCCS move. 74
7.9 An example of an MRB move that leads to a split delivery. 75
7.10 The Back Left Low, Back Low Left, Low Back Left, Low Left Back, Left Low Back, Left Back Low strategies. 77
7.11 A GLSM representation of the control flow of our algorithm. See the text for more details. 78

List of Tables

3.1 Taxonomy of *vehicle routing problems with heterogeneous fleet* 20
3.2 References to manuscripts on VRPs grouped by problem formulation and solution technique. 24

4.1 Classification of packing problems 28
4.2 Manuscripts on packing problems classified by problem formulation and solution technique. 30

5.1 Prohibition rules for TS(Ins). 43
5.2 Features of the two scenarios. 45
5.3 Features of instances. 45
5.4 Values (in €) of the different cost components for the best solutions. 50
5.5 Results on benchmarks Chu-H and B-H of the *Vehicle Routing Problem with Private fleet and Common carrier*. 51

6.1 Families of instances. 61
6.2 Results of the C3 component on family CS1. 62
6.3 Results of the C2 and C3 components on family CS2. 62
6.4 Results of the C1 component on family CS3. 62
6.5 Results of the C1 and C2 components on family CS4. 63
6.6 Features of the families IMM and BR1–BR15. 63
6.7 Results for family IMM. 64
6.8 Results for family BR. 66

7.1 Features of the new real-world instances 79
7.2 Results on our instances with different constraints (H2 in m^3 and z in km). 80
7.3 Computational results for 3L-CVRP on Gendreau et al. [91] instances (time is in seconds). 82

1 Introduction

1.1 Scientific and industrial context

Nowadays companies have to face an ever-growing competition, that inevitably leads not only to analyze the market in which they act, but also to make internal processes effective and efficient as possible.

In this context, logistics engineering has been developed with the aim to systematically organize everything related to the purchase, transport, storage and distribution of products. Although, in terms of management control, logistics is considered a cost center, actually many activities of the customer service, which add up value to a product, are related to the logistics. These processes are strategic and their organization and planning is a key element for the success (or survival) of a company in the global market. Indeed, every company faces the problem of optimally managing the resources available in such a way to maximize profits and customer satisfaction. This has pushed companies to advanced computer systems for designing and scheduling integrated logistics. This trend has increased the attention of the academic community to optimization problems that arise from real-world logistic applications.

The capacitated vehicle routing problem (CVRP) [see 163, 127, 117] and container loading problem (CLP) [see 72, 20] belong to this class of problems. In short, the aim of the VRP is to design a path for each vehicle such that all customers are visited once and the total distance is minimized. In the CLP the goal is to load a container with boxes, such that the loaded volume is maximized.

Gendreau et al. [91] introduced the *three-dimensional loading capacitated vehicle routing problem* (3L-CVRP), which is the union of VRP and CLP. In this case, the goal is to plan not only the optimal vehicle routes, but also the loading of cargo within them. The 3L-CVRP belongs to a new class of problems called *routing problems with loading constraints*, which has recently been introduced in literature [107].

In spite of conceptual simplicity of these issues, they are quite difficult to solve in practice because real-world problems require to model complex constraints and to solve quickly large-scale cases.

In fact, modeling a problem requires first a careful analysis of inputs, in order to understand what is really crucial for the description of the problem; then it is necessary to make a clear distinction between the constraints of the problem, the properties that a solution must have and the objective to optimize. The difference between these three notions can sometimes be very subtle and it can depend on how the problem is modeled. A sharp and efficient modeling of the problem is the prerequisite for subsequent good performance of the algorithm. In case of real-world problems, this activity can be a very hard task that requires expertise and an intense collaboration between algorithm developers and customers.

Another difficulty in solving this kind of problems arises from the fact that the number of possible solutions from which the best one needs to be selected grows exponentially with the size of the problem instances.

For these reasons, it is often preferred to use approximate algorithms, which are able to obtain sub-optimal solutions in a reasonable time, but without any performance guarantee. In addition, they generally are flexible enough to handle additional side constraints.

Among all the approximate algorithms, metaheuristics are among the most promising ones because of their domain-independent quality. A metaheuristic can be defined as an high-level strategy which typically guides problem-specific heuristics or iterative improvement algorithms, to increase their performance.

The *stochastic local search algorithms* (SLS) belong to this algorithm class. They explore the space of candidate solutions in a not systematic way, but rather with a trial and error approach. They consist essentially in constructing one or more initial candidate solutions and trying to improve them by local changes defined by an opportune neighborhood relation between solutions. The stochastic element of these algorithms is the chance to choose randomly the move to perform.

This work aims to study, develop and validate models for solving optimization real-world problems in integrated logistics.

1.2 Objectives and contributions

The objective of this work is to develop models and metaheuristic search methods to solve complex real-world problems in the domain of computational logistics.

Our first goal is therefore to extend the problem formulations available in the literature in the routing and packing areas, in order to include several constraints that arise in practical situations. In fact, the gap between problem models presented in literature and those that occur in the real-world situations can be large. With this aim, we revised and extended the CVRP, the CLP and the 3L-CVRP.

The classical CVRP has been extended in order to deal with case of a heterogeneous fleet, a multi-day planning horizon, a complex carrier-dependent cost function for vehicles, and the possibility of leaving orders unscheduled.

Starting from the CLP, we define a new problem which considers multiple containers of different dimensions, box rotations and bearable weight of items. In addition, our problem manages the case of boxes which must be delivered in different places (*multi-drop*), thus setting additional constraints on the order of loading the boxes in the container.

We finally address the problem which is a combination of the two previous. Combining two difficult problems leads to a considerable increase of difficulty, but on the other hand it allows to obtain a better solution of the corresponding logistic target. Respect to the 3L-CVRP we redefine the important notions of stability of the cargo, fragility of items, loading and unloading policies. We also consider the case of split the demand of a customer among multiple vehicles.

Our second goal is to design effective metaheuristic approaches that consider and solve the whole problems without any simplification. We investigate different LS algorithms based on the combination of different neighborhoods and techniques. In particular we experimented tabu search (TS) and simulated annealing (SA) algorithms and the outcome is that our solvers are able to solve complex problems and also to compete fairly well with state-of-the-art solvers developed *ad hoc* for the simpler problem. In addition, we made available to the community a set of challenging real-world instances provided by industrial partners, that could potentially become a benchmark set for future researches.

1.3 Scientific publications connected with the thesis

Several chapters presented in this thesis are based on papers we have published or submitted for publication to conferences or journals.

The design and development of a tabu search algorithm for solving the vehicle routing problem described in Chapter 5, is based on the following two papers:

- S. Ceschia and A. Schaerf. (2009). Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. In *4th Multidisciplinary*

International Conference on Scheduling: Theory and Applications (MISTA 2009), pages 594–605, Dublin, Ireland.

- S. Ceschia, L. Di Gaspero and A. Schaerf, A. (2011). Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. *Journal of Scheduling*, 14(6):601–615.

The problem definition and the solution approach adopted for multiple container loading problem given in Chapter 6, is published in:

- S. Ceschia and A. Schaerf. (2009). Local search techniques for 3-dimensional packing. In *8th Metaheuristic International Conference (MIC 2009)*, Hamburg, Germany.
- S. Ceschia and A. Schaerf. (2011). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, pages 1–20. Online first. doi:10.1007/s10732-011-9162-6.

An article on the simulated annealing algorithm developed for the problem described in Chapter 7 which combines routing and packing aspects, is submitted to a journal. The preliminary work on this topic has been the subject of a short conference article.

- S. Ceschia, A. Schaerf and T. Stützle. (2011). Local search for a routing-packing problem. In *9th Metaheuristics International Conference (MIC 2011)*, Udine, Italy.
- S. Ceschia, A. Schaerf and T. Stützle. (2011). Local search for a routing-packing problem. Submitted for publication on a journal.

In the last chapter, we give an outline of our works on timetabling and healthcare which do not deal with logistics, but that are equally included in this thesis because they share the same techniques. The publications connected with these issues are:

- S. Ceschia, L. Di Gaspero and A. Schaerf. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, 39(7):1615–1624.
- S. Ceschia and A. Schaerf. (2011a). Local search and lower bounds for the patient admission scheduling problem. *Computers & Operations Research*, 30:1452–1463.

1.4 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 describes the background and it is divided into three parts. The first part introduces combinatorial optimization problems. The second part provides the basic concepts of local search algorithms and gives the details of our implementation of tabu search and simulated annealing. In the last part the experimental environment is reported and we list the software tools used to design, analyze and configure our solvers.

The techniques described in Chapter 2 serve as a basis for designing effective algorithms for the problems we addressed in the following chapters.

Chapter 3 provides a survey of some vehicle routing problems, whose review is preparatory the fully understand the problems subsequently tackled. For each problem, we give a mathematical formulation and we cover exact methods, heuristics and metaheuristics proposed in the literature.

The same has been done in Chapter 4, where we investigate different three dimensional packing problems and the solution approaches presented in literature.

Chapter 5 introduces the *vehicle routing problems with time windows and carrier-dependent costs* (VRPTWCDC), a real-world routing problem which takes into account a heterogeneous fleet, a multi-day planning horizon, a complex cost function for vehicles, and the possibility of not visiting all customers. We develop a TS algorithm and we carry out the experimental analysis of different neighborhood relations on a set of real-world instances. We also test the proposed solver on public benchmarks of the *vehicle routing problem with private fleet and common carrier* (VRPPC) [23].

Chapter 6 focuses on the *multiple container loading problem with bearing weights and multi-drop* (MCLPBWMD), which is a variant of the container loading problems (CLP). It considers several practical features, such as multiple containers, box rotations, and bearable weight and the possibility that the boxes must be delivered in different places (*multi-drop*). We engineer a hybrid solution approach which combines local search to a specific heuristic procedure which works on the actual arrangement of boxes into containers. The solver has been tested on both new instances and benchmarks of the CLP.

Chapter 7 is concerned with an integrated real-world problem in logistics (RPP), which can be seen as a complex variant of the *three-dimensional loading capacitated vehicle routing problem* (3L-CVRP). It includes many several features, some of which are described in our works on VRPPC (Chapter 5) and MCLPBWMD (Chapter 6). The chapter follows the same structure of the previous ones by giving the problem formulation and the solution approach; then the solver is tested and analyzed on new real-world instances, and finally some experimental results on public benchmarks are shown to asses the quality of our approach compared to other techniques.

An outline of the thesis is shown in Figure 1.1: the CVRP and 3D-PP blocks are enclosed in a double rectangle because they represent a class of problems, whereas the others are extensions or variants of these two original ones. The figure also highlights the relationship between the problems considered: the VRPTWCDC and MCLPBWMD are variants of the CVRP and 3D-PP respectively. The 3L-CVRP is a union of a CVRP and a 3D-BPP. Finally, the RPP redefine some aspects of the 3L-CVRP, and it can be considered as the combination of the VRPTWCDC and the MCLPBWMD.

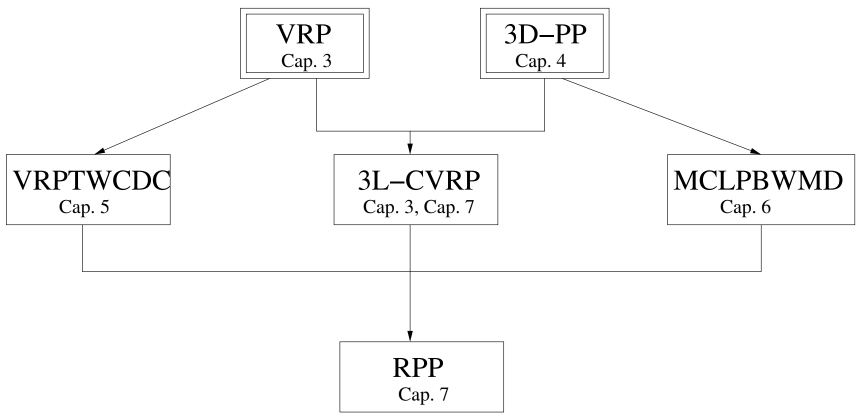


Figure 1.1: Outline of the thesis.

Chapter 8 summarily describe our research on other problems that do not concern the logistic field but share the same techniques, thus are not included in the core part of this thesis. In particular we deal with the *post enrollment course timetabling problem* (PE-CTT) and the *patient admission scheduling* (PAS).

The last chapter concludes the thesis with a summary of the main contributions followed by directions for future research.

2 Background

2.1 Combinatorial optimization problems

Generally speaking a problem can be defined as “a question for which you have to find the right answer, using mathematics or careful thought”. In mathematics, when the set of possible answers (or solutions) is finite, the problem is called combinatorial. Indeed, a combinatorial problem consists of finding a solution, among the discrete set of all possible ones, which satisfies certain constraints.

Depending on the target, combinatorial problems can be divided in three main classes. The goal of a *decision problem* is to answer a question whether or not a solution exists. *Search problems* aim to find any solution that satisfies the imposed constraints. Finally, for *optimization problems*, a criterion which assesses the quality of solutions is defined, and the objective is to find the solution that satisfies all the constraints and optimizes this quality criterion. A combinatorial optimization problem can be either a maximization or a minimization problem and, more formally, it consists of:

- a set of instances $\mathcal{I}$ of the problem Π,
- a finite set of possible solutions $\mathcal{S}$ and a finite set of feasible candidate solutions $\mathcal{F} \subseteq \mathcal{S}$, for each instance $i \in \mathcal{I}$,
- an objective function $F(i, s)$ that assigns to each solution $s \in \mathcal{S}$ of the instance $i \in \mathcal{I}$ a value called *solution value* or objective function value of s.

The issue is to find an optimal solution, that is a solution $s^* \in \mathcal{F}$ such that $F(s^*) \leq F(s)$ for all $s \in \mathcal{F}$ for a minimization problem, or $F(s^*) \geq F(s)$ for a maximization one.

The objective function is often expresses in terms of constraints (*soft constraints*), which capture the optimization goal and can not be all satisfied simultaneously. Therefore the constraints imposed to a problem can be of two types:

- *Soft constraints* can be violated but a penalty is added to the objective function in order to deteriorating the solution quality.
- *Hard constraints* individuate features that a final solution must have. They must be always satisfied and they are responsible for restricting the set of candidate solutions $\mathcal{S}$ to the set of feasible solutions $\mathcal{F}$.

It has been demonstrated that most of the combinatorial optimization problems of practical interest belong to the class of NP-hard problems, therefore (unless P=NP), they can not be solved in polynomial time. For this reason, there is much interest in approximate algorithm that are able to obtain near-optimal solutions in a reasonable computation time.

2.2 Local search algorithms

Local search methods (LS) are approximate algorithms that do not guarantee to find a feasible (optimal) solution, neither provide any provable solution quality and provable run time bounds. Indeed, they are defined *incomplete* or *non-exhaustive* because the space of the candidate solutions $\mathcal{F}$ is traversed in a non systematic manner, but moving from the current solution to a neighboring one based on the local knowledge [105].

The fundamental idea behind this approach is to iteratively generate and evaluate candidate solutions; in case of optimization problems the evaluation of a solution means to compute the value of the respective objective function. New solutions are generated by performing local changes, which are defined by an opportune neighborhood, on the current solution. If the algorithm makes use of randomized choices in this process, it belongs to the family of *stochastic local search algorithms* (SLS).

The *local* aspect of LS is in the fact that at each step of the iterative process decisions are made on the basis of a limited amount of information that covers only solutions partially different from the current one.

2.2.1 Basic concepts of local search

Given a combinatorial optimization problem Π, a LS algorithm for solving an instance $i \in \mathcal{I}$ of the problem can be defined by the following entities:

Search Space: the search space $\mathcal{S}(i)$ of an instance $i \in \mathcal{I}$ is the finite set of candidate solutions $s \in \mathcal{S}$, while $\mathcal{F}(i) \subseteq \mathcal{S}(i)$ is the set of feasible solutions.

Neighborhood Relation: given a solution $s \in \mathcal{S}$, $\mathcal{N}(s) \subseteq \mathcal{S}$ is the set of neighboring solutions of s. $\mathcal{N}(s)$ is also called the *neighborhood* of s.

Cost Function: a cost function $F(s)$ is a relation that associates to each solution $s \in \mathcal{S}$ a positive value, which rates the solution and assesses its quality.

The neighborhood is usually implicitly defined by a move which partially modifies the current solution and establishes the possible transitions between it and its neighbors.

The cost function is used to evaluate the neighbors and thus to drive the search toward promising areas of the solution space. If soft constraints are imposed, the cost function is defined as the weighted sum of the value of the objective function and the distance to feasibility (which accounts for the hard constraints). In order to favor feasibility over optimality, an high weight is usually imposed to hard constraints.

Based on this definition, Figure 2.1 shows the algorithm outlines of a generic LS procedure [62].

The choice of the initial solution, the stop criterion, the condition of selection and acceptance of a move, depend on the specific LS technique.

The initial solution can be built by an *ad hoc* greedy algorithm or it can be randomly generated. Some of most common stop criteria are based on the specific characteristics of a solution, on a maximum number of iterations, or on a timeout. The selection of a move can be done at random, or looking for the best or the first improving move in the neighborhood. Finally, a move can be accepted only if it improves the current solution or also if it worsening, under some condition.

LS algorithms include constructive heuristics, iterative improvement heuristics and metaheuristics. Constructive methods generate candidate solutions by iteratively extending partial solutions. Iterative improvement techniques try to improve the current solution by performing local changes. Metaheuristics are high level search strategies that are domain independent.

Metaheuristics can be defined as general-purpose algorithms that are used to guide the underlying problem specific heuristics (typically iterative improvement algorithms). Some of the most popular metaheuristics include *hill climbing* (HC), *simulated annealing* (SA) [115], *tabu search* (TS) [96, 97], *iterative local search* (ILS) [125], *greedy randomized adaptive search* procedures (GRASP) [83], *ant colony optimization* (ACO) [68] and *evolutionary algorithms* (EA) [8].

```
procedure $LSProcedure$(SearchSpace $\mathcal{S}$, Neighborhood $\mathcal{N}$, CostFunction $F$)
begin
  $i := 0$;
  $s_0 := InitialSolution(\mathcal{S})$;
  while not $(StopCriterion(s_i, i))$ do
  begin
    $m := SelectMove(s_i, F, \mathcal{N})$;
    if $(AcceptableMove(m, s_i, F))$
    then  $s_{i+1} := s_i \odot m$;
    else  $s_{i+1} := s_i$;
    $i := i + 1$;
  end;
  return $s_i$;
end;
```

Figure 2.1: A general local search procedure

In this thesis, we focus on simulated annealing and tabu search algorithms, therefore in the following section only these techniques are described in detail; we refer the reader to [1, 105] for a complete review of the other LS algorithms.

2.2.2 Simulated annealing

The origin of simulated annealing (SA) lies in the physical annealing process [115, 169]. In the literature, many variants of SA have been proposed [see, e.g., 1, 105]. The version used throughout this thesis, which is shown in Figure 2.3 as a flow chart and in Figure 2.2 as pseudocode, is the one with probabilistic acceptance and geometric cooling.

In detail, at each iteration of the search process a random neighbor is selected. The move is performed either if it is an improving one or according to an exponential time-decreasing probability. If the cost of the move is $\Delta F > 0$, the move is accepted with probability $e^{-\Delta F/T}$, where T is a time-decreasing parameter called *temperature*. At each temperature level a number N_σ of neighbors of the current solution is sampled and the new solution is accepted according the above mentioned probability distribution. The value of T is modified using a *geometric* schedule, i.e., $T_{i+1} = \beta \cdot T_i$, in which the parameter $\beta < 1$ is called the *cooling rate*. The search starts at temperature T_0 and stops when it reaches T_{min}.

The procedure shown in Figure 2.3 and Figure 2.2 has four parameters: start temperature T_0, stop temperature T_{min}, cooling rate β, and number of neighbors sampled at each temperature N_σ. The search for adequate parameter values has been the subject of many practical and theoretical studies over the years [see, e.g., 167, 111, 112].

Firstly, the initial temperature T_0 can be set based on trial runs. In this case, an initial probability of acceptance is defined and T_0 is set in proportion to the maximal difference in cost between any two neighboring solutions, in order to find approximately the fraction of accepted moves during the initial stage.

The number of neighbors N_σ sampled at each temperature can be fixed depending on the size of the neighborhoods or the time granted, or it can vary depending on the current temperature T.

Concerning the way to update the temperature, i.e., the cooling schedule, we can basically distinguish between static and dynamic schedules: In the first case β is fixed, whereas in the latter one it is adaptively changed during execution of the algorithm. Interested readers can refer to [166, 162] for further information.

Finally, the final value of the temperature T_{min} that determines the stop of the search process can be fixed at some small value, which may be related to T_0 or to the maximum cost difference between two neighboring solutions [1]. Other stopping criteria can be used to state if the system

```
procedure SAProcedure(SearchSpace $\mathcal{S}$, Neighborhood $\mathcal{N}$, CostFunction $F$, $T_0$, $T_{min}$, $\beta$, $N_\sigma$)
begin
    $T := T_0$;
    $s := RandomState(\mathcal{S})$;
    $s_{best} := s$;
    while $(T \geq T_{min})$ do
    begin
        $n := 0$;
        while $(n < N_\sigma)$ do
        begin
            $m := RandomMove(s, \mathcal{N})$;
            $\Delta F := F(s \oplus m) - F(s)$;
            if $(\Delta F \leq 0)$
            then
                $s := Apply(s, m)$;
                if $(F(s) < F(s_{best}))$
                then $s_{best} := s$;
                end;
            else
                $r := Random(0, 1)$;
                if $(r < e^{\frac{-\Delta F}{T}})$
                then $s := Apply(s, m)$;
                end;
            end;
            $n := n + 1$;
        end;
        $T := T \cdot \beta$;
    end;
    return $s_{best}$;
end;
```

Figure 2.2: The pseudocode of the SA procedure

is *frozen*, based on the improvement of the cost function in a given interval of iterations.

2.2.3 Tabu search

The key idea of tabu search (TS) [96, 97] is to use the memory of the past search to escape from local minima. In its simplest version, at each step of the search process a subset of the neighborhood is explored and the neighbor that gives the minimum cost value becomes the new solution independently of the fact that its cost value is better or worse than the current one.

The subset is induced by the *tabu list* (*tl*), i.e. a list of the moves recently performed, whose inverses are currently forbidden and thus excluded from the exploration. In many cases, the inverse is not a single move, but rather a set of moves determined by the values of a collection of attributes that are considered tabu. The tabu mechanism is introduced to prevent local search to return to recently visited solutions and to avoid cycling. Nevertheless, the drawback of this short-term memory approach is that it can also forbid movements towards attractive, unvisited candidate solutions. For this reason, an *aspiration criterion*, which permits to override the tabu status of a move, is allowed under some conditions. The search process usually stops after ii_{max} iterations without an improvement. Figure 2.5 and Figure 2.4 display the flow chart and the pseudo code of our implementation of the TS algorithm.

One of the key issues of TS is to define the *tabu tenure* (*tt*), that is the number of iterations that a move should be considered tabu or, in other words, the length of the tabu list. The basic algorithm is based on a fixed-length tabu list, however in literature different improvements have been proposed which employs a tabu list of variable length.

We make use of the *robust tabu search* scheme, as it is called by Taillard [156], in which the length of the tabu list is *dynamic*. This is obtained by assigning at random to each performed move the number of iterations in which it will remain in the tabu list. In detail, we set two values

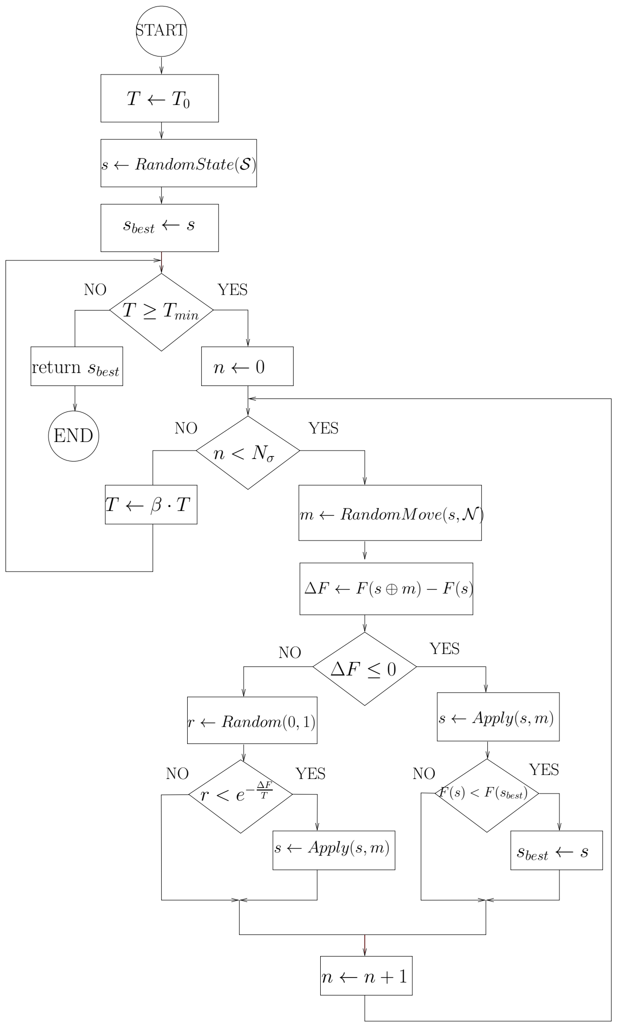

START
$T \leftarrow T_0$
$s \leftarrow RandomState(\mathcal{S})$
$s_{best} \leftarrow s$
NO
$T \geq T_{min}$
YES
return s_{best}
$n \leftarrow 0$
END
NO
$n < N_\sigma$
YES
$T \leftarrow \beta \cdot T$
$m \leftarrow RandomMove(s, \mathcal{N})$
$\Delta F \leftarrow F(s \oplus m) - F(s)$
NO
$\Delta F \leq 0$
YES
$r \leftarrow Random(0,1)$
$s \leftarrow Apply(s,m)$
NO
$r < e^{-\frac{\Delta F}{T}}$
YES
NO
$F(s) < F(s_{best})$
YES
$s \leftarrow Apply(s,m)$
$s_{best} \leftarrow s$
$n \leftarrow n+1$

Figure 2.3: Flow chart of the SA procedure

tt and δ_{tt} and we assign to each accepted move a tabu tenure randomly selected between $tt - \delta_{tt}$ and $tt + \delta_{tt}$.

The tabu status of a move can be overruled by the aspiration criterion which makes a move acceptable even if it is tabu. In this thesis, we use a basic aspiration criterion which states that a move is accepted if it improves on the current best solution. This criterion is called *NB* by Hvattum [106], who claims that it is a "safe choice" that works well in general cases.

The procedure to select the best move in the neighborhood is quite sophisticated, therefore we have decided to show in Figure 2.7 and Figure 2.6 both the flow chart and the pseudo code of our implementation. Starting from the first move, the whole neighborhood is explored. If the cost of a neighbor solution is less than the current one and the move is not prohibited, the current move becomes the best one. A move is prohibited if it does not satisfy the aspiration criterion and it is not tabu. If the value of the cost function is equal than the current best solution and the move is not prohibited (or all moves are prohibited), the best move is updated with a probability equal to $1/(1 + number\_of\_bests)$. In the other cases, when all moves are tabu and no aspiration applies, the TS algorithm executes the best of all the moves, thus ignoring the tabu status.

```
procedure TSProcedure(SearchSpace $\mathcal{S}$, Neighborhood $\mathcal{N}$, CostFunction $F$, $ii_{max}$, $tt$, $\delta_{tt}$)
begin
    $ii := 0$;
    $tl := \{\}$;
    $s := RandomState(\mathcal{S})$;
    $s_{best} := s$;
    while ($ii \leq ii_{max}$) do
    begin
        $n := 0$;
        $m := BestMove(s, \mathcal{N})$;
        $s := Apply(s, m)$;
        if ($F(s) < F(s_{best})$)
        then
            $s_{best} := s$;
            $ii := 0$;
        else
            $ii := ii + 1$;
        end;
        $tt_m := Random(tt - \delta_{tt}, tt + \delta_{tt})$;
        $tl := UpdateTabuList(tl, m, tt_m)$;
    end;
    return $s_{best}$;
end;
```

Figure 2.4: The pseudocode of the TS procedure

2.3 Software tools and environment

All the source code of the projects hereinafter described is written in *C++* and it is compiled using the GNU C/C++ compiler, v. 4.4.3. All experiments have been performed on a 2.66Ghz quad-core PC with 4 GB RAM, running Ubuntu Linux x86_64 (rel. 10.04).

The software is based on the framework EasyLocal++ [63]. EasyLocal++ is an object-oriented framework that can be used as a general tool for the development and the analysis of LS algorithms in C++. The abstract classes that compose the framework specify and implement the invariant part of the algorithm, and are meant to be specialized by concrete classes that supply the problem-dependent part. It supports the design of combinations of basic techniques and/or neighborhood structures.

The parameters settings have been configured resorting the *nearly orthogonal latin hypercubes* (NOLH) spreadsheet made available by Sanchez [149] and a new implementation of *iterated F-race* [17] by López-Ibáñez et al. [124].

As an alternative to the common full factorial design of experiments, for which the number of runs increases dramatically as the number of factors increase, the NOLH algorithm belongs to the family of response surface methods, that allow us to fill the parameters' space using much less configurations [51]. It constructs latin hypercubes that have good space-filling and are nearly orthogonal. Given these properties, the NOLH algorithm is used as an experimental design that allows to screen a large number of parameters configuration identifying a modest number of dominant factors.

The main purpose of *iterated F-race* (irace) is to automatically configure optimization algorithms by finding the most appropriate settings.The program *irace* implements the iterated racing procedure, which is an extension of the F-race procedure proposed by Birattari [15]. Indeed, the F-race is based on racing and Friedman's non-parametric two-way analysis of variance by ranks. This was improved by performing several iterations of F-race that successively refine a probabilistic model of the parameter space.

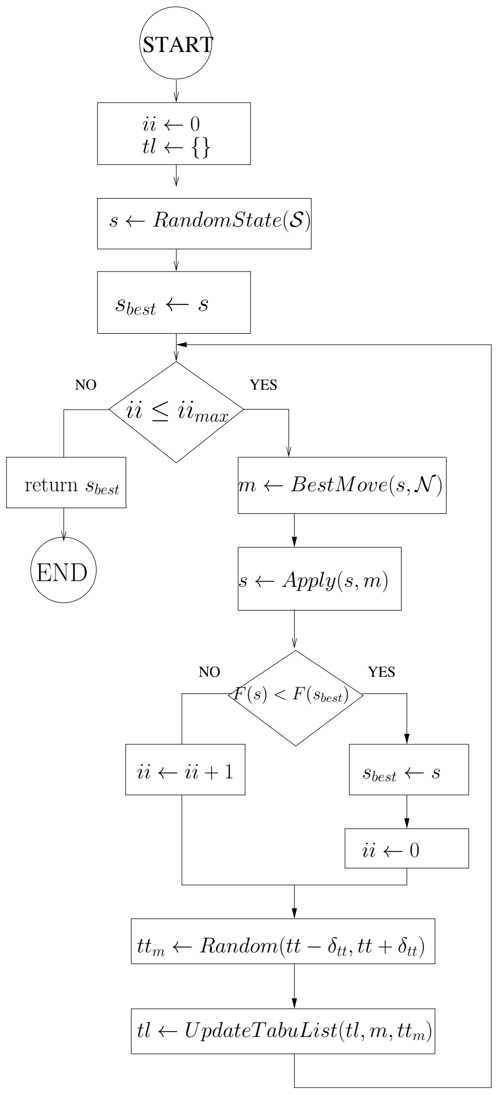

START
$ii \leftarrow 0$
$tl \leftarrow \{\}$
$s \leftarrow RandomState(\mathcal{S})$
$s_{best} \leftarrow s$
NO
YES
$ii \leq ii_{max}$
return s_{best}
END
$m \leftarrow BestMove(s, \mathcal{N})$
$s \leftarrow Apply(s, m)$
NO
YES
$F(s) < F(s_{best})$
$ii \leftarrow ii + 1$
$s_{best} \leftarrow s$
$ii \leftarrow 0$
$tt_m \leftarrow Random(tt - \delta_{tt}, tt + \delta_{tt})$
$tl \leftarrow UpdateTabuList(tl, m, tt_m)$

Figure 2.5: Flow chart of the TS procedure

```
procedure $BestMove$(State $s$, Neighborhood $\mathcal{N}$, CostFunction $F$, $tl$)
begin
    $m := FirstMove(s, \mathcal{N})$;
    $m_{best} := m$;
    $s_{best} := s \oplus m_{best}$;
    $all\_moves\_prohibited := ProhibitedMove(s, m, tl)$;
    $m := NextMove(s, \mathcal{N})$;
    while ($m \neq FirstMove(s, \mathcal{N})$) do
    begin
        $s := s \oplus m$
        if ($F(s) < F(s_{best})$)
        then
            if (!$ProhibitedMove(s, m, tl)$)
            then
                $s_{best} := s$;
                $m_{best} := m$;
                $number\_of\_bests := 1$;
                $all\_moves\_prohibited := false$;
            else if ($all\_moves\_prohibited$)
                then
                $m_{best} := m$;
                $number\_of\_bests := 1$;
                $all\_moves\_prohibited := false$;
                end;
            end;
        else if ($F(s) = F(s_{best})$)
            then if (!$ProhibitedMove(s, m, tl)$)
                then if ($all\_moves\_prohibited$)
                    then
                    $m_{best} := m$;
                    $number\_of\_bests := 1$;
                    $all\_moves\_prohibited := false$;
                end;
                else if ($Random(0, number\_of\_bests) = 0$)
                    then $m_{best} := m$;
                    end;
                $number\_of\_bests := number\_of\_bests + 1$;
                end;
            else if ($all\_moves\_prohibited$)
                then if ($Random(0, number\_of\_bests) = 0$)
                    then $m_{best} := m$;
                    end;
                $number\_of\_bests := number\_of\_bests + 1$;
                end;
            end;
            else if ($all\_moves\_prohibited$)
                then if(!$ProhibitedMove(s, m, tl)$)
                    $m_{best} := m$;
                    $s_{best} := s$;
                    $number\_of\_bests := number\_of\_bests + 1$;
                    $all\_moves\_prohibited := false$;
                end;
            end;
        end;
    $m \leftarrow NextMove(s, \mathcal{N})$;
    end;
    return $m_{best}$;
end;
```

Figure 2.6: The pseudocode of the *BestMove* procedure of TS

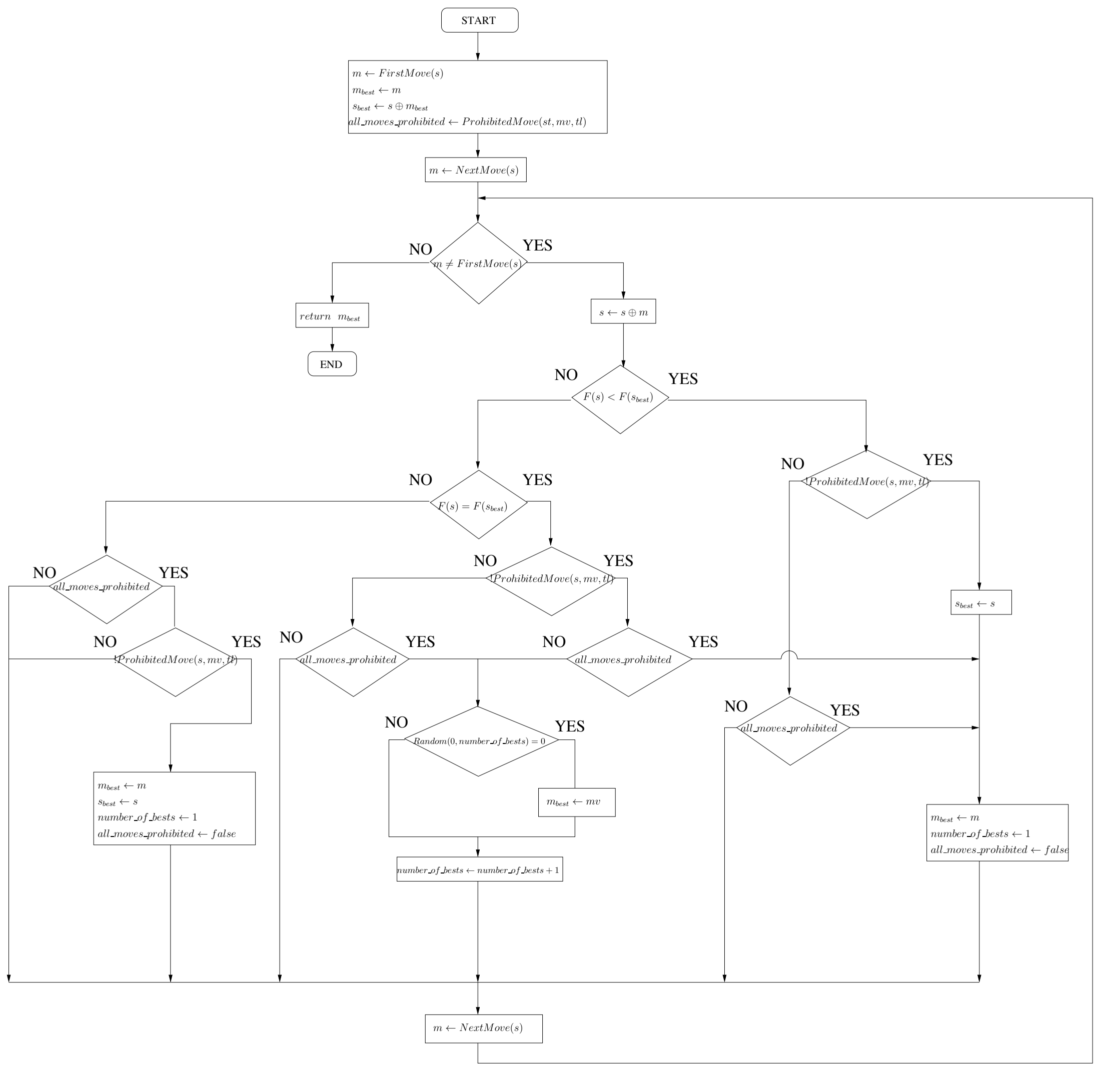

START
$m \leftarrow FirstMove(s)$
$m_{best} \leftarrow m$
$s_{best} \leftarrow s \oplus m_{best}$
$all\_moves\_prohibited \leftarrow ProhibitedMove(st, mv, tl)$
$m \leftarrow NextMove(s)$
NO
YES
$m \neq FirstMove(s)$
$return\ \ m_{best}$
END
$s \leftarrow s \oplus m$
NO
YES
$F(s) < F(s_{best})$
NO
YES
$F(s) = F(s_{best})$
NO
YES
$!ProhibitedMove(s, mv, tl)$
$s_{best} \leftarrow s$
NO
YES
$all\_moves\_prohibited$
NO
YES
$!ProhibitedMove(s, mv, tl)$
NO
YES
$!ProhibitedMove(s, mv, tl)$
NO
YES
$all\_moves\_prohibited$
NO
YES
$all\_moves\_prohibited$
NO
YES
$Random(0, number\_of\_bests) = 0$
$m_{best} \leftarrow mv$
NO
YES
$all\_moves\_prohibited$
$m_{best} \leftarrow m$
$s_{best} \leftarrow s$
$number\_of\_bests \leftarrow 1$
$all\_moves\_prohibited \leftarrow false$
$number\_of\_bests \leftarrow number\_of\_bests + 1$
$m_{best} \leftarrow m$
$number\_of\_bests \leftarrow 1$
$all\_moves\_prohibited \leftarrow false$
$m \leftarrow NextMove(s)$

Figure 2.7: Flow chart of the procedure for the selection of the best move in the TS algorithm.

I

State of the art

3

Vehicle Routing Problems: a review

3.1 Introduction

The *vehicle routing problem* (VRP) was first introduced by Dantzig and Ramser [56] in what they called the *truck dispatching problem.* It was formulated as a branch of the *traveling salesman problem* (TSP) with multiple vehicles and routes. Subsequently, many other extensions that include time windows, different depots, pick-up and delivery options, heterogeneous fleet and periodic routing have been proposed in literature.

In the following sections we describe some of these problem variants whose knowledge and comprehension is necessary to fully understand the complex real-world problem described in Chapter 5.

In Section 3.2, the different VRPs' are formally defined, while in Section 3.3 there is a brief overview of the solution approaches proposed in the literature.

3.2 Problems' definition and notation

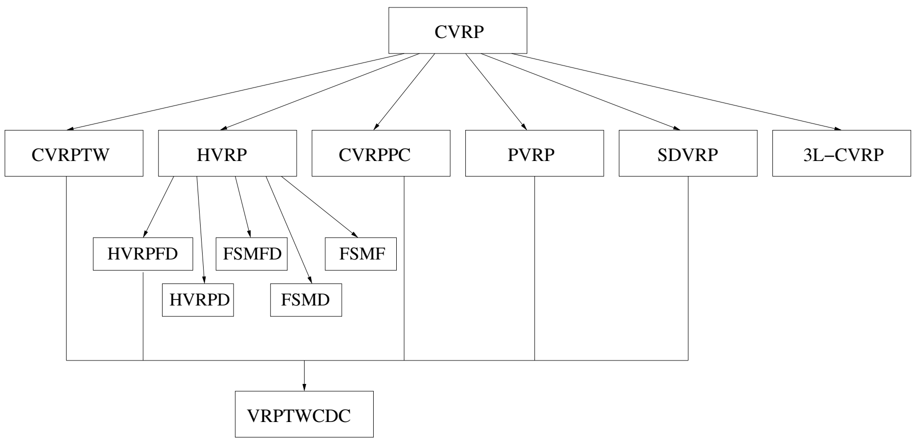


Figure 3.1: A classification of some vehicle routing problems and their interconnections

Figure 3.1 summarizes the problem variants described in the following sections and their interconnection. An arrow from problem Π_1 to Π_2 means that Π_2 is an extension of Π_1. As shown in the picture, the problem called VRPTWCDC, which will be fully defined in Chapter 6, is a further extension that takes some features from different VRPs.

3.2.1 Capacitated vehicle routing problem

The solution of a *capacitated vehicle routing problem* (CVRP) calls for the determination of a plan of routes, one for each vehicle, such that all customers are served and the total traveling costs are minimized. The vehicles are all identical and they must leave and return to a single central depot. Each customer has a demand, which is the weight of the goods to be delivered, that may not be split. There are only capacity restriction, i.e. the total demand of a route must not exceed the vehicle capacity.

Figure 3.2 gives an example of solution of a CVRP with twelve customers, three vehicles and a single central depot.

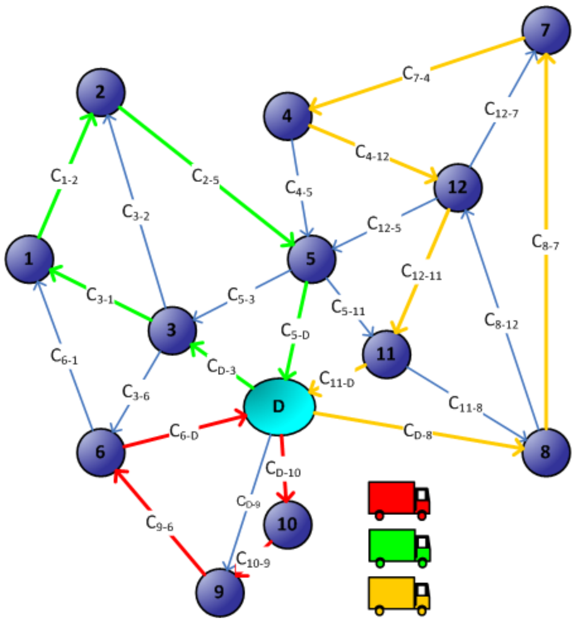


Figure 3.2: A solution of an instance of CVRP.

The CVRP is particularly suitable to be described using the terminology of graph theory. We call $G = (\mathcal{V}, \mathcal{A})$ the graph where $\mathcal{V}$ is the set of vertex (or nodes), each one corresponding to a customer, and $\mathcal{A}$ is the set of arcs that connect nodes. The graph is complete such that for each pair of customer i and j there exists a corresponding arc (i, j). The depot is considered as a special customer, and it is usually associated with vertex 0.

A cost c_{ij}, which represents the routing cost for traveling from node i to node j, is associated to each arc (i, j). When the routing cost matrix C is symmetric ($c_{ij} = c_{ji}$ for all arcs), the problem becomes a *symmetric vehicle routing problem*. In most of the practical cases, the C matrix satisfies the *triangle inequality* ($c_{ij} + c_{jk} \geq c_{ik}$ for all i, j, k), meaning that is more convenient to go directly from a node to another, rather than deviate and pass through intermediate nodes.

A positive demand q_i is associated to each customer i ($i = 1, \ldots, n$), while the depot has a dummy demand equal to zero ($q_0 = 0$). The fleet $\mathcal{K}$ is composed by all identical vehicles, each one with capacity W. We assume that the demand of each customer is lower than the capacity of vehicles ($q_i \leq W$ for all customers).

In literature, many different mathematical formulations have been proposed to model the CVRP. We use a three-index formulation already presented in [163] belonging to the family of the *vehicle flows formulations*, which is usually more adequate to express complex constraints that are common in real world oriented problems.

It uses two integer variables: x_{ijk} that expresses the number of times an arc (i, j) is traversed by a vehicle k, and y_{ik} that indicates if the customer i is visited by vehicle k. The objective function is:

$$\text{minimize} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} \sum_{k \in \mathcal{K}} x_{ijk} \tag{3.1}$$

subject to:

$$\sum_{k \in \mathcal{K}} y_{ik} = 1, \qquad \forall\ i \in \mathcal{V} \setminus 0 \tag{3.2}$$

$$\sum_{k \in \mathcal{K}} y_{0k} = |\mathcal{K}|, \tag{3.3}$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik} = y_{ik}, \qquad \forall\ i \in \mathcal{V} \setminus 0, \forall\ k \in \mathcal{K} \tag{3.4}$$

$$u_{ik} - u_{jk} + W x_{ijk} \leq W - q_j \qquad \forall\ i \in \mathcal{V} \setminus 0, i \neq j, \text{such that } q_i + q_j < W, \forall\ k \in \mathcal{K} \tag{3.5}$$

$$y_{ik} \in \{0,1\} \qquad \forall\ i \in \mathcal{V}, \forall\ k \in \mathcal{K} \tag{3.6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\ i,j \in \mathcal{V}, \forall\ k \in \mathcal{K} \tag{3.7}$$

$$q_i \leq u_{ik} \leq W \qquad \forall\ i \in \mathcal{V} \setminus 0, \forall\ k \in \mathcal{K} \tag{3.8}$$

$$\tag{3.9}$$

Constraints 3.2 state that each nodes must be included in exactly one route. Equation 3.3 imposes that the depot is the starting node of each route. Constraints 3.4 establish that the same vehicle arrives at node and leaves from it. Constraints 3.5 are a generalization to the three index formulation of the *subtour elimination constraints* proposed by Miller et al. [131].

The u_{ik} variable is a continuous variable that counts the load of the vehicle k after visiting customer i. If the variable x_{ijk} is equal to zero, the constraint is always satisfied because $u_i \leq W$ and $u_j \geq q_j$ are always true for definition (3.8). On the other hand, if $x_{ijk} = 1$, the resulting constraint is $q_j \leq u_j - u_i$, which imposes both restriction on capacity and connectivity of a route.

3.2.2 Vehicle routing problem with time windows

The *vehicle routing problem with time windows* (VRPTW) is probably one of the most studied extension of the CVRP. In this problem, each customer specifies a time interval (time window) and the visit must take place during that period. Once arrived, a vehicle must remain to the customer location for a service time. The vehicle can not arrive at a place after the latest time window, but if it arrives before the opening of the window, it is allowed to wait until the customer is ready. A dummy service time equal to zero is assigned to the depot and it is usually assumed that all vehicles leave the depot at time instant 0. The objective is to find a collection of routes that minimize the routing costs (or the duration of the routes) while satisfying all constraints about capacity of vehicles and time windows of customers.

More formally, we call $[e_i, l_i]$ the earliest and the latest time windows of customer i, s_i the service time that a vehicle must wait at customer location i and τ_{ij} the travel time needed to go from customer i to customer j. The variable t_{ik} represents the arrival time of vehicle k at location i. The objective function remains the same (3.1), but the following constraints, that guarantee a schedule feasible according to the time windows restrictions, have to be added to the previous ones (3.9):

$$x_{ijk}(t_{ik} + s_i + \tau_{ij} - t_{jk}) \leq 0, \qquad \forall\ i,j \in \mathcal{V}, \forall k \in \mathcal{K} \tag{3.10}$$

$$e_i \sum_{j \in \mathcal{V}} x_{ijk} \leq t_{ik} \leq l_i \sum_{j \in \mathcal{V}} x_{ijk} \qquad \forall\ i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{3.11}$$

$$e_i \leq t_{ik} \leq l_i \qquad \forall\ i \in \mathcal{V}, k \in \mathcal{K}, \tag{3.12}$$

$$\tag{3.13}$$

3.2.3 Heterogeneous vehicle routing problem

The *heterogeneous vehicle routing problem* (HVRP) addresses the problem of a fleet of vehicles with different capacities [99]. The vehicles are grouped in m different types ($\mathcal{M} = 1, \ldots, m$), and for each type $k \in \mathcal{M}$, m_k vehicles are available at the depot with capacity W_k. In addition, it is possible to associated a fixed cost F_k to vehicle each type, which represents the amortization cost. The objective of the problem is find a set of routes that minimize the sum of routing costs (variable and fixed), while satisfying all the capacity constraints.

As proposed by Baldacci et al. [10], the formulation 3.9 can be modified as follows to model the case of different types of vehicles:

$$\text{minimize} \sum_{j \in \mathcal{V} \setminus 0} \sum_{k \in \mathcal{M}} F_k x_{0jk} + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} \sum_{k \in \mathcal{M}} x_{ijk} \tag{3.14}$$

subject to:

$$\sum_{k \in \mathcal{M}} y_{ik} = 1, \qquad \forall\, i \in \mathcal{V} \setminus 0 \tag{3.15}$$

$$\sum_{j \in \mathcal{V} \setminus 0} x_{0jk} = m_k, \qquad \forall\, k \in \mathcal{M} \tag{3.16}$$

$$\sum_{j \in \mathcal{V}} x_{ijk} = \sum_{j \in \mathcal{V}} x_{jik} = y_{ik}, \qquad \forall\, i \in \mathcal{V} \setminus 0, \forall\, k \in \mathcal{M} \tag{3.17}$$

$$u_{ik} - u_{jk} + W_k x_{ijk} \leq W_k - q_j \qquad \forall\, i \in \mathcal{V} \setminus 0, i \neq j, \text{such that } q_i + q_j < W_k, \forall\, k \in \mathcal{M} \tag{3.18}$$

$$y_{ik} \in \{0,1\} \qquad \forall\, i \in \mathcal{V}, \forall\, k \in \mathcal{M} \tag{3.19}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{V}, \forall\, k \in \mathcal{M} \tag{3.20}$$

$$q_i \leq u_{ik} \leq W_k \qquad \forall\, i \in \mathcal{V} \setminus 0, \forall\, k \in \mathcal{M} \tag{3.21}$$

$$\tag{3.22}$$

Respect to the classical CVRP, the main difference is the cost function 3.14 that takes into account also a fixed cost of use of a vehicle. Notice that in this case $\mathcal{M}$ is not the set of all vehicle, as in CVRP, but the set of vehicle types.

Several variants of this general problem have been proposed in literature, depending on the size of the fleet (unlimited or limited) and the costs that are taken into account (the fixed cost might be considered or not, and the routing costs might depend on the vehicle type c_{ijk}). Baldacci et al. [10] proposed a possible classification, reported on Table 3.1, that summarizes the features of each problem variant:

Problem name	fleet size	fixed costs	routing costs
HVRPFD	limited	considered	dependent
HVRPD	limited	not considered	dependent
FSMFD	unlimited	considered	dependent
FSMD	unlimited	not considered	dependent
FSMF	unlimited	considered	independent

Table 3.1: Taxonomy of *vehicle routing problems with heterogeneous fleet*

where:

HVRPFD: heterogeneous vrp with fixed costs and vehicle dependent routing costs

HVRPD: heterogeneous vrp with vehicle dependent routing costs

FSMFD: fleet size and mix vrp with fixed costs and vehicle dependent routing costs

FSMD: fleet size and mix vrp with vehicle dependent routing costs

FSMF: fleet size and mix vrp with fixed costs

Recently, Liu and Shen [120] proposed the *fleet size vehicle routing problem* with *time windows* (FSMVRPTW) which considers the case of an unlimited heterogeneous fleet of vehicle that has to visit the costumers during their time windows. The objective functions sums the traveling costs and the fixed cost of vehicles.

3.2.4 Vehicle routing problem with private fleet and common carrier

The *vehicle routing problem with private fleet and common carrier* (VRPPC) describes a situation where, by hypothesis, the total demand exceeds the capacity of the internal fleet, so an external transporter is necessary (*common carrier*) [50]. In this case the problem is twofold: select customers that should be served by the external carrier and define routes of the internal fleet to serve remaining customers. In the VRPPC, the cost of serving a customer by the common carrier is fixed for each customer without referring to any routing and the fleet is homogeneous, thus $\mathcal{M}$ is the set of all identical vehicles as in Section 3.2.1.

Beside to the fixed cost of use of vehicles, the objective function includes a new component that counts the cost of the customers served by external carriers, which is totally independent of distances. The variable z_i is equal to one if the customer i is served by an external carrier, 0 otherwise. The mathematical formulation is:

$$\text{minimize} \sum_{j\in\mathcal{V}\setminus 0}\sum_{k\in\mathcal{M}} F_k x_{0jk} + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} c_{ij} \sum_{k\in\mathcal{M}} x_{ijk} + \sum_{i\in\mathcal{V}} \gamma_i z_i \tag{3.23}$$

subject to:

$$\sum_{k\in\mathcal{M}} y_{ik} + z_i = 1, \qquad \forall\, i \in \mathcal{V}\setminus 0 \tag{3.24}$$

$$\sum_{k\in\mathcal{M}} y_{0k} = |K|, \tag{3.25}$$

$$\sum_{j\in\mathcal{V}} x_{ijk} = \sum_{j\in\mathcal{V}} x_{jik} = y_{ik}, \qquad \forall\, i \in \mathcal{V}\setminus 0, \forall\, k \in \mathcal{M} \tag{3.26}$$

$$u_{ik} - u_{jk} + W x_{ijk} \leq W - q_j \qquad \forall\, i \in \mathcal{V}\setminus 0, i \neq j, \text{such that } q_i + q_j < W, \forall\, k \in K \tag{3.27}$$

$$y_{ik} \in \{0,1\} \qquad \forall\, i \in \mathcal{V}, \forall\, k \in K \tag{3.28}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{V}, \forall\, k \in K \tag{3.29}$$

$$q_i \leq u_{ik} \leq W \qquad \forall\, i \in \mathcal{V}\setminus 0, \forall\, k \in K \tag{3.30}$$

$$\tag{3.31}$$

Constraints 3.24 ensure that each customer is served either by a vehicle of the internal fleet or the external carrier. All the others constraints are the same proposed in the model 3.9 for the CVRP.

3.2.5 Period vehicle routing problem

The *period vehicle routing problem* (PVRP) [12] is a generalization of the CVRP and it can be considered a composition of a classical CVRP and an assignment problem. Indeed, the planning period is not a single day but multiple days, therefore a collection of routes have to be designed for each day of the horizon.

During the planning period, a customer has to be visited once or several times, and visits have to take place according to a set of allowed combinations of delivery days. For example, if a customer requires two visits over a horizon of 5 days, the combinations could be $Monday/Wednesday, Tuesday/Wednesday, Wednesday/Friday$.

Lets denote $\mathcal{D}$ the set of days in the the planning horizon, each customer i has associated a service frequency f_i and a set of allowable combinations of visit days $\mathcal{H}_i$. It is assumed that the demand of a customer q_i remains the same for all his deliveries.

We report the mathematical formulation of the problem proposed by Cordeau et al. [53]. The integer binary variable x_{ijkd} takes value 1 if the arc (i, j) is traversed by vehicle k on day d and the variable y_{ih} is positive if the visit combination $h \in \mathcal{H}_i$ is assigned to customer i. In addition the binary constant z_{hd} is equal to 1 if day d belongs to visit combination h, 0 otherwise. The objective function of the problem is:

$$\text{minimize} \sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} \sum_{k \in \mathcal{M}} x_{ijkd} \tag{3.32}$$

subject to:

$$\begin{aligned}
\sum_{h \in \mathcal{H}_i} y_{ih} = 1, &\quad \forall\, i \in \mathcal{V} \setminus 0 && (3.33)\\
\sum_{j \in \mathcal{V}} \sum_{k \in \mathcal{M}} x_{ijkd} - \sum_{h \in \mathcal{H}_i} z_{hd} y_{ih} = 0, &\quad \forall\, i \in \mathcal{V} \setminus 0, \forall\, d \in \mathcal{D} && (3.34)\\
\sum_{j \in \mathcal{V}} x_{ijkd} = \sum_{j \in \mathcal{V}} x_{jikd} &\quad \forall\, i \in \mathcal{V} \setminus 0, \forall\, k \in \mathcal{M}, \forall\, d \in \mathcal{D} && (3.35)\\
\sum_{i \in \mathcal{V} \setminus 0} x_{0jkd} \leq 1 &\quad \forall\, k \in \mathcal{M}, \forall\, d \in \mathcal{D} && (3.36)\\
u_{ikd} - u_{jkd} + W x_{ijkd} \leq W - q_j &\quad \forall\, i \in \mathcal{V} \setminus 0, i \neq j, && (3.37)\\
&\quad \text{such that } q_i + q_j < W, \forall\, k \in \mathcal{M}, \forall\, d \in \mathcal{D} && (3.38)\\
y_{ih} \in \{0, 1\} &\quad \forall\, i \in \mathcal{V}, \forall\, h \in \mathcal{H}_i && (3.39)\\
x_{ijkd} \in \{0, 1\} &\quad \forall\, i, j \in \mathcal{V}, \forall\, k \in \mathcal{M}, \forall\, d \in \mathcal{D} && (3.40)\\
q_i \leq u_{ikd} \leq W &\quad \forall\, i \in \mathcal{V} \setminus 0, \forall\, k \in \mathcal{M}, \forall\, d \in \mathcal{D} && (3.41)\\
& && (3.42)
\end{aligned}$$

Constraints 3.33 guarantee that only one allowed combination of days is assigned to each customer, while constraints 3.33 state that a customer has to be visited only on days corresponding to the selected combination. The other constraints are the usual ones, but int his case they must be verified for all days of the planning horizon $d = 1, \ldots |\mathcal{D}|$.

3.2.6 Split delivery vehicle routing problem

The *split delivery vehicle routing problem* (SDVRP) [70] is a variant of the capacitated CVRP, where each customer can be visited more than once and his/her demand can be greater than the capacity of a vehicle. Therefore the SDVRP can be considered a relaxed version of the CVRP, where the demand of a customer can be split between more vehicles.

As usual in the mathematical programming formulation, the integer variable x_{ijk} counts the number of times vehicle k travels from i to j, whereas the variable y_{ik} stores the quantity of the demand of i delivered by vehicle k. As presented by Archetti et al. [4] the mathematical model of the problem is:

$$\text{minimize} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} \sum_{k \in \mathcal{M}} x_{ijk} \tag{3.43}$$

subject to:

$$\sum_{k\in\mathcal{M}}\sum_{i\in\mathcal{V}} x_{ijk} \geq 1, \qquad \forall\, j \in \mathcal{V}\setminus 0 \tag{3.44}$$

$$\sum_{j\in\mathcal{V}} x_{ijk} = \sum_{j\in\mathcal{V}} x_{jik}, \qquad \forall\, i \in \mathcal{V}\setminus 0, \forall\, k \in \mathcal{M} \tag{3.45}$$

$$y_{ik} \leq q_i \sum_{j\in\mathcal{V}} x_{ijk}, \qquad \forall\, i \in \mathcal{V}\setminus 0, \forall\, k \in \mathcal{M} \tag{3.46}$$

$$\sum_{k\in\mathcal{M}} y_{ik} = q_i, \qquad \forall\, i \in \mathcal{V}\setminus 0 \tag{3.47}$$

$$\sum_{i\in\mathcal{V}} y_{ik} \leq W, \tag{3.48}$$

$$u_{ik} - u_{jk} + W x_{ijk} \leq W - y_{jk} \qquad \forall\, i \in \mathcal{V}\setminus 0, i \neq j, \text{such that } y_{ik} + q_{jk} < W, \forall\, k \in \mathcal{M} \tag{3.49}$$

$$y_{ik} \in \{0,1\} \qquad \forall\, i \in \mathcal{V}, \forall\, k \in \mathcal{M} \tag{3.50}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{V}, \forall\, k \in \mathcal{M} \tag{3.51}$$

$$1 \leq u_{ik} \leq W \qquad \forall\, i \in \mathcal{V}\setminus 0, \forall\, k \in \mathcal{M} \tag{3.52}$$

$$\tag{3.53}$$

Constraints 3.44 guarantee that each customers is visited at least once; constraints 3.45 are the flow conservation constraints; the subsequent constraints (3.46, 3.47, 3.48) regulate the allocation of the demand of a customer in different vehicles. Finally, constraints 3.49 are imposed to eliminate subtours.

3.2.7 Three-dimensional loading capacitated vehicle routing problem

The *three-dimensional loading capacitated vehicle routing problem* (3L-CVRP) [91] represents a combination of vehicle routing and three dimensional loading. It consists in finding a set of routes that satisfies the demand of all customers, minimizes the total routing cost and guarantees a feasible packing of items.

Differently from the CVRP where the only constraint is about the total weight of boxes that must not exceed the capacity of vehicles, in the 3L-CVRP the loading is explicitly taken into account. Indeed, in the 3L-CVRP the demand of a customer is not expressed as the total weight of items required, but as the set of rectangular three dimensional boxes of a given size. Some of the loading constraints, coming from the literature on 3D-BPP (Section 4.2), ensure that items no overlapping, integrity and orthogonality. Others constraints (Section 4.2.1), which arise from practical consideration in transportation, deal with rotations, fragility, stability and easy unloading of items. For sake of readability we do not report here a mathematical formulation of the problem but we refer the interested reader to the article by Moura and Oliveira [134].

We do not give here a detailed description of the problem formulation, as it will be provided in Section 7.2.1. In fact, the problem we addressed in Chapter 7 can be considered as an extension of the 3L-CVRP, and it is defined from it.

3.3 Solution techniques

The literature on the CVRP and VRPTW is very extensive, for a complete review the reader is referred to [163, 127, 117] and [34, 35], respectively. In following sections, we review only the works in the literature that deal with those variants of the CVRP described in the previous sections.

We grouped the different solution techniques in three main categories: exact approaches (including new mathematical formulations and lower bounds), *ad hoc* heuristics (constructive and improvement ones) and metaheuristics algorithms.

In Table 3.2 we report a summarizing classification of the manuscripts according to the problem treated and the solution technique adopted. The table shows that due to the difficulty of these

	exact methods	**heuristics**	**metaheuristics**
HVRP	[10, 99, 47, 176]	[99, 61, 148, 120]	[153, 155, 135, 90, 175, 159, 160, 118, 36]
VRPPC	[66]	[50, 23]	[24, 54]
PVRP	[69]	[22, 48, 157, 147, 43]	[53, 2, 102, 171]
SDVRP	[70, 109, 11, 7, 6]	[70, 46, 110]	[5]
3L-CVRP	–	–	[91, 158, 85, 173, 27, 133]
MP-VRP/PP-VRP	–	–	[67, 165, 177]

Table 3.2: References to manuscripts on VRPs grouped by problem formulation and solution technique.

variants of vehicle routing problems, heuristic (and metaheuristic) approaches are usually adopted, while exact technique are often used to provide lower bounds.

3.3.1 Exact methods

Baldacci et al. [10] have given an overview of the mathematical formulations, lower bounds and approaches used to solve **HVRP**s. To our knowledge, no exact algorithm has ever been developed for this class of problems but only different lower bounds have been proposed [99, 47, 176].

Diaby and Ramesh [66] solved exactly (for $n < 200$) the single vehicle case of the **VRPCC** formulated by Volgenant and Jonker [172].

The **SDVRP** is a relaxation of the CVRP, but nonetheless it seems to be even more difficult to solve exactly. Dror [69] introduced a mathematical formulation based on integer programming, proposed some valid inequalities and solved through a constraint relaxation branch and bound approach. Lower bounds have been proposed by Belenguer et al. [11] while Archetti et al. [7] studied the complexity of the problem for some classes of instances whose underling graph exhibit a special structure. Finally, column generation approaches have been presented by Jin et al. [109] and Archetti et al. [6].

3.3.2 Heuristics

The first study on **HVRP**, is due to Golden et al. [99] that proposed a constructive heuristics based on the *Clarke and Wright Saving Algorithm* [52], where the classical cost-saving procedure was adapted to take into account of the fixed cost of use of vehicles and the opportunity saving, that is the saving that might occur by replacing two small vehicle with a large one. Subsequently, Desrochers and Verhoog [61] extended this approach by developing a matching-based saving heuristic based where, at each iteration, the best merge between routes is selected by solving a weighted matching problem. Salhi and Rand [148] proposed a complex multi-step heuristic: an initial solution with all identical vehicles is initially built, then different improvement operators are applied.

In the first work about **HVRPTW**, Liu and Shen [120] developed a two phase algorithm: in the constructive stage, an insertion-based algorithm is used to build the routes, and in the following improvement phase, that intra and inter route perturbations are performed. Dullaert et al. [71] extended the sequential insertion algorithm used by Solomon [154].

The **VRPPC** was formally introduced by Chu [50] who solved it heuristically: he firstly applied a modified version of the classical saving procedure [52] followed by some local exchanges between routes. Bolduc et al. [23] have proposed a heuristic called *SRI* that is composed of three steps: the selection of customer served by the external carrier, the construction of the solution (routing) and the improvement, through the application of sophisticated exchanges.

The first works on the **PVRP** was carried out by Bodin et al. [22] and Russell and Igo [146] who solved heuristically a problem of waste collection. The problem was then formalized in [48].

In their two stage solution approach, firstly customers are assigned to delivery days according to meet service level requirements, then an interchange procedure is used to improve the solution. Other heuristics have been developed by Tan and Beasley [157], Russell and Gribbin [147], Chao et al. [43]. In particular, in [43] firstly visit combinations are assigned to customers by means of integer programming, then a modified cost-saving algorithm solves a VRP for each day, and finally some local improvements are performed.

Dror and Trudeau [70] introduced the **SDVRP** and proposed a heuristic algorithm; they also showed that allowing split deliveries can yield to substantial savings in the routing cost and in the number of vehicle used. Other heuristic approaches have been proposed in [46, 110].

3.3.3 Metaheuristics

The first metaheuristic approach for **HVRP** was proposed by Semet and Taillard [153] who developed a TS algorithm to solve a problem with several real-world features. Taillard [155] presented a heuristic column generation approach: a TS algorithm is used to solve a set of homogeneous VRPs, one for each type of vehicle, then the solutions obtained are selected and recombined by solving a set partitioning problem. Other TS approaches for this problem have been developed in [135, 90, 175]. All these algorithms are extensions to the heterogeneous case of the algorithm used for the classical CVRP. Tarantilis et al. [159, 160] have presented a list-based threshold accepting metaheuristic and more recently Li et al. [118] and Bräysy et al. [36] have developed a deterministic variants of SA.

Also for the **VRPPC**, the more recent works are on metaheuristics. Bolduc et al. [24] have presented a perturbation metaheuristic, called *RIP* (Randomized construction, Improvement, Perturbation), which essentially combines a descent method with diversification strategies. Recently, Côté and Potvin [54] have obtained the best known results on benchmarks using a TS approach.

Cordeau et al. [53] developed a TS algorithm to solve the **PVRP** based on the GENI heuristic [89]. Differently from all other heuristics used for this problem, infeasible solutions with respect to the capacity and duration constraints are allowed during the search, although they are strongly penalized in the cost function. A scatter search procedure has been proposed by Alegre et al. [2], while Hemmelmayr et al. [102] presented an application of a variable neighborhood search. Vidal et al. [171] designed a hybrid metaheuristic, which is mainly based on the GA paradigm, that addresses the multi-depot VRP (MD-VRP), the PVRP and the multi-depot VRP (MD-PVRP), obtaining the best known solutions for the benchmark instances of each problem class.

For the solution of the **SDVRP**, a meta-heuristic approach has been proposed by Archetti et al. [4] and subsequently used in [5] to show in which case allowing split deliveries is likely to be beneficial.

In the reminded of the section, we discuss the literature on the **3L-CVRP** more in detail, given that this problem is extensively examined in Chapter 7 and our solver is compared with most of the solution techniques here described. The literature on the 3L-CVRP shows that this problem is usually solved by means of a hierarchical solution approach where the master problem, the routing problem is solved by a metaheuristic technique, while the slave problem, the packing problem, is delegated to fast and simple packing heuristics, such as the *bottom left algorithm* [9] and the *touching perimeter algorithm* [122]. In the solution technique presented by Gendreau et al. [91], an outer TS algorithm works in the space of the routing problem moving customers from different routes, whereas an inner TS routine deals with the packing problem swapping items and iteratively invoking the loading heuristic. The search space is composed also by infeasible states, in which there are violations of the capacity of vehicles (exceeding weight) or the loading space of the vehicle is not sufficient to contain all items. Tarantilis et al. [158] implemented a hybrid TS-Guided Local Search algorithm. The TS explores the search space of the routing problem, while a bundle of six different packing heuristics are iteratively reapplied until a feasible loading is obtained. If no feasible loading is achieved, the items of the sequence are re-sorted for a maximum of three times (keeping fixed the customer sequence). In addition, as already mentioned, they introduced a new problem version called *Manual 3L-CVRP* that modifies the definition of the LIFO constraint as proposed by Gendreau et al. [91]. The ACO approach described by Fuellerer et al. [85] draws on some of the features of the Saving-based ACO [144], which was successfully used to solve the standard CVRP. To test the feasibility of a route as for loading constraints, two greedy packing heuristics are repeatedly applied and, if necessary, swaps among items of the same customer are performed. Recently, Wang et al. [173] developed a two phase TS algorithm. In the first phase they consider both hard and soft constraints, allowing the search process to go through infeasible states [as in 91]; in the second phase, they apply five different neighborhood operators (*2-opt*, *2-swap*, *move*, *crossover*, *splitting*), and only feasible solutions are generated and evaluated. In addition, the classical *bottom left algorithm* has been adapted to perform effectively also in the case of the supporting area constraint. Finally, Bortfeldt [27] presented a hybrid approach that uses a TS algorithm for routing and a tree search algorithm for packing. In the TS algorithm, he distinguishes two phases by applying different neighborhood structures and different quality measures: during the first phase, the aim is to reduce the number of vehicles, while in the second phase the goal is to reduce the total travel distance. The loading of boxes is carried out by means of a tree search procedure that uses a complex system for ranking boxes not already packed and for filtering the potential placement points, which are generated using the *extreme point-based heuristic* [55].

Moura and Oliveira [133] studied a different integrated vehicle routing and container loading problem. Their problem takes into account several additional real world features, such as customers' time windows, service times, strongly and weakly heterogeneous cargo, cargo's orientation, LIFO constraint and load stability. The objective function is a weighted sum of three components: the number of vehicles, the total travel time, and the wasted space in vehicles. They propose two solution approaches: in the first one, called *sequential method*, they solve simultaneously the routing and packing problem. In this case, two constraints (the LIFO constraint and the one that states that each customer belongs to exactly one route) are relaxed, thus the problem turns out to be a SDVRP. In the second solution approach, the *hierarchical method*, they first solve the routing problem and subsequently try to pack the items in the vehicles. Moreover, in the hierarchical method all constraints are considered. They extensively discuss the interdependency between the routing and the packing problem in different cases (number of customers per route, heterogeneity of cargo, density of goods). Their conclusion is that in case of routes with many customers, the routing aspects dominates the loading ones; on the other hand, when the demand of a customer is large so that it fills a big part of the vehicle, the loading problem becomes important. Finally, both problems appear to be relevant when the number of customers per route is small and the demand is weakly heterogeneous, thus an integrated solution approach is more appropriate. To our knowledge, the work by Moura and Oliveira [133] is the only one that considers the possibility of split the customer's demand in a routing-packing problem context.

In [67], Doerner et al. introduced the *multi-pile vehicle routing problem* (MP-VRP) which stems from a company that deliveries timber products. In this case, products that have the same dimensions are grouped together in pallets which can have fixed width, but length and height variable within certain values. The objective is to find a loading which respects the sequential constraints and minimizes the routing costs. The problem has been solved through TS and ACO [67], and variable neighborhood search and branch-and-cut [165].

Another practical problem that integrates routing and packing aspects is the *pallet-packing vehicle routing problem* (PPVRP) [177]. The key difference between the PP-VRP and the other models is that items must be firstly assigned and packed into pallets [20], which are then loaded in vehicles. Zachariadis et al. developed a solution technique based on TS to deal with the routing aspects, whereas the packing heuristic is related to the one described in [158]. In order to allow to unload easily items, all the demand of a customer must be stacked in the same pallet and the LIFO constraint is not imposed.

For a recent and comprehensive review on routing problems with loading constraints, the reader can refer to the survey of Iori and Martello [107].

4

Three dimensional packing problems: a review

4.1 Introduction

Packing problems have several applications, thus many variants of the problems have been developed depending on the constraints that are imposed. In this thesis we focus on the three dimensional packing problems (3D-PP) which are a generalization of the one dimensional case, where a set of items of a given weight have to be packed in the minimum number of bins of fixed capacity. Figure 4.1 presents a classification of the most important three dimensional packing problems, along with their interconnections. Each problem will be described in detail in the following section highlighting the distinctive features. Then we describe the more common packing strategies and the solution approaches adopted in literature.

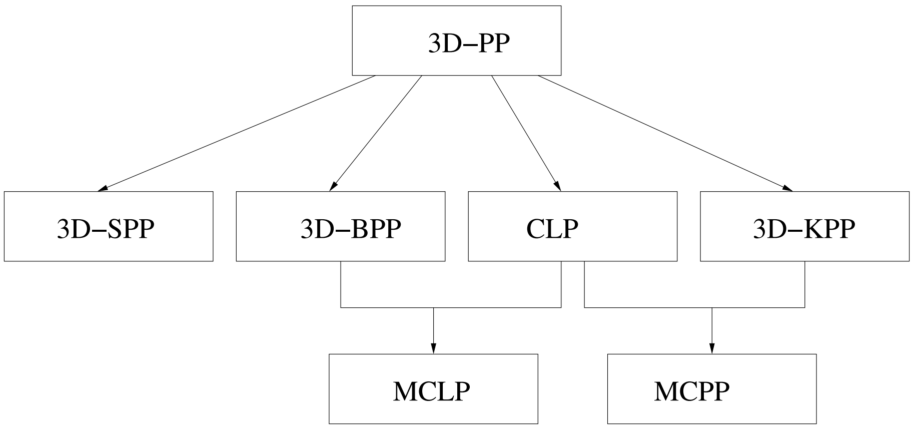


Figure 4.1: A classification of Three Dimensional Packing Problems and their interconnections

4.2 Problems classification

Three dimensional packing problems consist in packing a set of rectangular shaped items into containers. Each item i (with $i = 1, \dots, n$) is characterized by a width w_i, a length l_i and a height h_i, while the containers (or bins) have all the same dimensions W, L and H. A packing is feasible if the items are allocate completely inside a container without overlapping and in a orthogonal way, i.e. with their edges parallel to corresponding edge of the container. Depending on the objective function and side constraints, these problems can be divided in six categories:

- *Three dimensional bin packing problem* (3D-BPP): In this problem, the number of available containers is unlimited and the objective is to find the minimum number of them necessary to pack all items.
- *Three dimensional strip packing problem* (3D-SPP): In this variant of the problem, there is only one container available which has fixed width and height, but infinite length. Therefore the objective is to pack all boxes such that the required length is minimized.
- *Three dimensional knapsack packing problem* (3D-KPP): In this version, only one container with all fixed dimensions is available and a profit is associated to each item. The problem consist in choosing and feasible packing a subset of items such that the total profit is maximized.
- *Container loading problem* (CLP): In this problem, a single container has to be filled in such a way that the volume utilization is maximized. Indeed, the CLP can be considered a (3D-KPP) where the profit of a box is equal to its volume.
- *Multiple container loading problem* (MCLP): This problem is similar to the 3D-BPP, but the containers may have different dimensions and a fixed cost of use, thus the objective is to select the subset of containers with the minimum total cost.
- *Multiple container packing problem* (MCPP): In this case, a subset of items, each with a value associated, have to be packed in containers such that the total value of the selected items is maximized.

For the sake of readability we do no report the mathematical formulation of the different problem variants, that can be found in [45, 81, 136, 3, 79, 103, 113].

A first classification on packing problems was proposed by Dyckhoff [72], which has been recently improved by Wäscher et al. [174] introducing new categories that have become important in the last years. In particular, the container loading/packing problems (CLP, MCLP, MCPP) differ from the other problems also because they are often enriched with real-world constraints.

	# containers	**container dimensions**	**profit/cost**	**objective**
3D-BPP	unlimited	identical, fixed	no	minimize # containers
3D-SPP	1	variable length	no	minimize container length
3D-KPP	1	fixed	associated to items	maximize total profit
CLP	1	fixed	no	maximize volume utilization
MCLP	unlimited	different, fixed	associated to containers	minimize total cost
MCPP	unlimited	identical, fixed	associated to items	maximize total profit

Table 4.1: Classification of packing problems

4.2.1 Additional issues

Bischoff and Ratcliff [20] have described a variety of additional issues that may be important in many real world applications; the most significative of them are:

- *Weight limits:* Each item can be characterized also by its weight and the total weight of boxes must not exceed the container limits.
- *Weight distribution*: Several authors [e.g., 58, 30, 75] have considered the additional issue of balancing the weight inside the container.
- *Rotations*: Items may be rotated of 90° in any orthogonal direction, or an orientation can be fixed. For instance, if the vertical orientation can not be changed, only rotations on the width-length are allowed.

- *Maximum supported weight:* Depending on its construction and its contents, for each item it may be specified the maximum weight per unit of area that it can support. This value can be different for each side of the box [e.g., 143, 57, 18, 113].
- *Stability:* To ensure that the cargo do not move significantly during transport, stability constraints can be imposed. There is not a general definition of what it a stable cargo, however two measurements are usually adopted: *static stability* that states that each box must have its bottom supported by others boxes or the floor, and *dynamic stability* that is related to the capacity of the cargo to not move horizontally. In the first case the measure adopted is the minimum percentage of the base area supported; for the dynamic stability it is used the minimum percentage of boxes not surrounded by at least three sides [113].
- *Multi-drop:* If the container is transported by a vehicle and used to make deliveries, the boxes should be loaded in such a way that boxes of the same delivery are close, and unload and re-load operations are avoided.

4.3 Solution techniques

The subsequent section is divided in two parts. In the first one, we describe the strategies used to find the possible positions for placing an item in the loading area. Some of them are heuristics themselves and correspond to different ways to fill a containers; others are techniques used to cleverly find and reduce the set of admissible points for placing an item. These packing strategies are then included in more complex solution approaches, that are described in the second part of the section.

In the literature, several specific packing procedure have been proposed, however they can be grouped in the following macro-classes:

- *Wall-building:* This technique, introduced by George and Robinson [92], fills the container in layers across the depth. The first box of a layer determines its depth, the wall is then packed in horizontal (or vertical) strips; a single strip is constituted by several boxes placed sequentially and parallel to the container width (respectively, height) [19, 151, 140, 123, 132].
- *Stack building:* This heuristic first packs the boxes in vertical stacks and then arranges the obtained stacks on the floor of the container [94].
- *Horizontal layers.* In this case, the container is packed in horizontal layers from the floor upwards to the ceil of it [21, 20, 151].
- *Cuboid arrangements.* This approach fills the container with homogenous blocks made up of identical items with the same orientation [75, 32, 126, 114, 178].
- *Corner points:* The corner points are the possible positions where an item can be placed considering the current cargo. In two dimension, they can be individuated as the points where the slope of the envelope of items changes whereas, in the 3D case, the 2D algorithm is applied for each distinct value of depth of items. Given a fixed sequence of items to load, depending on the order in which candidate positions are selected and tested, different loading strategies can be obtained [128, 129, 55, 60, 3, 91, 158, 85, 173, 27, 177].
- *Overlapping graph:* In [82, 80] a graph-theoretical approach is presented. A graph is used to describe the relative position between items and to deduce necessary conditions that define feasible arrangements.

These specific loading strategies are then embedded in more complex solution approaches. Table 4.2 summarizes the main contributions in literature, which will be described in detail in the following sections. The manuscripts are grouped by problem formulation and the solution approach. Given that 3D-PPs are strongly NP-hard since they are a generalization of the one dimensional case [86], heuristic approaches are usually considered as shown Table 4.2. The main reason for this choice is that approximate algorithms are able to obtain good solutions in a reasonable time, and they are generally flexible to handle additional side constraints.

	exact methods	**heuristics**	**metaheuristics**
3D-BPP	[45, 81, 136, 3, 79, 103, 113, 128, 80, 33, 82]	[151, 55, 3]	[123, 78]
3D-KPP	–	–	[74]
3D-SPP	–	[92, 19, 31]	[29]
CLP	[152, 113]	[21, 139, 140, 75, 77, 101]	[30, 87, 32, 126, 26, 132, 137, 121]
MCLP	[152]	[108, 20, 75, 76, 114, 44, 161, 178]	[76, 44, 178]
MCPP	–	–	[142, 141, 150]

Table 4.2: Manuscripts on packing problems classified by problem formulation and solution technique.

4.3.1 Exact algorithms

Mathematical formulations of the **3D-BPP** are presented in [45, 81, 136, 3, 79, 103, 113]. Martello et al. [128] discussed some lower bounds and designed an exact branch and bound algorithm. Subsequently, new improved lower bounds have been computed by Fekete [80] and Boschetti [33]. A tree search algorithm which uses a graph-theoretic characterization has been proposed by Fekete et al. [82].

Scheithauer [152] proposed a linear programming relaxation for the **CLP** and **MCLP**, computed some bounds on these relaxed problem and proposed a column generation solution approach. Junqueira et al. [113] presented mixed integer linear programming that considers also stability and supporting constraints.

4.3.2 Heuristics

Constructive heuristics for the **3D-BPP** are presented by Scheithauer [151] and Crainic et al. [55]; both extend the *next/first/best fit decreasing* procedures commonly used for the 2D case. An other heuristic approach has been proposed by Almeida and Figueiredo [3].

Heuristic procedures for the **3D-SPP** are presented by George and Robinson [92], Bischoff and Marriott [19]. In [31], the authors presented a heuristic derived from branch-and-bound approach for the CLP of Pisinger [140].

For the **CLP**, several authors [21, 139, 140, 119] have proposed heuristics. Eley [75] and Fanslau and Bortfeldt [77] used tree search algorithms that load boxes arranged in cuboid blocks. In particular, Fanslau and Bortfeldt have recently published the best known results on available instances for both cases with and without the full support constraint. Recently He and Huang [101] proposed the *fit degree algorithm* whose underling idea is to pack items into corners or caves in the container in such a way that it is as close as possible to the other items.

The literature on the **MCLP** shows that this problem is often solved by adapting the heuristics used for the single container case. Possible strategies include the sequential approach where containers are filled in turn one by one [see 108, 20, 75, 76, 114, 44] the pre-assignment strategy, which first assigns boxes to containers and then performs the loading [161], and the simultaneous strategy, where a given number of containers are filled at the same time [75, 178].

4.3.3 Metaheuristics

For the **3D-BPP**, a TS algorithm was proposed by Lodi et al. [123] and a GLS technique has been used by Faroe et al. [78].

Bortfeldt and Gehring [29] presented a TS algorithm and a genetic algorithm (GA) for the **3D-SPP** that are an adaptation of the ones presented for the CLP in [28, 30]. The adaption of these original methods to the 3D-SPP was is performed in two ways: an open container can be considered and the objective is to minimize the necessary length, or a series of problems with decreasing container lengths is successively solved.

Egeblad and Pisinger [74] solved the **3D-KPP** using a SA algorithm combined with a particular representation of solutions, called *sequence triple*, which stores the relative box placements for each one of the three dimensions,

For the solution of the **CLP**, in [30, 87] Bortfeldt and Gehring use GA, whereas in [32] and [126] the authors presented TS and SA methods respectively. In [26] a TS approach is used also to cater for the multi-container case. GRASP approaches have been presented in [132, 137]. Recently, Parreño et al. [138] have applied a *variable neighborhood search* algorithm and Liu et al. [121] proposed and hybrid TS approach.

In [76, 44, 178], the **MCLP** was formulated as a set cover problem and solved using column generation techniques. However, the solution of the pricing problem, which is a CLP itself, is delegated to an approximate algorithm. Indeed, for solving the single CLP Eley [76] used the heuristic presented in [75], Che et al. [44] developed a bunch of three different GRASP heuristics, and Zhu et al. [178] proposed an approach which iteratively alternates a construction phase to an *hill climbing* algorithm.

In [142] and [141], the authors presented a GA approach for the **MCPP**, whereas Sang-Moon et al. [150] proposed an EA.

II

Real-world applications

5 The vehicle routing problems with time windows and carrier-dependent costs

5.1 Introduction

Vehicle routing is one of the most studied problems in optimization and many variants of the VRP have been introduced in the literature, some of which have been already discussed in Chapter 3. Nevertheless, despite the availability of this large set of classified formulations, often the practical problem that companies have to face is more complex than the standardized version discussed in scientific articles.

This is the case of the problem we came across, and thus in this work which has been published in [40], we consider a new version of the VRP problem. We decided to deal with its exact real-world formulation, without any concession to "judicious simplification", that would have allowed us to borrow results from existing successful solution techniques.

Our formulation, explained in detail in Section 5.2, includes a heterogeneous fleet, a multi-day planning horizon, a complex carrier-dependent cost function for vehicles, and the possibility of leaving orders unscheduled.

The problem formulation includes some non-linear constraints and cost components, thus the use of exact methods for its solution is quite impractical. Therefore, we resort to TS techniques that have shown to be effective on other variants of VRP. We also make use of a combination of different neighborhood relations. The experimental analysis is carried out on a set of real-world instances, and makes use of principled statistical tests to tune the parameters and to compare different variants.

The final outcome of the experimental analysis is that the most promising techniques are obtained by a combination of different neighborhood structures.

All the instances employed in the experiments, along with the best solutions found by our methods, are available on the web at the URL `http://www.diegm.uniud.it/ceschia/index.php?page=vrptwcdc`.

In order to evaluate objectively the performance of our solver, we also test it on public benchmarks of the *vehicle routing problem with private fleet and common carrier* (VRPPC) [23], which significantly resembles our problem and allows the comparison with other approaches. The outcomes of these comparisons show that our results are at the same level of the best ones in literature and we have been able to obtain a new best-known solution for one case.

The chapter is organized as follows. In Section 5.2 we present the problem formulation. The application of TS to the problem is illustrated in Section 5.3. Section 5.4 shows the experimental analysis on our instance and on benchmarks of the VRPPC. Finally, in Section 5.5 we draw some conclusions and discuss future work.

5.2 Problem formulation

We present our problem in stages by showing one by one the features (either previously published or original) that are included in the formulation and the various costs associated with the use of the vehicles.

5.2.1 Features of the problem

In Chapter 3 we have presented the mathematical formulations of different variants of VRPs, here we describe our formulation starting from the basic version of CVRP, which is characterized by the following entities and constraints:

Customers/Orders: The basic entity of the problem is the *customer*, who requires a supply of goods, called an *order*.

More formally we are given a set of n orders $\mathcal{O} = \{1, \ldots, n\}$, each issued by the corresponding customer. Multiple orders by the same customer are grouped together, so that in this basic formulation orders and customers are indistinguishable. As a consequence, in the following we will use the terms customer and order interchangeably unless stated explicitly.

A special customer, denoted by the number 0, represents the depot of the transportation company.

Each order i has associated a *demand* $q_i \geq 0$, which is the amount of goods to be supplied.

Fleet: The transportation of goods is performed by a fleet of vehicles $\mathcal{F} = \{1, \ldots, m\}$. In the original formulation all the vehicles are identical (i.e., they have the same capacity Q) and they are located at the same central depot (called *home depot*), where they have to return upon complete delivery.

Routes: A *vehicle route* (or simply a *route*) r is a sequence $\langle 0, v_1, \ldots, v_k, 0\rangle$ starting at the depot, visiting customers $v_1, \ldots, v_k \in \mathcal{O}$ in that order, and returning back to the depot. The orders served by a route r, is the set $\{v_1, \ldots, v_k\}$, which will be denoted by $ord(r)$. It is useful to define the *predecessor* $\pi(i, r)$ of a customer i w.r.t. the route r, as the previous customer in the sequence r.

We allow the possibility of empty routes, that is $r = \langle 0, 0\rangle$. In those cases, the vehicle is not used.

Load limits: An important constraint is that the load of each vehicle assigned to a route cannot exceed the vehicle capacity. If we define $q(r) = \sum_{i \in ord(r)} q_i$ as the total demand of route r, we impose that $q(r) \leq Q$.

Transportation costs: Each route has associated a transportation cost, denoted by $t(r)$. It can be either the road distance or a different measure of the total expenses of going on a given way from one customer to the following one (e.g., time, tolls, ...).

The solution of a VRP calls for the determination of a set of routes $\mathcal{R} = \{r_1, \ldots, r_m\}$, (also called a *routing plan*), one for each vehicle, that minimize the total transportation cost $\sum_{j=1}^{m} t(r_j)$ and additionally fulfills the following constraints:

1. the routes satisfies all orders, i.e., $\bigcup_{j=1}^{m} ord(r_j) = \mathcal{O}$;
2. each customer is visited only once, i.e., $ord(r_i) \cap ord(r_j) = \emptyset, 1 \leq i < j \leq n$;
3. the demands of all orders are fulfilled[1].

[1]In our formulation this constraint is enforced by construction since we define the total demand of a route as the sum of the single orders, therefore we implicitly do not allow partial deliveries.

The first extensions to the problem that we consider are represented by the so-called *service times* and *time windows*, discussed by Solomon [154]:

Service times: Each order is associated with a service time $s_i \geq 0$ needed to unload the goods from the vehicle. The vehicle must stop at the customer location for the service time.

Travel times: The time for traveling from one customer i to another customer j is estimated by a travel time τ_{ij}.

Time windows: Each customer and the depot are associated with a time interval $[e_i, l_i]$ (called *time window*) in which the service should take place. The depot time window includes all the time windows of the customers.

Earliest service time: All vehicles leave at the start time of the depot window (usually set to 0), and in case of early arrivals at the location of each customer, the vehicle is required to wait until the service can start.

More formally, given a route $r = \langle 0, \dots, j, i, \dots, 0 \rangle$, the *earliest service time* of order i on r is defined by $\alpha(i,r) = \max\{e_i, \delta(j,r) + \tau_{ji}\}$, where $\delta(j,r)$ is the *earliest departure time* from customer/depot j. This value is recursively defined as $\delta(0,r) = 0$ and $\delta(i,r) = \alpha(i,r) + s_i$.

Notice that for each order i on route r, this expression enforces by construction the fulfillment of the constraint $\alpha(i,r) \geq e_i$, which prevents early arrivals. Conversely, there is still the possibility of late arrivals, i.e., situations in which $\alpha(i,r) > l_i$. In practice these situations are usually allowed but they are treated as soft constraints and are penalized as described in Section 5.2.3.

Secondly, we consider the case of *heterogeneous* fleet [see, e.g., 153, 90] and the possibility to outsource part of the transportation to external carriers [172].

Heterogeneous vehicles: Vehicles are not identical as in the original problem but each vehicle j has its own capacity Q_j.

Carriers: Each vehicle j belongs to a *carrier*, denoted by $carr_j$, which is an external subcontractor of the transportation company.

Each carrier, including the company itself, uses a different function $t_j(r)$ to bill the routing costs to the transportation company, depending on the capacity of the vehicle employed and the length of the route (see Section 5.2.2 for details).

Moreover, in our problem, the planning period is not limited to a single day, but it spans over multiple days and each customer can place more than one order to be delivered in different days. In order to consider these features, we introduce the following entities:

Planning period: The planning period is composed by a number of consecutive days $\mathcal{D} = \{1, \dots, d\}$. Therefore, we have to design a set of routing plans $\mathcal{R}^* = \{\mathcal{R}_1, \dots, \mathcal{R}_d\}$, one for each day of the planning period.

Each vehicle performs only one route per day (it must return to the depot at the end of each day) and the same fleet is available on all the days of the planning period. We denote by r_{jk} the route travelled by vehicle j on day k.

Multiple orders: Each customer can issue different orders in different days. Therefore, at this stage orders and customers become different but related entities. For each order i we now define the (unique) customer associated to it, which will be denoted by $cust(i)$.

Delivery dates: As a consequence of introducing a multi-day perspective, there is also the possibility of specifying an interval of days $[\eta_i, \theta_i]$ in which the order i should be delivered.

Similarly to time windows case, delivery dates are treated as a soft constraint and their violations are penalized as described in Section 5.2.3.

Additional features of the problem we consider, concern the limitations of using some types of vehicles in particular situations and the possibility to leave orders out of the schedule.

As for the vehicle limitations, sometimes, due to site topology and road barriers, there might be impossible to use some vehicles to serve certain customers. Other limitations could regard the area of operation of some carriers, in the sense that they do not accept to deliver in specific regions (e.g., too far from their headquarters). Both alternatives are modeled by the following constraint:

Site reachability: It is given a compatibility matrix ρ, such that order i can be served by vehicle j only if $\rho_{ij} = 1$.

Since some of the real-world instances could be over-constrained in terms of the number of orders to be delivered, we give the possibility of define a priority on orders. Therefore we distinguish between *mandatory* orders, which must be served in a solution, and *optional* orders, which can be excluded. These concepts are captured in the following:

Mandatory/Optional orders: The set of orders $\mathcal{O}$ is partitioned into two sets $\mathcal{M}$ and $\mathcal{P}$ (where $\mathcal{O} = \mathcal{M} \cup \mathcal{P}$, $\mathcal{M} \cap \mathcal{P} = \emptyset$). Orders in $\mathcal{M}$ are mandatory, and must be delivered; orders in $\mathcal{P}$ are optional, therefore they can be discarded at a given cost γ_i.

The original constraints on VRP solutions must be adapted to deal with the new elements added in these stages. The constraints are modified as follows:

1. the routes satisfy all mandatory orders: $\mathcal{M} \subseteq \bigcup_{j=1}^{m} \bigcup_{k=1}^{d} ord(r_{jk}) \subseteq \mathcal{O}$;
2. each order is delivered at most once: $ord(r_{jk}) \cap ord(r_{j'k'}) = \emptyset, 1 \leq j < j' \leq n, 1 \leq k < k' \leq d$;

Finally, since, by regulation, drivers must take breaks during their activity, a set of mandatory *rests* of drivers must be set [98]:

Driving rests: After a long consecutive working period, drivers should take a rest of a given minimum duration. In our case, a rest of 45 minutes after 4 hours and 30 minutes of consecutive driving is imposed by law.

Ergo, the earliest arrival time for the delivery of order i on route $r = \langle 0, \ldots, j, i, \ldots, 0 \rangle$ must be modified in order to take account of the mandatory rests: $\alpha'(i, r) = \max\{e_i, \delta(j, r) + \tau_{ji} + \zeta(i, r)\}$, where $\zeta(i, r)$ can be either 0 or 45 minutes according to the working/rest times patterns of the predecessors of order i in route r.

It is important to observe that in our formulation also service times and waiting times are accounted as working times for computing rests.

5.2.2 Vehicle cost functions

Since we consider the possibility to rely on external carriers for deliveries, we have to deal with different ways to compute transportation costs, even within a single problem instance. As an example, some carrier companies could bill the transportation company for the service on the basis of the route, other carriers could consider the size of the delivered goods, etc. Therefore, in order to be general enough, we designed our solver so that an external code for computing these costs can be invoked.

In the cases we have examined we have identified some common criteria for computing the transportation costs. In practice, the following four cost functions are used (where $dist_{ij}$ is the road distance between customers i and j):

1. A fixed cost for the vehicle c plus a cost ξ_1 per travel unit (measured in €/Km). If we denote with $\|r\|$ the total distance traveled in route r, i.e., $\|r\| = \sum_{i \in ord(r)} dist_{\pi(i,r)i}$, we have:

$$t(r) = c + \xi_1 \cdot \|r\|$$

2. A fixed cost for the vehicle c plus a cost ξ_2 per load unit (measured in €/Kg), which is dependent on the farthest location. If we denote the maximum distance between the depot and a costumer in the route with $\|r\|^*$, i.e., $\|r\|^* = \max_{i \in ord(r)}\{dist_{0i}\}$, we have:

$$t(r) = c + \xi_2 \left(\|r\|^*\right) \cdot q(r)$$

3. A fixed cost for the vehicle c plus a cost ξ_1 per travel unit up to a predefined level of load L (dependent on the vehicle capacity), a cost per load unit ξ_2 dependent on the farthest location for larger loads. That is:

$$t(r) = \begin{cases} c + \xi_1 \cdot \|r\| & q(r) \leq L \\ c + \xi_2(\|r\|^*) \cdot q(r) & q(r) > L \end{cases}$$

4. A fixed cost for the vehicle c plus a cost ξ_3 per load unit, which is dependent both on the total load $q(r)$ and on the farthest location. That is:

$$t(r) = c + \xi_3 \left(\|r\|^*, q(r)\right) \cdot q(r)$$

Since the value of the load cost coefficients ξ_2 and ξ_3 depends on the distance of the farthest customer $\|r\|^*$, the carrier should define such a value for each customer (and, in the case of ξ_3, also for each load level). Normally, the carriers partition their area of operation in regions and specify the load cost coefficient for every region (each customer location belongs to a region). The load cost coefficient selected to compute the cost is the largest of the route, i.e. the one associated with the region of the farthest customer.

5.2.3 Constraints and objective function

Similarly to other optimization problems, constraints are split into two categories: *hard* and *soft* constraints. A legal solution to the problem must satisfy all the hard constraints, whereas soft constraints can be violated and they are included in the objective function to be minimized.

Summarizing, in our formulation we deal with the following hard constraints:

H1 The load of each vehicle must not exceed its capacity, i.e., $q(r_{jk}) \leq Q_j$, for $1 \leq j \leq m, 1 \leq k \leq d$.

H2 Vehicles must return to the depot before a *shutdown time* $\bar{l}_0$ (in our case, $\bar{l}_0$ is fixed to 1 hour after the end of the depot time window l_0). Notice that late returns within l_0 and $\bar{l}_0$ are possible but they will be penalized as explained later, whereas solutions with a return time to the depot after $\bar{l}_0$ are infeasible.

H3 The compatibility relation must be satisfied, i.e., an order i should be served by vehicle j for which the ρ_{ij} relation holds.

H4 All mandatory orders must be delivered, i.e., $\mathcal{M} \subseteq \bigcup_{j=1}^{m} \bigcup_{k=1}^{d} ord(r_{jk})$.

H5 The route timetable must obey the regulations on driving rests.

The other problem features described in Section 5.2.1 are considered as soft constraints and they become part of the objective function $F(\mathcal{R}^*) = w_{\mathsf{S1}} \cdot F_{\mathsf{S1}}(\mathcal{R}^*) + w_{\mathsf{S2}} \cdot F_{\mathsf{S2}}(\mathcal{R}^*) + w_{\mathsf{S3}} \cdot F_{\mathsf{S3}}(\mathcal{R}^*) + w_{\mathsf{S4}} \cdot F_{\mathsf{S4}}(\mathcal{R}^*)$, which is the linear combination of the following components:

S1 The delivery of an order on a day not included in its delivery days is penalized proportionally to its demand:

$$F_{\mathsf{S1}}(\mathcal{R}^*) = \sum_{k=1}^{d} \sum_{j=1}^{m} \sum_{i \in ord(r_{jk})} \left(1 - \chi_{[\eta_i, \theta_i]}(k)\right) \cdot q_i$$

where $\chi_I(x)$ is the characteristic function of interval I, i.e.,

$$\chi_I(x) = \begin{cases} 1 & x \in I \\ 0 & \text{otherwise} \end{cases}$$

S2 The delivery of an order after the end of its time window is penalized proportionally to the delay:

$$F_{\mathsf{S2}}(\mathcal{R}^*) = \sum_{k=1}^{d} \sum_{j=1}^{m} \sum_{i \in ord(r_{jk})} \max\{0, \alpha'(i, r_{jk}) - l_i\}$$

S3 Optional orders not delivered are penalized according to their cost γ_i:

$$F_{\mathsf{S3}}(\mathcal{R}^*) = \sum_{i \in \mathcal{O} \setminus \bigcup_{k=1}^{d} \bigcup_{j=1}^{m} ord(r_{jk})} \gamma_i$$

S4 The transportation costs for each vehicle is computed according to the carrier agreements t_j, in one of the forms described in Section 5.2.2:

$$F_{\mathsf{S4}}(\mathcal{R}^*) = \sum_{k=1}^{d} \sum_{j=1}^{m} t_j(r_{jk})$$

The weights of the various components are not fixed at some global level, but they are set by the operator for each specific case. To this regard, setting such weights is rather a complex task because the relative importance of the numerous components is difficult to establish. With the purpose of simplifying this process and having an immediate grasp of the costs, we decide to represent the costs directly in a real currency (€ in our case). Moreover, in order to deal with an objective function that can be represented in integer arithmetic and it is fine-grained enough, we set the cost unit to 1/1'000th of €.

5.3 Application of tabu search

First of all, it is important to observe that the presence of non-linear constraints (H5) and cost function components (the family of F_{S4} vehicle costs) makes it quite impractical to apply exact methods on this problem formulation. Therefore we resort to metaheuristic techniques for tackling the problem.

The solver we developed is based on the TS metaheuristic, which has been already described in Section 2.2.3. In order to apply TS to our VRP problem we have to define several features. We first illustrate the search space and the procedure for computing the initial state. Then, we define the neighborhood structure and the prohibition rules, and finally we describe the set of search components employed and the high-level strategies for combining them.

5.3.1 Search space, cost function, and initial solution

The local search paradigm is based on the exploration of a search space composed of all the possible complete assignments of values to the decision variables, possibly including also the infeasible ones. In our case, a state is composed by a set of routes, one for each vehicle on each of the planning days.

An order can appear in only one route (i.e., it is *scheduled*) or it can be left outside, in the set of *unscheduled orders*. Thus, for each scheduled order, the solution specifies the day when the order is delivered, the vehicle, and the position in the corresponding route (the arrival time at the client is deterministically computed, given its position in the route, according to the rules presented in Section 5.2.1).

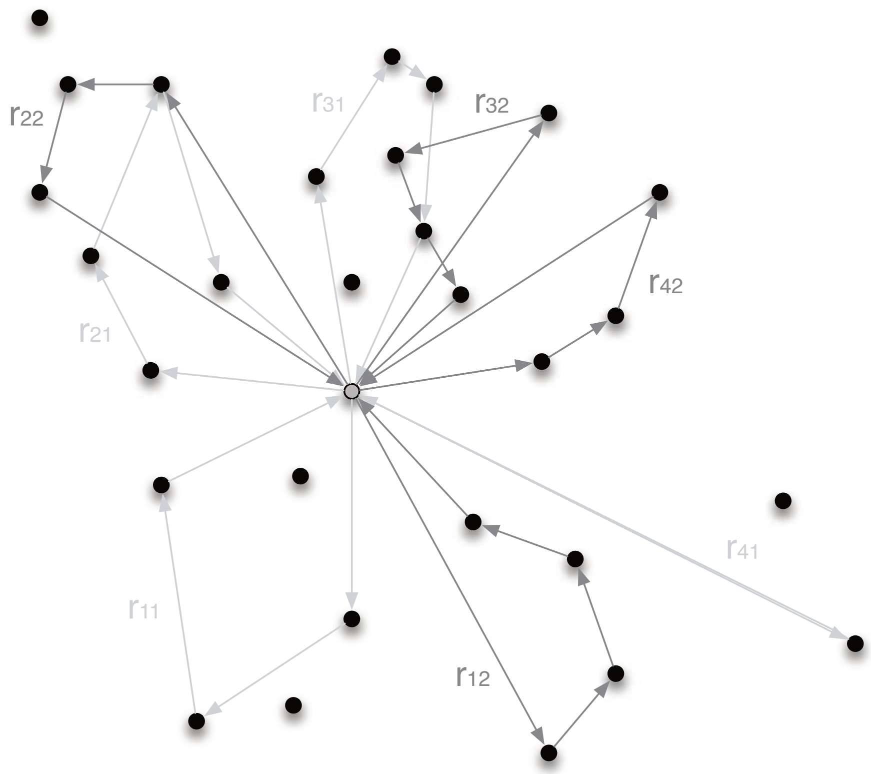


Figure 5.1: An example of a state composed of 4 routes for 2 days (r_{jk} identifies the route j on day k).

An example of a state is shown in Fig. 5.1, in which different gray levels are used to highlight routes performed on each of the two days composing the planning horizon (r_{jk} identifies the route j travelled on day k). Notice that some customers are left out of all routes whereas others are visited more than once because they place orders on different days.

The search space is restricted to states that satisfy constraints H3 (site reachability) and H4 (mandatory orders), whereas constraints H1 (vehicle capacity) and H2 (late return) can be violated and are included in the cost function with a high weight (H5 is satisfied by construction).

The cost function is thus the (monetary) sum of all soft constraints S1–S4, plus the *distance to feasibility* for H1 and H2 multiplied by a suitable high weight. For H1 the distance to feasibility is the sum of the quantities (measured in Kg) that exceed the vehicle capacity. For H2, there are several ways to define the distance to feasibility. Our choice is to count the number of orders (including the return to the depot) that are in a route that finishes after the shutdown time. This solution is more effective than summing up the delays w.r.t. the shutdown time, because it creates smoother trajectories from solutions with many violation toward the total elimination of them. For example, in the case of two orders of the same client that are late, if we only count the delays, in order to obtain an improvement we would need to move both orders at the same time (which is not done by our neighborhoods). Conversely, in our solution, every single order removed from the route improves the cost function independently of the fact that the total delay is reduced.

The initial solution is constructed at random, but satisfying some of the constraints. That is, we create a state of the search space that satisfies the constraints about the site reachability (H3), the driving rests (H5) and the delivery day (S1). This is made by assigning each order $i \in \mathcal{O}$ to a randomly selected feasible day $k \in [\eta_i, \theta_i]$ of the planning horizon $\mathcal{D}$ and to a random vehicle j, chosen among the compatible ones ($\rho_{ij} = 1$).

Once the day and the vehicle are selected, the route r is unequivocally identified, so we can assign the order to a random position in the selected route. The fulfillment of constraint H5 is enforced by construction. In addition, in the initial solution, all orders are scheduled, so that constraints H4 and S3 are also satisfied completely.

5.3.2 Neighborhood relations

The neighborhood of a solution is usually implicitly defined by referring to a set of possible moves, which define transitions between solutions. A move is composed by *attributes* that identify the resources involved in the move. In our problem, we are dealing with the assignment of an order to three kinds of resources: the day, the vehicle and the position in the route.

We consider the following three neighborhood relations:

Insertion (Ins): This neighborhood is defined by the removal of an order from a route and its insertion in another one in a specific position. An order can also be inserted in the list of the unscheduled ones (the position is not meaningful in this case) or put back from this list to a route. The list of unscheduled orders is in practice treated as an additional special route, with the main difference that the position of orders in this sequence is irrelevant.

A move m of type Ins is identified by five attributes $m = \langle o, or, op, nr, np\rangle$ where o represents an order, or and op the old route and the old position in the old route, and np and nr, the new route and new position in the new route, respectively.

Inter-route swap (InterSw): This neighborhood is defined by exchanging an order with another one belonging to a different route. A move m of this type is identified by six attributes $m = \langle o1, o2, r1, r2, p1, p2\rangle$, where $o1$ and $o2$ are orders, $r1$ and $r2$ are the routes of $o1$ and $o2$, and $p1$ and $p2$ are the positions of the orders in the routes.

Intra-route swap (IntraSw): This neighborhood is defined by exchanging an order with another one belonging to the same route. A move m of this type is identified by five attributes $m = \langle o1, o2, r, p1, p2\rangle$, where $o1$ and $o2$ are orders, r is the route, $p1$ and $p2$ are the positions of the orders in the route.

Notice that, given the state, some of the attributes are dependent from each other. For example, given an Ins move $m = \langle o, or, op, nr, np\rangle$, the order o identifies the pair (or, op) and vice versa. It is however useful to have all of them in the representation of the move for the definition of the prohibition rules.

5.3.3 Prohibition rules

In the seminal version of TS, for the purpose to prevent cycling in the search trajectory, when a move m is in the tabu list, the move m' that would lead back to the same state (i.e., the *inverse* of m) is prohibited.

Nevertheless, in many cases there is a further risk that the search remains trapped in the proximity of some local minimum and iterates chaotically around it. In these cases, it is necessary to have some diversification mechanisms for "pushing" the search away from the minimum. This is obtained by generalizing the prohibition behavior with the definition of a general relation (called *prohibition rule*) between pairs of moves (m_t, m_e) that states that move m_e is excluded from the neighborhood by the fact that move m_t is in the tabu list. This enables the possibility that the presence of a move m_t in the tabu list results in the prohibition of a large set of moves, rather than the single inverse one. The prohibition rules are based on the values of the attributes of the two moves, the one in the tabu list m_t and the one under evaluation m_e.

It is quite difficult to tell *a priori* which is the most suitable prohibition rule for a given neighborhood, therefore for each of them we have defined and tested several ones, of different restrictive levels. They are compared experimentally in Section 5.4.3.

Rule	Condition	Description of tabu moves	Strength
PR1	$o_e = o_t \wedge or_e = nr_t$	moves removing the order o_t from the route nr_t	0.088%
PR2	$nr_e = or_t \wedge or_e = nr_t$	moves putting back any order from nr_t to or_t	0.485%
PR3	$o_e = o_t \wedge nr_e = or_t$	moves reinserting the order o_t in the route or_t	1.470%
PR4	$o_e = o_t$	moves involving the same order o_t	1.471%
PR5	$or_e = nr_t$	moves removing any order from the route nr_t	5.583%
PR6	$nr_e = or_t$	moves reinserting any order into the route or_t	7.978%

Table 5.1: Prohibition rules for TS(Ins).

For the Ins neighborhood, assuming that the move $m_t = \langle o_t, or_t, op_t, nr_t, np_t \rangle$ is in the tabu list and the move $m_e = \langle o_e, or_e, op_e, nr_e, np_e \rangle$ is the move to be evaluated, we consider the six alternatives shown in Table 5.1. The last column shows the so-called *tabu strength*, which is defined as the (average) percentage of the entire neighborhood that a single move in the list prohibits.

Similarly, we have tested several prohibition rules also for the two neighborhoods IntraSw and InterSw. However, prohibition rules for these two neighborhoods have a more limited influence, and thus we report only the ones which proved to be the most effective for our instances. Specifically, for IntraSw, if $m_e = \langle o1_e, o2_e, r_e, p1_e, p2_e \rangle$ is the move to be tested and $m_t = \langle o1_t, o2_t, r_t, p1_t, p2_t \rangle$ is in the tabu list, the condition

$$o1_e = o1_t \vee o2_e = o2_t \vee o1_e = o2_t \vee o2_e = o1_t$$

is imposed. It forbids to make a move where any of the two orders of m_e is equal to any of those of m_t. Its tabu strength is 7.613%. The same condition is used also for InterSw. If $m_e = \langle o1_e, o2_e, r1_e, r2_e, p1_e, p2_e \rangle$ is the move to be tested and $m_t = \langle o1_t, o2_t, r_t, r_t, p1_t, p2_t \rangle$ is the move in the tabu list, then

$$o1_e = o1_t \vee o2_e = o2_t \vee o1_e = o2_t \vee o2_e = o1_t$$

is imposed. For InterSw the tabu strength of this prohibition rule is 5.933%.

We make use of the *robust tabu search* scheme [156] in which the length of the tabu list is *dynamic*, as it is described in Section 2.2.3. In detail, we set two values tt and δ_{tt} and we assign to each accepted move a tabu tenure randomly selected between $tt - \delta_{tt}$ and $tt + \delta_{tt}$. In all our experiments δ_{tt} is set to 2 (based on preliminary experiments), whereas tt is subject to tuning.

The basic aspiration criterion which states that a move is accepted if it improves on the current best solution is applied.

5.3.4 Search techniques

On the basis of the three neighborhood relations defined, we come up with three basic TS techniques, that we call TS(Ins), TS(IntraSw), TS(InterSw).

It is important to observe that the search space is not connected under the IntraSw and InterSw neighborhood structures. Indeed, both neighborhoods do not change the number of orders in a route, thus there is no trajectory that goes from a state with a given number of orders in a route to a state with a different one. Consequently, TS(IntraSw) and TS(InterSw) must be used in combination with TS(Ins), since they are not effective when used by themselves.

We use a sequential solving strategy for combining TS algorithms based on different neighborhood functions, as proposed (among others) by Di Gaspero and Schaerf [64] under the name of *token-ring* search. Token ring works as follows: Given an initial state and a set of algorithms, it makes circularly a run of each algorithm, always starting from the best solution found by the previous one. The overall process stops either when a full round of the algorithms does not find an improvement or the time granted is elapsed. Each single technique stops when it does not improve the current best solution for a given number of iterations (*stagnation*).

We identify the following five strategies, applying the token ring to the basic neighborhood structures. In the following, token ring is denoted by the symbol $\triangleright$.

1. TS(Ins)
2. TS(Ins) ▷TS(IntraSw)
3. TS(Ins) ▷TS(InterSw)
4. TS(Ins) ▷TS(IntraSw) ▷TS(InterSw)
5. TS(Ins) ▷TS(InterSw) ▷TS(IntraSw).

We also consider solvers that use the union of many neighborhoods: The algorithm based on this compound neighborhood, denoted by the $\oplus$ symbol in [64], selects at each iteration a move belonging to any of the neighborhoods as part of the union. We therefore have three more strategies:

6. TS(Ins $\oplus$ IntraSw)
7. TS(Ins $\oplus$ InterSw)
8. TS(Ins $\oplus$ IntraSw $\oplus$ InterSw)

Finally, we experiment two further techniques that use both union neighborhoods and token ring search:

9. TS(Ins) ▷TS(IntraSw $\oplus$ InterSw)
10. TS(Ins) ▷TS(Ins $\oplus$ IntraSw $\oplus$ InterSw)

All these ten strategies will be analyzed and compared experimentally in the next section.

5.4 Experimental analysis

In this section, we first introduce the benchmark instances and the general settings of our analysis, and then we move to the experimental results. We analyze our techniques in stages, starting from the simplest algorithm which uses one single neighborhood, then moving to the more complex ones. Finally we show the results of the best configuration of our solver on benchmarks of the VRPPC.

5.4.1 Benchmark instances

Two cases coming from different real-world scenarios provided by our industrial partner, beanTech s.r.l., are at our disposal for the experimental part. The main features of these two cases are shown in Table 5.2. In order to highlight the importance of the various components of the problem, starting from each case we have created 9 different instances, named with the letters from A to I. The A instances are the original ones, whereas instances B–I are obtained by perturbing one specific feature, so as to produce instances that are realistic but specifically biased toward stressing the use of a particular feature.

Table 5.3 shows the resulting 18 instances, along with the values of a set of indicators that describe the features (perturbed values are in bold). In detail, the columns are defined as follows:

- **Days (D):** The number of days in the planning horizon.
- **Filling Ratio (FR):** The ratio between the total demand and the total capacity of the vehicles multiplied by the number of days.
- **Time Windows (TW):** The average ratio between the time window of the orders and the time window of the depot.

Inst.	#Orders	#Customers	#Vehicles	#Carries	#Regions
case1	139	44	7	4	19
case2	166	56	7	4	22

Table 5.2: Features of the two scenarios.

Inst.	D	FR	TW	DW	C	MO	SO	OUT
case1-A	3	27.47	90.91	89.47	87.97	14.04	10.12	T
case1-B	3	27.47	90.91	89.10	87.97	**100.00**	11.09	T
case1-C	**2**	41.21	90.91	76.27	87.89	13.56	9.77	T
case1-D	**4**	20.60	90.91	64.74	88.83	10.26	7.39	T
case1-E	3	27.47	**73.12**	89.47	87.97	14.04	10.12	T
case1-F	3	27.47	**40.51**	89.47	87.97	14.04	10.12	T
case1-G	3	27.47	90.91	89.47	100	14.04	10.12	**F**
case1-H	3	27.47	90.91	**33.33**	87.97	14.04	10.12	T
case1-I	3	27.47	90.91	**33.33**	88.45	14.89	12.27	T
case2-A	3	47.41	90.91	66.30	85.87	16.67	11.06	T
case2-B	3	47.41	90.91	65.84	85.71	**100.00**	12.29	T
case2-C	**2**	71.12	90.91	80.25	85.36	17.28	12.29	T
case2-D	**4**	35.56	90.91	53.47	86.38	14.81	9.21	T
case2-E	3	47.41	**81.41**	66.30	85.87	16.67	11.06	T
case2-F	3	47.41	**41.62**	66.30	85.87	16.67	11.06	T
case2-G	3	47.41	90.91	66.30	100	16.85	11.19	**F**
case2-H	3	47.41	90.91	**33.33**	84.87	19.12	14.64	T
case2-I	3	47.41	90.91	**33.33**	85.88	17.05	11.31	T

Table 5.3: Features of instances.

- **Day Window (DW):** The average number of available days of the orders divided by the total number of days.
- **Compatibility (C):** The density of the compatibility matrix between vehicles and orders.
- **Mandatory Orders (MO):** The percentage of mandatory orders.
- **Space Occupancy (SO):** The average percentage of space taken by an order in a vehicle.
- **Outsourcing (OUT):** If this indicator is set to F (False), it means that there are no external carriers and the routing costs depend only on the total distance travelled; if the indicator is set to T (True), there are external carriers and consequently different ways to compute costs.

Instances H are modified in such a way that each orders should be dispatched on the first day of its delivery dates (*as soon as possible*); alternatively for instances I each order should be dispatched on the last day of its delivery dates (*as last as possible*). For all instances the cost γ_i of not delivering an optional order i ($i \in \mathcal{P}$) is set equal to its demand q_i.

Summarising, we tested our algorithms on a benchmark composed of 18 instances, which have been made available through the web at the URL `http://www.diegm.uniud.it/ceschia/index.php?page=vrptwcdc`.

5.4.2 General settings and implementation

All the algorithms have been implemented in the C++ language, exploiting the EasyLocal++ framework [63]. The experiments have been performed on the environment detailed in Section 2.3.

The stopping criterion of the basic algorithms is the detection of stagnation, which can occur at different times. Therefore, in order to compare different combinations in a fair way, we decide to set a maximum number of iterations equal to 1000 for each basic TS(·) component of the token ring strategy. Moreover, we also impose a maximum of 3 full rounds or 1 idle round as the stopping criterion of the whole token ring procedure. We run 100 trials for each configuration of the parameters.

For each individual TS(·) component of the search strategy, the maximum number of iterations from the last improvement is set to 500 for the Ins neighborhood and all the unions including it, and 300 for the other neighborhoods.

For the experiments we set $w_{\mathsf{S1}} = 30$, $w_{\mathsf{S2}} = 10$, $w_{\mathsf{S3}} = 250$ and $w_{\mathsf{S4}} = 1$.

5.4.3 Results on prohibition rules for TS(Ins)

The first set of experiments focuses on the Insertion neighborhood and compares different prohibition rules for TS(Ins).

Obviously, for each prohibition rule the best tabu list length can be different. Therefore we have to tune the length for each rule independently and then compare the prohibition rules among themselves, each with its best setting.

Fig. 5.2 shows, in form of box-and-whiskers plots, the results obtained on the two original instances for different values of the tabu length and for three prohibition rules, namely PR1, PR4, and PR5. We select to show PR5 because it provides the best results, and PR1 and PR4 because they correspond to diverse tabu strength levels.

The figures highlight that the best prohibition rule is PR5, and this is confirmed by performing a statistical comparison of the different results, yielding to p-values of the pairwise two-sided *Student* t-test [170] that are inferior to 0.0001.

They also show an interesting phenomenon regarding the tabu list length for prohibition rules PR1 and PR4: The curves have two different minima. Our interpretation is that the first one is related to the depth of local minima in the search space and it represents the "normal" behavior of TS, the second "spurious" minimum is due to the situation in which most of the moves are tabu, and the search alternates between performing non tabu moves and tabu moves. Anyway, this situation provides an effective diversification and, in our case, this second minimum is indeed the lowest one. This phenomenon is confirmed by the fact that for PR1 and PR4 in such conditions about 30% of the moves performed are tabu. It is worth remarking that this anomalous behavior occurs only on the less performing prohibition rules.

The results obtained for the other instances and prohibition rules are similar to those shown in the figures and are omitted.

5.4.4 Results on inter-route swap and intra-route swap neighborhoods

As already noticed, TS(InterSw) and TS(IntraSw) cannot be used alone because the search space is not connected under their neighborhood relations. Consequently, it would be meaningless to tune those algorithms starting from random solutions and instead we apply the tuning procedure to the strategies TS(Ins)▷TS(InterSw) and TS(Ins)▷TS(IntraSw).

In Fig. 5.3 we show the outcome of the experiments. In this case, for TS(InterSw) and TS(IntraSw) we can see that the results are not really affected by the length of the tabu list. We select the value 10 for TS(InterSw) and 15 for TS(IntraSw) in the following experiments, although there is no significant statistical difference with the other values. In fact their role is to diversify during the search process, moving on plateaux of the search space.

5.4.5 Results on composite solvers

Our final experiment on the problem concerns the comparison of composite solvers. For each component of the solvers, the parameters are set to the best values found in the previous experiments.

We perform a F-Race selection [16] based on Friedman two-way analysis of variance by ranks as implemented in the EasyAnalyzer framework [65]. At each step of the selection procedure each solver is tested on one instance (among the available ones) and it is assigned a rank according

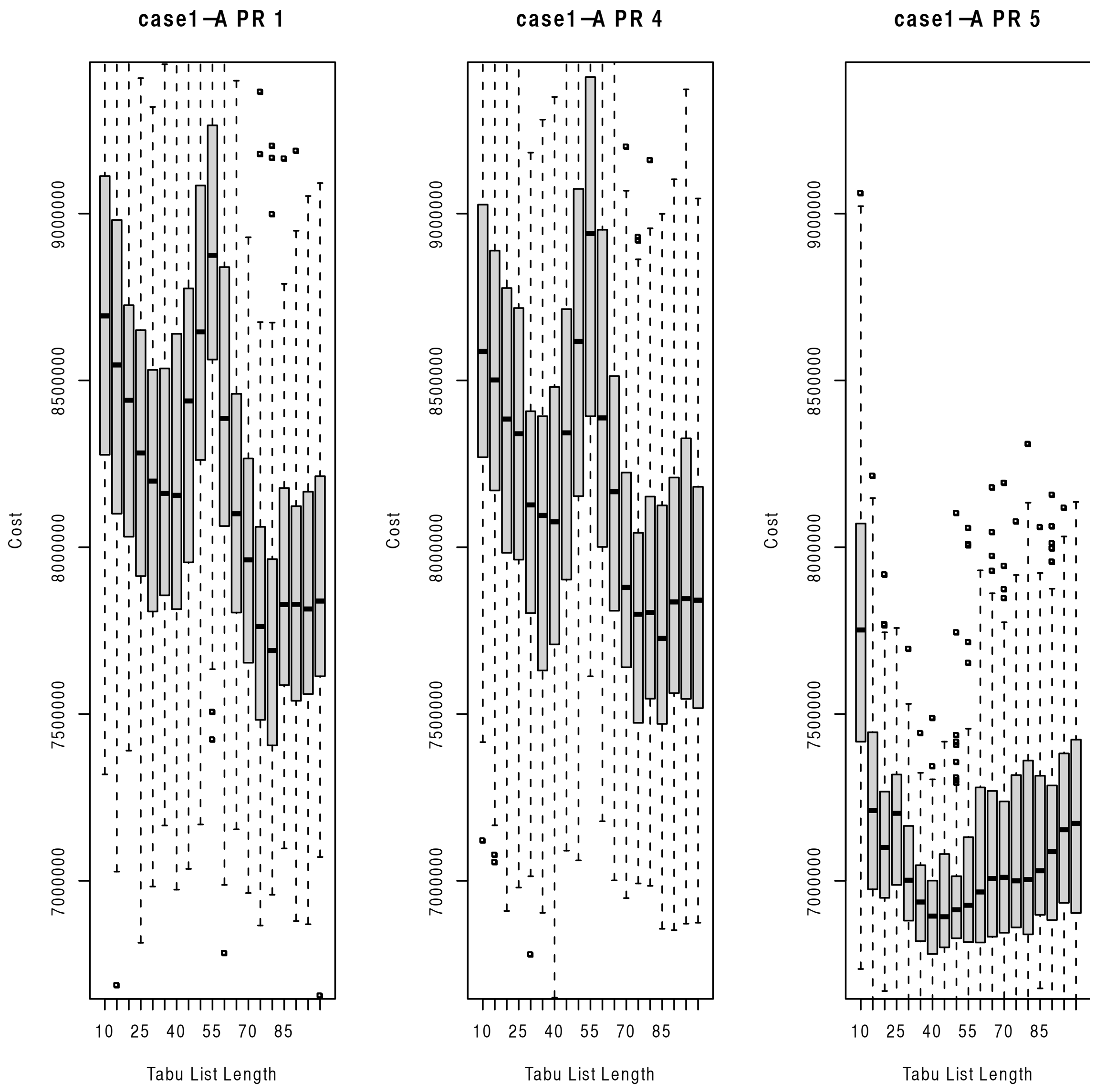


Figure 5.2: Results for different tabu list lengths of TS(Ins) for case1-A with different prohibition rules.

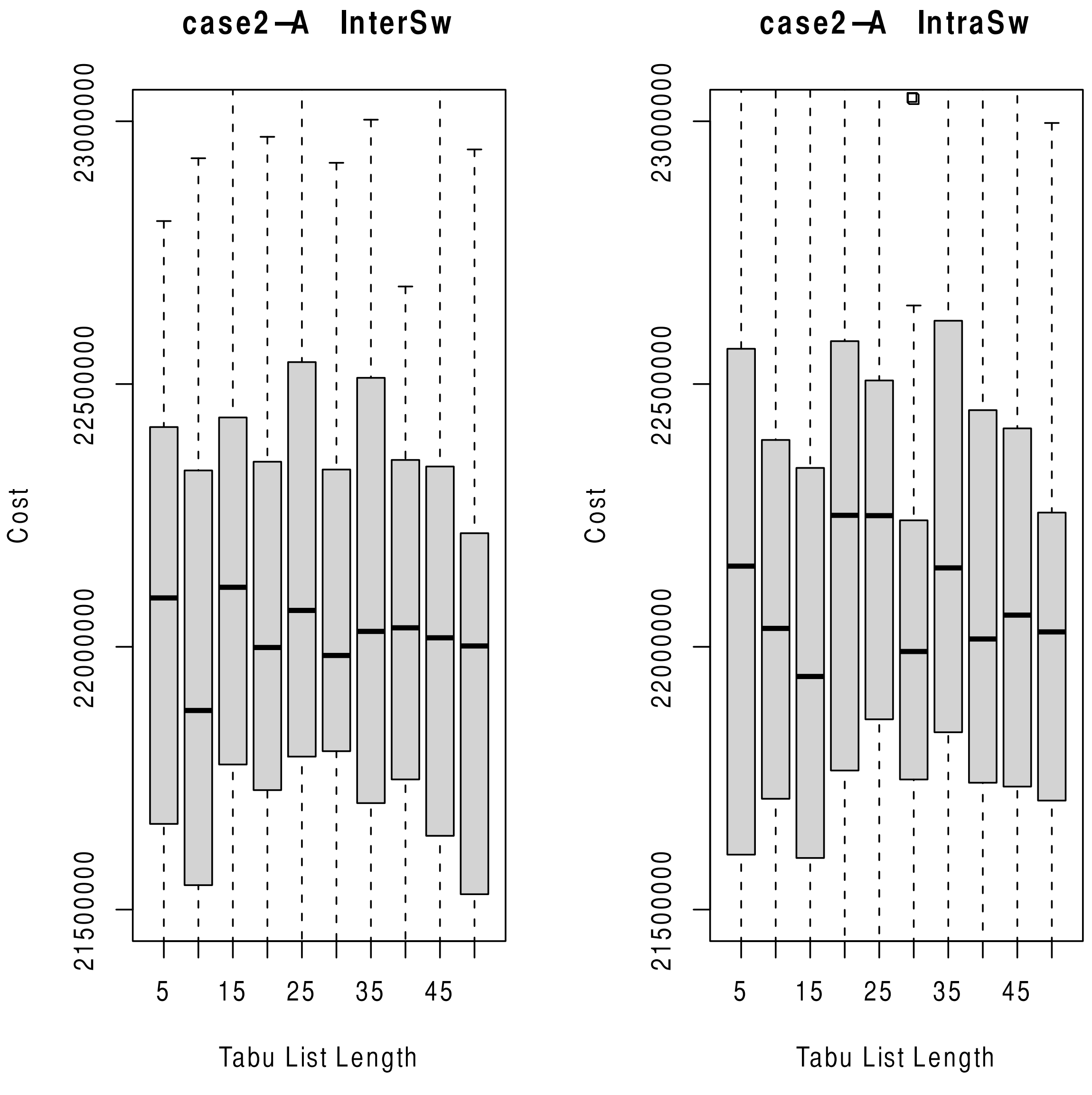


Figure 5.3: Results for different tabu list lengths of TS(IntraSw) and TS(InterSw) for case2-A.

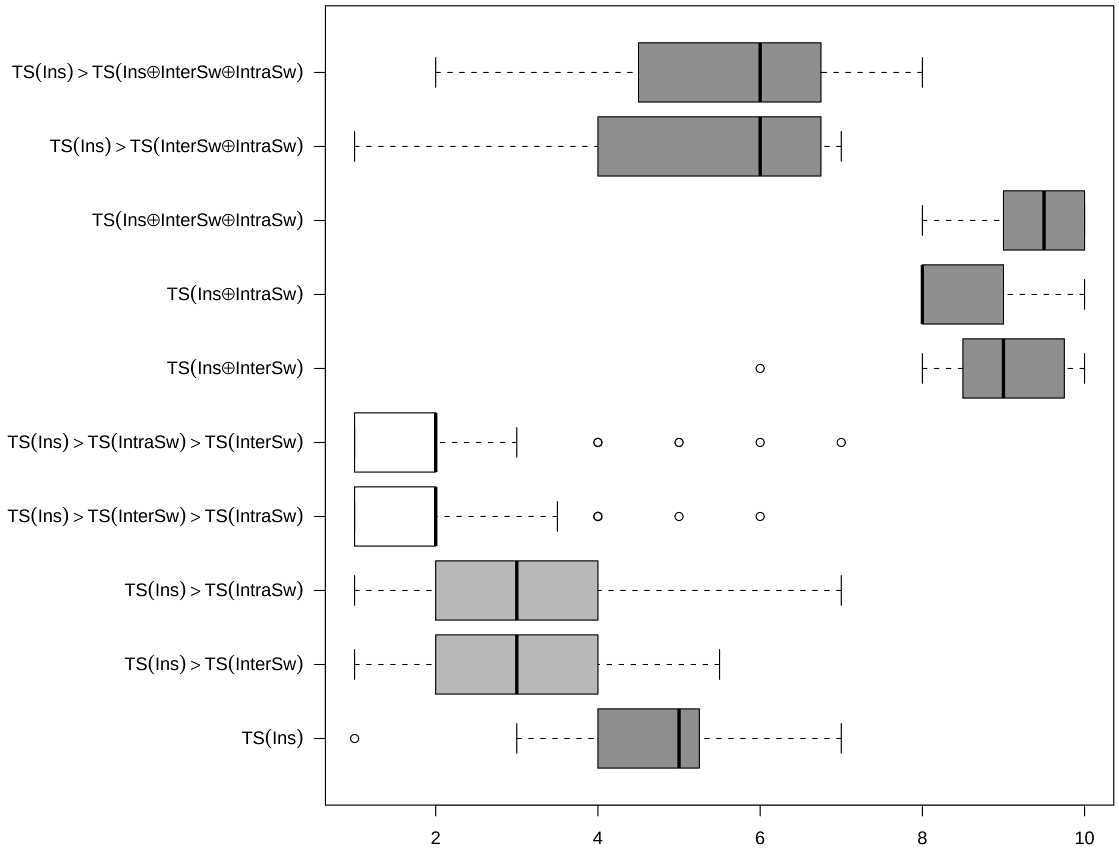


Figure 5.4: Results of the F-Race selection procedure on the composite solvers.

to the value of the cost function. This way we are able to compare the results on different instances independently on the inherent difference in the cost functions. As soon as statistical evidence that a given solver is inferior with respect to the others is collected, it is discarded and the comparison proceeds only among the remaining ones.

We decide to set $p < 0.05$ as for the confidence level employed in the F-Race procedure and we allow for at most 100 replicates.

In this case, in order to compare different solvers in a fair way, we decide to add a timeout mechanism that stops the overall solver when the total time granted is elapsed. Each solver is allowed to run for 500 seconds.

The results are presented in Fig. 5.4, in from of box-and-whiskers plots, showing the distribution of the ranks obtained by each configuration. Moreover, boxes are filled with a gray level which is proportional to the stage of the F-Race procedure in which the corresponding algorithm has been discarded (the darker, the sooner). This way, the algorithms that were found as equally good at the end of the procedure are denoted by white boxes.

The final outcomes of the selection procedure report that only two solvers survive the F-Race, that are TS(Ins) $\triangleright$TS(IntraSw) $\triangleright$TS(InterSw) and TS(Ins) $\triangleright$TS(InterSw) $\triangleright$TS(IntraSw), revealing that all three moves are necessary for obtaining high quality solutions.

Interestingly enough, the solver that uses the union of the three moves does not achieve good results and it has been discarded early by the F-Race procedure. In our opinion, the explanation of this fact is twofold: On the one hand, the use of IntraSw and InterSw moves in the initial stage of the search leads to bad quality local minima, because it tends to optimize single routes before spreading the orders correctly in the various routes. On the other hand, in the later stage of the search, the presence of a large set of IntraSw and InterSw moves having small improvements

Inst.	S1 Date Window	S2 Time Window	S3 Order Outside	S4 Vehicle Fixed Cost	S4 Vehicle Travel Cost	H2 Late Return	Total Cost
case1-A	46.95	101.20	375.25	2700	3205.250	0	6428.650
case1-B	28.38	172.63	0.00	3200	3633.178	1	7034.188
case1-C	291.72	89.33	187.50	3200	3431.114	0	7199.664
case1-D	161.64	110.02	147.50	2600	3703.168	0	6722.328
case1-E	8.43	74.18	40.00	2600	3264.802	0	5987.412
case1-F	24.45	145.21	1039.50	2600	3099.078	0	6908.238
case1-G	14.76	99.93	1149.75	1500	2810.880	0	5575.320
case1-H	714.00	155.81	477.25	2800	3370.175	0	7517.235
case1-I	799.92	74.07	673.25	2500	3874.780	0	7922.020
case2-A	637.68	150.74	11568.75	3200	5197.776	0	20754.946
case2-B	791.82	371.61	0.00	4500	7801.087	8	13464.517
case2-C	316.47	120.85	12631.00	3200	5022.794	0	21291.114
case2-D	583.17	127.72	11501.25	3100	5368.957	0	20681.097
case2-E	462.84	83.88	11847.00	2700	5185.437	0	20279.157
case2-F	437.94	271.39	12263.25	3200	5042.965	0	21215.545
case2-G	237.87	151.19	12860.00	2100	3580.944	0	18930.004
case2-H	1792.44	150.15	11538.50	3000	5390.057	0	21871.147
case2-I	1668.75	213.28	11671.50	3000	5272.349	0	21825.879

Table 5.4: Values (in €) of the different cost components for the best solutions.

prevents the search from making the "disruptive" Ins moves necessary to find deeper local minima. Conversely, the token-ring solvers, by focusing on each single move type, allow the search to perform a more effective diversification at different stages of the search.

Table 5.4 shows the costs of the different components of the objective function for the best solution for each instance. The table reports also the number of (hard) violations: the corresponding value represents the number of orders that are in some routes that return after the shutdown time (H2).

Unsurprisingly, the main cost comes up from the traveling of the vehicles and their fixed cost of use. Among the other components, the most relevant is the one related to unscheduled orders (S3). However, the other two that are related to delivery in the wrong day (S1) and delivery after the end of the time window (S2) are not negligible.

In addition, looking at the different results for instances A–I of the same case, it is evident that the tightness of a specific "resource" (time window, day, ...) makes the cost of the corresponding component higher in the best solution.

Indeed, we can see that if all orders are mandatory (B), the solver is not able to find a feasible solution, and the cost component related to vehicles (S4) increases, because the solver leads towards solutions that use all available resources. The reduction of the planning horizon (C) causes a larger number of deliveries on a wrong day (S1) and higher value of vehicle costs; on the other hand, its extension brings to solutions with less orders unscheduled.

Results highlight that the constraint related to time windows is really tight and its relaxation (E) or restriction (F) has a strong effect on all other cost components. In addition, as we could expect, the lowest value of the total cost comes out for the G, confirming the idea that using the internal fleet is cheaper (if available). Finally, results of the last two cases show that if we schedule all orders only on the first day (H) or on the last day (I) of the planning horizon, that impacts on every cost component, and we obtain solutions with the highest total cost.

We must remark that the effect of perturbations is more evident for case1 than for case2; this is probably due to a higher filling ratio (see Section 5.4.1) for the latter, that leaves less freedom during the search process.

Inst.	CPLEX	Chu (2005)		Bolduc et al. (2007)		Bolduc et al. (2008)		Ceschia et al. 2011	
		z	sec	z	sec	z	sec	z	sec
Chu-H-01	387.5*	387.5	0.02	387.5	0.00	387.5	0.35	387.5	0.11
Chu-H-02	586.0*	631.0	0.03	586.0	0.02	586.0	1.90	586.0	0.70
Chu-H-03	823.5*	900.0	0.08	826.5	0.03	826.5	3.50	823.5	1.96
Chu-H-04	1389.0*	1681.5	0.06	1389.0	0.08	1389.0	5.85	1389.0	7.79
Chu-H-05	1441.5	1917.0	0.28	1444.5	0.09	1441.5	10.40	1441.5	16.93
B-H-01	423.5*	503.0	0.02	423.5	0.02	423.5	1.85	423.5	0.11
B-H-02	476.5*	476.5	0.05	476.5	0.02	476.5	3.65	476.5	0.76
B-H-03	777.0*	884.0	0.11	804.0	0.03	778.5	4.75	777.0	2.13
B-H-04	1521.0*	1737.0	0.06	1564.5	0.09	1521.0	15.85	1521.0	7.81
B-H-05	1609.5	1864.5	0.16	1609.5	0.13	1609.5	12.90	**1578.0**	16.30

Table 5.5: Results on benchmarks Chu-H and B-H of the *Vehicle Routing Problem with Private fleet and Common carrier.*

5.4.6 Comparison with benchmarks of the VRPPC

In order to evaluate the performance of our solver also on public benchmarks, we adapt it to solve the *vehicle routing problem with private fleet and common carrier.*

We test our solver on two instance families, namely Chu-H [50] and B-H [23]. For these instances the number of customers ranges from 5 to 29, the internal fleet is heterogeneous and the external carrier cost was set equal to 6 times the distance between the depot and the corresponding customer. We run one of the two best solvers, namely TS(Ins) ▷TS(IntraSw) ▷TS(InterSw), for 400 trials, and for each instance we set the tabu list length equal to a fifth of the number of orders.

Table 5.5 compares the best solution values obtained by Chu [50], Bolduc et al. [23], and Bolduc et al. [24] with ours published in [40]. The first column reports the solution values computed by Bolduc et al. after a maximum of 150 hours of computation time of CPLEX (v. 9.0). The values marked with * are proven optimal solutions.

The outcome is that our solver is able to obtain the optimum or the best solution value for all instances, and besides, for instance B-H-05 it has found a new best known result.

5.5 Summary

We have modeled a highly complex version of the classical vehicle routing problem, arising from a real world situation and we have proposed an approach based on TS for its solution. To this aim, we have investigated the use of different neighborhoods. The experimental analysis shows that the best results are obtained by a combination of all of them. Finally, we demonstrate that on public benchmark instances of *vehicle routing problem with private fleet and common carrier* our results are very competitive.

6

The multiple container loading problem with bearing weights and multi-drop

6.1 Introduction

In this chapter, we consider a real-world container loading problem, arising from an industrial application, that includes several practical features, such as multiple containers, box rotation, and bearable weight. In addition, the problem takes into account the possibility that the boxes must be delivered in different places (*multi-drop*), thus setting additional constraints on the order of the boxes in the container.

According to the classification of Section 4.2, our problem belongs to the family of container loading problems. However, we first try to classify our problem in the existing literature, reaching the conclusion that it represents a combination of features that has not been explored already. Therefore, no previous approaches are available and, consequently, no benchmark instances.

We solve the problem using a local search approach which works of an indirect search space composed of sequences of boxes, rather than on their physical position. The actual placing of the boxes is performed by a heuristic procedure that loads containers according to the constraints, exploiting as much as possible the presence of sub-sequences of boxes of the same type.

We test our solver on a set of real-world instances provided by our industrial partner beanTech s.r.l. (`http://www.beantech.it`). The outcome is that our solution techniques have been able to find very good solutions on a large variety of practical cases, which improve significantly upon the previous heuristic solution developed by beanTech.

All instances and solutions are available from our dedicated web site `http://satt.diegm.uniud.it/3DPacking`. The web site contains also an application for validating and visualizing new solutions, so as to allow everybody to perform a fair comparison with ours.

In order to have a more measurable assessment of the quality of our solver, we test it also on available benchmarks from the literature. The benchmarks refer to much simpler problems with respect to the one we address, and therefore we solve them by adapting our software, mainly discarding some of the features.

The paper describing this work has been recently published in the *online first* version, therefore all experimental results reported here refer to [39].

6.2 Problem description

We introduce the problem in two stages: we first describe (in Section 6.2.1) what we call the *basic problem*, which does not consider the multi-drop feature. In fact, the latter is the most complex feature to be dealt with and it modifies the model significantly; consequently, it is presented and modeled separately (in Section 6.2.2).

6.2.1 Basic problem

The main entities involved in the problem are *container types* and *box types*:

Container types: each container type is characterized by: dimensions, number of containers, weight limit, and fixed cost of use.

Box types: each box type is characterized by: dimensions, allowed rotations, number of boxes, weight, cost, and bearable weight for each face.

The specific features that are involved in the different formulations have been classified by Bischoff and Ratcliff [20]. According to their terminology, our problem comprises the following ones:

Box rotations (BR): the boxes may be rotated in orthogonal directions; possible rotations are stated for each box type.

Load bearing strength (LBS): the maximum weight per unit area which a box can uphold depends on its type and its vertical orientation.

Full support (FS): a box should not be placed on top of another with a smaller base area; more specifically, both dimensions of the base of the box above must be smaller or equal of the ones of the box below.

Container weight limit (CWL): the sum of the load must not exceed the weight limit of the container.

Using the classification system proposed by Wäscher et al. [174], our problem could be described as a *three-dimensional regular multiple heterogeneous knapsack problem/ three-dimensional regular multiple bin packing problem*, that means that:

- we deal with three-dimensional items,
- at times we use all the containers available to load the maximum volume of boxes so that some boxes are left outside (*output maximization*), at others we use only a subset of containers to load all the boxes (*input minimization*),
- we have many boxes of many different dimensions (*strongly heterogeneous assortment of small items*), and
- multiple containers of different dimensions (*weakly heterogeneous assortment of several large objects*),
- boxes have a rectangular shape (*regular shape of small items*)

The objective function f is the combination of the following components:

C1. Boxes not loaded: cost of boxes that do not fit in the containers.

C2. Container cost: fixed cost for using each container.

C3. Empty linear space: linear space, in the depth direction, that is empty (available for loading unforeseen items).

More specifically, we define f as follows:

$$f = w_1 f_{C1} + w_2 f_{C2} + w_3 f_{C3}$$

where f_{Ci} is the actual cost of the component Ci, and w_i is the corresponding weight.

In order to simplify the complex process of setting the weights w_i and, at the same time, to have an immediate grasp of the costs, we decide to represent the costs directly in a real currency, Euro (€) in our case. To the aim of evaluating the monetary costs of each component, we had to interview our industrial partner and the domain experts. Some costs are relatively easy to be established, such as the cost of renting a truck; on the contrary, some others are rather intangible and we could estimate only the approximate cost. For example, the missed delivery of a box leads to the dissatisfaction of the client thus involving the risk of loosing the client, whose cost is difficult to be interpreted in monetary terms.

It is worth noticing, that the objective function is almost *hierarchical*. In fact, intuitively it is more important to deliver the boxes (C1) than to save a container (C2). In turn, saving a container is more important than having a large empty linear space in all containers (C3). However, the function cannot be treated as totally hierarchical because there are situations in which costs related to C2 are more prominent than those of C1. For example, we prefer to leave a few boxes undelivered rather than moving one container only for them.

To design an objective function that is fine-grained enough, but can be represented in the integer domain (so as to use the faster integer arithmetic), we set the unit of cost to the *thousandth of €* as already done in Chapter 5 for the VRPTWCDC.

6.2.2 Complete problem

In the complete problem, boxes loaded in the container may be delivered to different destinations. The input data are extended by including for the box types also the identifier of the destination $d \in D$, where D is the set of destinations. In the case of boxes of the same dimensions, allowed rotations, number of boxes, weight, cost, and bearable weighs, but different destinations d_1 and d_2, we create separate box types having all data (but the destination and the quantity) identical.

There are two implications of this extension. First, we now aim also to load the containers in such a way that they make the minimum number of stops; this is obtained by preferring solutions that group boxes to the same destination in the same container(s). Second, containers with boxes with multiple destinations must be loaded in such a way that boxes belonging to the earlier destinations can be unloaded without having to move the other ones.

Regarding the first issue, it is clear that, besides the other constraints and objectives, there is also an underlying vehicle routing problem (VRP) to be solved. However, for the practical situations of our industrial partner, distances are very limited and thus the traveling costs are not taken into account. The additional objective that we include is then to minimize the total number of stops of the containers. Ideally, each container should go to one single destination, whereas destinations can be served by many containers.

We therefore introduce a new cost component:

C4. Container stops: sum of the number of distinct destinations of the boxes for each container.

Regarding the second issue, we model it by imposing the constraint that all boxes of a destination should be able to be unloaded *without moving any* of the ones of the subsequent destination.

Multi-drop (MD): if a container carries consignments of different destinations, the boxes should be loaded in such a way that it is possible to set an order of delivery such that no box to a later destination needs to be moved to unload the boxes of the earlier ones.

An example of a container loaded under the multi-drop constraint is shown in Fig. 6.1, where the three destinations are highlighted by the different gray levels.

As will be explained in Section 6.3, this is obtained by allowing only loading sequences that have boxes strictly ordered by destination. Since the loading strategy places the sequences in layers starting from the bottom of the container towards the door, this ensures that constraint MD is always satisfied.

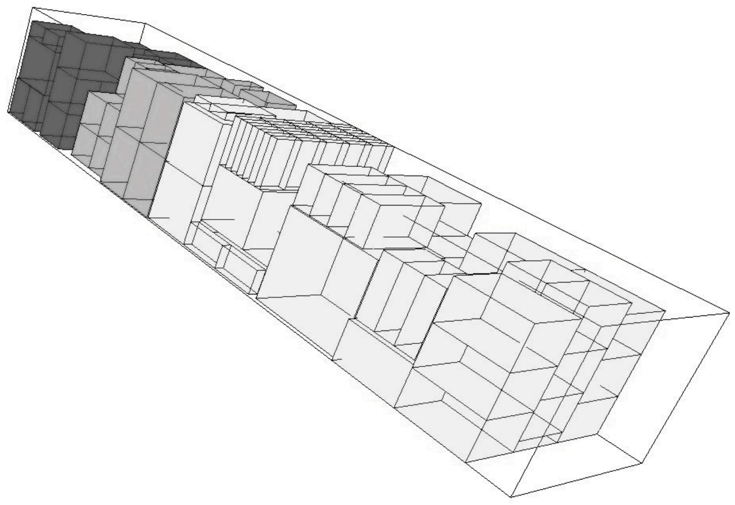


Figure 6.1: Multi-drop container loading.

6.3 Solution for the single destination problem

We first introduce the solution technique for the problem with a single destination, as described in Section 6.2.1. The multi-drop case is dealt with in Section 6.4.

Our solution technique is not a "pure" local search approach in the sense that local search works on the space of sequences of boxes to be loaded. The actual load is obtained by means of a specialized procedure, called *loader*, which is invoked at each iteration for all containers involved in the move. The loader inserts the boxes in the container using a deterministic heuristic strategy which produces a load that is feasible according to all our constraints.

6.3.1 Preprocessing

In some instances, the number of available containers is largely in excess with respect to the boxes to be loaded. In these cases, the inclusion of all containers in the search space would only result in a waste of time for the solver.

The task of the preprocessor is to reduce the set of available containers to a set that any reasonable solution could use. The preprocessor works only on the basis of the volumes of the containers and the total volume of the boxes, without taking into account the actual boxes.

It makes use of a parameter, called α, which is an estimation of the maximum ratio between the volume of a container and the volume of the boxes it actually carries. The value of α is based on runs on known instances, however it can be adjusted in subsequent solutions of a single instance.

The preprocessor uses a simple greedy algorithm, which works as follows.

1. Let V_t be the total volume of the boxes. Set the volume V to be loaded to $V = \alpha V_t$.
2. Select the container with the lowest specific cost defined as the ratio of the fixed cost of use to the volume of the container; set $V = V - V_c$ where V_c is the volume of the container.
3. If $V < 0$ exit, otherwise go back to Step 2.

Containers that are not selected are removed from the input data of the solver.

6.3.2 Search space

We use a local search procedure in which a state in the search space is composed by a set of sequences, one for each container. Each element of a sequence is a *block*, which is a triple composed

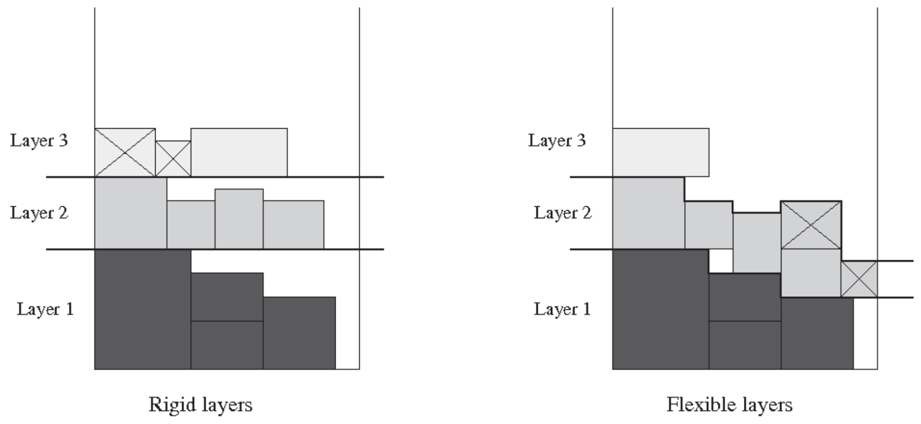


Figure 6.2: Loader with rigid and flexible layers.

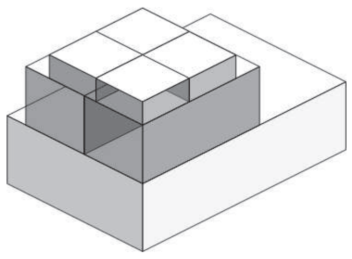


Figure 6.3: Stack construction with aside boxes.

by box type, rotation and number of boxes. A block is also assigned an integer-valued identifier (which is unique in the state).

In any state of the search space all input boxes must be included in exactly one block. However, it is possible that the loader leaves some boxes *in blocks at the end of a sequence* outside the container. This can happen either because there is no physical room for the box or because the weight limit of the container is exceeded. We call the set of unloaded boxes the *tail* of the container, and the sum of the costs of all tails constitute the component C1 of the objective function.

6.3.3 Loader

The loader is based on the wall building approach by George and Robinson [92] mentioned in Section 4.3. It fills the container in a number of *layers* across the depth of it. Every layer is divided in vertical *stacks* and each stack is then packed by consecutively inserting boxes, keeping fixed the loading order of the blocks.

The depth of a layer is set to be the depth of the first box that is inserted in that layer. For all the following boxes, if they do not fit in that depth they are moved to a new layer.

Similarly, if a box does not fit on top of the previous one in a stack, it is moved to a new stack or (if there is not enough room) to a new layer. A box might not fit in a stack either because its height exceeds the container ceiling, or because the base area of the box below is insufficient in at least one dimension, or because the weight is not bearable by one of the boxes below in the stack.

As already proposed by George and Robinson [92] and Moura and Oliveira [132], the layers are *flexible* in the sense that boxes are allowed to slide down in the previous layer if there is enough room. Fig. 6.2 shows the loading of a container viewed from the top, and it helps to clarify this point. The sliding down of the boxes of layer 2 has created the space for the two boxes marked with the cross, which are inserted in layer 2 as well.

Only for boxes belonging to the same block (same dimensions and rotation), we try to put a box *aside* the previous one if the composite base area is inside the one of the box below and the weight is bearable. See in Fig. 6.3 an example in which two boxes are side by side above a single one, and four boxes are in turn above the merge of these two.

We do not put aside boxes of different types because of stability problems; that is, the boxes above might not have an horizontal base to lay on. Note that boxes of different types could in principle be placed aside if they are on the highest layout (just below the ceiling of the container). However, experiments with this option showed that this does not lead to better solutions.

Notice that the loader could be improved by selecting the next box to be placed in the container using some sort of *best fit* strategy, instead of using the strict order induced by the sequence. The loader however is not the “optimizer”, but it is just a module of the overall procedure. Indeed, our idea is precisely that the loader should be simple and fast, and the discovery of improvements is totally demanded to the main local search procedure that performs the changes in the sequence given to the loader.

6.3.4 Cost function

All constraints are enforced by construction, thus the cost function that guides the search is the same as the objective function of the problem; it is a weighted combination of C1, C2, and C3 (remember that C4 is related only to the multi-drop case).

However, an auxiliary component C5 is added to it in order to lead the search toward states that use less containers. Specifically, the component C5 sums up for each container the *square* of its volume minus the *square* of the volume of the boxes loaded. We use a quadratic term in order to privilege solutions with asymmetric loading of containers, so that it would eventually lead to freeing completely the least loaded ones, rather than keeping the load balanced among them.

It is worth noticing that components C2 and C3 are not sufficient to obtain this behavior. In fact, they come into play only on the removal of the last block from a container and the last block of the layer, respectively. In the other states, they do not “push” in the direction to have a container with less blocks. Notice also that some containers are already removed by the preprocessor; however, the preprocessor works on a rough overestimation of the loading level (α), whereas here we work on the actual situation.

Indeed, experiments without C5 show a substantial increase of the number of containers used in the final solutions, and a corresponding increase of the total cost.

6.3.5 Neighborhood relation

The neighborhood consists in moving a block, or a portion of it, in a different container and/or different position, possibly with a different orientation. Fig. 6.4 shows an example of a move in which part of a block is moved from container 1 to container 2.

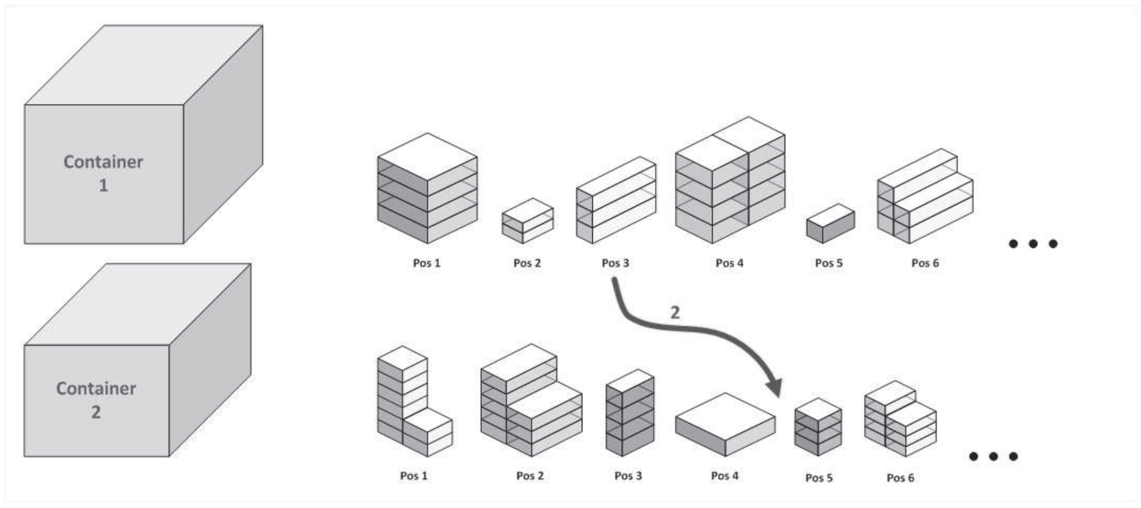
A move is thus represented by five attributes: block identifier, new container, new position, new rotation, and quantity of boxes moved. The quantity varies from 1 to the size of the block.

When only a portion of a block is moved (i.e., the quantity is less than the size of the block), the block is split and thus a new block with a new identifier is created. Conversely, when a block is moved and the two blocks before and after it are compatible (i.e., same box type and rotation), they are merged summing up their quantities and keeping the identifier of the block before. Similarly if a block is moved to a position adjacent to a compatible one, they are merged, keeping the identifier of the one already there.

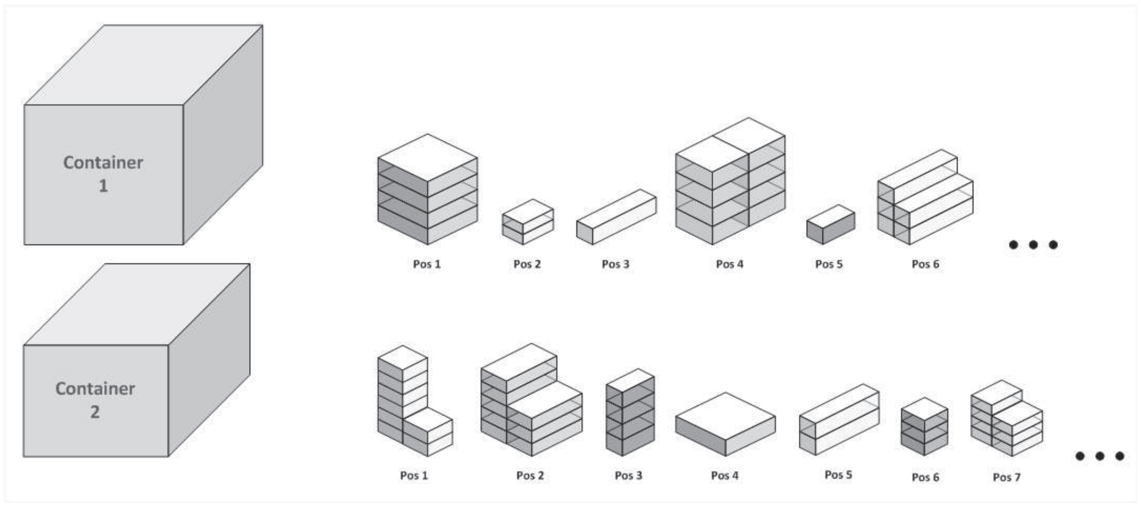
The neighborhood is *restricted* in such a way to remove moves that are clearly useless for the local search. In detail, we exclude the following types of moves:

Tail-to-tail moves: a move that transfers a block from the tail of a container to the tail of another one. This kind of moves do not change the cost of the objective function.

Duplicated moves: a move that leads to the same state of another move. This are the moves that move a full block after the one that immediately follows it: the same movement is already obtained by moving the second block one position backward (Figure 6.5).



Initial state



Final state

Figure 6.4: An example of move.

Null moves: a move that leads to the same state as the current one. These are the moves that either the new container, the new position, and the new rotation are all the same of the current ones of the block or they move part of a block just after the block itself (which then is rebuild by merging).

When a move is evaluated, the loader is called only on the two containers (or one if the move is internal to one container) involved in the move. In addition, only the part of the load starting from the stack that includes the box being moved is recomputed. The preceding part is not affected by the move and thus it is left unchanged.

6.3.6 Initial solution

The initial solution is obtained by creating one single block for each box type. The container, the rotation, and the position in the container of each block is selected at random. The rotation is selected only among the feasible ones for that box type.

The rationale behind this choice of having one block per type is that intuitively we should keep the blocks as big as possible, because it is easier to pack boxes of the same type when they are

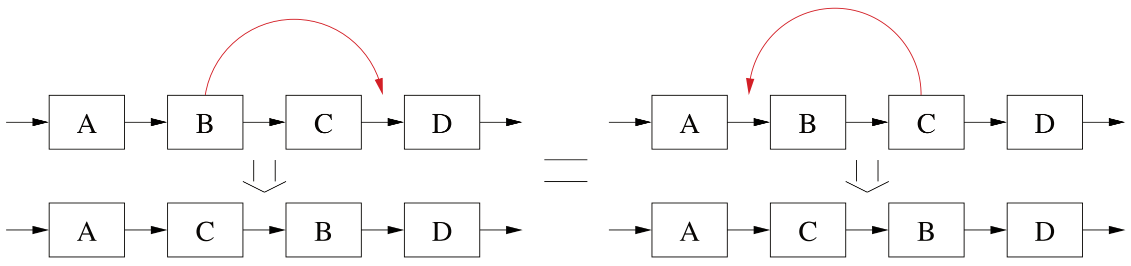


Figure 6.5: An example of duplicated move

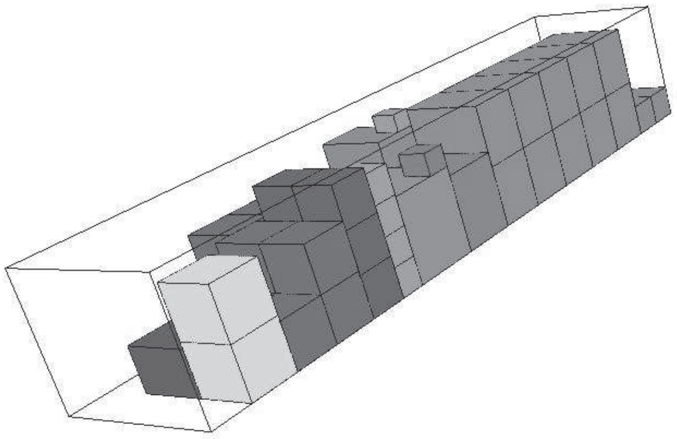


Figure 6.6: Example of initial solution.

close to each other. The local search procedure would then split the blocks, when this leads to better solutions. Figure 6.6 shows an example of initial solution.

6.3.7 Metaheuristics

We developed two main solvers: one based on a TS algorithm and the other on a SA algorithm. The details of our implementation of the algorithms are described in Section 2.2.2 and 2.2.3. Both solvers are equipped with the neighborhood described in Sections 6.3.5 only.

6.4 Solution technique for the multi-drop problem

For the multi-drop case, the idea is to solve a multi-drop instance by solving a sequence of *sub-instances* obtained by adding one destination at the time.

Letting $D = \{d_1, \ldots, d_n\}$ be the set of destinations, we sequentially solve n sub-instances, such that sub-instance i (with $i = 1, \ldots, n$) is obtained considering only the box types belonging to the destinations d_j, such that $j \leq i$.

Each sub-instance i is solved by the technique presented in Section 6.3, with some modifications to take care of the MD constraint and the cost component C4. The modifications are the following:

Reserved containers: If in the final solution of sub-instance i a given container c is fully loaded, then c is *reserved* to the destinations that are in c in that solution. This means that during the solution of all sub-instances j, with $j > i$, only box types of those destinations can be inserted in c.

Incremental initial solution: The initial solution of sub-instance i is obtained by starting from the best solution of instance $i - 1$ and adding to it the blocks of destination d_i at random

Family		Conditions		Size				Cost
Name	#I			#CT	#C	#BT	#B	components
CS1	71	$f_{c1} = 0$	$f_{c2} = \gamma$	1–3	1–100	1–124	4–2760	C3
CS2	12		$f_{c2} < \gamma$	1–2	100–200	10–90	23–212	C2 and C3
CS3	31	$f_{c1} > 0$	$f_{c2} = \gamma$	1-3	1–100	5–97	27–1612	C1
CS4	3		$f_{c2} < \gamma$	1-2	2–5	2–47	82-1439	C1 and C2

Table 6.1: Families of instances.

only at the end of the sequences of non-reserved containers. For sub-instance 1, the initial solution is obtained in the same way of Section 6.3.6.

Neighborhood restrictions: The local search procedure takes into account the multi-drop constraint by prohibiting to move a box of destination d_i in container c in position p if one of the following conditions holds:

- the container c is reserved and d_i does not belong to its reservation list,
- a box of destination d_j (with $j < i$) is in container c in a position which follows p.

This technique proved experimentally to be more performing than solving the overall problem with a single local search step.

6.5 Experimental analysis

In this section, we first describe the instances used in the experiments. Secondly, we show the results of our methods on these instances. Lastly, we show the comparison between our methods and the ones in the literature on simpler problems.

6.5.1 Instances

We experiment on a set of 117 real-world instances coming from the clients of our industrial partner beanTech. All instances are available at the URL `http://satt.diegm.uniud.it/3DPacking`.

The instances exhibit different container types and a high variability in terms of number of box types and quantity of each type. This large variability can be expressed in terms of the ratio between the total volume of the boxes and the total volume of the containers, that we call θ. Normally if $\theta \ll 1$, we expect that all boxes fit into the containers, and consequently, $f_{C1} = 0$ and the other components become significant. Conversely, when $\theta \geq 1$, the cost function is dominated by f_{C1} and the other components are almost negligible. For the values $\theta < 1$ (but $\theta \not\ll 1$) it depends on the specific instance whether f_{C1} is zero or not.

A further distinction can be made between instances in which all containers are always used, for which f_{C2} is fixed to the total cost of the containers (called γ) and the ones for which f_{C2} varies from run to run, and is thus meaningful.

We thus classify our instances in four families, called CS1–CS4, based on the two conditions that the best known solution has $f_{C1} = 0$ or not, and $f_{C2} = \gamma$ or not, respectively. Table 6.1 shows for each family the number of instances belonging to it, the ranges in terms of containers (#C), container types (#CT), box types (#BT), and total boxes (#B), and the cost components that are most meaningful for that family.

6.5.2 Parameter setting

We now discuss the settings of the parameters of the metaheuristics.

Our TS implementation employs a dynamic short-term tabu list so that a move is kept in the tabu list for a random number of iterations in the range $[tt_{min}, tt_{max}]$ and the search stops after a fixed number of *idle* iterations (ii_{max}). Therefore the parameters to set are tt_{min}, tt_{max} and ii_{max}.

	C3			
Technique	Avg	Dev	t-test	Running times (secs)
SA	**338.256**	**9.177**	–	**39.41**
TS	351.999	10.235	0.044	83.05

Table 6.2: Results of the C3 component on family CS1.

	C2		C3			
Technique	Avg	Dev	Avg	Dev	t-test	Running times (secs)
SA	**0.088**	**0.0082**	**158.802**	**4.421**	–	**38.95**
TS	0.096	0.016	161.263	4.557	0.16	86.42

Table 6.3: Results of the C2 and C3 components on family CS2.

The SA algorithm requires a starting temperature T_0, a final temperature T_{min} and a number of neighbors sampled at each temperature level N_σ. The value of T is modified using a *geometric* schedule, i.e., $T' = \beta \cdot T$, in which the parameter $\beta < 1$ is the cooling rate.

A tuning phase has been performed to obtain the configuration of values that would be the best for the entire data set CS1–CS4. The settings that obtained the best results are the following:

for SA: $T_0 = 50$, $T_{min} = 0.001$, $\sigma_N = 1000$, and $\beta = 0.998$

for TS: $tt_{min} = \lfloor \delta \rfloor / 5$, $tt_{max} = tt_{min} + 2$, and $ii_{max} = 500$, where δ is the number of blocks in the current solution; as for the inverse relation, a move is prohibited if it involves the same block of a move in the tabu list.

Notice that δ varies from state to state, so that the tabu list length is *adaptive*. The results of the following section are obtained with the above configurations.

6.5.3 Experimental results

We present in Tables 6.2–6.5 the results separately for each family. As mentioned in Section 6.2.1, we chose to express the cost of the components of the objective function in €.

To compare the results we use the *Student's t-test*, provided that the underlying distributions can be assumed to be normal and independent [170]. If the calculated p-value is below the threshold chosen for statistical significance (typically $p < 0.05$), then the *null hypothesis*, which states that the two groups do not differ, is rejected in favor of an alternative hypothesis, which states that an algorithm is superior to another on the proposed instances.

The column t-test shows the p-value of the comparison between best configuration (marked with a dash) and the other one.

Looking at Tables 6.2–6.4, it is clear that for these instances SA is the technique that works better than TS, although the confidence is not always high enough to reach a definitive conclusion. On the other hand, TS preforms well only on instances of family CS4 which is made of only 4 instances.

The reason for these bad performances of TS is in our opinion mainly due to the computational cost of the full exploration of the neighborhood, which evidently turned out to be less effective than the random selection of SA.

	C1			
Technique	Avg	Dev	t-test	Running times (secs)
SA	**2572.517**	**167.133**	–	**52.78**
TS	2662.959	164.012	0.060	100.31

Table 6.4: Results of the C1 component on family CS3.

	C1		C2			
Technique	Avg	Dev	Avg	Dev	t-test	Running times (secs)
SA	715.346	119.590	0.183	0	0.277	97.39
TS	**695.717**	**156.689**	**0.183**	**0**	–	**153.37**

Table 6.5: Results of the C1 and C2 components on family CS4.

Family		Size			Features					Objectives			
Name	#I	#C	#BT	#B	BR	LBS	FS	MD	CWL	C1	C2	C3	C4
IMM	47	2–55	2–5	47-150	$\surd$	—	$\surd$	—	—	—	$\surd$	—	—
BR	1500	1	3–100	69–476	$\surd$	—	$\surd$	—	—	$\surd$	—	—	—

Table 6.6: Features of the families IMM and BR1–BR15.

The low value of the component C2 in Table 6.3 that computes the fixed cost of using the containers, can be explained by the fact that, for the instances of family CS2 there are always available *in-house* containers (owned by the company), whose cost was set to the fictitious value of 1 thousandth of € by the operator of the company.

6.5.4 Comparison with related work

Given that the problem is new, it is not possible to compare with other researchers on this problem. However, in order to have an assessment of the quality of the solver, we use it to solve simpler problems for which previous results are available.

To this aim we use two public benchmarks: the family IMM proposed by Ivancic et al. [108], and BR defined by Bischoff and Ratcliff [20]. Our results reported here have been submitted to Journal of Heuristics in 2010 and recently published in [39].

Table 6.6 summarizes for each family the number of instances (#I), the ranges of the instances in terms of containers (#C), box types (#BT), and total boxes (#B). The following columns represent the problem features, and the symbol $\surd$ means that it is present in the family. Finally, we have the cost components, and here $\surd$ means that it has been considered by the papers that solve the instances of that family. It is evident that the families IMM and BR cover only a small subset of the features of this work set and exhibit a much smaller variability of size.

We start discussing the results of IMM instances, which consider one container type (i.e., all containers are identical), and as objective the number of containers used (i.e., C2 with all costs equal to 1).

We report in Table 6.7 the number of containers used in the best solution found by our SA solver, along with the results in the literature. A brief description of the approaches with which we compare is in Section 4.3. The average running time for a trial of a single instance for SA is 289.37 secs. The best known result is shown in bold; in addition, when it is proven optimal Eley [by 76] it is marked with *. The values marked with the symbol $\flat$ have been inferred by us, as in [75] it is reported a value that is lower than the lower bound subsequently proven by Eley [76] himself.

Keeping in mind that we can obtain only an approximate comparison because we do not have information about the running times and the distributions of all the other solvers, looking at the bottom line of Table 6.7, it is clear that

Our solver has been able to obtained the best known result for most of the instances. Indeed, our results improve significantly on the original works by Ivancic et al. [108] and Bischoff and Ratcliff [20], and upon the recent works by Bortfeldt [26] and Eley [76]. However, our solver has been recently outperformed by the solution technique of Zhu et al. [178].

All our best solutions to the IMM instances, along with the input files in our format, are available at our web site `http://satt.diegm.uniud.it/3DPacking` for verification.

The BR dataset is divided into 15 families of 100 instances each, depending on the number of

Test case	Ivancic et al. (1989)	Bischoff and Ratcliff (1995)	Bortfeldt (2000)	Eley (2002)	Eley (2003)	Ceschia and Schaerf (2011)	Zhu et al. (2011)
IMM1	26	27	**25***	26	**25***	**25***	**25***
IMM2	11	11	**10**	**10**	**10**	**10**	**10**
IMM3	20	21	20	22	20	**19**	**19**
IMM4	27	29	28	30	**26***	**26***	**26***
IMM5	65	61	**51**	**51**	**51**	**51**	**51**
IMM6	**10***	**10***	**10***	**10***	**10***	**10***	**10***
IMM7	**16***	**16***	**16***	**16***	**16***	**16***	**16***
IMM8	5	**4***	**4***	**4***	**4***	**4***	**4***
IMM9	**19***	**19***	**19***	**19***	**19***	**19***	**19***
IMM10	**55***	**55***	**55***	**55***	**55***	**55***	**55***
IMM11	18	19	18	18	**17**	**17**	**17**
IMM12	55	55	**53***	**53***	**53***	**53***	**53***
IMM13	27	**25**	**25**	**25**	**25**	**25**	**25**
IMM14	28	**27***	28	**27***	**27***	**27***	**27***
IMM15	**11***	**11***	**11***	12	**11***	**11***	**11***
IMM16	34	28	**26***	**26***	**26***	**26***	**26***
IMM17	8	8	**7***	**7***	**7***	**7***	**7***
IMM18	3	3	**2***	**2*♭**	**2***	**2***	**2***
IMM19	**3***	**3***	**3***	**3*♭**	**3***	**3***	**3***
IMM20	**5***	**5***	**5***	**5*♭**	**5***	**5***	**5***
IMM21	24	24	21	26	**20**	**20**	**20**
IMM22	10	11	9	9	**8***	**8***	**8***
IMM23	21	22	**20**	21	**20**	**20**	**20**
IMM24	6	6	6	6	6	6	**5**
IMM25	6	**5**	**5**	**5**	**5**	**5**	**5**
IMM26	**3***	**3***	**3***	**3***	**3***	**3***	**3***
IMM27	5	5	5	5	5	5	**4**
IMM28	**10**	11	**10**	**10**	**10**	**10**	**10**
IMM29	18	**17**	**17**	18	**17**	**17**	**17**
IMM30	24	24	**22**	23	**22**	**22**	**22**
IMM31	13	13	13	14	13	13	**12**
IMM32	5	**4***	**4***	**4***	**4***	**4***	**4***
IMM33	5	5	5	5	5	**4**	**4**
IMM34	9	9	**8**	9	**8**	**8**	**8**
IMM35	3	3	**2***	**2***	**2***	**2***	**2***
IMM36	18	19	**14***	**14***	**14***	**14***	**14***
IMM37	26	27	**23***	**23***	**23***	**23***	**23***
IMM38	50	56	**45**	**45**	**45**	**45**	**45**
IMM39	16	16	**15**	**15**	**15**	**15**	**15**
IMM40	9	10	9	9	**8**	**8**	**8**
IMM41	16	16	**15**	**15**	**15**	**15**	**15**
IMM42	**4***	5	**4***	**4***	**4***	**4***	**4***
IMM43	**3***	**3***	**3***	**3***	**3***	**3***	**3***
IMM44	4	4	**3**	4	4	**3**	**3**
IMM45	**3**	**3**	**3**	**3**	**3**	**3**	**3**
IMM46	**2***	**2***	**2***	**2***	**2***	**2***	**2***
IMM47	4	**3***	**3***	**3***	**3***	**3***	**3***
All cases	763	763	705	721	699	696	**693**

Table 6.7: Results for family IMM.

box types (shown in parentheses in Table 6.8). All instances have one single container and one single objective, namely C1. No cost is associated to the boxes, therefore the objective function is based on the box volume (volume utilization).

The results are shown in Table 6.8. Again, the comparison is approximate because we do not have access to all data about the distributions and the running times of all the others. In any case, the outcome is that our solver improves on the original results of Ratcliff and Bischoff [143] and achieves results quite close to the heuristic of Eley [75] and the GRASP approach of Moura and Oliveira [132]. Unfortunately though, it is clearly outperformed by Bortfeldt and Gehring [28] and Bischoff [18], and the recent results by Fanslau and Bortfeldt [77] and Zhu et al. [178].

However, all the techniques reported in Table 6.8 are specifically designed and tuned for the case that involves only one container, whereas our technique solves the more general problem that deals with bearing weights, different container types, multi-drop and other features, as previously described in Section 6.2.2. In addition, our solution is designed to work equally well in the case of weakly heterogeneous cargoes, with large quantities of identical boxes, and strongly heterogeneous ones composed of many different box types.

For instance, Eley takes advantage of the fact that the loading is weakly heterogeneous by considering a limited number of potential arrangement of pre-designed homogeneous blocks of boxes. Fanslau and Bortfeldt extend the classical block build approach in order to face also with a situation of strongly heterogeneous cargo. At each stage of the solution process their technique generates the packing patterns that come out from the arrangement of blocks made up of different box type and orientation. Nevertheless, this procedure could result impracticable in case of simultaneous loading of multiple containers, thus the number of possible arrangements for each residual space would explode.

It is clear that these approaches would not be applicable in our case, because our instances have a much larger variability in terms of number and types of boxes and containers, as clearly shown in Table 6.1.

Other authors [e.g., 126] have solved these instances finding also higher percentages of volume occupation. For example, Parreño et al. [138] reach an average filling of 94.53 for cases BR1-7. However, they consider a different notion of stability, in which the base of a box does not need to be fully supported by the box below, but its base can be also partly outside it. Same comment can be drawn also for the cutting variant of the algorithm proposed in [77], which was able to obtain the current best results with an average volume utilization of 93.8 through all the family BR. For this reason, they are not included in Table 6.8.

6.6 Summary

We have developed a set of techniques for solving a very complex packing problem, including many features and cost components. Our solvers have been able to deal with a broad assortment of different practical situations, ranging from cases in which there is an unlimited availability of containers to cases in which it is not possible to load all the boxes.

With respect to the product currently in list for beanTech, the solver captures a much richer model as it adds the multi-drop feature, the bearing strength, and the costs of containers and boxes. In addition, it improves or equals its results on all instances. The average improvement is about 13% on the tested instances. For the sake of measurability, all the CS instances are available on the web, along with our solutions and a validator for new solutions.

The experiments have shown that on our complete problem SA outperforms TS in most cases. They also demonstrate that on public benchmarks our results are very competitive for the multi-container cases, whereas they still worse than some in the literature for the single container ones.

Test case (# box types)	Bischoff and Ratcliff (1995)	Gehring and Bortfeldt (2002)	Eley (2002)	Moura and Oliveira (2005)	Bischoff (2006)	Fanslau and Bortfeldt (2010)	Ceschia and Schaerf (2011)	Zhu et al. (2011)
BR1(3)	83.79	88.10	88.05	89.07	89.39	**94.51**	90.31	93.57
BR2(5)	84.44	89.56	88.44	90.43	90.26	**94.73**	90.75	93.87
BR3(8)	83.94	90.77	89.23	90.86	91.08	**94.74**	90.52	94.14
BR4(10)	83.71	91.03	89.24	90.42	90.90	**94.41**	89.84	93.86
BR5(12)	83.80	91.23	88.99	89.57	91.05	**94.13**	89.16	93.51
BR6(15)	82.44	91.28	88.91	89.71	90.70	**93.85**	88.73	93.39
BR7(20)	82.01	91.04	88.36	88.05	90.44	**93.2**	87.68	92.68
Cases BR1–7	83.45	90.43	88.75	89.73	90.55	**94.22**	89.57	93.57
BR8(30)		90.26		86.13		**92.26**	86.36	
BR9(40)		89.50		85.08		**91.48**	85.51	
BR10(50)		88.73		84.21		**90.86**	84.49	
BR11(60)		87.87		83.98		**90.11**	83.58	
BR12(70)		87.18		83.64		**89.51**	83.21	
BR13(80)		86.70		83.54		**88.98**	82.41	
BR14(90)		85.81		83.25		**88.26**	81.91	
BR15(100)		85.48		83.21		**87.57**	80.77	
Cases BR8–15		87.69		84.13		**89.88**	83.53	
All cases		88.97		86.74		**91.91**	86.35	

Table 6.8: Results for family BR.

7

A routing and packing problem

7.1 Introduction

In Section 3.2.7 we have summarily introduced the three-dimensional loading capacitated vehicle routing problem (3L-CVRP), which can be considered as a combination of two optimization problems: the capacitated vehicle routing problem and the three-dimensional bin packing problem.

The work described in this chapter aims of solving an integrated real-world problem in logistics, which can be seen as a complex variant of the 3L-CVRP. Our project includes many real world features, some of which have been already described in Chapter 5 for the VRPTWCDC and Chapter 6 for the MCLPBWMD in isolation, but have never been considered together. Indeed, as is shown in Figure 1.1, the problem addressed in this chapter can be considered the natural extension and integration of the two previous ones.

As our problem is a combination of two strongly NP-hard problems, it appears to be appropriate to tackle it by using metaheuristics techniques, in particular to solve large instances in reasonable computational time. We thus propose a LS approach based on a combination of simulated annealing and large-neighborhood search that considers the overall problem in one single stage, by including neighborhood relations that modify both the routes and the container loadings.

The algorithm was tested both on a set of real-world instances and on available benchmarks from the literature. All instances and best results are available at `http://www.diegm.uniud.it/ceschia/index.php?page=vrclp` for future comparisons.

A preliminary work on this problem has been presented on the *9th Metaheuristic Conference (MIC 2011)* [41].

7.2 Problem description

We introduce the problem in stages. Starting from the 3L-CVRP (Section 7.2.1), we describe progressively the new features in order to be able to fully define the complete problem that we deal with (Section 7.2.2).

7.2.1 3L-CVRP formulation

The 3L-CVRP formulation is provided by Gendreau et al. [91]. We report it here in order to make the paper self-contained and facilitate comparisons to the model we are proposing. The main entities involved in the problem are:

Clients: It is given a set of customers $\mathcal{V} = \{0, 1, \ldots, n\}$ each one corresponding to a vertex of the graph $G = (\mathcal{V}, \mathcal{A})$. The depot is treated as a special customer and it is identified with vertex 0. Each edge (i, j) of the graph has an associated cost c_{ij}, which is the cost of traveling from customer i to customer j.

Items: Each customer $i \in \mathcal{V} \setminus \{0\}$ requires a supply of m_i items whose total weight is q_i. The demand of customer i is made up of *items*, that is, three-dimensional rectangular boxes each one having width w_{ik}, height h_{ik}, and length l_{ik} (with $k = 1, \dots, m_i$). Therefore, the total volume needed by customer i is $s_i = \sum_{k=1}^{m_i} w_{ik} h_{ik} l_{ik}$. In addition, a flag f_{ik} is associated to each item, such that if $f_{ik} = 1$ the corresponding item is fragile. The total number of items for all customers is $M = \sum_{i=1}^{n} m_i$.

Vehicles: The delivery of items is performed by a fleet $\mathcal{F} = \{1, \dots, v\}$ of identical vehicles, each one with weight capacity Q. The three-dimensional rectangular loading space has width W height H and length L, so the total loading volume available for each vehicle is $S = W \cdot H \cdot L$.

The solution of a 3L-CVRP calls for the determination of a set of routes that minimizes the total transportation cost and satisfy all the typical constraints of a vehicle routing and packing problem, as follows:

1. the number of routes must be at most equal to the size of the fleet, i.e. one route per vehicle;
2. each route must start and end at the depot;
3. each customer must be visited exactly once;
4. the demands of all customers must be fulfilled;
5. for each route the total demand weight must not exceed the weight capacity of the vehicle;
6. items must be stowed completely in the vehicle;
7. no two items can overlap;
8. the loading must be orthogonal, i.e. the edges of an item must lie parallel to the edges of the vehicle.

Furthermore, for each route the loading must be feasible according to these additional conditions:

Fixed vertical orientation (C1): Items can be rotated by 90° on the *width-length* plane, keeping fixed the vertical orientation.

Fragility (C2): Non-fragile items cannot be placed on the top of fragile ones.

Minimum supporting area (C3): The base of each item must be supported by other items or by the vehicle's floor at least by a minimum supporting area, which is proportional to the bottom side of the item.

LIFO policy (C4): The loading order is the inverse of the customers' visit order. In such a way it is possible to unload items of a customer without moving items belonging to other customers that will be visited later. The unloading procedure is performed through straight movements parallel to the length and height plane.

7.2.2 Complete problem

We now describe the formulation of the complex real-world problem that we tackled, which extends the 3L-CVRP in several directions.

Weakly heterogeneous cargo and heterogeneous fleet

Firstly, the 3L-CVRP does not consider the possibility of having a cargo composed by large groups of items of the same dimensions and a heterogeneous fleet.

Indeed in the 3L-CVRP benchmarks, there are only a few large items, all different from each other. However, in many real cases, there is a large number of identical items to be loaded. As examples, we refer to the real-world instances described in our work on MCLPBWMD [39] and those used by Moura and Oliveira [133]. In addition, the fleet may be composed by vehicles that are not identical. This leads to the definition of new problem entities:

Item types: Each item i (with $i = 1, \dots, M$) belongs to an item type j (with $j = 1, \dots, it$), which is characterized by: dimensions (w_j, l_j, h_j), allowed rotations (uw_j, ul_j, uh_j), number of items of this type (μ_j), weight (wg_j) and bearable weight for each face (bw_j, bl_j, bh_j).

The allowed rotations are denoted by the binary-valued parameters uw_j, ul_j, uh_j, which are given a value of 1 if the corresponding side can be positioned upright, and 0 otherwise. The bearing weight is the maximum load, expressed in units of weight per area, which may be placed on the top surface of the item when the corresponding edge is placed upright. Its role in the problem formulation will be explained in Section 7.2.2.

Vehicle types: Each vehicle i (with $i, \dots, v$) belongs to a vehicle type j (with $j = 1, \dots, vt$), which is characterized by: dimensions (W_j, L_j, H_j), number of vehicles of this type (ν_j), weight capacity (Q_j) and possible fixed cost of use (C_j).

In case of weakly heterogeneous cargo, an efficient loading can be usually obtained by grouping items of the same type in homogeneous blocks [75, 77]. Nevertheless, in our model items belonging to different customers can be of the same type, thus a block homogeneous with respect to the item type could be heterogeneous with respect to deliveries (i.e. customers). Therefore the packing strategy has to be modified in order to take into account of different deliveries for a single block, so as to not violate the C4 constraint.

Load bearing strength

The definition of item fragility and the corresponding constraint C2 is rather simplistic. It needs to be reformulated by means of introducing the concept of maximum supported weight (that could be proportional to the weight of the item) whose notion has been already introduced in Section 4.2.1. Indeed, Figure 7.1 highlights a doubtful situation: box 1, which is not fragile, is supported at least for the 75% of its base area by box 3, which is not fragile too. Nevertheless, box 2 is as high as box 3 and it is pushed near box 2 and under box 1. This placement generates a fragility violation because box 2 is fragile and box 1 is not, although it could reasonably be expected that box 1 is mainly supported by box 3, independently of box 2.

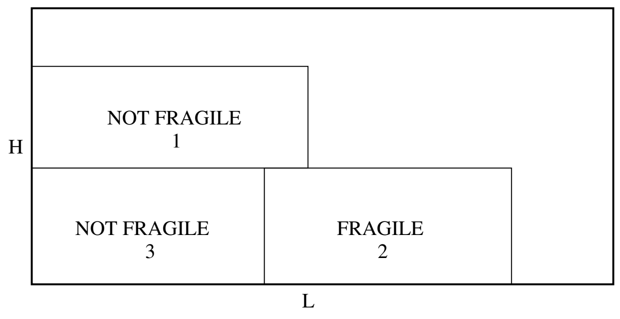


Figure 7.1: Violation of the Fragility constraint.

Therefore, the qualitative notion of fragility is replaced by the quantitative one of bearing strength, which leads us to replace the C2 constraint with the following one:

Load bearing strength (C5): There is a maximum weight per unit area which a box can uphold depending on its type and its vertical orientation.

Cargo stability

The proposed definition of stability based only on the minimum supporting area between one item and the underlying one can actually lead to skewed item stacks that are actually unstable (see Figure 7.2). Therefore, the notion of stability has been reformulated and the constraint C3 was changed to:

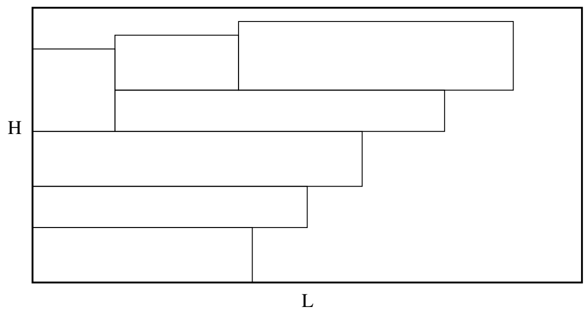


Figure 7.2: Skewed item stack

Robust stability (C6): A minimum supporting area has to be guaranteed for all items below the current one in the stack. This means that for each item, the constraint C3 is applied not only to the underlying item, but also to all items below it.

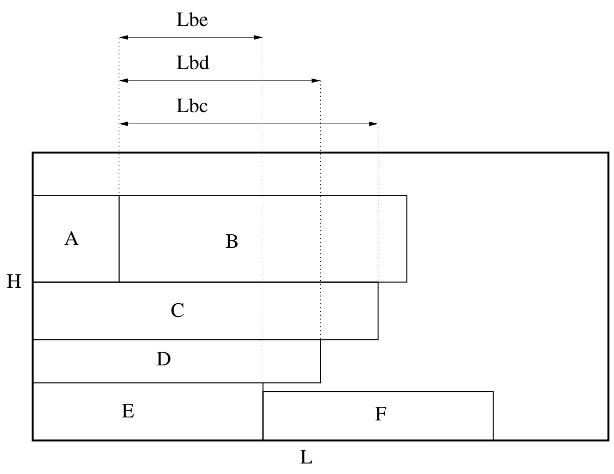


Figure 7.3: The Robust Stability constraint

Figure 7.3 highlights the difference between the standard Minimum supporting area (C3) constraint and the Robust stability (C7) constraint. Assuming that all items in the Figure 7.3 have the same width, the supporting area depends only on the length, so the label Lij marks the part of the item j that supports item i. The C3 constraint establishes that the stability of box B has to be calculated considering only box C, which is immediately below it. In contrast, the C6 constraint requires that the stability of the box B is computed taking into account items E and D, too. In this case, L_{be} is too short thus box B is not stable for C6 while L_{be} is long enough to guarantee stability according to C3.

The LIFO constraint

The LIFO constraint, as formulated in the 3L-CVRP, is not realistic, in the sense that it does not capture correctly the physical constraints that allow the human operator or machine to unload an item without having to move the others.

In detail, the LIFO constraint proposed by Gendreau et al. [91] establishes that any item of a customer visited later than the current one, must not be placed above a box of the current customer or between it and the rear of the vehicle. A variant of this formulation has been already proposed by Tarantilis et al. [158] leading to the problem called Manual 3L-CVRP (or M3L-CVRP) in order to capture situations where boxes are manually unloaded. In this case, boxes are not necessarily elevated before pulling them out of the vehicle, thus it is possible to place an item of a subsequent customer above (without contact) a box of the current customer. Figure 7.4 shows an example of loading that is feasible for the M3L-CVRP and infeasible for the original 3L-CVRP; in fact, the tour represented below the loading area indicates that the first customer to be visited is customer 1, whose box has above it a box of a customer that is later in the tour (customer 2). This situation would incur a violation of the classical LIFO constraint, but it is correct for the M3L-CVRP because a human operator can simply slide out the box without touching the one above.

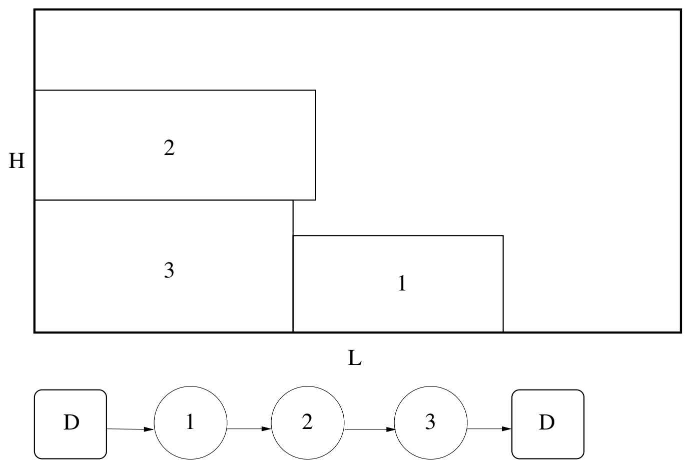


Figure 7.4: Manual 3L-CVRP: a human operator can simply slide out the box 1 without touching box 2.

The LIFO constraint requires to be further refined in order to deal with real cases. In fact, Figure 7.5 draws a situation that is considered feasible for both 3L-CVRP and M3L-CVRP, but in reality it is totally unworkable. The designed tour establishes that customer 1 is the first one to be visited, therefore box 1 needs to be unloaded before the others. However, both a human operator and a forklift are not able to unload it without moving the others, since they are at the rear door of the vehicle and box 1 is too far away at the bottom side. In this case, any subsequent placement of the box in a position closer to the rear door would incur in a fragility or stability violation.

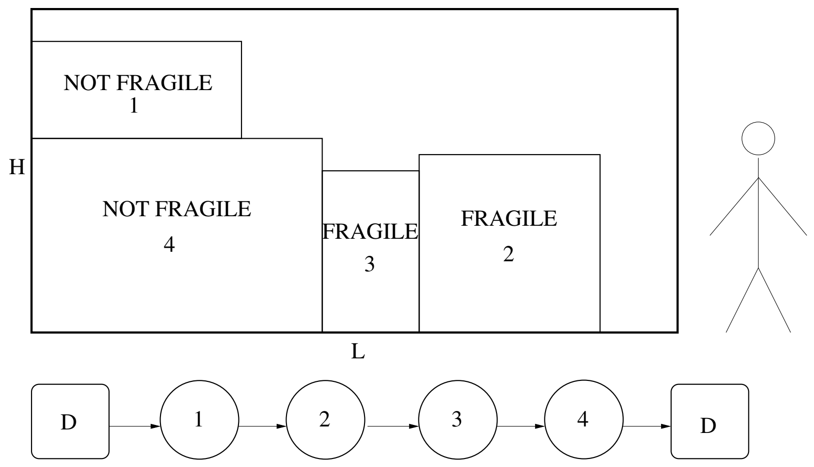


Figure 7.5: Situation impracticable for unloading box 1

We, thus, introduce a new loading constraint:

Reachability (C7): An item is considered reachable if the distance between it and a human operator or a forklift is less than or equal to a fixed length λ. It is supposed that the operator is placed as close as possible to items inside the vehicle, i.e. the position with the minimum length with respect to the current loading.

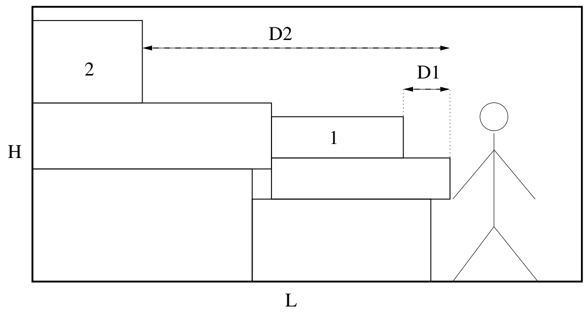


Figure 7.6: The Reachability constraint

Figure 7.6 shows the position of an operator inside a vehicle and how distances are computed (D_j is the distance between an operator and item j). In this case, both items 1 and 2 could be unloaded without moving other items, thus this loading does not violate the standard LIFO constraint (C4). However, due to the physical limitations of the operator, only item 1 can be unload, while box 2 is unreachable.

In case of manual unloading λ is set to 50 cm (approximately the length of a human arm), otherwise it is set to a higher value, typically 100 cm. This way, our solver is implicitly able to manage cases where both constraints about reachability and manual unloading are enforced, and cases where only the reachability constraint is applied.

Split deliveries

Finally, as already mentioned, it is possible that a customer has a demand that is greater than the vehicle capacity; in this case, the demand needs to be split and be delivered by more than one vehicle (we do not allow that a customer is visited more than once by the same vehicle).

As it will be clear in Section 7.3, this is a significant change that leads to a completely different search space and much more complex neighborhood operators. In fact, this routing problem falls into the category of CVRP with *split deliveries*, which uses approaches that are quite different from those of the classical CVRP (see Section 3.2.6).

7.3 Solution technique

In this section, we present our solution technique, which is based on LS. Firstly we define the search space, the cost function, the initial solution, and the neighborhood relations; then we briefly describe the metaheuristic technique we used.

7.3.1 Search space

Differently from the approaches to the 3L-CVRP cited in Section 4.3 [91, 158, 85, 173, 27], we design a one-stage local search solver, which works on the items-vehicles space rather than on the customers-vehicles one. The solver is responsible for

- generating the customers' routes,

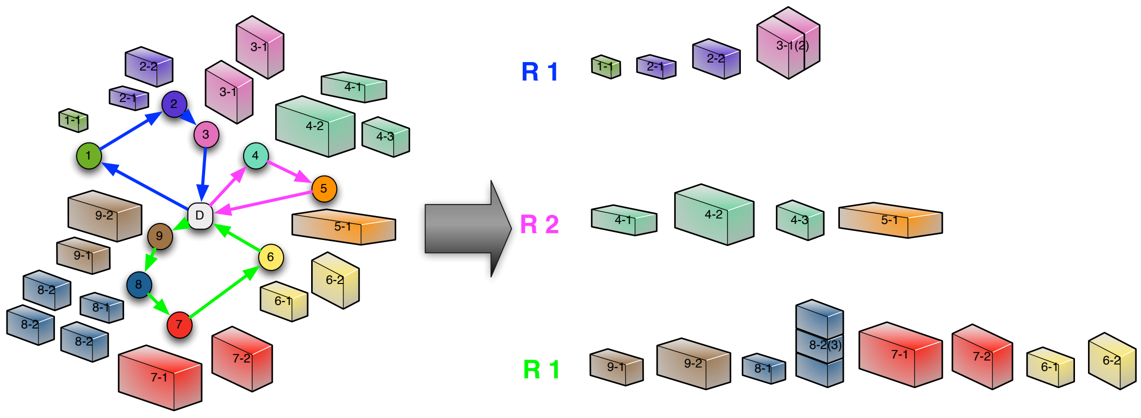


Figure 7.7: The search space consist of sequences of blocks of boxes.

- setting the order in which the items of each sequence will be loaded, and
- selecting the loading heuristic that will be used to determine the actual loading of items in the vehicle.

A state in the search space is then a set of sequences, one for each vehicle, where each element of a sequence is an item or a block of items of the same type (Figure 7.7). Items belonging to the same customer are consecutive, in such a way each items' sequence corresponds to a unique customers' sequence.

7.3.2 Cost function

The cost function is a weighted sum of the objective function and the *distance to feasibility*. In fact, the solver can accept a move that leads to an infeasible state, although violations of hard constraints are penalized by multiplying their degree of violation with a suitably high weight.

We allow two kinds of infeasibility: *exceeding weight* (H1) and *unloaded volume* (H2). The first one represents the total weight that exceeds each vehicle's capacity and the second one represents the total volume of items that do not fit into the vehicles. As already proposed by Gendreau et al. [91], we set the value of the weight of H1 in dependency of the size of the instance and we fix it to $\mathtt{HW} \cdot 100 \cdot \bar{c}/Q$, where $\bar{c}$ denotes the average edge distance and HW is a multiplicative constant. Conversely, the weight of H2 is set to HW. Preliminary experiments have shown that the best choice for HW is 3, thus this value has been used during all experimentation.

All other hard constraints are enforced by construction.

7.3.3 Initial solution

The initial solution is constructed at random. Firstly, we randomly choose a customer and we assign her to a random vehicle. Then, we iteratively select at random a box type and we create a block of homogeneous items. We append it, with a random rotation, to the sequence of blocks of the selected vehicle. After that, we rebuild from each sequence of blocks the corresponding sequence of customers.

7.3.4 Neighborhood relations

We implemented three neighborhood relations: the first two remind the Ins, IntraSw and InterSw developed in Section 5.3 for the VRPTWCDC, whereas the last one is an adaption of neighborhood used in Section 6.3.5 for the MCLPBWMD. In detail, the neighborhood relations are the following:

Move Customer & Change Strategy (MCCS): This neighborhood is defined by the removal of a customer from a route and its insertion in another one in a specific position. As a consequence, all the blocks of items belonging to the selected customer are removed from the old sequence and then inserted in the new one, keeping fixed their relative position and their orientation. In addition, the loading strategy of the new route can be changed. An MCCS move is identified by six attributes $\langle c, or, op, nr, np, st\rangle$ where c represents a customer, or and op the old route and the old position in the old route, and np and nr, the new route and new position in the new route, respectively. Lastly, st is the loading strategy selected for the new route (Figure 7.8). If the customer is not moved (i.e., if $or = nr \wedge op = np$), an MCCS move performs only a change of strategy.

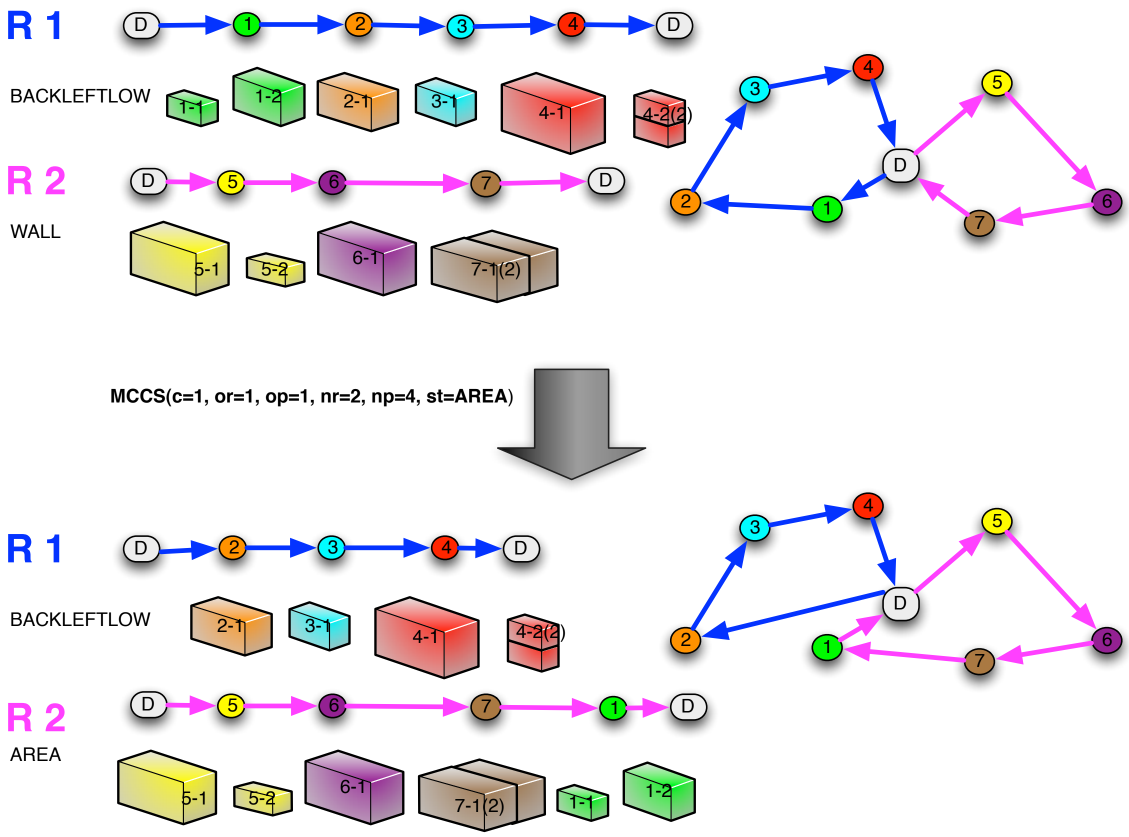


Figure 7.8: An example of MCCS move.

Swap Customers (SC): This neighborhood is defined by exchanging a customer with another one. A move m of type SC is identified by six attributes $\langle c_1, c_2, r_1, r_2, p_1, p_2\rangle$ where c_1 and c_2 are customers, r_1 and r_2 are routes, p_1 and p_2 are the positions of the customers in the corresponding route. Blocks of items belonging to the customers involved in the move are swapped keeping fixed their reciprocal position and their orientation. If r_1 is the same as r_2, the move turns out to be an intra-route exchange.

Move & Rotate Block (MRB): This neighborhood is defined by the removal of a block of homogeneous items from the blocks' sequence of a route and its insertion, possibly with a new orientation, into another route. A move of MRB type is represented by 7 attributes $\langle bt, or, op, nr, np, r, qty\rangle$, where bt is the box type identifier, or and op are the old sequence and the old position in the old sequence, np and nr, the new sequence and new position in the new sequence, r is the rotation and qty is the number of boxes of the block that are involved in the move (Figure 7.9). A block can be completely removed from the old route, or just partially. In the latter case, the original block is divided in two ones: the past that remains in the old route, and the new block composed by *qty* boxes, which is inserted in the new route.

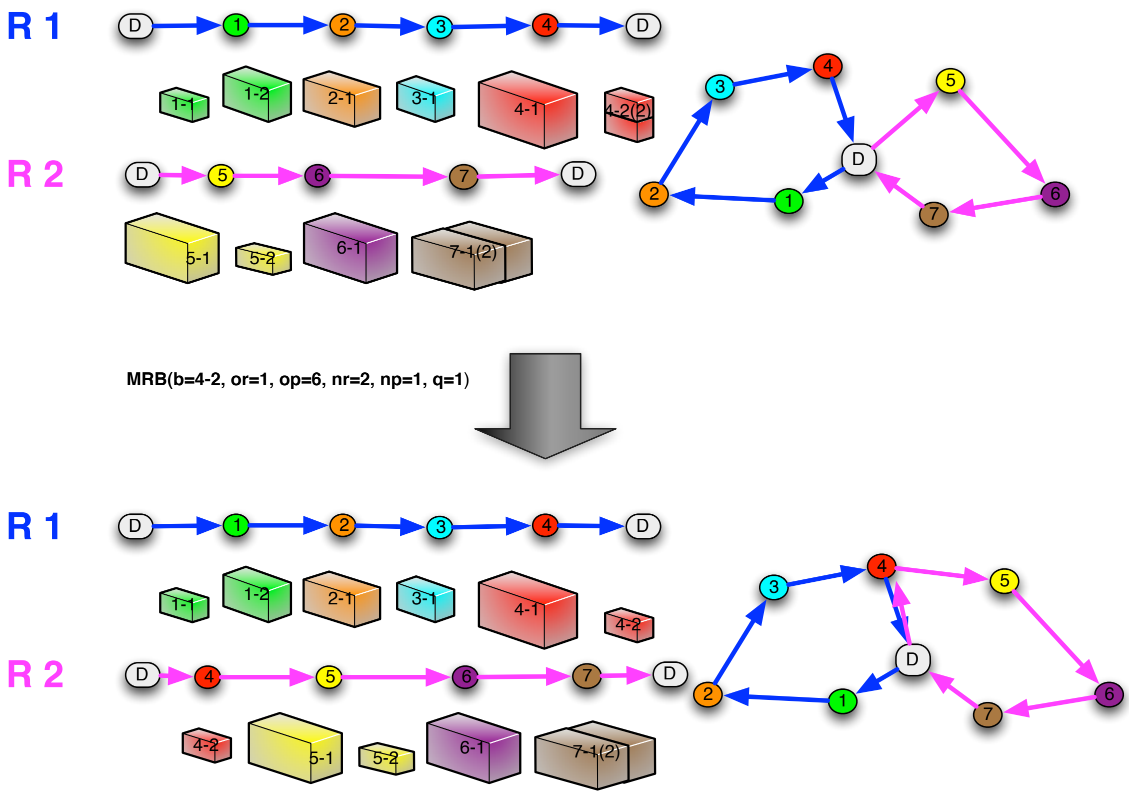


Figure 7.9: An example of an MRB move that leads to a split delivery.

Notice that if the new route is different from the old one, an MRB move brings about the splitting of the demand. In Figure 7.9, a single item of customer 4 is moved from route 1 to route 2, thus the demand is split in two vehicles and the customer is visited twice.

As already proposed for the MCLPBWMD in Section 6.3.3, once a move is performed, a packing heuristic is called for the blocks' sequences involved in the move. The packing heuristic takes care of all the constraints about the loading and returns the total volume loaded. If some items do not fit in the vehicle, their volume contributes to the cost component H2. The packing heuristic does not modify the blocks' sequence, thus, the C4 constraint is always guaranteed.

7.3.5 Loading heuristics

In section 4.3 we have described diverse packing strategies, here we have implemented nine loading heuristics. Eight of these are variants of the extension to the three-dimensional case of the *bottom left algorithm* [9] and the *touching perimeter algorithm* [122]. These heuristics individuate the *normal position* [49], i.e., with the item's bottom edge touching either the floor of the vehicle or the top edge of another item, and its left edge touching either the left edge of the vehicle or the right edge of another item. The detection of normal positions in the three-dimensional case is not trivial [see 128, 55, 129]; for this task we applied the procedure proposed by Martello et al. [128] with some modifications that take into account the C3 constraint.

For each item, the choice of the packing position among all the possible normal positions in a vehicle depends on the specific strategy. To give an intuition of how each strategy works, in Figure 7.10a we introduce an easy example of loading with only three items: all the normal positions are labelled with capital letters. The difference between strategies is in the order in which the normal positions are tested.

In this example there are seven candidate normal positions (A, B, C, D, E, F, G) where it is possible to place a new item. If we select the Back Low Left strategy, the first position that will be tested is A; if this placement is feasible (all loading constraints are satisfied), the box is placed with its bottom left corner in A. Otherwise, the box is rotated and the we check if the A position is now feasible. In case of infeasibility, the next normal position (B) is tested with the box in the original orientation, and so on.

For the strategies based on the *bottom left algorithm* (Back Left Low, Back Low Left, Low Back Left, Low Left Back, Left Low Back, Left Back Low), Figure 7.10b describes the order of selection of candidate positions and the direction of loading (vertical walls, horizontal layers, from the left side, from the back ...). The feasibility of a position is evaluated with the current orientation; if it is not feasible, a rotation on the width-length plane is applied to the item.

The Area and Area No Walls strategies derive from the *touching perimeter algorithm*: the first one selects the position with highest score, defined as the percentage of the item's area that touches either the vehicle and other items already packed; the second one does not consider the wall of vehicles in the score. Each position is evaluated only for the current orientation because the candidate positions are ordered for decreasing score, which is computed considering the current orientation.

The other packing heuristic (Wall Building) is based on the *wall building approach* introduced by George and Robinson [93] for the container loading problem and already used by us for the MCLPBWMD (Section 6.3.3).

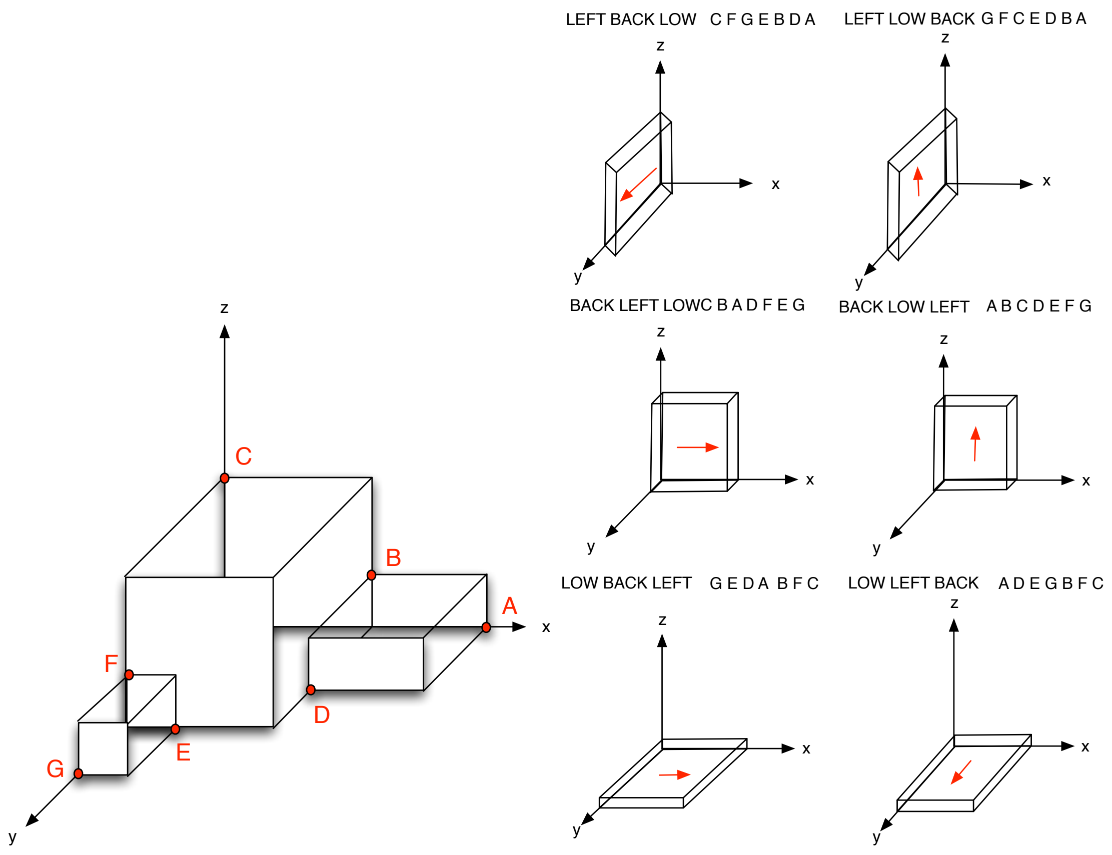
7.3.6 Metaheuristics

We use a sequential solving approach which alternates the SA algorithm and a large-neighborhood search, called *intensifier*, as shown in Figure 7.11. The algorithm control is illustrated in the style of the *generalized local search machines* (GLSM) proposed by Hoos [104]. Nodes represent search strategies and arrows represent transitions of the control from one search strategy to another. In Figure 7.11, `F` denotes the value of the cost function of the best solution found in the current state, whereas `F_best` refers to the best solution found so far in the overall solution process. Each component starts from the best solution of the previous one, and the overall process makes a fixed number of rounds R.

The SA algorithm has been described in detail in Section 2.2.2. The novelty is the fact that it is called more than once in the solution process. Therefore, starting from the second round the initial temperature is not set to T_0 but to a lower value, in order to avoid to "destroy" completely the previous solution. We thus define a new parameter, called ρ, such that the initial temperature of all rounds but the first is set to T_0/ρ. The SA component uses as neighborhood the union of MCCS, MRB, and SC moves. The random selection prescribed by SA is performed selecting first the neighborhood used, and then the specific move.

In order to improve the effectiveness of the SA component, it is useful to set the weight of the hard constraint violation `HW` to a relatively low value, given that the selection relies on variation of the cost function. As a consequence, it is possible that the whole process terminates with a solution that has violations of hard constraints even though a feasible solution could be reached. For this reason, we introduce a new component in the process, called *REPAIR* in Figure 7.11, whose aim is to eliminate violations. In the *REPAIR* state, `HW` is set to a particularly high value and a simple *hill climbing* (HC) algorithm is invoked.

We also apply a special-purpose operator, called *intensifier* that uses as neighborhood a composition of the two basic neighborhoods MCCS and MRB. At each iteration, the neighborhood composed by one move of MCCS and one of MRB is explored and the first improving one is selected. In order to reduce the size of the composite neighborhood, we consider only pairs of moves


z
C
B
A
x
F
D
E
G
y
LEFT BACK LOW C F G E B D A
LEFT LOW BACK G F C E D B A
BACK LEFT LOWC B A D F E G
BACK LOW LEFT A B C D E F G
LOW BACK LEFT G E D A B F C
LOW LEFT BACK A D E G B F C
z
x
y

(a) Normal positions (b) Order of selection of candidate positions

Figure 7.10: The Back Left Low, Back Low Left, Low Back Left, Low Left Back, Left Low Back, Left Back Low strategies.

that concerns the same customer. The aim of this operator is to intensify the search process in promising areas.

As shown in Figure 7.11, the intensifier is launched only when SA gives no improvement in the cost function. This choice is motivated by the fact that the intensifier is computationally expensive and therefore it should be launched only when SA is clearly stuck. Once it has been launched, the intensifier is invoked iteratively as long as it finds improvements in the cost function.

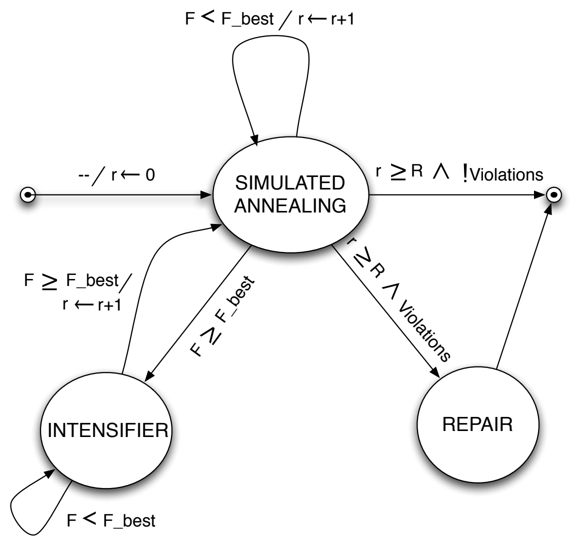


Figure 7.11: A GLSM representation of the control flow of our algorithm. See the text for more details.

7.4 Experimental analysis

In this section, we first describe the setting of the parameters used in the experiments. Then we give a description of our new instances and we show our results. Lastly, we compare our approach with the best ones in the literature on the 3L-CVRP, using the instances introduced by Gendreau et al. [91] as benchmarks.

7.4.1 Parameter settings

The software is written in *C++*, it uses the framework EasyLocal++ [63]. For the automatic tuning of the solver we use the *irace* package, provided by López-Ibáñez et al. [124]. More details about the environmental settings can be found in Section 2.3.

The SA algorithm has five parameters to tune: start temperature T_0, stop temperature T_{min}, cooling rate α, the number of neighbors sampled at each temperature σ_N, and restart temperature ratio ρ. Moreover, given that the solver uses different neighborhoods, we add two parameters γ_{MCCS} and γ_{MRB}, which are the probability of drawing a move of type MCCS and MRB, respectively. The probability of drawing a move of type SC is set to $\gamma_{\mathsf{SC}} = 1 - \gamma_{\mathsf{MCCS}} - \gamma_{\mathsf{MRB}}$.

We decide to tune $\delta = T_0/T_{min}$ instead of T_{min}, which turned out to provide a better distribution of the configurations than using T_{min} directly. In addition, in order to give a similar amount of time to the same instance for each parameter configuration, we let the parameters δ and α vary, and we compute σ_N in such a way to have for the SA component exactly the same number of SA iterations I for each benchmark. In detail, the number of neighbors sampled for each temperature is $\sigma_N = I/\log_\alpha(\delta)$. For each instance, being v the number of vehicles and n the number of customers, the total number of iterations granted is $I = 10^4 \times v \times n$. The full solver

Instance	n	bt	M	vt	v	φ	θ	Gini index
SD-CSS1	11	36	254	1	5	45.17	18.82	98.23%
SD-CSS2	25	15	350	1	13	59.57	29.78	88.29%
SD-CSS3	33	9	285	1	26	41.55	31.77	76.86%
SD-CSS4	37	13	312	1	12	56.58	17.86	89.49%
SD-CSS5	41	47	7035	2	13	5.10	1.57	95.39%
SD-CSS6	43	97	8060	1	35	28.53	22.70	93.21%
SD-CSS7	45	14	284	2	10	62.57	13.60	88.44%
SD-CSS8	48	70	3275	3	36	37.12	27.27	97.19%
SD-CSS9	56	45	1725	1	23	48.10	19.41	97.44%
SD-CSS10	60	29	1840	1	20	23.30	7.64	88.26%
SD-CSS11	92	34	3790	2	13	43.79	6.12	88.67%
SD-CSS12	129	10	745	1	50	53.67	20.64	97.34%
SD-CSS13	129	63	2880	1	35	34.21	9.21	96.53%

Table 7.1: Features of the new real-world instances

stops when it has performed four rounds or one *idle* round, i.e., a round without an improvement of the best result.

Preliminary experiments show that α is not significant. This is not surprising, because in our setting σ_N is a function of the other parameters. Therefore, α only determines the entity of the single step in the temperature and not the actual slope of the cooling trajectory, which is determined by δ. We therefore set α to the fixed value 0.9999. We also experimented with different values for the probabilities $\gamma_{\mathsf{MCCS}}, \gamma_{\mathsf{MRB}}$, but there was no strong evidence in favor of some values. We therefore set $\gamma_{\mathsf{MCCS}} = 0.4$, and γ_{MRB} and γ_{SC} to 0.3.

For the benchmark instances, we set a tuning budget for I/F-Race of 4000 experiments. We use the following domains for the parameter settings: $T_0 \in [10, 10^5]$, $\delta \in [10^2, 10^6]$ and $\rho \in [10^{-1}, 10^3]$, and the set of training instances consist of instances 1–11 published in [91]. For our instances, we keep the same parameter domains and we use as training instances the set {SD-CSS2, SD-CSS3, SD-CSS4, SD-CSS7, SD-CSS12}, which are the fastest to be solved.

The outcome of the I/F-Race procedure is that the best configurations are the following: $T_0 = 12746.634$, $\delta = 28990$ so that ($T_{min} = 439.680$, and $\rho = 210$ for the 3L-CVRP benchmarks, and $T_0 = 297143$, $\delta = 88298$ (resulting in $T_{min} = 3.36$), and $\rho = 810$ for our instances. The results for this configuration are presented in Table 7.2 and Table 7.3 in comparison with previous work.

7.4.2 New instances

Table 7.1 shows the features of each instance in terms of the number of customers (n), box types (bt), the total number of boxes (M), vehicle types (vt) and the total number of vehicles (v). The terms φ and θ indicate in percentage the ratio between the total volume of boxes and the total volume of available vehicles and the ratio between the average customer demand (in m^3) and the average volume of vehicles, respectively.

The instances are very diverse for the number of customers, the number of box types and the number of boxes. In particular it is worth noticing that the number of boxes in seven of the thirteen instances is greater than one thousand. The terms φ and θ give an intuition of how difficult is the packing subproblem and of the number of nodes per route, respectively.

The *Gini index* [95] is a measure of the heterogeneity of boxes from the relative frequencies associated with a box type. A value of 0 expresses that all the boxes are equal, whereas a value of 100 % that all items are different. We highlight this aspect because different levels of box heterogeneity need different packing strategies in order to perform the loading effectively.

Table 7.2: Results on our instances with different constraints (H2 in m^3 and z in km).

	3L-CVRP formulation		No split delivery		split delivery	
Instance	H2	$z_{3L-CVRP}$	H2	z	H2	z_{SD}
SD-CSS1	0	5708.57	0	5848.4	0	6420.88
SD-CSS2	0	12033.24	0	12090.49	0	13167.83
SD-CSS3	51.22	–	51.22	–	0	16645.03
SD-CSS4	0	11398.58	0	11820.99	0	12607.49
SD-CSS5	0	11836.71	1.09	–	0	26185.16
SD-CSS6	0	19939.81	3.91	–	0	36676.21
SD-CSS7	0	11809.04	0	12548.78	0	15654.43
SD-CSS8	0	23183.1	15.96	–	0	49579.63
SD-CSS9	0	17724.8	0	18889.95	0	32602.95
SD-CSS10	0	12945.89	0	17859.39	0	27923.13
SD-CSS11	0	26900.47	0	33070.26	0	43790.43
SD-CSS12	0	34807.29	0	35176.91	0	36433.19
SD-CSS13	0	28060.23	0	35897.29	0	66279.74

7.4.3 Experimental results

We applied our solver to different problem formulations: the *3L-CVRP* formulation, described in Section 7.2.1, the complete problem formulation, described in Section 7.2.2, and the complete problem formulation without split deliveries. The results are shown in Table 7.2. Average running times vary between 300 seconds and 10000 seconds depending on the average number of boxes for vehicle.

For some instances, it has been impossible to find a feasible solution, without split delivery. For such cases, we report the level of violation of the H2 constraint (the H1 constraint is never violated). In order to give a measure of the quality of a solution that is real-valued, the cost z is expressed in total km travelled and the value of H2 violations in m^3 of boxes that were not possible to load in the vehicles.

Comparing the *3L-CVRP* formulation and the complete one without split deliveries, it is clear that the first one is able to obtain solutions with lower cost, thus we can conclude that the new constraints about fragility, stability and reachability have a no negligible impact on the solution process.

For the remainder of this section we focus on the results of the complete formulation with split deliveries. Looking at the results for the complete formulation with split deliveries, we notice that for instance SD-CSS3 this formulation is the only one able to obtain a feasible solution. A possible explanation is that this instance has the highest θ value, which is the ratio between the average customer demand and the average volume of vehicles, and the solution of this case might require to split the demand of customers to different vehicles.

For most instances, the use of the larger space with split deliveries allows us to obtain a feasible solution (SD-CSS3, SD-CSS5, SD-CSS6, SD-CSS8). On the other hand, when the instance is large in terms of the number of boxes and vehicles, exploring a larger space is not effective, resulting in solutions of worse quality. Our experimental results also suggest to use an adaptive strategy in the complete formulation that switches between the possibility of using or not split delivery. One may start by forbidding split deliveries. If feasible solutions without split deliveries are found, these are typically of a better quality w.r.t. the total distance travelled. If feasible solutions are not obtained relatively quickly, for example, after the first round through the phases simulated annealing and intensifier, we may switch to the formulation considering split deliveries to increase the chance to find feasible solutions.

7.4.4 Comparison with related work

Our solver has been tested also on the 27 instances proposed by Gendreau et al. [91], which have been derived from CVRP instances [164] by specifying the customer demand in the form of rectangular three-dimensional items. A detailed description of the instances' features can be found in the work by Gendreau et al. [91]. For all benchmarks, the supporting area factor is set to 0.75, as in previous works.

For the 3L-CVRP, the MRB neighborhood is restricted to exclude moves that lead to split deliveries. This is done by allowing to reinsert an item only in a position where it is adjacent to another one belonging to the same customer. Thus, the MRB move can only modify the reciprocal position of items of the same customer and change their orientation.

Table 7.3 summarizes the features of the instances and shows the computational results. The best result is shown in bold face. For instances by Gendreau et al. [91], in Table 7.3 we compare our solver with the reported results of the TS by Gendreau et al. [91], the Guided TS by Tarantilis et al. [158], the ACO by Fuellerer et al. [85] and the Hybrid TS by Bortfeldt [27] (see Section 3.3 for more details about these techniques). We do not report the results by Wang et al. [173] because they use a larger number of vehicles, thus they are not comparable.

Since the TS is deterministic, it was invoked only once for each instance and this value is reported. For the Guided TS we have only the cost of the best solutions available. ACO, Hybrid TS and our SA are invoked 10 times with different random seeds on each run, and the average and the minimum cost are reported.

The outcome is that our solver improves on the original results by Gendreau et al. [91] and those of Tarantilis et al. [158]. It performs worse, however, than the ACO [85] and the Hybrid TS [27] approaches although in one case (instance 7) is able to find the best known result. However, all the techniques reported in Table 7.3 are specifically designed and tuned for the case of the 3L-CVRP and for these instances, whereas our technique solves the more general problem that deals with split deliveries, weakly heterogeneous cargos, and stability constraints as described below.

7.5 Summary

In the last few year, the interest of the research community for composite and structured problems has increased and different problem formulations and models have been proposed. In this chapter we have addressed a complex problem, which combines routing and packing issues, and considers several features that arise in real-world situations. Our problem extends and enriches the *three-dimensional loading capacitated vehicle routing problem* [91] by redefining some constraints with respect to the notion of stability, loading and unloading policy and the heterogeneity of the cargo. It also considers the possibility of managing a heterogeneous fleet and to split the demand of a customer in more vehicles.

We have presented a local search approach based on SA and large-neighborhood search that solves the integrated problem in one single stage. In fact, a solution is represented as sequences of items making suitable the use of neighborhood operators that modify both routes and arrangements in vehicles.

The solution approach has been tested on 13 new real-world instances, that exhibit great diversity in size and features and the effect of introducing split deliveries has been analyzed and discussed. All instances are available at `http://www.diegm.uniud.it/ceschia/index.php?page=vrclp`.

In addition, the solver has been tested on benchmark instances of Gendreau et al. [91] for the 3L-CVRP. Although our solver is not designed for that problem, it reaches better or equal solutions than two algorithms specifically designed for the 3L-CVRP; but it is inferior to two other recent metaheuristics algorithms for the 3L-CRVP.

Table 7.3: Computational results for 3L-CVRP on Gendreau et al. [91] instances (time is in seconds).

				Gendreau et al.		Tarantilis et al.		Fuellerer et al.			Bortfeldt			Us		
I	n	M	v	z	sec	z_{min}	sec	z_{avg}	z_{min}	sec	z_{avg}	z_{min}	sec	z_{avg}	z_{min}	sec
1	15	32	4	316.32	129.5	321.47	13.2	305.35	304.13	11.2	**302.02**	302.02	41.60	305.10	302.03	100.62
2	15	26	5	350.58	5.3	334.96	11.5	**334.96**	334.96	0.1	**334.96**	334.96	0.30	**334.97**	334.97	55.19
3	20	37	4	447.73	461.1	430.95	540.6	409.79	399.68	88.5	**404.34**	388.10	159.10	439.35	414.51	213.82
4	20	36	6	448.48	181.1	458.04	323.5	440.68	440.68	3.9	**437.94**	437.19	12.40	447.43	440.69	139.39
5	21	45	6	464.24	75.8	465.79	99.6	453.19	450.93	22.7	**447.49**	443.61	170.50	460.13	450.16	179.59
6	21	40	6	504.46	1167.9	507.96	1212.4	**501.47**	498.32	17.5	**501.47**	498.16	15.30	507.24	498.33	165.36
7	22	46	6	831.66	181.1	796.61	364.8	797.47	792.13	51.4	791.03	769.68	62.80	**755.79**	752.75	225.08
8	22	43	8	871.77	156.1	880.93	230.0	**820.67**	820.67	56.2	824.00	810.89	98.90	822.04	820.22	193.22
9	25	50	8	666.10	1468.5	642.22	982.2	**635.50**	635.50	15.3	663.39	630.13	11.20	654.79	630.14	216.01
10	29	62	8	911.16	714.0	884.74	1308.4	841.12	840.75	241.2	**829.31**	820.35	139.80	856.79	841.43	427.44
11	29	58	8	819.36	396.4	873.43	522.5	821.04	818.87	172.4	**815.35**	800.52	118.80	828.71	805.93	383.93
12	30	63	9	651.58	268.1	624.24	294.6	**629.07**	626.37	46.2	636.10	610.23	12.80	637.89	621.58	393.54
13	32	61	8	2928.34	1639.1	2799.74	2193.1	2739.80	2739.80	235.4	**2701.10**	2679.86	232.90	2747.96	2718.34	460.28
14	32	72	9	1559.64	3451.6	1504.44	4581.3	1472.26	1466.84	623.8	**1419.84**	1368.42	312.20	1494.01	1436.51	574.77
15	32	68	9	1452.34	2327.4	1415.42	2528.3	1405.48	1367.58	621.0	**1357.01**	1331.27	299.90	1410.01	1369.48	556.86
16	35	63	11	707.85	2550.3	**698.61**	4256.5	**698.92**	698.92	12.8	704.24	698.61	2.40	702.34	698.63	442.43
17	40	79	14	920.87	2142.5	872.79	2096.0	**870.33**	868.59	11.8	969.59	876.22	1.70	882.65	866.42	549.87
18	44	94	11	1400.52	1452.9	1296.59	2275.2	1261.07	1255.64	2122.2	**1233.99**	1206.67	315.10	1272.62	1238.71	1103.97
19	50	99	12	871.29	1822.3	818.68	2509.0	781.29	777.18	614.3	**755.05**	757.04	419.20	817.77	786.13	1239.86
20	71	147	18	732.12	790.0	641.57	1940.9	611.26	604.28	3762.3	**598.23**	585.65	432.10	625.15	612.42	2977.67
21	75	155	17	1275.20	2370.3	1159.72	2823.4	1124.55	1110.09	5140.0	**1108.40**	1091.33	452.30	1160.77	1133.68	3143.04
22	75	146	18	1277.94	1611.3	1245.35	2685.6	1197.43	1194.18	2233.6	**1175.82**	1162.11	428.60	1220.57	1188.51	2752.97
23	75	150	17	1258.16	6725.6	1231.92	4659.1	1171.77	1158.51	3693.4	**1138.19**	1144.46	430.50	1191.70	1149.33	3030.60
24	75	143	16	1307.09	6619.3	1201.96	4854.1	1148.70	1136.80	1762.8	**1127.76**	1114.54	413.30	1194.01	1162.37	2934.60
25	100	193	22	1570.72	5630.9	1457.96	5725.8	**1436.32**	1429.64	8619.7	**1436.63**	1398.42	463.50	1477.09	1455.30	5978.76
26	100	199	26	1847.95	4123.7	1711.93	6283.1	**1616.99**	1611.78	6651.2	1630.63	1590.31	436.70	1662.95	1627.00	5720.78
27	100	198	23	1747.52	7127.2	1646.44	9915.7	1573.50	1560.70	10325.8	**1541.99**	1528.43	441.60	1628.01	1587.04	5110.95
Avg				1042.26	2058.9	997.20	2415.9	966.67	960.87	1746.5	**958.74**	939.97	219.46	982.88	960.84	1454.5

8
Other Problems

During the PhD course, the techniques and the software tools described in Section 2.2 and Section 2.3, have been profitably applied to other optimization problems, that belong to the domains of timetabling and healthcare. These themes do not concern the core topics of this thesis, therefore in this chapter we only give a brief presentation of our works on these side issues.

8.1 Timetabling problems

Timetabling problems are widespread in many human activities and their solution is a hard optimization task that can be profitably tackled by optimization methods. Educational timetabling is a sub-field of timetabling that considers the scheduling of meetings between teachers and students.

A large number of variants of educational timetabling problems have been proposed in the literature, which differ from each other based on the type of institution involved (university, school, or other), the type of meeting (course lectures, exams, seminars, . . .), and the constraints imposed.

The *university course timetabling* (CTT) problem is one of the most studied educational timetabling problems and consists in scheduling a sequence of events or lectures of university courses in a prefixed period of time (typically a week), satisfying a set of various constraints on rooms and students. Many formulations have been proposed for the CTT problem over the years. Indeed, it is impossible to write a single problem formulation that suits all cases since every institution has its own rules, features, costs, and fixations.

Nevertheless, two formulations have recently received more attention than others, mainly thanks to the two timetabling competitions, ITC 2002 and ITC 2007 [130], which have been used by them as competition ground. These are the so-called *curriculum-based course timetabling* (CB-CTT) and *post-enrolment course timetabling* (PE-CTT). The main difference between the two formulations is that in the CB-CTT all constraints and objectives are related to the concept of *curriculum*, which is a set of courses that form the complete workload for a set of students. On the contrary, in PE-CTT this concept is absent and the constraints and objectives are based on the student enrolments to the courses.

In [42] we focused on the PE-CTT problem and we designed a single-step metaheuristic approach based on Simulated Annealing, working on a composite neighbourhood composed of moves that reschedule one event or swap two events.

We experimented our solver on all the instances that have been made publicly available (up to our knowledge). The outcome of our experimental analysis was that our general solver, properly engineered and tuned, was able to outperform most of the solvers specifically designed and tuned for a single specific formulation and/or a specific set of instances.

In order to ensure reproducibility, the source code has been made available at the website `http://satt.diegm.uniud.it`, along with the best solution found for each instance.

8.2 Healthcare problems

Healthcare is surely one of the most important application domain of optimization in general and of metaheuristics in particular. Many papers have been devoted to healthcare, for example to nurse and physician rostering problems [37, 145], and more generally to timetabling problems in hospitals [see, e.g., 100].

The *patient admission scheduling* (PAS) problem consists in assigning patients to hospital rooms in such a way to maximize both medical treatment effectiveness and patients' comfort. PAS has been defined by Demeester et al. [59], and further studied by the same research group [168, 14].

In the proposed PAS formulation, each patient has fixed admission and discharge dates and one or more treatments to undergo. Each room is characterized by its equipments and location and may be more or less suitable for a specific patient; and this results in a matrix of patient/room compatibility costs that contributes to the objective function. In addition, there is a room gender policy that forbids, for normal rooms, the simultaneous presence of male and female patients. Finally, patients should possibly not change the room during their stay; a room change is called a *transfer* and is penalized in the objective function. The problem then consists in assigning patients to rooms for each day of their stay in hospital, minimizing all the mentioned costs and respecting the capacity constraint and the gender policies of the rooms. The planning horizon h is expressed in days and varies from 2 weeks ($h = 14$) up to 3 months ($h \simeq 90$).

Unfortunately, this *long-term* version of the problem has little usefulness for most practical cases where patients might arrive at unpredictable times (urgent and emergency patients) [116, 88]. Furthermore, it is also frequent that the discharge date of the patients is unknown, because it might depend of the progress of the gradual recovery [73, 84]. In these cases, the long-term solution of the problem can provide only a provisional assignment that needs to be subsequently modified several times.

In [38], we proposed a local search approach to the PAS problem that makes use of different search spaces and neighborhood relations. In addition, we developed a relaxation procedure to compute lower bounds (using CPLEX v. 12), which are useful to assess the quality of the solutions more objectively. The outcome of our work is that our results are better than the ones obtained by Bilgin et al. [13], and in some cases also quite close to the lower bounds.

We also introduced a *short-term (daily)* version in which the admission dates become known to the solver only a fixed number of days (called *forecast level*) before the patient arrivals. We also solved the short-term version of the problem and we analyzed the behavior of the solver as a function of the forecast level.

Conclusions

In this final chapter we draw the conclusions about the research lines pursued and the issues that are still opened. Since the detailed discussion about each original contribution is normally included as a final section of the corresponding chapter, in the following we outline only some general discussion about the different topics of this work.

This thesis introduces new problems in the logistic domain which include several practical constraints that arise in real-world situations. For tackling these issues we propose metaheuristic approaches based on a complex combination of neighborhood relations.

We collected and published on the web a set of real-world instances, in cooperation with our current partners. Differently from the benchmark datasets available in literature, these instances exhibit a great diversity in size and features, making them particularly challenging. The web site contains also an application for validating new solutions, so as to allow everybody to perform a fair comparison with ours.

We performed an extensive experimental analysis on both real-world instances and public benchmarks, making use of principled statistical tests to tune the parameters and to compare different algorithm variants. In particular, we identified the most important parameters of the method, select a set of configurations, and eventually compare them using reliable statistical tests. For the choice of the configurations for tuning the parameters, we used some recently proposed techniques for the design of experiments, such as *nearly-orthogonal latin hypercubes* [51] and I/F-race [17].

The general outcome is that metaheuristics techniques are particularly suitable to solve this typology of problems, due to the actual complexity of the problem which makes it impossible to solve large instances in short time and to the ability to model complex constraints, sometimes non-linear, in a easy and flexible way. Indeed, the comparison of our solution techniques with the ones specifically designed and tuned for the standard problems presented in the literature shows that our results are in general competitive with the ones of the state-of-the-art solvers.

We consider this work as the core of a more inclusive project whose objective is to built an application able to effectively solve the most common vehicle routing problems and three dimensional packing problems. Our aim is to obtain a flexible solver that could be adapted to a large variety of situations, producing solutions (non necessarily optimal) in short time. The solver should be flexible enough to be able to adapt automatically to different problem formulations. In particular, it shall be able to attach or leave out cost components and other features at run-time based on the analysis of the specific instance.

For the future we plan to complete the application that could be used in practice by potential clients, either interactively (on-line) or in batch mode. The solver will be the algorithmic core of the application, which needs to be complemented by a graphical front-end and a persistent managing tool.

From our experience with practitioners and industrial partners, we have learnt that in real-world applications the cost function which rates a solution is usually rather complex, including many objectives which are often conflicting. In our approaches we combine all of the objectives into a single aggregate objective function, which is the weighted linear sum of the objectives; in order to make the quality of a solution immediately evaluable, comparable and easy to understand, we decided to represent the value of the cost function in a real currency. However, determining the appropriate weight for each cost component, especially for those which are related to intangible aspects, can be a difficult and subjective process, given that it depends on the view of the decision manager. To overcome this problem, we would like to go towards a multi-objective perspective that would allow us to work on the frontier of Pareto optimal solutions. Indeed in multi-objective approaches, the potential users do not need to define the importance of each objective a priori by setting the weights, but they have to choose among different (Pareto optimal) solutions by quantifying the trade-offs in satisfying the different objectives.

We also aim to collect a even larger set of instances that are very different in size and features, so that it would also be possible to analyze the relationship between performances and instance properties. As a by-product, we hope that the published instances will be used as new challenging benchmarks for future researches by other groups. In order to obtain this result, we plan the undertake several necessary tasks, as suggested by Bonutti et al. [25] for another optimization problem. In summary, we plan to publish the description of the input and output formats, the solution validator, and a web application to upload results and lower bounds.

In conclusion, we strongly believe that the integrated solution of routing and loading problems constitutes a promising research area that opens new possibilities of solving real-world problems in transportation. Further developments in this field could concern both the enrichment of the problem model and the design of new solution techniques. In Chapter 7, we have presented our problem formulation which already includes some new practical features (split deliveries, heterogenous fleet, load bearing strengths, robust stability, reachability), nevertheless we think that much is still to be done to actually capture the practical problems that companies have to face (vehicle compartments, items incompatibilities, load balancing, unloading policies, regulations, pickup and delivery, . . .). As for the solution techniques, in our opinion new perspectives will be open for this class of problems by hybrid approaches that combine different metaheuristics or integrate AI/OR techniques into metaheuristics.

A Notation

A.1 Combinatorial optimization problems (Section 2.1)

Π	problem
$\mathcal{I}$	set of instances
$\mathcal{S}$	set of candidate solutions
$\mathcal{F}$	set of feasible candidate solutions
$F(i, s)$	objective function of the solution $s \in \mathcal{S}$ of the instance $i \in \mathcal{I}$
s^*	optimal solution

A.2 Local search algorithms (Section 2.2)

$\mathcal{S}(i)$	search space of an instance $i \in \mathcal{I}$
$\mathcal{N}(s)$	neighborhood of the solution $s \in \mathcal{S}$
$F(s)$	cost function of the solution $s \in \mathcal{S}$
m	move that identifies a neighborhood
ΔF	variation of the cost function $(F(s \oplus m) - F(s))$

A.2.1 Simulated annealing (Section 2.2.2)

T	temperature
T_0	start temperature
T_{min}	stop temperature
N_σ	number of neighbors sampled at each temperature level
β	cooling rate
ρ	parameter such that the initial temperature of all rounds but the first is set to T_0/ρ
I	total number of iterations, i.e., the total number of moves selected
δ	temperature range, equal to T_0/T_{min}

A.2.2 Tabu search (Section 2.2.3)

tl	tabu list
tt	tabu tenure
δ_{tt}	variation of the tabu tenure, the length of the tabu list ranges between $tt - \delta_{tt}$ and $tt - \delta_{tt}$
ii_{max}	maximum number of iterations without an improvement

A.3 Vehicle routing problems

$\mathcal{V}$	set of vertex (or nodes) of a graph
$\mathcal{A}$	set of arcs that connect nodes in a graph
G	graph individuated by arcs and vertices
(i,j)	arc of a graph connecting node i with node j
C	cost matrix of arcs
c_{ij}	cost associated to the arc (i,j)
q_i	demand associated to customer i (weight of goods)
K	set of all identical vehicles
W	capacity of vehicles (total weight)
x_{ijk}	integer variable which expresses the number of times an arc (i,j) is traversed by a vehicle k
y_{ik}	integer variable which indicates if the customer i is visited by vehicle k
u_{ik}	continuos variable that counts the load of the vehicle k after visiting customer i

A.3.1 Vehicle routing problem with time windows (Section 3.2.2)

$[e_i, l_i]$	earliest and latest time windows of customer i
s_i	service time that a vehicle must wait at customer location i
τ_{ij}	travel time needed to go from customer i to customer j
t_{ik}	time variable that represents the arrival time of vehicle k at location i

A.3.2 Heterogenous vehicle routing problem (Section 3.2.3)

$\mathcal{M}$	set of types of vehicles
m	number of different types of vehicles
m_k	number of vehicles available of type $k \in \mathcal{M}$
W_k	capacity of a vehicle of type k
F_k	fixed cost associated to each vehicle of type k
c_{ijk}	cost of traversing arc (i,j) with a vehicle of type k

A.3.3 Vehicle routing problem with private fleet and common carrier (Section 3.2.4)

z_i	integer value which indicates if the customer i is served by an external carrier

A.3.4 Period vehicle routing problem (Section 3.2.5)

$\mathcal{D}$	set of days in the the planning horizon
f_i	service frequency of customer i
$\mathcal{H}_i$	set of allowable combinations of visit days for customer i
x_{ijkd}	integer variable which expresses the number of times arc (i,j) is traversed by vehicle k on day d
y_{ih}	integer variable which indicates if the visit combination $h \in \mathcal{H}_i$ is assigned to customer i
z_{hd}	binary constant equal to 1 if day d belongs to visit combination h

A.3.5 Split delivery vehicle routing problem (Section 3.2.6)

y_{ik}	integer variables which stores the quantity of the demand of i delivered by vehicle k

A.4 Three dimensional packing problems

n	number of items
w_i	width of item i
l_i	length of item i
h_i	height of item i
W	width of a container (or bin)
L	length of a container
H	height of a container

A.5 The vehicle routing problems with time windows and carrier-dependent costs

A.5.1 Problem formulation (Section 5.2)

$\alpha(i,r)$	earliest service time of order i on r
c	fixed cost for the vehicle
$carr_j$	the carrier to which vehicle j belongs
$cust(i)$	customer of order i
$\delta(j,r)$	earliest departure time from customer j on route r
d	number of days in the planning period ($\|\mathcal{D}\|$)
$\mathcal{D}$	the planning period
$dist_{ij}$	road distance between customers i and j
$[e_i, l_i]$	time window of customer i
$[\eta_i, \theta_i]$	delivery dates of order i
$\zeta(i,r)$	working/rest times patterns of order i in route r
$\mathcal{F}$	set of vehicles
γ_i	cost of the optional order i, if it is not fullfilled
$\bar{l}_0$	shutdown time
L	threshold level of load for changing vehicle cost function
m	number of vehicles ($\|\mathcal{M}\|$)
$\mathcal{M}$	set of mandatory orders
n	number of orders ($\|\mathcal{O}\|$)
ξ_1	cost per travel unit (€/Km)
ξ_2	cost per load unit (€/Kg) dependent on the farthest location
ξ_3	cost per load unit dependent both on the total load and on the farthest location
$\mathcal{O}$	set of orders
$ord(r)$	set of orders served by a route r
$\pi(i,r)$	predecessor of a customer i in the route r
$\mathcal{P}$	set of optional orders
q_i	demand associated to order i
Q_J	capacity of vehicle j
$q(r)$	total demand of route r
$\\|r\\|$	total distance traveled in route r
$\\|r\\|^*$	maximum distance between the depot and a costumer in the route r

r_{jk}	route travelled by vehicle j on day k
$\mathcal{R}_j$	routing plan of day j
ρ_{ij}	reachability relation that indicates if order i can be served by vehicle j
s_i	service time associated to order i
τ_{ij}	time for traveling from one customer i to another customer j
$t_j(r)$	transportation cost of vehicle j associated to each route r
v_i	ith order to be visited
$\chi_I(x)$	characteristic function of interval I

A.5.2 Application of tabu search (Section 5.3)

m	move
o	order involved in the move
or	old route of order o
op	position of order o in the old route or
nr	new route fro order o
np	position in the new route for order o
Ins	Insertion move: removal of an order from a route and its insertion in another one
$o1$	first order involved in the move
$o2$	second order involved in the move
$r1$	route of the first order $o1$
$r2$	route of the second order $o2$
InterSw	Inter-route swap: swap of two orders in different routes
IntraSw	Intra-route swap: swap of two orders in the same route
m_t	move in the tabu list
m_e	move under evaluation which might be excluded from the neighborhood by the fact that move m_t is in the tabu list

A.6 The multiple container loading problem with bearing weights and multi-drop

D	set of destinations
n	number of destinations ($\|D\|$)
α	estimation of the maximum ratio between the volume of a container and the volume of the boxes it actually carries
V_t	total volume of the boxes
V_c	volume of the container
V	estimation of the actual volume of boxes to be loaded
θ	ratio between the total volume of the boxes and the total volume of the containers
γ	total cost of the containers
δ	number of blocks in the current solution

A.7 A routing and packing problem

A.7.1 3L-CVRP formulation (Section 7.2.1)

$\mathcal{V}$	set of customers (or vertex or nodes)
n	number of customer

$\mathcal{A}$	set of arcs that connect nodes in a graph
G	graph individuated by arcs and vertices
(i, j)	arc of a graph connecting node i with node j
c_{ij}	cost associated to the arc (i, j)
m_i	number of items that composed the demand of customer i
q_i	demand associated to customer i (total weight of items)
w_{ik}	width of item k belonging to customer i
h_{ik}	height of item k belonging to customer i
l_{ik}	length of item k belonging to customer i
s_i	total volume of items belonging to customer i
f_{ik}	flag which indicated if item k of customer i
M	total number of items
$\mathcal{F}$	set of vehicles
v	number of vehicles ($\lvert\mathcal{F}\rvert$)
Q	capacity of a vehicle (total weight)
W	width of a vehicle
H	height of a vehicle
L	length of a vehicle
S	total loading volume of a vehicle

A.7.2 Complete problem (Section 7.2.2)

it	number of item types
w_i	width of item type i
h_i	height of item type i
l_i	length of item type i
uw_j	binary-valued parameter which indicates if the width can be positioned upright
ul_j	binary-valued parameter which indicates if the length can be positioned upright
uh_j	binary-valued parameter which indicates if the height can be positioned upright
μ_j	number of items of type j
wg_j	weight of items of type j
bw_j	maximum load which may be placed on the top surface of the item when the width side is placed upright
bl_j	maximum load which may be placed on the top surface of the item when the length side is placed upright
bh_j	maximum load which may be placed on the top surface of the item when the height side is placed upright
vt	number of vehicle types
W_j	width of a vehicle of type j
H_j	height of a vehicle of type j
L_j	length of a vehicle of type j
ν_j	number of vehicles of type j
Q_j	weight capacity of a vehicle of type j
C_j	cost of use of a vehicle of type j
λ	fixed length to verify the reachability constraint

A.7.3 Solution technique (Section 7.3)

$\bar{c}$	average edge distance

MCCS	Move Customer & Change Strategy move: removal of a customer from a route and its insertion in another, possibly changing the packing strategy
c	customer involved in the move
or	old route of customer c
op	position of customer c in the old route or
nr	new route
np	position on the new route nr
st	packing strategy
SC	Swap Customers move: exchanging a customer with another one
c_1	first customer involved in the move
c_2	second customer involved in the move
r_1	route of the first customer $c1$
r_2	route of the second customer $c2$
p_1	position of customer c_1 in route r_1
p_2	position of customer c_1 in route r_1
MRB	Move & Rotate Box: removal of a block from the blocks' sequence of a route and its insertion, possibly with a new orientation, into another route
bt	box type identifier
or	old route of the block
op	position of the block in the old route or
nr	new route
np	position of the block in the new route nr
r	rotation of the block
qty	number of boxes of the block that are involved in the move
φ	ratio between the total volume of boxes and the total volume of available vehicles
θ	ratio between the average customer demand (in m^3) and the average volume of vehicles
R	number of rounds of the SA(+intensifier) algorithm
γ_{MCCS}	probability of drawing a move of type MCCS
γ_{MRB}	probability of drawing a move of type MRB

List of Abbreviations

1D-BPP	One dimensional bin packing problem
3D-BPP	Three dimensional bin packing problem
3D-KPP	Three dimensional knapsack packing problem
3D-SPP	Three dimensional strip packing problem
3L-CVRP	Three-dimensional loading capacitated vehicle routing problem
ACO	Ant colony optimization
AI	Artificial intelligence
CLP	Container loading problem
CVRP	Capacitated vehicle routing problem
EA	Evolutionary algorithms
FSMD	Fleet size and mix vehicle routing problem with vehicle dependent routing costs
FSMF	Fleet size and mix vehicle routing problem with fixed costs
FSMFD	Fleet size and mix vehicle routing problem with fixed costs and vehicle dependent routing costs
FSMVRPTW	Fleet size vehicle routing problem with time windows
GA	Genetic algorithm
GLS	Guided local search
GLSM	Generalized local search machines
GRASP	Greedy randomized adaptive search
HC	Hill climbing
HVRP	Heterogeneous vehicle routing problem
HVRPD	Heterogeneous vehicle routing problem with vehicle dependent routing costs
HVRPFD	Heterogeneous vehicle routing problem with fixed costs and vehicle dependent routing costs
ILP	Integer linear programming
ILS	Iterative local search
IP	Integer programming
irace	Iterated $F-$race
LIFO	Last in first out
LS	Local search

MCLP	Multiple container loading problem
MCLPBWMD	Multiple container loading problem with bearing weights and multi-drop
MCPP	Multiple container packing problem
MD-PVRP	Multi-depot periodic vehicle routing problem
MD-VRP	Multi-depot vehicle routing problem
MP-VRP	Multi-pile vehicle routing problem
NOLH	Nearly orthogonal latin hypercubes
NP-hard	Non-deterministic polynomial-time hard
OR	Operations Research
PAS	Patient admission scheduling
PE-CTT	Post-enrolment course timetabling
PP-VRP	Pallet-packing vehicle routing problem
PVRP	Period (or periodic) vehicle routing problem
SA	Simulated annealing
SDVRP	Split delivery vehicle routing problem
SLS	Stochastic local search
TS	Tabu search
TSP	Traveling salesman problem
VNS	Variable neighborhood search
VRP	Vehicle routing problem
VRPPC	Vehicle routing problem with private fleet and common carrier
VRPTW	Vehicle routing problem with time windows
VRPTWCDC	Vehicle routing problems with time windows and carrier-dependent costs

Bibliography

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization.* John Wiley & Sons, Chichester, 1997.

[2] J. Alegre, M. Laguna, and J. Pacheco. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. *European Journal of Operational Research*, 179(3):736–746, June 2007. ISSN 03772217.

[3] A. D. Almeida and M. B. Figueiredo. A particular approach for the three-dimensional packing problem with additional constraints. *Computers & Operations Research*, 37(11):1968–1976, Nov. 2010.

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

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

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

[7] C. Archetti, D. Feillet, M. Gendreau, and M. Grazia Speranza. Complexity of the VRP and SDVRP. *Transportation Research Part C: Emerging Technologies*, 19(5):741–750, Jan. 2011.

[8] T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, USA, 1996. ISBN 0195099710.

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

[10] R. Baldacci, M. Battarra, and D. Vigo. Routing a heterogeneous fleet of vehicles. *The vehicle routing problem: latest advances and new challenges*, 43(1959):3–27, 2008.

[11] J. M. Belenguer, M. C. Martinez, and E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48(5):801–810, Sept. 2000.

[12] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[13] B. Bilgin, P. Demeester, and G. Vanden Berghe. A hyperheuristic approach to the patient admission scheduling problem. Technical report, KaHo Sint-Lieven, Gent, 2008.

[14] B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, and G. Vanden Berghe. One hyper-heuristic approach to two timetabling problems in health care. *Journal of Heuristics*, 2011. Online first `http://dx.doi.org/10.1007/s10732-011-9192-0`.

[15] M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective.* PhD thesis, Université Libre de Bruxelles, Belgium, 2004.

[16] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[17] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. *F-Race and iterated F-race: An overview.* Springer, Berlin, 2010. ISBN 3642025374.

[18] E. Bischoff. Three-dimensional packing of items with limited load bearing strength. *Journal of Operational Research*, 168(3):952–966, 2006.

[19] E. E. Bischoff and M. D. Marriott. A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44(2):267–276, 1990.

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

[21] E. E. Bischoff, F. Janetz, and M. S. W. Ratcliff. Loading pallets with non-identical items. *European Journal of Operational Research*, 84(3):681–692, 1995.

[22] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211, 1983.

[23] M.-C. Bolduc, J. Renaud, and F. Boctor. A heuristic for the routing and carrier selection problem. *European Journal of Operational Research*, 183(2):926–932, 2007.

[24] M.-C. Bolduc, J. Renaud, F. Boctor, and G. Laporte. A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society*, 59:776–787, 2008.

[25] A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 2010. Online first: http://dx.doi.org/10.1007/s10479-010-0707-0.

[26] A. Bortfeldt. A heuristic for multiple container loading problems. *OR Spectrum*, 22:239–261, 2000. In German.

[27] A. Bortfeldt. A hybrid algorithm for the capacitated vehicle routing problem with three dimensional loading constraints. Technical report, FernUniversität in Hagen, Dec. 2010.

[28] A. Bortfeldt and H. Gehring. A tabu search algorithm weakly heterogeneous container loading problem. *OR Spektrum*, 20:237–250, 1998. In German.

[29] A. Bortfeldt and H. Gehring. Two metaheuristics for strip packing problems. In D. Despotis and C. E. Zopounidis, editors, *Proceedings of the Fifth International Conference of the Decision Sciences Institute*, volume 2, pages 1153–1156, Athens, 1999.

[30] A. Bortfeldt and H. Gehring. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161, 2001.

[31] A. Bortfeldt and D. Mack. A heuristic for the three-dimensional strip packing problem. *European Journal of Operational Research*, 183(3):1267–1279, 2007.

[32] A. Bortfeldt, H. Gehring, and D. Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5):641–662, 2003.

[33] M. Boschetti. New lower bounds for the three-dimensional finite bin packing problem. *Discrete Applied Mathematics*, 140(1-3):241–258, May 2004.

[34] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, Feb. 2005.

[35] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, Feb. 2005.

[36] O. Bräysy, W. Dullaert, G. Hasle, D. Mester, and M. Gendreau. An effective multi-restart deterministic annealing metaheuristic for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, 42(3):371–386, 2008.

[37] E. K. Burke, P. De Causmaeker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004.

[38] S. Ceschia and A. Schaerf. Local search and lower bounds for the patient admission scheduling problem. *Computers & Operations Research*, 30:1452–1463, 2011.

[39] S. Ceschia and A. Schaerf. Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, pages 1–20, 2011. Online first: http://dx.doi.org/10.1007/s10732-011-9162-6.

[40] S. Ceschia, L. Di Gaspero, and A. Schaerf. Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. *Journal of Scheduling*, 14(6):601–615, 2011.

[41] S. Ceschia, A. Schaerf, and T. Stützle. Local search for a routing-packing problem. In *9th Metaheuristics International Conference (MIC 2011)*, Udine, Italy, July 25–28 2011.

[42] S. Ceschia, L. Di Gaspero, and A. Schaerf. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, 39(7):1615–1624, 2012.

[43] I.-M. Chao, B. L. Golden, and E. Wasil. An improved heuristic for the period vehicle routing problem. *Networks*, 26(1):25–44, Aug. 1995.

[44] C. H. Che, W. Huang, A. Lim, and W. Zhu. The multiple container loading cost minimization problem. *European Journal of Operational Research*, 214(3):501–511, Nov. 2011.

[45] C. Chen, S. Lee, and Q. Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76, 1995.

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

[47] E. Choi and D.-W. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34(7):2080–2095, 2007.

[48] N. Christofides and J. E. Beasley. The period routing problem. *Networks*, 14(2):237–256, 1984.

[49] N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44, Jan. 1977.

[50] C.-W. Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 127(3):657–667, September 2005.

[51] T. M. Cioppa and T. W. Lucas. Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49(1):45–55, 2007.

[52] G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568 – 581, 1964.

[53] 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, Sept. 1997.

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

[55] T. Crainic, G. Perboli, and R. Tadei. Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20(3):368–384, Jan. 2008.

[56] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, Oct. 1959.

[57] A. Davies. *Approaches to the container loading problem.* PhD thesis, University of Wales, Swansea, 2000.

[58] A. P. Davies and E. E. Bischoff. Weight distribution considerations in container loading. *European Journal of Operational Research*, 114(3):509–527, 1999.

[59] P. Demeester, W. Souffriau, P. De Causmaecker, and G. Vanden Berghe. A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine*, 48(1):61–70, January 2010.

[60] E. den Boef, J. Korst, S. Martello, D. Pisinger, and D. Vigo. Erratum to the three-dimensional bin packing problem: Robot-packable and orthogonal variants of packing problems. *Operations Research*, 53(4):735–736, July 2005.

[61] M. Desrochers and T. Verhoog. A new heuristic for the fleet size and mix vehicle routing problem. *Computers & Operations Research*, 18(3):263–274, Jan. 1991.

[62] L. Di Gaspero. *Local Search Techniques for Scheduling Problems: Algorithms and Software Tools.* PhD thesis, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, 2003.

[63] L. Di Gaspero and A. Schaerf. EasyLocal++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.

[64] L. Di Gaspero and A. Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modeling and Algorithms*, 5(1):65–89, 2006.

[65] L. Di Gaspero, A. Roli, and A. Schaerf. EasyAnalyzer: an object-oriented framework for the experimental analysis of stochastic local search algorithms. In T. Stützle, M. Birattari, and H. Hoos, editors, *Engineering Stochastic Local Search Algorithms (SLS-2007)*, number 4683 in Lecture Notes in Computer Science, pages 76–90. Springer-Verlag, 2007.

[66] M. Diaby and R. Ramesh. The distribution problem with carrier service: a dual based approach. *ORSA Journal on Computing*, 7(1):24–35, 1995.

[67] 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.

[68] M. Dorigo and T. Stützle. *Ant Colony Optimization.* The MITT Press, July 2004. ISBN 9780262042192.

[69] M. Dror. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254, May 1994.

[70] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2): 141, 1989.

[71] W. Dullaert, G. Janssens, K. Sörensen, and B. Vernimmen. New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, pages 1232–1238, 2002.

[72] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, January 1990.

[73] W. Eaton and G. A. Whitmore. Length of stay as a stochastic process: A general approach and application to hospitalization for schizophrenia. *The Journal of Mathematical Sociology*, 5(2):273–292, 1977.

[74] J. Egeblad and D. Pisinger. Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36(4):1026–1049, Apr. 2009.

[75] M. Eley. Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141(2):393–409, 2002.

[76] M. Eley. A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1):45–60, 2003.

[77] T. Fanslau and A. Bortfeldt. A tree search algorithm for solving the container loading problem. *INFORMS Journal of Computing*, 22(2):222–235, 2010.

[78] O. Faroe, D. Pisinger, and M. Zachariasen. Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing*, 15(3):267–283, July 2003.

[79] G. Fasano. Non-linear approximations for solving 3d-packing mip models : a heuristic approach. *Optimization*, pages 1–7, 2010.

[80] P. Fekete. New classes of fast lower bounds for bin packing. *Small*, 31:11–31, 2001.

[81] S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In R. Burkard and G. Woeginger, editors, *Algorithms ESA '97*, volume 1284 of *Lecture Notes in Computer Science*, pages 144–156. Springer Berlin / Heidelberg, 1997.

[82] S. P. Fekete, J. Schepers, and J. C. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587, May 2007.

[83] T. a. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, Mar. 1995.

[84] A. J. Forster, I. Stiell, G. Wells, A. J. Lee, and C. van Walraven. The effect of hospital occupancy on emergency department length of stay and patient disposition. *Academic emergency medicine : official journal of the Society for Academic Emergency Medicine*, 10(2):127–33, Feb. 2003.

[85] 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, 2010.

[86] M. R. Garey and D. S. Johnson. *Computers and Intractability—A guide to NP-completeness.* W.H. Freeman and Company, San Francisco, 1979.

[87] H. Gehring and A. Bortfeldt. A parallel genetic algorithm for solving the container loading problem. *Int. Transactions on Operational Research*, 9(4):497–511, 2002.

[88] P. Gemmel and R. Van Dierdonck. Admission scheduling in acute care hospitals: does the practice fit with the theory? *International Journal of Operations & Production Management*, 19(9):863–878, 1999.

[89] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, pages 1086–1094, 1992.

[90] M. Gendreau, G. Laporte, C. Musaraganyi, and E. D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 26 (12):1153–1173, 1999.

[91] 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.

[92] J. A. George and D. F. Robinson. A heuristic for packing boxes into a container. *Computers and Operations Research*, 7(3):147–156, 1980.

[93] J. A. George and D. F. Robinson. A heuristic for packing boxes into a container. *Computer & Operations Research*, 7(3):147–156, 1980.

[94] P. C. Gilmore and R. E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965.

[95] C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121):124–126, 1921. ISSN 00130133.

[96] F. Glover. Tabu search. Part I. *ORSA Journal of Computing*, 1:190–206, 1989.

[97] F. Glover. Tabu search. Part II. *ORSA Journal of Computing*, 2:4–32, 1990.

[98] A. Goel and T. Vidal. A hybrid genetic algorithm for combined vehicle routing and truck driver scheduling. In *The IX Metaheuristics International Conference (MIC 2011)*, pages 575–577, Udine, Italy, July 25–28 2011.

[99] B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66, Jan. 1984.

[100] E. Hans, G. Wullink, M. van Houdenhoven, and G. Kazemier. Robust surgery loading. *European Journal of Operational Research*, 185(3):1038–1050, 2008.

[101] K. He and W. Huang. An efficient placement heuristic for three-dimensional rectangular packing. *Computers & Operations Research*, 38(1):227–233, Jan. 2011.

[102] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, June 2009.

[103] M. Hifi, I. Kacem, S. Nègre, and L. Wu. A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics*, 36:993–1000, Aug. 2010.

[104] H. H. Hoos. *Stochastic local search-methods, models, applications.* PhD thesis, Darmstadt University of Technology / Germany, 1998.

[105] H. H. Hoos and T. Stützle. *Stochastic Local Search – Foundations and Applications.* Morgan Kaufmann Publishers, San Francisco, CA (USA), 2005. ISBN 1-55860-872-9.

[106] L. M. Hvattum. On the value of aspiration criteria in tabu search. In *The VIII Metaheuristics International Conference (MIC 2009)*, Hamburg, Germany, July 13th–16th 2009.

[107] M. Iori and S. Martello. Routing problems with loading constraints. *TOP*, 18(1):4–27, July 2010.

[108] N. Ivancic, K. Mathur, and B. Mohanty. An integer programming based heuristic approach to the three-dimensional packing problem. *Journal of Manufacturing and Operations Management*, 2(4):268–298, 1989.

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

[110] M. Jin, K. Liu, and B. Eksioglu. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*, 36(2):265–270, Mar. 2008.

[111] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 37 (6):865–892, 1989.

[112] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.

[113] L. Junqueira, R. Morabito, and D. Sato Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39:74–85, July 2012.

[114] M. Kang, C. Jang, and K. S. Yoon. Heuristics with a new block strategy for the single and multiple containers loading problems. *Journal of the Operational Research Society*, pages 1–13, December 2008.

[115] S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[116] R. Kusters and P. M. Groot. Modelling resource availability in general hospitals design and implementation of a decision support model. *European Journal of Operational Research*, 88 (3):428–445, Feb. 1996.

[117] G. Laporte. Fifty Years of Vehicle Routing. *Transportation Science*, 43(4):408–416, 2009.

[118] F. Li, B. Golden, and E. Wasil. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34(9):2734–2742, 2007.

[119] A. Lim, B. Rodrigues, and Y. Yang. 3-D container packing heuristics. *Applied Intelligence*, 22(2):125–134, Mar. 2005.

[120] F. H. Liu and S. Y. Shen. The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 50(7):721–732, 1999.

[121] J. Liu, Y. Yue, Z. Dong, C. Maple, and M. Keech. A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 38(4):797–807, Apr. 2011.

[122] 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, Jan. 1999.

[123] A. Lodi, S. Martello, and D. Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420, 2002.

[124] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The `irace` package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[125] H. Lourenço, O. Martin, T. Stützle, F. Glover, and G. Kochenberger. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science.* Kluwer Academic Publishers, Boston, 2003. ISBN 1-4020-7263-5.

[126] D. Mack, A. Bortfeldt, and H. Gehring. A parallel hybrid local search algorithm for the container loading problem. *Int. Transactions on Operational Research*, 11(5):511–533, 2004.

[127] Y. Marinakis and A. Migdalas. Annotated bibliography in vehicle routing. *Operational Research*, 7(1):27–46, January 2007.

[128] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48:256–267, 2000.

[129] S. Martello, D. Pisinger, D. Vigo, E. D. Boef, and J. Korst. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Trans. Math. Softw.*, 33(7), March 2007.

[130] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

[131] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7(4):326–329, Oct. 1960.

[132] A. Moura and J. F. Oliveira. A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 20:50–57, 2005.

[133] A. Moura and J. F. Oliveira. An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, 31(4):775–800, 2008.

[134] A. Moura and J. F. Oliveira. An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, 31(4):775–800, October 2009.

[135] I. H. Osman and S. Salhi. Local search strategies for the vehicle fleet mix problem. In V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, editors, *Modern Heuristic Search Methods*, chapter 8, pages 131–153. John Wiley & Sons, 1996.

[136] M. Padberg. Packing small boxes into a big box. *Mathematical Methods of Operations Research*, 52:1–21, 2000.

[137] F. Parreño, R. Alvarez-Valdes, J. Oliveira, and J. Tamarit. A maximal-space algorithm for the container loading problem. *INFORMS Journal of Computing*, 20(3):412–422, 2008.

[138] F. Parreño, R. Alvarez-Valdes, J. Oliveira, and J. Tamarit. Neighborhood structures for the container loading problem: a VNS implementation. *Journal of Heuristics*, 16(1):1–22, February 2010.

[139] D. Pisinger. A tree search heuristic for the container loading problem. *Ricerca Operativa*, 28:31–48, 1998.

[140] D. Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141:382–392, 2002.

[141] G. R. Raidl. The multiple container packing problem: a genetic algorithm approach with weighted codings. *ACM SIGAPP Applied Computing Review*, 7(2):22–31, 1999.

[142] G. R. Raidl and G. Kodydek. Genetic algorithms for the multiple container packing problem. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 875–884, London, UK, 1998. Springer-Verlag.

[143] M. Ratcliff and E. Bischoff. Allowing for weight considerations in container loading. *OR Spectrum*, 20:65–71, 1998.

[144] M. Reimann, K. F. Doerner, and R. F. Hartl. D-Ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591, Apr. 2004.

[145] L.-M. Rousseau, M. Gendreau, and G. Pesant. A general approach to the physician rostering problems. *Annals of Operations Research*, 115:193–205, 2002.

[146] R. Russell and W. Igo. An assignment routing problem. *Networks*, 9(1):1–17, 1979.

[147] R. A. Russell and D. Gribbin. A multiphase approach to the period routing problem. *Networks*, 21(7):747–765, Dec. 1991.

[148] S. Salhi and G. K. Rand. Incorporating vehicle routing into the vehicle fleet composition problem. *European Journal of Operational Research*, 66(3):313–330, May 1993.

[149] S. M. Sanchez. NOLH designs spreeadsheet. `http://diana.cs.nps.navy.mil/SeedLab/`, 2005. Visited on May 13, 2011. Last updated on April 7, 2006.

[150] S. Sang-Moon, S.-W. Lee, G.-T. Yeo, and M.-G. Jeon. An effective evolutionary algorithm for the multiple container packing problem. *Progress in Natural Science*, 18(3):337–344, 2008.

[151] G. Scheithauer. A three-dimensional bin packing algorithm. *Elektronische Informationsverarbeitung und Kybernetik*, 27(5/6):263–271, 1991.

[152] G. Scheithauer. LP-based bounds for the container and multi-container loading problem. *International Transactions in Operational Research*, 6(2):199–213, Mar. 1999.

[153] F. Semet and E. Taillard. Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41:469–488, 1993.

[154] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[155] E. Taillard. A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO Recherche Opérationnelle*, 33(1):1–14, 1999.

[156] E. D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4-5):443–455, 1991.

[157] C. Tan and J. Beasley. A heuristic algorithm for the period vehicle routing problem. *Omega*, 12(5):497–504, Jan. 1984.

[158] C. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. A hybrid metaheuristic algorothm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on intelligent trasportation systems*, 10(2):1524–9050, June 2009.

[159] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis. A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *Journal of the Operational Research Society*, 54(1):65–71, 2003.

[160] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, 152(1):148–158, 2004.

[161] J. Terno. An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, 123(2):372–381, June 2000.

[162] J. Thomson and K. Dowsland. General cooling schedules for simulated annealing-based timetabling system. In *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling (ICPTAT-95)*, pages 345–363, 1995.

[163] 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, Philadelphia, PA, USA, 2001. ISBN 0-89871-498-2.

[164] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. S.I.A.M., Philadelpia, PA (USA), 2002.

[165] F. Tricoire, K. F. Doerner, R. F. Hartl, and M. Iori. Heuristic and exact algorithms for the multi-pile vehicle routing problem. *OR Spectrum*, 33(4):931–959, Aug. 2009.

[166] E. Triki, Y. Collette, and P. Siarry. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research*, 166(1):77–92, Oct. 2005.

[167] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, Kluwer Academic Publishers Group, 1987.

[168] W. Vancroonenburg, M. Mısır, B. Bilgin, P. Demeester, and G. Vanden Berghe. A hyper-heuristic approach for assigning patients to hospital rooms. In *Proceedings of the 8^{th} International Conference on the Practice and Theory of Automated Timetabling (PATAT-2010)*, pages 553–555, 2010.

[169] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(l):41–51, 1985.

[170] W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. Statistics and Computing. Springer, 4^{th} edition, 2002.

[171] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Technical Report CIRRELT-2011-05, CIRRELT, 2011.

[172] T. Volgenant and R. Jonker. On some generalizations of the travelling-salesman problem. *Journal of the Operational Research Society*, 38(11):1073–1079, 1987.

[173] L. Wang, S. Guo, S. Chen, W. Zhu, and A. Lim. Two natural heuristics for 3D packing with practical loading constraints. In B.-T. Zhang and M. Orgun, editors, *PRICAI 2010: Trends in Artificial Intelligence*, volume 6230 of *Lecture Notes in Computer Science*, pages 256–267. Springer Berlin / Heidelberg, 2010.

[174] 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.

[175] N. A. Wassan and I. H. Osman. Tabu search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*, 53(7):768–782, 2002.

[176] H. Yaman. Formulations and Valid Inequalities for the Heterogeneous Vehicle Routing Problem. *Mathematical Programming*, 106(2):365–390, July 2005.

[177] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. The pallet-packing vehicle routing problem. *Transportation Science*, Oct. 2011. Online first http://dx.doi.org/10.1287/trsc.1110.0373.

[178] W. Zhu, W. Huang, and A. Lim. A prototype column generation strategy for the multiple container loading problem. In *9th Metaheuristics International Conference (MIC 2011)*, pages 1–32, 2011.