Multi-Objective Tools for the Vehicle Routing Problem with Time Windows

Juan Castro-Gutierrez

Thesis submitted to The University of Nottingham

for the degree of Doctor of Philosophy

March 2012

Dedicated to my parents Juan Luis and Nieves

Abstract

Most real-life problems involve the simultaneous optimisation of two or more, usually conflicting, objectives. Researchers have put a continuous effort into solving these problems in many different areas, such as engineering, finance and computer science. Over time, thanks to the increase in processing power, researchers have created methods which have become increasingly sophisticated. Most of these methods have been based on the notion of Pareto dominance, which assumes, sometimes erroneously, that the objectives have no known ranking of importance.

The Vehicle Routing Problem with Time Windows (VRPTW) is a logistics problem which in real-life applications appears to be multi-objective. This problem consists of designing the optimal set of routes to serve a number of customers within certain time slots. Despite this problem's high applicability to real-life domains (e.g. waste collection, fast-food delivery), most research in this area has been conducted with *hand-made* datasets. These datasets sometimes have a number of unrealistic features (e.g. the assumption that one unit of travel time corresponds to one unit of travel distance) and are therefore not adequate for the assessment of optimisers. Furthermore, very few studies have focused on the multi-objective nature of the VRPTW. That is, very few have studied how the optimisation of one objective affects the others.

This thesis proposes a number of novel tools (methods + dataset) to address the above-mentioned challenges: 1) an agent-based framework for cooperative search, 2) a novel multi-objective ranking approach, 3) a new dataset for the VRPTW, 4) a study of the pair-wise relationships between five common objectives in VRPTW, and 5) a simplified Multi-objective Discrete Particle Swarm Optimisation for the VRPTW.

Acknowledgements

The research presented in this thesis was undertaken under the supervision of Dr. Dario Landa-Silva to whom I am deeply grateful for his invaluable input and guidance. I am also profoundly thankful to Professor José Andrés Moreno Pérez for his patience, continuous encouragement, support and friendship as co-advisor.

I wish to express my sincere gratitude to the members of the Group of Intelligent Computing (GCI) at the University of La Laguna (Spain). Thanks to Marcos Moreno for introducing me to field of Heuristics, and to him and Belén Melián to give me the opportunity to work with them as a research assistant. Thanks also to Julio Brito for sharing his knowledge with me.

I want to thank all my beloved friends in Tenerife. Thanks to Miguel Ángel (Miki) for being always there for me, for being like a brother. Thanks to Jonatan (Jony) for sharing dreams, happiness, sadness, films, ... Thanks to Santi for his friendship and encouraging conversations. Thanks also to Carlos and Javi for their friendship and the collections of good memories I have from our undergraduate times. Thanks to Inés, Elsa and Eva for their unconditional friendship and support.

I want to express my gratitude to all the friends I have made through these Ph.D. years. Thanks to Elkin for being my brother in the UK, for those long and inspiring conversations, for being there in both good and bad moments. Thanks to those special housemates and friends, such as Tiago and Daniel Muller to whom I am deeply grateful for sharing their life with me. Thanks to Daniel Soria for his sincere friendship and continuous support. Thanks also to Pawel, Claudio, María, Fran, Madda, Ben, Pilar, Javier, Alberto, and a long etcetera of good friends who have shared pieces of their

lives with me.

I would also like to thank '*mijn geliefde en lieve vrouw*' Hilde for her support and love, for her advice and for proof-reading this work, for helping me to look always at the bright side and for cheering me up in those hard moments.

This work would have not been possible without the continuous love, support and encouragement of my parents Juan Luis and Nieves, my brothers Javier and Sergio, and my uncle José Castro.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Most real-world problems are multi-objective in nature. Multi-objective optimisation problems have a number of objectives that are usually in conflict, so that improving one objective leads to worsening another. A large number of tools (methods + datasets) have been proposed to tackle multi-objective optimisation problems. In particular, a big effort has been put into creating elaborate solving methods. Several optimisation frameworks have been put forward to address the design and implementation of such tools. Thus, multi-objective optimisation frameworks, such as ParadiEO-MOEO [166] or HeuristicLab [252], provide a toolbox upon which to create, test and compare different methods. Many solving methods for multi-objective optimisation involve the notion of cooperation (e.g. Particle Swarm Optimisation [150], Ant Colony Optimisation [74]). Even though most successful multi-objective optimisation frameworks support cooperative algorithms, they do not seem to provide explicit mechanisms to work with cooperative systems (e.g. full control over the communication topology, type of information to be shared).

Another important topic in multi-objective optimisation is how to rank solutions with multiple objectives. There is a variety of methods to discriminate solutions in multi-objective spaces. Two well-known methods are Lexicographic ordering and Pareto dominance. In the lexicographic approach, the decision maker assigns a priority to

each objective. Solutions are then compared objective by objective according to these priorities. A limitation of the lexicographic method is that the decision-maker must express preferences in order to establish the ordering and this is not always easy. Pareto dominance gives the same importance to all objectives. A limitation of Pareto dominance is therefore the lack of flexibility in the trade-off between improvements and detriments to different objective values during the search process. The lack of flexibility in Pareto dominance leads to scalability problems. In fact, some studies [138, 152, 203] have concluded that Pareto dominance cannot succeed in dealing with optimisation problems with more that three objectives.

Road transport problems are examples of multi-objective optimisation problems. These problems have a serious impact on society. Research has shown that society is negatively affected by inefficient road transport. In the EU a quarter of the final energy consumption and 93% of greenhouse gas emissions comes from the road transport sector [137]. Congestion costs are estimated to be as high as approximately 1% of the GDP in Western Europe, which is commonly accepted as too high [127]. In addition, transport causes noise pollution and loss of biodiversity, which have become prominent issues [170]. Transport route planning and control were identified as one of the most effective policy actions to reach a sustainable transport system [127]. Road transport problems are usually addressed as Vehicle Routing Problems (VRP). The VRP is the problem of finding a set of least-cost routes to serve a number of customers. VRP belongs to the most intensively studied problems in operations research. The most well-known variant of the VRP is the Vehicle Routing Problem with Time Windows (VRPTW) which considers a series of time slots in which customers must be served. The VRPTW has many applications to real-life problems, such as waste collection [220], fast-food routing [218] and many others [118]. Interestingly, most VRP and VRPTW solving techniques have been assessed using *hand-made* datasets. These datasets sometimes present unrealistic features and are not entirely adequate for the assessment of optimisers, especially when dealing with multiple objectives [36]. Furthermore, very little research exists regarding the relationships between common objectives in VRP and VRPTW. That is, there is not much information available on how the optimisation

of a particular objective affects the others. This information might be useful to rule out objectives when the relationship is positive, or to divide the problem into sub-problems when the objectives are not related [204].

The work in this thesis presents a number of tools (methods + datasets) for multi-objective optimisation. In particular, we focus on the application of these tools to the Vehicle Routing Problem with Time Windows (VRPTW) with multiple objectives.

1.2 Thesis Structure

This thesis is organised as follows:

Chapter 2 provides the required background for this thesis. This chapter is divided into four sections which provide relevant information about the Vehicle Routing Problem with Time Windows (VRPTW), Multi-Objective Optimisation (MOO), Cooperation in Multi-Agent Systems (MAS), and Particle Swarm Optimisation (PSO).

Chapter 3 introduces CODEA, a software framework to develop cooperative Multi-Agent Systems (MAS) to tackle Multi-objective Optimisation Problems (MOP). This chapter first provides an in-depth overview of some successful optimisation frameworks for multi-objective optimisation. Secondly, it presents CODEA and describes its structure and how each component works. Finally, it discusses four other studies which have benefited from CODEA.

Chapter 4 presents the Dynamic Lexicographic Approach (DLA). DLA is a novel ranking method to discriminate solutions with multiple objectives. This chapter describes DLA and compares its performance to that of Pareto Dominance and Lexicographic approach. Experiments are carried out using a canonical Multi-Objective Discrete Particle Swarm Optimisation (MOPSO) with eight objectives.

Chapter 5 examines a number of benchmarking dataset problems for the Vehicle Routing Problem with Time Windows. In particular, this chapter focuses on the Solomon's dataset by providing an in-depth study of its most important features. Special emphasis is put on the analysis of some unrealistic features of this dataset, and how these fea-

tures might affect its suitability for the assessment of multi-objective optimisation algorithms. This chapter also specifies a number of ways to overcome Solomon's dataset's unrealistic features.

Chapter 6 presents MOVRPTW, a real-world based dataset for the assessment of the Vehicle Routing Problem with Time Windows (VRPTW). This dataset is introduced as an alternative to the Solomon's dataset. This chapter details the characteristics of the MOVRPTW dataset, and explains how its features might help to better assess multi-objective optimisation methods.

Chapter 7 focuses on achieving a better understanding about the multi-objective nature of the VRPTW. This chapter first studies the relationships that occur between pairs of objectives throughout the optimisation process. It then compares the multi-objective suitability of Solomon's dataset to that of the MOVRPTW dataset. This comparison is based on the correlation values in non-dominated sets found by a multi-objective genetic algorithm using both datasets.

Chapter 8 presents a simplified Multi-objective Discrete Particle Swarm Optimisation (MODPSO) to tackle the Vehicle Routing Problem with Time Windows (VRPTW). This chapter first provides a brief overview of PSO algorithms related to our MODPSO. It then presents a high-level description of our algorithm and the way it is applied to VRPTW. The performance of this MODPSO is compared to that of NSGA-II using Solomon's and MOVRPTW datasets.

Chapter 9 addresses the conclusion of our research. We summarise the aim of our investigation and state a conclusion for each chapter. Moreover, we specify the most promising lines of investigation according to our experiences and results.

1.3 Contribution of this thesis

This thesis contributes to the field of multi-objective optimisation by providing:

- An object-oriented software framework to develop systems based on groups of agents that cooperate to tackle complex multi-objective optimisation problems.

Two versions of this framework are presented: CODEA $v2$ and CODEA $v3$. CODEA $v2$ is the first version of CODEA that supports multi-objective optimisation. CODEA $v3$ is the product of the hybridisation of CODEA $v2$ and ParadisEO-MOEO [166]. In particular, CODEA $v3$ presents a number of competitive features with respect to those provided by state-of-the-art MOO frameworks.

- A new ranking approach to discriminate solutions with multiple objectives. The Dynamic Lexicographic Approach (DLA) eliminates the need for the Decision-Maker (DM) to establish fixed priorities among the competing objectives, which is often difficult. At the same time, DLA offers more flexibility to navigate constrained combinatorial search spaces than Pareto dominance, which treats all objectives with no relative order of importance. Results in a Particle Swarm Optimisation (PSO) applied to the Vehicle Routing Problem with Time Windows (VRPTW) with eight objectives indicate that DLA is superior to Pareto dominance and the ordinary lexicographic approach.

- A novel and challenging real-world based dataset for the assessment of the VRPTW with multiple-objectives. This dataset is based on a real test case of a distribution company. The company provided the geographic locations, information about time windows profiles and type of demands of more than 1000 customers. Customers' locations were used to calculate travel distances and travel times with the Google Maps database. In addition to the MOVRPTW dataset, we provide a configurable dataset generator. This generator allows the specification of a number of parameters, such as the size of the instance, the demands profile and the time windows profile.

- An in-depth study of the multi-objective nature of the Vehicle Routing Problem with Time Windows (VRPTW), in particular the conflicting relationships between 5 common objectives: number of vehicles, total travel distance, makespan, total waiting time, and total delay time. This study investigates the (conflicting) nature of various objectives in the VRPTW and shows that some of the classic test instances are not suitable for conducting a proper multi-objective study. It also

shows that the MOVRPTW dataset is more adequate for the assessment of multi-objective methods.

- A simplified Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) for the Vehicle Routing Problem with Time Windows (VRPTW). This algorithm consists of only two components: 1) a Discrete Particle Swarm Optimisation [54], and 2) the notion of Dominance Depth Fitness Assignment introduced by Goldberg [116]. No other mechanism is used to boost convergence or diversity. Results show that this simplified MODPSO outperforms NSGA-II [65] in two VRPTW datasets with 5 common objectives.

1.4 Articles resulting from this thesis

The following publications are related to the work presented in thesis:

1. J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimization approach. In N. Krasnogor, B. Melián-Batista, J. Pérez, J. Moreno-Vega, and D. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, volume 236 of *Studies in Computational Intelligence*, pages 103–114. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-03210-3

2. J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Dynamic lexicographic approach for heuristic multi-objective optimization. In *Proceedings of the Workshop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009)*, pages 153–163, Seville (Spain), 2009

3. J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Codea - an agent based multi-objective optimization framework. In *VII Congreso Español sobre Metaheuristicas, Algoritmos Evolutivos y Bioinspirados (MAEB 2010)*, pages 319–327, Valencia, Spain, September 2010

4. J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Improved dynamic lexicographic ordering for multi-objective optimisation. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 31–40. Springer Berlin / Heidelberg, 2011

5. J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 257–264, 2011

6. R. Qu, Y. Xu, J. Castro-Gutierrez, and D. Landa-Silva. Particle swarm optimization for the steiner tree in graph and delay-constrained multicast routing problems. *Journal of Heuristics*, pages 1–26, 2012

7. J. Castro-Gutierrez and D. Landa-Silva. Dynamic lexicographic approach for many-objective combinatorial optimisation. *Preparing for submission*, 2012

8. J. Castro-Gutierrez and D. Landa-Silva. Investigating pair-wise relationships between objectives in the vehicle routing problem with time windows. *Preparing for submission*, 2012

9. J. Castro-Gutierrez and D. Landa-Silva. A simplified multi-objective discrete particle swarm optimisation for the vehicle routing problem with time windows. *Preparing for submission*, 2012

10. J. Castro-Gutierrez and D. Landa-Silva. Asynchronous cooperative multi-agent search for the university course time tabling problem. *Preparing for submission*, 2012

CHAPTER 2

Literature Review

Summary

This chapter introduces a number of background concepts related to these research areas which frame our contribution. We introduce each topic and provide an in-depth explanation and relevant citations.

2.1 Introduction

This chapter is organised as follows:

- Section 2.2 introduces the Vehicle Routing Problem (VRP) and one of its most popular variants, the Vehicle Routing Problem with Time Windows (VRPTW). We first introduce both VRP and VRPTW and then review some common solving techniques.
- Section 2.3 presents a number of key concepts concerning Multi-Objective Optimisation (MOO). Besides, we provide an overview of the most common techniques used in MOO.
- Section 2.4 briefly introduces the concept of cooperative search within the context of multi-agent systems.
- Section 2.5 introduces Particle Swarm Optimisation (PSO). This sections covers

the original PSO, and two variants: discrete PSO and Multi-Objective PSO.

2.2 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a very well known combinatorial optimisation problem. The goal is to design a least-cost route-plan to serve a number of customers. The main interest in this problem resides on its applicability in the field of road transport. This section presents the Vehicle Routing Problem (VRP), and one of its extension the Vehicle Routing Problem with Time Windows (VRPTW).

The following defines a VRP instance [243]:

- There is a set of $N+1$ *vertices*, $V = \{v_0, v_1, \ldots, v_N\}$. These vertices denote the customers and depot geographic locations.
- There is a set of *customers* (which is a subset of the *vertices*) $C = \{v_1, \ldots, v_N\}$. Each customer $v_i \in C$ requires a certain amount of goods q_i (demand) and has a location given by (x_i, y_i).
- There is a depot, $v_0 \in V$, from which goods are dispatched. It is assumed that the depot has an unlimited amount of goods. The depot has location given by (x_0, y_0), and has null demand, $q_0 = 0$.
- There is a fleet of delivery vehicles with the same capacity, $Q \geq maxq_i : i = 1, \ldots, N$.

The VRP is to find a set of K circuits or routes (each corresponding to a delivery vehicle) to serve all customers at the minimum cost. The cost is usually given by the sum of travel distances of all routes. The travel distance between a pair of customers (vertices) $v_i, v_j \in V$ is denoted by $d_{i,j}$ and is often calculated using the Euclidean distance:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.2.1}$$

There are a number of hard constraints in the VRP:

- All K circuits must start and end at the depot.

- Each customer must be served by exactly one delivery vehicle.
- The sum of demands of the customers in each route cannot exceed the maximum capacity Q of a delivery vehicle.

The Vehicle Routing Problem (VRP) has a number of variants. Some of the most known VRP variants are: Multi-Depot VRP (MDVRP), in which goods can be delivered from more than one depot; Periodic VRP (PVRP) in which the planning period is more than one day; Split Delivery VRP (SDVRP) where a single customer can be served by different vehicles; Stochastic VRP (SVRP) in which one or several components of the problem are random; VRP with Backhauls (VRPB) where customers can demand or return some commodities and VRP with Time Windows (VRPTW) in which customers must be served within a time slot. An in-depth review of the VRP and its variants can be found in [119]. The next sub-section focuses on the VRPTW as it is a central topic in this thesis.

2.2.1 Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) is one of the most studied extensions of the VRP. This extension considers the following:

- Each pair of customers (vertices) $v_i, v_j \in V$ has a *travel time*. This is the time that takes to go from customer v_i to customer v_j, and is denoted by $t_{i,j}$.
- Each customer has a *time window* (time slot) $[a_i, b_i]$. This time window represents the time by which each customer wants to be served.
- Each customer has a *service time*. This is the time that takes to unload the goods, once the delivery vehicle has arrived at the customer's location.

Time windows are hard constraints in the VRPTW. It is possible to arrive earlier to a customer's location, in which case the delivery vehicle must wait until the time window opens (*waiting time*), but it is not allowed to arrive after it closes. However, due to the high applicability to real-world scenarios, many authors treat time windows as

soft constraints. Thus, arriving late at a customer's location is allowed, but a penalty is introduced in the objective function.

2.2.2 Solution Techniques for VRP and VRPTW

In this section we discuss some solution techniques for both VRP and VRPTW. Most solution techniques for VRP and VRPTW fall in three categories: *exact methods*, *heuristics* and *metaheuristics*.

Exact Methods –

These methods provide optimal solutions but can be computationally expensive or simply intractable. There are two main lines of research in relation to exact methods for VRPTW [58]. One line concerns the generation of inequalities to make the LP relaxation stronger, and the other considers mathematical decomposition techniques. The LP relaxation consists of replacing the constraint stating that each variable takes value 0 or 1 by a weaker constraint stating that each variable takes a value between 0 and 1. There are three main exact methods to solve the VRPTW: *Lagrangian relaxation*, *column generation* and *branch-and-cut*.

Lagrangian relaxation methods approximate problems by relaxing their hard constraints. These methods place these hard constraints into the objective function and assign weights (penalties) to them. These weights are known as Lagrangian multipliers. Some works using Lagrangian relaxation for VRPTW are those by Fisher [96], Fisher et al. [98] and Kohl and Madsen [154].

"Column generation is intimately related to Lagrangean relaxation and can be seen as a special way to update the multipliers associated with the relaxed constraints" [58]. Some works using column generation include those by Desrochers et al. [70], Kohl et al. [155], Cook and Rich [55], Kallehauge et al. [147], Danna and Le Pape [62], Feillet et al. [90] and Chabrier [38].

A Branch-and-cut algorithm is an exact method that combines: 1) a cutting plane

method [245], and 2) a branch-and-bound [157] algorithm. A cutting plane method is a procedure to find integer solutions in linear programming problems. It first solves the non-integer linear program, and then checks if the solution found is also an integer solution. If the solution found is not integer, a new constraint that excludes the non-integer solution is added to the program. The method repeats this process until the optimal integer solution is found. Branch-and-bound is another technique for solving integer linear programs. It systematically enumerates solutions in a search tree and uses bounds to discard nodes that do not lead to the optimal solution. The first exact method for the VRPTW was a branch-and-bound algorithm by Kolen et al. [156]. Some works have been proposed in this area to address the VRPTW, such as Bard et al. [8] and Lysgaard [174].

Reviews on formulations and exact algorithms for the VRPTW can be found in Cordeau et al. [56, 59], Kallehauge [146], El-Sherbeny [85] and Baldacci et al. [7].

While realistic Travelling Salesman Problem (TSP) instances with several thousands of nodes can be solved optimally [121], VRP instances with more than a hundred nodes are currently impractical [108]. For this reason, we find an extensive amount of literature on Heuristic and Metaheuristic methods for the VRP. These solving techniques are discussed in the next two subsections.

Heuristics –

Heuristic methods are solving techniques that produce good quality solutions with a modest computing time. Most heuristic approaches in this area include constructive and improvement steps.

Constructive heuristics build feasible routes step by step. Sequence of customers are assigned to routes using the cost as the main criteria. The most commonly used constructive heuristics are:

- *Pure constructive heuristics* focus on the creation of routes producing results very fast. We distinguish two types:
  - *Saving heuristics* first assign a different route to each customer. At each

iteration, two routes are merged into a single route using a saving measure. This process is then repeated until no merge is possible without incurring in a violation. This strategy was proposed by Clarke and Wright in [45] and it is one of the most known heuristics for the VRP. This algorithm works for both directed and undirected graphs. However, Vigo [247] stated that the performance gets worse in the directed case. Another criticism to this algorithm is its tendency to produce worse routes as the optimisation process progresses. A way to overcome this issue was proposed by Gaskell [106] and Yellow [262]. They changed the equation of savings stated in the original algorithm $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ to a new one with a route shape parameter $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$. This parameter λ attributes the importance in the selection to the distance between nodes rather than their distance respect to the depot. Another variation of this algorithm is the *Matching-based savings algorithm*. This algorithm changes the saving function to $s_{pq} = t(S_p) + t(S_q) - t(S_p \bigcup S_q)$, where S_x is the vertex set of the route x and $t(S_x)$ is the length of TSP optimal solution on S_x. More information about *Saving heuristics* can be found in the works by Desrochers and Verhoog [69], Altinkemer and Gavish [3] and Wark and Holt [256]. Experiments in [59] indicate that the approach of Clarke and Wright [45] is, due to its simplicity, the best constructive heuristic in terms of speed. However, the best results in terms of the objective function value are obtained with the approach by Wark and Holt [256]. The Saving heuristics was extended to the VRPTW by Solomon [233]. This extension considered both the spatial and temporal closeness of customers by setting a maximum waiting time per route.

– *Insertion heuristics* are also very fast algorithms to build route-plans. Given a set of unvisited customers C^*, a customer $c \in C^*$ is en-routed if the operation produces a feasible route at least cost. Two well known algorithms using this approach were proposed by Mole and Jameson [180] and Christofides et al. [43]. The first approach expands a route step by step using a generali-

sation of the Clarke and Wright [45] algorithm. The second method operates in two steps. The first step uses a sequential insertion algorithm to find a set of feasible routes. The second step uses a parallel insertion approach. As the last step, these two insertion heuristics apply a 3-Opt [167] procedure to improve the quality route-plan. Chrisofides et al. [43] carried out comparisons between these algorithms. According to their results, the algorithm by Christofides et al. [43] gets better route-plans in less time compared to the approach by Mole and Jameson [180]. Solomon [233] proposed a number of extensions of insertion heuristics for the VRPTW. For example, the Solomon's insertion algorithm consists of adding customers to a route according to temporal closeness. This algorithm creates a new route when no feasible insertion is possible. Another heuristic by Solomon [233], called $I1$, first selects the farthest customer respect to the depot as a *seed* customer. This customer is then assigned to an empty route and unvisited customers are inserted into this route according to a saving measure. The saving measure takes into account the distance and the extra time required to visit a customer. When a feasible insertion is no longer possible, the algorithm creates a new route with a new seed customer. Dullaert and Bräysy [79] presented a modification of this algorithm for routing problems with a few customers per route (between 5 and 25). This modification is in fact an improvement that overcomes the Solomon's $I1$ underestimation of additional time required to insert a customer between the depot and the first customer being served. This new proposal produces good results for problems with up to 15-20 customers per route. Ioannou et al. [143] used a criteria selection that takes into account both visited and unvisited customers. Potvin and Rousseau [198] proposed a parallel version of the Solomon's I1. This approach had a regret measure to chose the next customer to be en-routed. This regret measure calculates the cost that occurs when a customer is not chosen at each step in the route building process.

- *Two phase construction heuristics*. These techniques, as their name suggests, de-

compose the problem in two phases. Depending on the order to execute them, we find two approaches: Cluster-first, Route-second and Route-first, Cluster-second.

– *Cluster-first, Route-second* first creates groups of customers in clusters using some criteria in the first step, and optimises the resulting routes in the second. The optimisation carried out in the second part often uses methods borrowed from TSP solving techniques. There are three main categories for Cluster-first, Route-second methods: *elementary clustering*, *truncated branch-and-bound* and *petal algorithms*.

  Examples of *elementary clustering* methods are the *sweep algorithm*, *generalised assignment based algorithm* and *location based heuristics*. Gillet and Miller [112] proposed the sweep algorithm. This algorithm creates clusters of feasible customers (or groups of customers which do not cause constraint violations) by sweeping them using an angle translation centred at the depot. All customers in the same cluster are assigned to the same route. Once the routes are created, a post-optimisation process tries to improve the route-plan by moving customers between routes. The Fisher and Jaikumar [97] *(generalised assignment based)* algorithm is a modification of the Clarke and Wright's [45] algorithm. This method performs the clustering by solving the Generalised Assignment Problem (GAP). GAP consists of assigning a number of tasks (customers) to a number of agents (vehicles). Once the clusters are found and a route is formed for each cluster, TSP-based algorithms are employed to improve the routes. A known example of the *location based heuristics* is the algorithm by Bramel and Simchi-Levi [16]. In this case, the clustering process is done by creating a number of route seeds. Each seed is calculated solving a *capacitated location problem*. The number of seeds created is equal to the size of the fleet. This algorithm, together with the one by Fisher and Jaikumar [97], requires the setting of the number of vehicles a priori. The effectiveness of these three algorithms is comparable to that of the insertion algorithms. However, the approach proposed by Bramel and Simchi-Levi [16]

produces better solutions at the cost of more computing time.

Another method in the *Cluster-first, Route-second* class is the *truncated branch-and-bound*. This algorithm is a mere simplification of an exact algorithm that solves the VRP. An example of this approach was proposed by Toth et al. [43]. Compared to the proposal of the exact version described in [42], Toth et al.'s method stores a single search tree with K (number of vehicles) levels. This family of algorithms tends to produce better solutions than constructive methods. Other results indicate that these procedures are faster and more effective than sweep algorithms [59].

Petal algorithms are the last type of algorithms in the *Cluster-first, Route-second* family. This algorithm consists of a natural extension of the sweep algorithm explained above. In the first part, it creates a long list of feasible routes (called *petals*) with a very similar method to the sweep algorithm [112]. In the second part, a set of final routes are selected by solving a set of partitioning problem. The use of this approach can be found in the work of Foster and Ryan [101], Ryan et al. [219] and Renaud et al. [211]. Petals algorithms usually perform better than the sweep algorithms [59].

– *Route-first, Cluster-second* methods relax the problem constraints, such as vehicle capacities and/or time windows, to build a *big tour* (TSP tour). This big tour is then split into feasible trips. Some relevant works in this topic are those by Beasley [11], Haimovich and Kan [123] and Bertsimas and Shimchi-Levi [16]. These methods were first applied to the VRPTW by Solomon [233]. This approach does not give remarkable results in comparison to others [59].

Improvement heuristics apply operators to find better neighbouring routes. These operators work by moving nodes (customers) within the route-plan. There are basically two types of moves: intra-route *(if only a single route is considered)* and inter-route *(if more than one route is considered at the same time)*. The operators that works in **intra-**

route improvements are mainly inherited from those used for TSP. One of the most general mechanisms was proposed by Lin [167] and is called the λ-operator (λ-opt). The λ-opt eliminates λ edges from a route and creates other λ edges, usually after trying all possible combinations. A number of alternatives to this operator appear in the VRP literature. For example, a dynamic λ that reports on average the best results was suggested by Lin and Kernighan [168]. Another alternative consists of moving strings of consecutive nodes to other locations in the route, this approach is called *Or*-opt and was proposed in [191]. It is worth noting that many of the heuristics already explained in this chapter use these operators. The **inter-route** improvements have also been well studied in the literature. Using the classification by Van Breedam [22], it is possible to highlight four types of operators: string cross, string exchange, string relocation and string mix. Given a route-plan R and two routes within R, $r_1, r_2 \in R$:

- The *string cross* operator exchanges two edges in two different routes r_1 and r_2. That is, if $(i-1, j)$ and (m, n) are edges in r_1 and r_2 respectively, we first remove two edges $(i-1, i)$ and $(j, j+1)$ from the first route, and two edges $(m-1, m)$ and $(n, n+1)$ from the second route. Then, four new edges are introduced $(i-1, m)$, $(n, j+1)$, $(m-1, i)$ and $(j, n+1)$.

- The *string exchange* operator swaps a pair of nodes in different routes. In this case, if i is a node in r_1 and j is a node in r_2, the edges $(i-1, i)$, $(i, i+1)$, $(j-1, j)$ and $(j, j+1)$ are replaced by $(i-1, j)$, $(j, i+1)$, $(j-1, i)$ and $(i, j+1)$.

- The *String relocation* operator moves nodes from one to another route. Thus, if i is the node in r_1 to be relocated, and $(j, j+1)$ is the destination edge in r_2, the edges $(i-1, i)$, $(i, i+1)$ and $(j, j+1)$ are replaced by $(i-1, i+1)$, (j, i) and $(i, j+1)$.

- The *String mix* operator selects the best move between *string exchange* and *string relocation*.

Improvement movements for VRPTW are based on those used for TSP and VRP. These operators concern edge-interchanges and are known as k-opt. The most common k-opt are 2-*opt*, 3-*opt* and the *inter-route improvements*.

- Replacements involving two edges (2-opt). These operators remove two edges in a route-plan and replace them with another two. As a result, this process produces a new feasible solution. The 2-opt was proposed for the TSP by Croes [61]. This algorithm inverts sub-sequences of customers, thus it is sometimes difficult to obtain route-plans that satisfy time window constrains. A modified version of this algorithm called 2-opt* was proposed for VRPTW by Potvin and Rousseau [199]. This approach replaces two edges from two different routes but with the aim of preserving the visiting order.

- Replacements involving three edges (3-opt). These operators remove three edges in a route-plan and replace them with another three. The 3-opt was proposed for the TSP by Lin [167]. This algorithm does not invert the order of customers in the intervening edges. This makes it easier to find a feasible solution after each interchange. A special case of this operator is the above-mentioned *Or*-opt [191]. Gendreau et al. [109] proposed the operator GENI. This operator relocates a customer in another route in between the pair customers who are closest to it, even if they are not consecutive.

- Inter-route improvements. This operator consists of interchanging two customers on two different routes. Ejection chains [113] have produced the best results in this area. These algorithms move a customer from one route to another. In case of producing an infeasible route, a customer from the latter route is moved to another route. This process continues until no customer needs to be ejected. The cyclic *k*-transfers proposed by Thompson and Psaraftis [241] interchange a number of customers between routes on a cyclical basis. The CROSS interchanges by Taillard et al. [236] interchange two sequences of customers on two different routes.

As most real-life VRPTW problem instances require the use of heuristics [21], these methods have drawn much attention from the OR community. A standard topic in this area is the development of heuristic solutions to tackle larger and more realistic problems [108]. However, we find that most algorithms in the literature are assessed using

hand-made datasets which might have unrealistic features [36]. In my opinion, some effort should be put into the development of more realistic and challenging datasets to further improve solution techniques in this area. This would enable better assessment and thereby improve heuristic essential attributes such as flexibility and robustness [21].

Metaheuristics –

These methods are general search strategies that promote a global exploration of problem search spaces. Many metaheuristics have been developed to tackle the Vehicle Routing Problem and its variants for more than 15 years. In this subsection, we divide the most common approaches into three classes as in [59]: *Local search approaches*, *Population Search approaches* and *Learning mechanism approaches*. Note that many approaches discussed in this subsection may be part of more than one class.

Local search approaches explore the search space of a problem by changing some attributes of a solution systematically. Given a solution x, these methods explore the neighbourhood of x, denoted as $N(x)$ by moving from one to another solution until a stop criterion is met. A number of local search based algorithms exist such as Simulated Annealing, Deterministic Annealing and Tabu Search. We will focus on these three metaheuristics, since they have been widely applied to the VRP(TW).

- *Variable Neighbourhood Search (VNS)* is a well-established metaheuristic that systematically changes neighbourhoods in a local search. Rousseau et al. [215] proposed three constraint programming based operators to perform a local search. These operators were combined in a Variable Neighbourhood Descent (VND) framework. The first operator (LNS-GENI - Large Neighbourhood Search - GENI) removes a subset of customers and subsequently reintroduces it at better locations in partial solutions. The second operator (NEC - Naive Ejection Chain) uses ejection chains to create new solutions. The third operator (SMART - SMAll RouTing) removes arcs and subsequently solves smaller VRP instances. Bräysy [19] presented a four-phase approach to tackle VRPTW. The first phase uses an heuristic based on the Solomon's insertion heuristic [233] to create set of initial solu-

tions. The second phase uses an ejection-based approach to reduce the number of routes. The third and fourth phases seek the improvement of the solutions in terms of travel distance by using Variable Neighbourhood Decent (VND). In this last phase, the waiting time is considered to avoid local optima. Results showed that this approach outperforms other local search techniques and metaheuristics.

- *Simulated Annealing (SA)* is a well established metaheuristic that explores the search space as a probabilistic local search. This algorithm randomly draws a solution x' from its neighbourhood $N(x')$. If its fitness value is not worse than the one of the current solution x_0, the current solution is updated with x'. Otherwise, this new solution is accepted with probability p or rejected with $1 - p$. This probability p is calculated by means of a factor (*temperature*) that decreases throughout the iterations. One of the best Simulated Annealing algorithms for the VRP was proposed by Osman [192] in 1993. A $2 - Opt$ to define the neighbourhood and a special rule to update the temperature were his major contributions. However, this was not enough to beat the best contemporary tabu search.

- *Deterministic Annealing (DA)* is a very similar approach to Simulated Annealing, but dropping the probabilistic acceptance component. There are two main versions which were introduced by Dueck in [78, 77]. The first accepts a new solution $x_{new} = x$, if $f(x) < f(x_{old}) + \theta_1$ (assuming minisation), where θ_1 is parameter specified by the user. The second approach will update the current solution $x_{new} = x$, if $f(x) < \theta_2 f(x_{old})$, where θ_2 is also a user controlled parameter.

- *Tabu Search (TS)* is another local search based algorithm that maintains a list of solution features to avoid re-visiting solutions. The memory structure used by this method is called *tabu list* and acts as a short-term memory. At iteration t, the cycling is prevented by avoiding those solutions in $N(x_t)$ contained in the *tabu list*. The first application of Tabu Search to VRPTW was proposed by Potvin et al. [201]. This algorithm used the Solomon's $I1$ in order to initialise route-plans, and 2-opt* and Or-opt to improve them. One of the most successful implementation of the Tabu Search is the one by Taillard et al. [236]. This Tabu Search uses $1 - interchange$ and a TSP-based periodic re-optimisation mechanism

in the routes. Other important works for VRPTW involving Tabu Search are those by Chiang et al. [41], Cordeau et al. [57], Bräysy and Gendreau [20] and Lau et al. [159].

Population search approaches are search techniques that keep a set of individuals following some pre-established rules. Each member in the population holds a solution that changes over the generations during the search process.

- *Genetic Algorithm (GA)* is population-based solving method that evolves a number of individuals mimicking natural evolution. Thangliah [239] proposed a GA cluster-first route-second based method to tackle the VRPTW. This method used an adaptive genetic based clustering method and a local post-optimisation procedure to move customers between clusters and enforce feasibility. Homberger and Gehring [131] presented two evolutionary strategies for solving the VRPTW. These strategies were based on the (μ, λ)-evolution strategy by Schwefel [226]. This strategy starts with a population of μ individuals. Some of these individuals are selected and recombined, producing an offspring with $\lambda > \mu$ individuals. These new individuals are then mutated and the λ fittest individuals are kept in the new population. The fitness of the individuals is assessed, first, according to the number of vehicles and, second, to the total distance travelled. Both algorithms used 2-opt, Or-opt and 1-interchange operators for the mutation of individuals. The same authors proposed two works involving a two-phase hybrid algorithm consisting of GA+TS in [107, 132]. The first phase used an Evolutionary Strategy to minimise the number of vehicles, and the second phase used a Tabu Search to minimise the total distance traveled. Another hybrid was proposed by Berger et al. [14]. That approach consists of evolving two populations simultaneously. Each population aimed at the minimisation of one of the two objectives: total travel distance and temporal constraint violation. The initialisation of the population was carried out by using a sequential insertion heuristic. This work also introduces a number of operators based on key features of heuristics such as Solomon's $I1$ and large neighbourhood search [229]. Mester and Bräysy [177] proposed a two-phase approach consisting of a Guided Local Search (GLS) [250] and

evolution strategies metaheuristics. In the first phase, a guided local search used a number of operators including 2-opt*, Or-opt and 1-interchange. In the second phase, the evolution strategy local search removes a number of customers from the current solution and re-inserts them in optimal positions within the route-plan. Other important contributions in this area include the works by Alvarenga et al. [4], Ghoseiri and Gehring [111], and Ursani et al. [246].

- *Particle Swarm Optimisation (PSO)* is a population-based algorithm that evolves a number of individuals inspired by the movement of swarms such as flock of birds or fish schooling. An in-depth review of PSO is provided in Section 2.5, as this algorithm will be used in this research.

Learning mechanism approaches are all those algorithms that somehow benefit from the experience gained from previous iterations. The two most remarkable approaches in this class are *Neural Networks* and *Ant Colony Optimisation*.

- *Neural Networks (NN)* are models that simulate biological neural networks. It consists of a network of processing units, referred as *neurones*, whose complexity emerges as a result of the global interaction. A very limited number of publications has been carried out on the application of NN to the VRP. The most common application to the VRP has been done by using *Self-Organising Maps*, see for instance the work of Ghaziri [110].

- *Ant Colony Optimisation (ACO)* is a bio-inspired algorithm proposed by Dorigo [74]. This algorithm mimics the complex system that arises from an ant colony searching for food. An example of a well-performing ACO-based algorithm for the VRP is D-Ants [209]. D-Ants first applies a $2-opt$ and a variant of the Clarke and Wright's [45] heuristic with a saving cost equation based on the reinforcement given by the learning process. Another important contribution was proposed by Potvin et al. [200] for the VRP with backhauls and time windows. This algorithm produces solutions close to the optimum (1% gap) with up to 100 nodes instances. Other publications using ACO applied to the VRPTW include the works by Gambardella et al. [104], Tan et al. [259], Hu et al. [136] and Zhang et al. [264].

Metaheuristic techniques usually provide solutions with much better quality than those generated with heuristics. However, this improvement in the quality of solutions is at the expense of: 1) higher CPU and memory consumption, 2) more complex development, and 3) the need for tuning of parameters. The choice between using heuristics or metaheuristics therefore, depends on the relative importance of time versus quality. The literature shows an increasing trend for creating more elaborate metaheuristic methods. This raises three important questions:

1. Are all components in these algorithms necessary to achieve a good performance?
2. Is there a way to ease the development of all these solving techniques?
3. Are we using the right datasets to test our algorithms?

This PhD thesis aims to address these open questions throughout a series of studies:

1. It shows that a canonical Particle Swarm Optimiser (PSO) with very simple operators is able to evolve poor infeasible solutions to feasible regions (see Chapter 8).
2. It presents an object-oriented framework for the design of cooperative systems of agents [33].
3. It analyses weaknesses in the characteristics of the most common dataset for assessment of the VRPTW, and present a new dataset to overcome those weaknesses [36].

Researchers have addressed most transport routing problems as single-objective optimisation problems. However, the vast majority of real-world problems are multi-objective in nature. Some important objectives found in real-world problems include: minimising the total travel time and/or travel distance, minimising the number of vehicles, maximising customers' satisfaction with on-time deliveries and maximising drivers' satisfaction with workloads balancing. An excellent summary and classification of these and other objectives are provided in [145]. New challenges in multi-objective VRP include objectives related to environmentally friendly transportation [94], and the use of real-time traffic information to avoid congestions [44].

Detailed surveys on solving techniques for VRP and its variants can be found in the works by Golden et al. [119], Pereira et al. [195] and El-Sherbeny [85].

2.3 Multi-objective Optimisation

Multi-objective optimisation is the process of simultaneously optimising two or more conflicting objectives subject to a number of constraints.

In mathematical terms, a minimisation multi-objective optimisation problem (note that we assume it is minimisation problem without loss of generality), can be written as:

$$minimise\, \vec{y} = f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})) \tag{2.3.1}$$

subject to:

$$\vec{x} = (x_1, x_2, \ldots, x_m) \in X. \tag{2.3.2}$$

where $\vec{x}$ is the vector of decision variables, X is the decision feasible region and $\Re^m$ is the decision space; $\vec{x} \in X \subset \Re^m$. The objective vector is denoted by $\vec{y} = (y_1, y_2, ..., y_n)$, $Y = f(X)$ is the objective feasible region and $\Re^n$ is the objective space; $\vec{y} \in Y \subset \Re^n$.

Definition If there exists a solution $\vec{x^*} \in X$ that minimises all objective functions simultaneously, $\vec{x^*}$ is *ideal vector*.

In other words, a feasible solution $\vec{x^*} \in X$ is *ideal solution* if there is no other $\vec{x} \in X$ and $i \in \{1, 2, \ldots, n\}$ such that:

$$f_i(\vec{x}) < f_i(\vec{x^*}). \tag{2.3.3}$$

2.3.1 Pareto Efficiency

Usually, there is not a single solution that optimises all objectives at the same time; i.e. an ideal solution. One way to address multi-objective optimisation problems is by using the concept of *Pareto Efficiency* [83]. Mathematically:

Definition Given two decision vectors $\vec{x_1}, \vec{x_2} \in X$, $\vec{x_1}$ is said to *dominate* $\vec{x_2}$ (denoted as $\vec{x_1} \prec \vec{x_2}$) iff:

$$\forall i \in \{1, 2, \ldots, n\} : f_i(\vec{x_1}) \leq f_i(\vec{x_2}),$$
$$\exists j \in \{1, 2, \ldots, n\} : f_j(\vec{x_1}) < f_j(\vec{x_2}).$$

Definition Given two decision vectors $\vec{x_1}, \vec{x_2} \in X$, $\vec{x_1}$ is said to *cover* $\vec{x_2}$ (denoted as $\vec{x_1} \preceq \vec{x_2}$) iff $\vec{x_1} \prec \vec{x_2}$ or $f(\vec{x_1}) = f(\vec{x_2})$.

Definition A vector of decision variables $\vec{x^*} \in X$ is *non-dominated* iff there is no other $\vec{x} \in X$, such that $\vec{x} \prec \vec{x^*}$.

Definition The *Pareto optimal Set* $\mathcal{P}^*$ is described as:

$\mathcal{P}^* = \{\vec{x} \in X : \vec{x}$ is Pareto Optimal$\}$.

Definition The *Pareto front* $\mathcal{PF}^*$ is defined as: $\mathcal{PF}^* = \{\vec{y} = f(\vec{x}) \in Y : \vec{x} \in \mathcal{P}^*\}$.

2.3.2 Multi-objective approaches

Theory in multi-objective optimisation has models to evaluate and establish a preference on the solutions in our *Pareto optimal set*. There are two main theories based on [53]:

- *Multi-attribute utility theory* (MAUT) [148] focuses on the composition of mathematical functions to aid the process of selecting the best combination among objectives in a set of solutions. Specifically, this technique enables the decision maker to quantify the desirability of certain alternatives. The outcome will be a function that represents the decision maker preference.
- *Multi-criteria decision aid theory* (MCDA) [217] is an alternative approach to study multi-objective classification problems. MCDA approaches focuses on the model development aspects that are related to the modelling and representation of the decision makers preference, value and judgment policy [76].

2.3.3 Methodologies for Multi-Objective Optimisation

A large number of methods have been proposed to tackle Multi-objective Optimisation Problems (MOPs). This review does not attempt to provide an in-depth analysis of all publications in this area because that is out of the scope of this thesis. We provide instead a broad and concise overview of existing Multi-objective Approaches.

One way to categorise optimisation methods concerns the moment in which we establish the preference [84]. We can distinguish three families:

- *Priori optimisation methods* refer to selecting the preferred solutions *a priori*, that is, before the execution of the optimisation method. The drawback of this methodology is the high complexity involving the creation of a model to specify a desirable trade-off.
- *Progressive optimisation methods* refer to asking the decision maker to direct the search. This way, the algorithm prompts the user with questions in order to decide how to search the space. The disadvantage of this procedure lies on the requirement of the time of the decision maker. However, this might not be a problem if the execution time is not long, but if that is not the case, this method could be unaffordable.
- *A posteriori optimisation methods* aim at collecting a number of solutions regularly spaced in the solution space to be post-processed. Given this set of solutions, the decision maker is able to select the most preferred ones. The drawback of this methodology is that finding those solutions might be difficult and computationally impracticable.

Most multi-objective solution methods fall in one of these three families of methods. Solution methods that do not fit in any of these categories are hybrid approaches (a combination of techniques using above-mentioned families). We now briefly describe a number of well-known multi-objective optimisation methods. For each method, we specify to which of the above families they belong. According to the nature of the method, we can find:

Scalar methods: Most methods in this category fall in the first (*A priori*) and third (*A posteriori*) families. Some well-known examples are the:

- *Weighted sum of objective functions* [47] involves the transformation of a multiple objectives problem into a single-objective problem. This method is set *a priori* when the decision maker establishes a preference by weighting objectives. All objectives are merged into a single one as a weighted composition. It must be noted that the *weighted-sum approach* can be also used as *a posteriori* or as *progressive* solution method. The main advantage of this technique is that is relatively easy to implement and it is not computationally expensive. However, this method has a number of drawbacks. For example, it can be difficult to determine the weights. Besides, due to the transformation into a single-objective problem, the optimisation method will only provide one solution. This problem can be overcome by modifying the weights in the weighted function and re-running the algorithm again (at a higher computational cost). The most serious pitfall of this approach is that it cannot generate some portions of the Pareto Front when this is non-convex.

- *Lexicographic ordering* [95] is a similar model in which the decision maker provides the order in which the objectives will be compared. Thus, when two solutions are compared, the objective with the highest importance (priority) is used first. If both solutions are indifferent for this objective, the algorithm proceeds to compare the objective with the second higher priority. This process is repeated until: 1) one objective is better than another, 2) no objective is left to be compared. This approach is very easy to implement, computationally cheap and easier to set up compared to *weighted approaches*. The *Lexicographic ordering* is useful when there is a clear precedence of importance among objectives. Furthermore, its performance is good compared to other methods when the number of objectives is small (2 or 3 objectives).

- *Target vector approaches* require the decision maker to create a vector of goals for each objective function. Thus, the search process is directed to approximate the solutions to these goals imposed *a priori*. Unlike *weighted sum approaches*, *target*

vector approaches are able to generate portions of concave Pareto front under certain conditions. Example of this class of techniques are *Goal Programming* [63], *Goal attainment* [263] and the *min-max algorithm* [124].

- *ϵ-constraint* methods consists of selecting (*a priori*) a primary objective to minimise, while the rest of objectives are rewritten as constraints. This way, each objective is bounded by a permitted quantity ϵ. It is worth noticing that it is possible to generate the entire Pareto front by changing the ϵ bounds. Although this technique is quite simple, depending on the number of objectives, it may be computationally expensive.

Interactive methods belong to the *progressive* family. Decision makers operate these methods to guide the search. The product of this interactive process is a single solution. Examples of these methods are:

- *Surrogate-Worth Trade-off method* (SWT) is based on the above mentioned *ϵ-constraint* method, but embedded in an interactive process. This method was developed by Haimes and Hall [122] who conceived it to approximate an underlying function. The interaction takes place when the decision maker sets the direction of the search based on a number of parameters that size trade-off levels. The drawback of this method is related to the parameter setting.

- *Step Method* [13] (STEM) uses preference information given by the decision maker to restrict the search space step by step. In this method, the DM has to specify which objectives have an acceptable value and which do not. Then, in order to improve the objectives with unacceptable values, the DM provides bounds by which the objectives with acceptable values can be worsened. STEM was one of the first iterative method proposed for multi-objective optimisation. One criticism to this methods is that it does not capture the trade-off between objectives. Another criticism is that the best-compromise solution does not exists if it is not found in a certain number of steps [50].

Fuzzy methods are designed to deal with uncertainty and imprecision, characteristics

of human knowledge. This class of methods tend to be placed in the *a priori* family. Two well known examples of Fuzzy methods are those by Sakawa and Yano [221] and Reardon [208]. The former method uses fuzzy logic at all levels (parameters of the problems and constraints). The advantage of this method is that the DM's preference is taken into account in relation to the solution. Thus, the DM can obtain more or less solutions depending on how demanding he/she is. The drawback of this method is that uses complex mathematical rules that are hard to apply [53]. The latter method (*Reardon method*) is an easier approach which uses a simple membership function to penalise the objective function when it is out of certain bounds. This method is easy to use and provides good results in some test functions.

Decision aid methods belong to *a priori* family, as the preference is established before the execution of the multi-objective optimisation method. Well known methods under this category are *ELECTRE (ELimination and Choice Expressing REality)* and *PROMETHEE (Preference Ranking Organisation METHod for Enrichment Evaluations)*. The first method was introduced by Roy [216] when he was facing some problems applying the weighted approach. Initially, it was proposed to select the best action from a set of possible actions, but later on, it was extended to be applied to choosing, ranking and sorting problems. *PROMETHEE* is another outranking method proposed by Brans [18] which uses pair-wise comparisons to find the best action or decision . This method requires a very low level of mathematical complexity what makes it transparent to both decision makers and non experts (i.e. stakeholders).

Techniques based on metaheuristics fall in the *a priori* and *a posteriori* families. This group of methods are eligible to separate the *multi-objective optimisation treatment* (i.e. *weighted approach*) from the *optimisation method*. This explains why many single-objective metaheuristics have been easily extended to deal with multi-objective decisional spaces. Some of the first multi-objective optimisation algorithms proposed in this area are: *PASA (Pareto Archived Simulated Annealing)* [86] and *MOSA (Multi-Objective Simulated Annealing*) [244], *VEGA (Vector Evaluated Genetic Algorithm)*[225], *MOGA (Multiple Objective Genetic Algorithm)* [100], *NSGA (Non-dominated Sorting Genetic Algorithm)* [234] and the *NPGA (Niched Pareto Genetic Algorithm)* [133]. However, in the literature, there

now exist many more metaheuristic approaches to tackle multi-objective problems.

PASA [86] and MOSA [244] are based on the Simulated Annealing [24] algorithm. PASA [86] uses an aggregation approach and an archive to store non-dominated solutions. This algorithm only works with a solution at a time. Thus, in order to generate the Pareto trade-off surface, it is necessary to re-run the search starting from solutions within the non-dominated archive. MOSA [244] uses an acceptance probability for each objective function to accept or reject solutions depending on the temperature. Then, these probabilities are merged using weighting approaches.

Many approaches have been proposed to deal with multi-objective problems using Genetic Algorithms. One of them is *VEGA*, proposed by Schaffer [225]. This algorithm was the first evolutionary multi-objective algorithm. It extends GAs to tackle multi-objective problems without merging the objective functions. The main difference with other GA lies in the selection step. At each generation, a number of groups (sub-populations) are generated according to one objective function. This makes this algorithm very simple to implement. However, since we are taking one objective at a time, the population might end up with average values for all objectives.

MOGA was proposed by Fonseca and Fleming [100]. This approach uses Pareto dominance in order to compute the efficiency of the individuals. Thus, individuals are ranked according to the number of population members they dominate. This algorithm is easy to implement, as well as efficient, but in some cases it does not provide a diverse set of solutions.

NSGA was proposed by Srinivas and Deb [234]. NSGA is based on the above-mentioned MOGA. This approach promotes diversity by using a different manner to compute the individuals' efficiency. This process is carried out by classifying the individuals with a dummy efficiency value to normalise the likelihood of reproduction. This efficiency parameter, to which all the objective function are reduced, is indeed what makes NSGA efficient. A problem with this algorithm is its sensitiveness respect to the sharing factor (θ_{share}) that represents the crowdedness of each category. Another important feature in the NSGA is the *non-dominated sorting mechanism*. This procedure ranks a population of solutions using the concept of Pareto dominance. This approach consists of divid-

ing the population in fronts according to non-domination relationships. When this process is over, all solutions contained in the first front are said to belong to the best non-dominated set. Each front is assigned with a fitness value according to its depth or closeness to the Pareto front. The non-dominated sorting mechanism received some criticism for its high complexity $O(MN^3)$, where M is the number of objectives and N is the size of the population. In order to overcome this issue, a new version of this algorithm with complexity $O(MN^2)$ was introduced in NSGA-II [65]. This new version is more efficient in terms of computational time and avoids the use of the sharing factor with elitism and crowded comparison operators.

NPGA was presented by Horn and Nafpliotis [133]. This algorithm is based on NSGA differing only in the selection procedure. It starts by selecting two random members of the population. Then, these two individuals are compared with a subset of the population. In case of a tie, fitness sharing is used to break the deadlock. The advantage of this method is the partial use of the Pareto ranking scheme. It makes the method run faster comparisons. However, as *NSGA*, its performance relies in a number of parameters that must be set.

A large number of publications have been put forward since the proposal of these first algorithms. Evolutionary Multi-objective Optimisation Algorithms (EMOAs) have caught the interest of many researchers due to their ability to find multiple Pareto optimal solutions in a single run [65]. Some state-of-the-art EMOAs include: SMS-EMOA [17], MO-CMA-ES [140] and ϵ-MOEA [66].

S-Metric Selection EMOA (SMS-EMOA) [17] combines the hypervolume metric [271] as a selection criteria to discard the lesast fit individuals, and the non-dominated sorting [65] as a ranking criterion. Thanks to the use of the hypervolume as a selection criteria, the performance of this algorithm is independent to the number of objectives. The Multi-Objective Covariance Matrix Adaptation Evolution Strategy (MO-CMA-ES [140]) is another algorithm that uses the hypervolume and non-dominated sorting mechanisms. However, in this case MO-CMA-ES employs a non-dominated sorting that uses either the contributing hypervolume or the NSGA-II's crowding distance [65]. The ϵ-MOEA [66] divides the search space in a number of hyper-boxes (grid)

such that each section of the grid only contains one solution. This algorithm updates an external archive for non-dominated solutions using the notion of ϵ-dominance [160]. All of the above mentioned EMOAs outperform the NSGA-II. In particular SMS-EMOA not only outperforms NSGA-II, but also achieves better results than the other algorithms respect to the hypervolume indicator.

Many successful state-of-the-art EMOAs, such as SMS-EMOA [17] or ϵ-MOEA [66], are a combination of pre-existing components. In my opinion, a major advance in this area could be the use of a unified framework upon which to develop and test these methods. A number of frameworks have been proposed for this purpose, such as ParadisEO [25] and HeuristicLab [252, 251]. The adoption of these frameworks could speed up the improvement of existing solving techniques, and thereby broaden the knowledge in this area. This could provide, among other things, a common "arena" in which to compare algorithms, a set of fine-grained tools that can be easily combined (e.g. non-dominated sorting mechanism [65], hypervolume metric [271]), and a set of predefined benchmark test problems.

2.3.4 Quality Metrics for Multi-Objective Optimisation

An active field of research in multi-objective optimisation focuses on how to assess multi-objective algorithms. One methodology to assess the performance of multi-objective algorithms is the *quality metrics*. Quality metrics (or quality indicators) usually help to measure the performance of these algorithms according to their final non-dominated solution set. However, one trend in the area of Multi-Objective Evolutionary Algorithms (MOEA) concerns the adoption of quality indicators as selection mechanisms (i.e. IBEA [270], or the more recent SPAM [276]). Some existing metrics that we have used throughout our studies are:

Hypervolume Metric [271], also known as S-metric or Lebesgue metric, is one of the most important unary indicators to assess the quality of Pareto front approximations. The hypervolume measures the size of the space covered by a Pareto front approximation set $\mathcal{A} \subset Y$ and a reference (or nadir) point $z \in Y$. In a maximisation problem, the reference point z is usually set to the origin. While in a minimisation problem, z is

usually set to a point dominated by all solutions in the approximation set $\mathcal{A}$.

Formally, given an objective space with n dimensions, the hypervolume of an approximation set $\mathcal{A} \in Y$, denoted by $M_H(\mathcal{A}) \in \Re$, can be calculated as [271]:

$$M_H(\mathcal{A}) = \lambda\Big(\bigcup_{(x_1,\ldots,x_n)\in\mathcal{A}} [x_1, z_1] \times \cdots \times [x_n, z_n] \Big) \tag{2.3.4}$$

where $\lambda(\cdot)$ is the standard Lebesgue measure.

Figure 2.1 shows (on the left) the graphical representation of this metric for the minimisation of two objectives: f_1 and f_2. In this example, the approximation set $\mathcal{A}$ contains 5 non-dominated solutions: $\{x_1, x_2, x_3, x_4, x_5\}$. Since this is a minimisation problem, the reference point z is set to point dominated by all solutions in $\mathcal{A}$. This way, the hypervolume is represented by the grey region delimited by the non-dominated solutions x_1, x_2, x_3, x_4, x_5 and z.

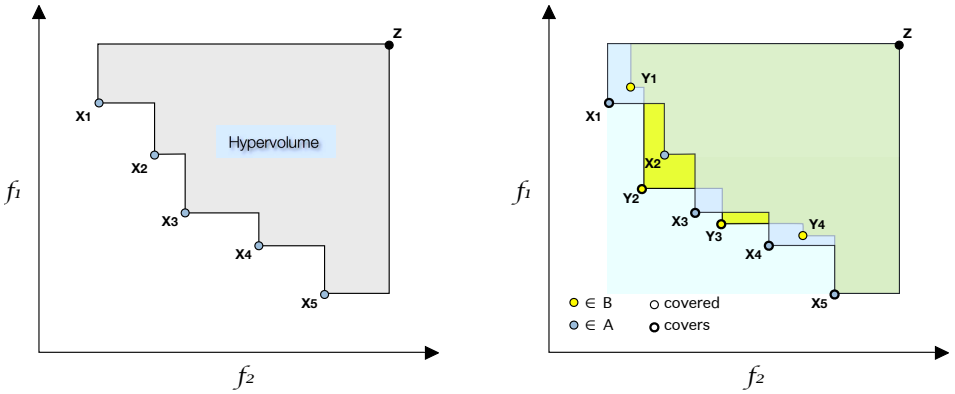


Figure 2.1: Graphical representation of the hypervolume metric (on the left) and the coverage metric (on the right). In this coverage metric example, $M_C(\mathcal{A}, \mathcal{B}) = 2/5$ because $Y1$ and $Y4$ are covered and $\mathcal{A}$ has 5 solutions, while $M_C(\mathcal{B}, \mathcal{A}) = 1/4$ because $X2$ is covered and $\mathcal{B}$ has 4 solutions.

Coverage Metric [273] is a binary performance indicator that compares to what extent a solution set $\mathcal{B}$ is covered by another solution set $\mathcal{A}$. Given two Pareto front approximation sets $\mathcal{A}$ and $\mathcal{B}$, the coverage of $\mathcal{A}$ over $\mathcal{B}$, denoted by $M_C(\mathcal{A}, \mathcal{B}) \in \Re$, is equal to the number of solutions in $\mathcal{B}$ that are covered by solutions in $\mathcal{A}$ to the cardinality of $\mathcal{B}$. Formally,

$$M_C(\mathcal{A}, \mathcal{B}) = \frac{|\{b \in \mathcal{B} : \exists a \in \mathcal{A}, a \preceq b\}|}{|\mathcal{B}|} \tag{2.3.5}$$

This function maps the ordered pair $(\mathcal{A}, \mathcal{B})$ to $[0, 1]$. If $M_C(\mathcal{A}, \mathcal{B}) = 1$, it means that all solutions in $\mathcal{B}$ are covered by solutions in $\mathcal{A}$. Conversely, $M_C(\mathcal{A}, \mathcal{B}) = 0$ indicates that no solution in $\mathcal{B}$ is covered by solutions in $\mathcal{A}$. This metric therefore shows the algorithm that provides the approximation set with the larger coverage to be better. It should be noted that $M_C(\mathcal{A}, \mathcal{B})$ is not necessarily equal to $1 - M_C(\mathcal{B}, \mathcal{A})$, and we must therefore calculate both.

Figure 2.1 depicts (on the right) the graphical representation of this metric. In this case, there are two approximation sets: $\{x_1, x_2, x_3, x_4, x_5\} \in \mathcal{A}$ and $\{y_1, y_2, y_3, y_4\} \in \mathcal{B}$. The values for this metric are $M_C(\mathcal{A}, \mathcal{B}) = 2/5$ and $M_C(\mathcal{B}, \mathcal{A}) = 1/4$. Thus, according to this metric the non-dominated set $\mathcal{A}$ is better than $\mathcal{B}$.

A large number of reviews have been published explaining these and other quality indicators. These reviews include the works by Knowles and Corne [153], Grosan et al. [120], Lizarraga et al. [169] and Cheng at al. [40].

2.4 Cooperation

Multi-Agent Systems (MAS) [92] is a research area that studies systems of interacting intelligent agents. These systems are useful to solve complex problems by cooperation. Cooperation is a fundamental research topic within the context of multi-agent systems.

Figure 2.2 provides a high-level topology of cooperation in the context of MAS [73]. At the top of the hierarchy we have *independent* and *cooperative* systems. In *independent* MAS each agent perform its own operations independently of the other agents. As Figure 2.2 depicts, independent MAS can be of the type *discrete* and *emergent cooperation*. Discrete MAS contain agents with no operations in common. Emergent cooperation occurs when agents cooperate unintentionally. For example, RoboCup soccer [186] is a MAS in which each agent pursues its own agenda. As a result of the global interaction

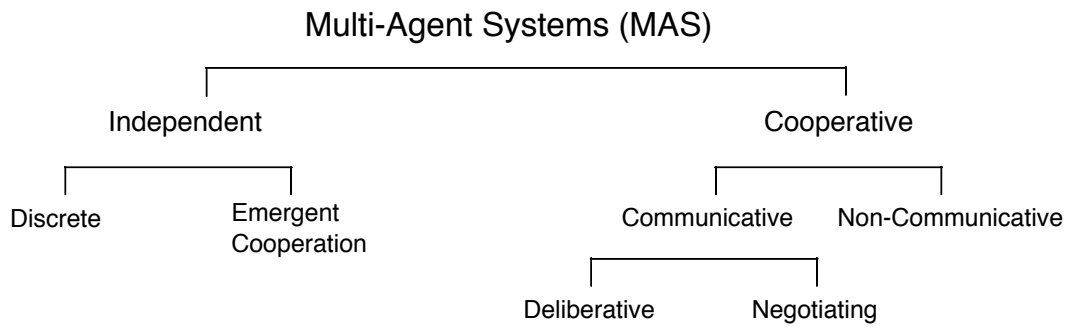


Figure 2.2: Classification of Multi-Agent Systems (MAS) based on [73].

involving agent's independent behaviours, a cooperative system emerges. Cooperative systems (top of the topology in Figure 2.2) can be *communicative* and *non-communicative* depending on wether or not an interchange of signals or information takes place. In non-communicative schemes, cooperation occurs as the result of agent's observations (e.g. robots that observe each other's behaviour [210]). Communicative cooperative MAS can be *deliberative* and *negotiating*. In the former scheme, agents plan their actions as a group. The latter is similar to deliberative schemes but it adds competition.

Cooperative MAS can be also categorised as *centralised* and *decentralised* [260]. A MAS is centralised if there is an entity controlling and/or supervising what others agents do. The complement of this scheme is a system in which agent run their operations independently or in a decentralised way. Another classification for cooperative MAS concerns the synchrony in their operations [237]. This way, MAS can be *synchronous* if agents have to wait for one another to perform certain operations, or *asynchronous* if no agent is forced to wait for the others.

This thesis contributes to the area of multi-agent systems with CODEA, a library of classes that supports all of the above-mentioned schemes (see Chapter 3). However, we have focused our research on the hybridisation of *independent* > *emergent cooperation* and *cooperative* > *communicative* systems (Figure 2.2) and *decentralised* cooperation systems. That is, MAS in which agents are independent as they pursue their own agenda, and cooperate using sending and receiving messages. An example of this system is Swam Intelligence (SI) [72]. Swarm Intelligence is a MAS in which each agent performs simple operations independently and in a decentralised fashion. In particular,

we have focused part of our research on the development of a Particle Swarm Optimisation (PSO) algorithm in CODEA. The next section discusses PSO existing variants, and states our contribution to this field.

2.5 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a stochastic swarm intelligence-based technique developed by Kennedy and Eberhart [81, 150]. This algorithm is inspired by the social behaviour of bird flocking or fish schooling.

PSO consists of a number of particles moving in the problem solution space. Each particle knows its current position x^{id}, its best position so far b^{id}, the best positioned particle in the neighbourhood g^{id}, and the best location achieved by the whole swarm up to this time g. Using this information, the particles in the swarm are able to update their positions. This position is updated by applying the equations 2.5.1 and 2.5.2.

The first equation (2.5.1) updates the velocity of each particle v^{id} and has four basic components, each one indicated with an over-brace:

- The first component (v^{id}) from the left encourages particles to move with the same speed and direction as in the previous iteration (also called inertial component).
- The second component $(g^{id} - x^{id})$ pushes particles towards the best location in the neighbourhood at the current iteration. This component is responsible for avoiding premature convergence.
- The third component $(b^{id} - x^{id})$ is called cognition and directs particles to their own best locations.
- The fourth component $(g - x^{id})$ keeps particles together by creating an attraction force to the best position found by the whole swarm up to time.

Each of the four components in Equation 2.5.1 is multiplied by coefficients w, c_1, c_2 and c_3, respectively. These coefficients establish the weight or importance of each compo-

nent. In addition, components 2 to 4 are multiplied by independent random numbers r_1, r_2 and $r_3 \in$ U[0,1] respectively, to induce a stochastic behaviour in the individuals. Once the velocity is updated using this equation, the new position of the particle is calculated using Equation 2.5.2. This equation simply sums up the previous position x^{id} of the particle with the new velocity v^{id}.

$$v^{id} = w \cdot \overbrace{v^{id}} + c_1 \cdot r_1 \cdot \overbrace{(g^{id} - x^{id})} + c_2 \cdot r_2 \cdot \overbrace{(b^{id} - x^{id})} + c_3 \cdot r_3 \cdot \overbrace{(g - x^{id})} \quad (2.5.1)$$

$$x^{id} = x^{id} + v^{id} \quad (2.5.2)$$

Although the behaviour of each particle might look quite simple, this is indeed the main purpose of the algorithm. The particles are not intended to perform complex operations, but it is the interaction within the individuals in the swarm what makes the intelligence emerge. This simplicity has made PSO one of the most successful optimisation algorithms. An extensive survey of PSO application can be found in [196].

2.5.1 Discrete Particle Swarm Optimisation

The original PSO throws particles that fly in continuous spaces. For this reason, we cannot apply the algorithm as it is to discrete-based space problems. In a discrete space the *common* concept of velocity loses its sense. This makes it necessary to redefine how particles move.

In 1997, Kennedy and Eberhart proposed a discrete variant of the original Particle Swarm Optimisation called DPSO [151]. This new algorithm, also known as Quantum-PSO, used a binary codification to encode the position of a particle. The velocity was reinterpreted by means of the binary digits flipped from one state to another. This way, zero bits flipped means no move and all of them flipped alludes the opposite situation, maximum velocity.

They related this concept of velocity to the position in terms of probabilities. Thereby, if $v_{id} = 0.2$, there is a 20% chance that x_i takes value 1, and 80% chance for x_1 to be 0. Since

the velocity is a bitset, its real value could be out of the bounds of a probability measure $[0, 1]$. To overcome this situation, the authors proposed to use a sigmoid function that maps $\Re \rightarrow [0, 1]$. With all this, given a random number r in $U[0, 1]$, if $r < sigmoid(v_{id})$ then $x_{id} = 1$ or 0 otherwise. This first approach to the DPSO is limited to binary-valued solutions elements. Furthermore, the introduction of a probability based velocity combined with the sigmoid function makes it difficult to see how particles move from one to another solution. The intuitiveness of the PSO in continuous spaces was sacrificed in order to keep unchanged the velocity equation 2.5.1.

Since this first DPSO, a number of alternatives have been proposed to adapt the PSO to discrete spaces. Most approaches differ in:

1. the way they encode the particles' positions
2. the way the swarm moves
3. the application problem

This review provides an overview on those proposals applied to combinatorial problems related to the mainstream of this dissertation: Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). This review identifies and contextualises publications based on the three above-mentioned criteria. Tables 2.1 and 2.2 summarise the publications cited in this review for TSP and VRP problems, respectively. The works in this table are in chronological order. The first column provides the reference to the work. The second column specifies the type of encoding used to define a particle's position:

1. binary (if positions are defined as a bit set)
2. vector (if positions are defined as a sequence of integers that represent the turn in which customers are served)
3. fuzzy matrices (if positions are defined as matrices in which each element represents the probability of choosing an edge in a directed graph)
4. set of edges (if positions are denoted by a number of edges of a directed graph)

5. real-valued vector (if positions are described as a vector of real-valued numbers that require of a transformation to obtain the resulting sequence)

6. a set of coordinates (if positions are defined as locations using Cartesian coordinates)

The third column describes how particles move:

1. transformation (if some decoding process is needed to obtain the resulting solution)

2. customisation (if the utilised encoding requires of an accommodation of Equations 2.5.1 and / or 2.5.2)

3. operators (if Equations 2.5.1 and / or 2.5.2 are interpreted as a number of operations carried out on the chosen encoding)

Finally, the fourth column states to which problem the DPSO is applied.

Table 2.1: Classification of some existing DPSO applied to TSP publications according to: 1) *Encoding* of each particle, 2) *Movement* of the particles move, and 3) *Application* problem tackled. References are listed in chronological order.

Publication	Encoding	Movement	Application
Clerc [46]	vector	customisation	TSP
Onwubolu and Clerc [189]	vector	customisation	TSP
Secrest [227]	vector	operators	TSP
Pang et al. [193]	fuzzy matrices	transformation	TSP
Wang et al. [255]	vector	operators	TSP
Shi et al. [230]	vector	operators	TSP
Goldbarg et al. [115]	vector	operators	TSP
Zhong et al. [268]	set of edges	operators	TSP
Shi et al. [231]	vector	operators	Gen-TSP
Fan [88]	vector	operators	TSP
Hoffmann et al. [130]	vector	operators	TSP

TSP is the first combinatorial problem that was approached by DPSO. Clerc [46] proposed a DPSO algorithm in which each particle's position is defined as an array of customers identifiers (customers ids). In that work, equations 2.5.1 and 2.5.2 are adapted

Table 2.2: Classification of some existing DPSO applied to VRP publications according to: 1) *Encoding* of each particle, 2) *Movement* of the particles move, and 3) *Application* problem tackled. References are listed in chronological order.

Publication	Encoding	Movement	Application
Chen et al. [39]	binary	transformation	VRP
Ai and Kachitvichyanukul [1, 2]	real-valued vector	transformation	VRP
Marinakis et al. [175]	vector	customisation	VRP
Zhao et al. [266]	vector	operators	VRPTW
Qing et al. [205]	vector	operators	VRPTW
MirHassani and Abolghasemi [178]	sets of coordinates	transformation	Open VRP

by redefining how each operator works for the proposed encoding. This approach was tested on a 17 nodes TSP instance with different parameters. This DPSO was also applied to a simplification of a real test case involving automated drilling operations [189]. Another application to a real-world problem was carried out by Secrest [227]. This work presents a hybrid DPSO + Ant System approach to solve the TSP in surveillance missions. In that application (as in the previous work), each particle holds a solution (tour). The move of the swarm is carried out using a number of permutation operators (e.g. swaps, 3-opts) and tour buildings operators (e.g. insertions) based on the original PSO equations. Another proposal to the TSP was presented by Pang et al. [193]. That work used binary fuzzy matrices to represent both position and velocity. The velocity equation 2.5.1 was accommodated to work with matrices. Results indicated that the proposed algorithm is a valuable alternative to the DPSO. Wang et al. [255] designed a DPSO introducing the concepts: Swap Operator (SO) and Swap Sequence (SS). These Swap Sequences (SS) are chains of SOs to turn a solution (tour) into another. That work inspired Fan [88] to create an extension of this algorithm by incorporating a crossover operator, a reverse operator and a noise factor to propel diversity. That new algorithm was successfully tested on instances ranging $14 - 51$ nodes. Shi et al. [230] proposed a DPSO aimed at tackling larger TSP instances. This DPSO also used the concept of Swap Operator to redefine the velocity equation. In [231] the authors added an uncertain strategy and extended it to tackle the generalised TSP. Other approaches dealing with large instances are Golbarg et al. [115] ($51 - 7397$ nodes) and Zhong et al. [268] ($14 - 200$ nodes). The work by Goldbarg et al. [115] uses a permu-

tation based encoding and changes velocity equation operators by means of two local searches and a path-relinking operator. Zhong et al. [268] propose a DPSO using an encoding based on sets of edges. Velocity equation operators are redefined to suit this new encoding and a new coefficient (mutation factor) is incorporated to avoid local optima. In a recent work, Hoffmaan et al. [130] provide the first theoretical study of DPSO for TSP. The authors analyse the work by Clerc [46] and Wang [255] and proposed a new DPSO in which the difference between tours is based on edge exchanges. That DPSO produces good results on several TSP instances ranging $52 - 105$ nodes.

Discrete Particle Swarm Optimisation has been also applied to the Vehicle Routing (VRP) family of problems many times. Chen et al. [39] proposed a hybrid approach DPSO + Simulated Annealing (SA) to tackle the VRP. A quantum theory based DPSO (previously presented in [197] and tested in benchmark test functions) was used to assign customers to routes, and a SA was incorporated to avoid local optima. Velocity was encoded as a vector in which each component has a value between $[0, 1]$. Each of these components denotes the probability of that element taking value 0. This way, a binary encoding was used to describe the position of each particle. This approach provided the best performance compared to a GA with 2-opt and SA. Ai and Kachitvichyanukul [2] presented another DPSO for the VRP. This proposal used two real-valued encodings (SR-1 and SR-2). SR-1 has a length of $n + 2m$ dimensions (n and m are the number of customers and vehicles, respectively). SR-1 had already been proposed in a previous work by the same authors in [1]. In that work, SR-1 is extended using a 2-opt local search to improve the quality of solutions. SR-2 has a length of $3m$ dimensions (m is the number of vehicles). That DPSO is compared to the one by Chen et al. [39]. Results indicate that both encodings within that DPSO are better in terms of solution quality and computational time. Zhao et al. [266] developed a DPSO for the VRP with Time Windows (VRPTW) using a number of heuristics for the initialisation of solutions and solution generations. Particle positions were encoded using vectors and the movement of the swarm was done by means of operators. Experiments compared two DPSOs (global and local) with three construct algorithms. Results indicated that the best combination is a global DPSO with Solomon insertion heuristic which: 1) as-

signs customers to the best feasible position (distance/time); and 2) selects customers according to a maximum saving concept. A similar work was proposed by Qin Zu et al. [205]. That work applied a very basic implementation of the DPSO to some short made-up instances of the VRPTW. Marinakis et al. [175] proposed another alternative to tackle the VRP. Their work consisted of the hybridisation of a DPSO with other three algorithms. That hybrid approach uses a *multiple neighbour search GRASP* to initialise the population. Both positions and velocities are represented as vectors with the same length. Positions are encoded using a vector of real number in $[0, 1]$. The authors use equation 2.5.2 to update positions and a *Path Relinking* strategy to calculate the position of each particle. Finally, that hybrid DPSO uses a *neighbourhood search strategy* to improve solutions. Results in Christofides dataset [75] produced very competitive results. MirHassani and Abolghasemi [178] has recently proposed two DPSOs for the Open VRP with different neighbourhood sizes: nbest (if there are a number of small neighbourhoods) and gbest (if the neighbourhood is the entire swarm). Positions are encoded as a set of coordinates and Equation 2.5.2 is accommodated to this encoding for each neighbourhood size. In order to improve the solutions, a one-point method is used during the decoding process. Results revealed a fair performance in 15 instances ranging $19 - 72$ nodes.

Some contributions that describe applications of DPSO to other problem domains are: Data Mining [60], n-Queen Problem [135], Steiner Tree Problem [54], Scheduling Problems [6].

Tables 2.1 and 2.2 show that most application of DPSO to TSP and VRP(TW) (VRP and VRPTW) problems have used vector-based encodings and operator-based movements. This is probably because vector-based encodings ease the re-utilisation of heuristic methods (see Section 2.2.2). The literature also shows that some effort have been put into solving large TSP instances (see for example [230, 115, 268]). However, we do not find many works addressing large scale VRP(TW) instances. It is our belief that this will be a new research avenue in DPSO applied to VRP(TW). For this purpose, we think that the availability of a VRP(TW) instance generator could motivate the further improvement in this area.

2.5.2 Multi-Objective Particle Swarm Optimisation

Since the Particle Swarm Optimisation was proposed by Kennedy and Eberhart in 1995 [150], this algorithm has gained many adepts due to its simplicity and efficiency. This is the reason many researchers are pursuing to extend this technique to use it in different scenarios. This part of the literature review focuses on the multi-objective paradigm of the Particle Swarm Optimisation.

In a Multi-Objective Particle Swam Optimisation (MOPSO), a number of difficulties arise inherent to the multi-objectives paradigm. While in a single-objective PSO it is remarkably easy to select the leader as the particle with the best fitness, in a multi-dimensional objective space this is no longer possible. To begin with, it is necessary to find a metric to classify solutions. However, this is not enough when our goal is to offer a set of solutions to the decision maker. To be successful in dealing with multi-objective problems, it is also important to control the diversity of the population. Without a control of the diversity within swarm, our algorithm is likely to stagnate in local optima very fast. This survey outlines a number of proposals explaining how scientists in this area have overcome these difficulties in order to adapt PSO into MOPSO.

In this section, we will extend the survey of Reyes-Sierra and Coello [213] by including later work, but using a different taxonomy. We will organise this survey around the three key aspects of MOPSO: 1) *selection of the leader*, 2) *archives* and 3) *diversity*. However, it is worth noticing that these three mechanisms have evolved in an intimate relationship. This way, sometimes it is difficult to discern for each contribution what is indeed related to a mechanism or another.

Selection of the leader

The first proposal in this area was by Moore and Chapman [181]. They emphasised the importance of using Pareto dominance in the MOPSO paradigm. However, many works between 2002 and 2004 focused on the most straightforward adaptation of PSO to multi-objective decisional spaces. In order to avoid defining special mechanisms to select leaders in multi-objective problems, these approaches resorted to other schemes

different to Pareto dominance. For example, **aggregating approaches** such as the ones proposed by Parsopoulos and Vrahatis [194] and Baumgartner et al. [10]. The first of those works uses three types of aggregating functions, a conventional model and two dynamic approaches. The second work divides the swarm in n sub-swarms according to a number of special weights. Each of these sub-swarms have a different leader that leads the group depending on the weights.

An example using **lexicographic** ordering was put forward by Hu and Eberhart [134]. Using a dynamic neighbourhood that changes at each iteration, the local leader (*lBest*) is selected by fixing a number of objectives. In case of using two objectives, the first is fixed, while the second is used to select the leader among the neighbouring particles.

In the context of **Pareto dominance**, the *selection of leaders* is an aspect that requires a careful consideration. Most contributions dedicated to MOPSO redefine the concept of leader given in the original version of the PSO. Since many particles within the swarm might be in different non-dominated positions, new mechanisms have to be developed in order to chose the most appropriate ones to guide the population. The most successful *quality* measures are related to *density metrics*. The density metrics provide information about crowdedness, useful to know which particles are in the least populated section of the objective space. Two well-known density metrics found in the literature are the *Nearest neighbour density estimator* and the *Kernel density estimator*. The first metric, proposed by Deb et al. [65], simply calculates the perimeter of the cuboid of a given particle with respect to its nearest neighbours. The second was introduced by Goldberg and Richarson in [117] and it consists of counting the number of particles around a population member using a given radius.

Other works have relied on **random** selection of the leader from a *repository* of non-dominated solutions. Toscano and Coello in [202] propose a multi-swarm MOPSO, so that each sub-swarm has a set of leaders. These leaders are *randomly* selected to guide the swarm. Alvarez-Benitez et al. [5] proposed three different techniques in order to select the leaders: *Rounds*, *Prob* and *Random and Rounds*. The Rounds technique one was intended to promote convergence, Prob to promote diversity, and Random and Rounds to be a compromise between the first two. Another interesting approach using *random*

selection was put forward by Li [164]. He proposed *maximinPSO* which uses a derived function from the maximin strategy to compute the Pareto front. Thus, the leaders are randomly selected from a special *repository*. Other previous works using pure random schemes to select leader can be found in the survey by Reyes and Coello [213].

Some authors have used the **Roulette-wheel selection** mechanism to *select the leader* in their algorithms. This selection technique is widely used in GA. It consists of building a number of sub-segments spanned in $[0, 1]$, so that the length of each sub-segment is proportional to its probability. Then, a random number r is generated (from an uniform distribution $U[0, 1]$) and the *attribute* whose sub-segment spans r is selected. This method tends to be supported by an aid-method that specifies the length of each sub-segment. Coello et al. in [49] and [51] presented works using this technique. It was based on dividing the fitness space in identical hypercubes and counting the number of particle in each. This process was carried out in order to establish the probability of each sub-segment. The same mechanism based on the fitness of the particles was used by Ho et al. in [129]. Salazar-Lechuga and Row [222] and Yang et al. [261] based the selection on the already mentioned *kernel density estimator*.

A number of methods have exclusively relied on **density measures** to *select the leader* of the swarms. An interesting contribution is the one introduced by Mostaghim and Teich [183] who suggested the *sigma method*. That method consists of dividing the objective space by using a number of concentric imaginary lines fired up from the origin of coordinates. Although that method is efficient and relatively easy to implement, it is limited to work only with objectives having positive values. Another work with a similar aim was put forward by Villalobos-Arias et al. [248]. Their approach was based on creating a number of *stripes* on the decision space using minimal points. Therefore, the leader for each stripe is selected minimising the weighted sum of the minimal points in the objective function. A very similar work was proposed by Toscano-Pulido et al. [242] using a so-called *Hyper-plane distribution*. Recently, Zhao and Suganthan [265] proposed a MOPSO that selects leaders: *lbest* (local best) and *gbest* (global best), from the top fronts in a non-dominated external archive. The authors also introduce a *two-lbests* based MOPSO. That algorithm assigns a *bin* to each solution according to

the objective function range in the external archive. Thus, in order to intensify local exploration, this MOPSO selects solutions located in neighbouring bins.

Another interesting approach using a completely different technique is the one by Fieldsend and Singh [93]. A special structure called **dominated tree** is built to store good quality (non-dominated) positions. Thus, the selection of the particles is based on the structure itself. A composite point is first selected based on dominance relations. Secondly, its closest particle in the objective space is then selected as the leader.

Another recent line of research is the interactive MOPSO. Interactive MOPSOs select *lbest* and *gbest* by means of **user-interaction**. Mostaghim and Teich [183] proposed the first guiding MOPSO in which a number of predefined solutions are provided to be *gbest* particles. Similarly, Wickramasinghe and Li [258] present a MOPSO that uses user-preference solutions (feasible or infeasible) as an input for *gbest*. As a further step towards usability, Hettenhausen et al. [128] proposed a heatmap-based visualisation tool to aid the decision maker in choosing the leaders at each iteration. Recently, Mostaghim et al. [184] studied an interactive MOPSO based on a **Desirability Function** (DF) to select *gbest* and *pbest*. Their MOPSO has a hybrid behaviour that alternates between a normal MOPSO and DF-MOPSO when a desirability index within the population is achieved.

Archives

An important issue that must be addressed when solving multi-objective problems is the storage of non-dominated solutions. The ideal case would be to store them all as the search process moves forward. However, this is rarely possible computationally speaking. In most scenarios, these *repositories* or *archives* of non-dominated points must be stored in data structures of fixed sizes. This increases the complexity adding an extra overhead when we have to decide what to do if the *archive* is full and a new non-dominated solution is found. In the last 10 years, a number of methods have been proposed to deal with these situations.

Most techniques to fix the size of the *archive* are intended to maintain a certain degree

of *diversity* within the swarm. A well known technique consists of **dividing the decision space** in hypercubes with identical dimensions. Then, the algorithm that controls the size of the *archive* counts the number of particles inside each hypercube. This density metric is then used to delete the particle in the archive whose hypercube is more crowded. Examples of this approach can be found in the works of Coello et al. [49, 51] and Bartz et al. [9].

Other authors relied on using well-known **density estimators** as the *kernel density* or *nearest neighbor*. Examples of the first algorithm can be found in Srinivasan and Hou [235] and Mostaghim and Teich [182]. The *nearest neighbour estimator* is used by Li [164] on the hybridisation of a PSO with the known NSGA-II [64]. Raquel and Naval [207] used a similar method called *crowding distance* in their approach to MOPSO. The *crowding distance* is equivalent to the *nearest neighbour estimator*, but calculating the volume rather than the perimeter. *Crowding distance* is used together with ϵ*-dominance* [160] in an improved version of the MOPSO presented by Sierra and Coello [212].

In a recent study, Helbig and Engelbrecht [126] propose three techniques to maintain an archive for dynamic multi-objective optimisation problems (DMOOP). The first technique consists of clearing or reseting the archives, which might be inconvenient in case of little changes in the Pareto optimal front. The second technique involves the re-evaluation of solutions contained in the archive and removal of the least suitable. And the third technique first re-evaluates all solutions in the archive. If a solution becomes dominated, a hill-climbing method is applied. If the solution is still dominated, it is removed from the archive. A number of benchmark functions are used to compare the performance of these three archiving models. Results indicate that the one involving the use of a hill-climbing method is superior.

Diversity

As it is stated in the survey by Reyes and Coello [213], diversity within the swarm has proven to be one of the key aspects to a good multi-objective particle swarm optimiser. Many factors related to this issue might affect the tendency of the particles to stagnate in a local minima. The *selection of the leader* and the way the algorithm manages

the *archive(s)* have a direct impact on the convergence. Also, Hu and Eberhart [134] highlighted the importance of selecting a larger population size for multi-objective environments. Other mechanisms rely on the **inertial component** [31] on the PSO (Eq. 2.5.1) or using different types of **communication topologies** [33].

Besides these mechanisms and more in the context of MOPSO, many authors have opted to add **turbulence** or **craziness** factors to encourage the diversity among the particles in the swarm. These factors are indeed operators that somehow modify the velocity and/or position of particles. So far, the mutation operator has been usually performed on the decision variables. Some examples are the works of Fieldsend and Singh [93], Coello et al. [49], Ho et al. [129], Mostaghim and Teich [182], Ishida et al. [144], Li [164] and Mohamed et al. [179]. In a more elaborated work on this topic by Reyes and Sierra [212], two mutation techniques, borrowed from Evolutionary Algorithms, are implemented: *Uniform mutation* and *Non-uniform mutation*. These operators are used in order to create three sub-swarms that share leaders. One of them does not implement any mutation and the other two implement the mutations proposed.

Another interesting contribution was introduced by Quintero et al. [223] who combined PSO and SS (Scatter Search) [176] in order to achieve a good compromise between exploration (SS) and exploitation (PSO). **Scatter Search** is mainly embedded to provide diversity, and PSO to speed up the convergence. Additionally, the authors implemented a *parameter-based mutation* used in NSGA-II [64]. Their experiments show it produces very competitive results compared to NSGA-II.

Some researchers have also used **elitism** in order to avoid using special operators to promote diversity. Srinivasan and Seow [235] presented (*PS-EA*), a hybridisation of a PSO algorithm and a Evolutionary Algorithm. This way, the issue of the diversity is covered by the own nature of the EA using elitism.

A work introduced by Zheng and Liu [267] presents VMAPSO (*A Hybrid Vertical Mutation and self-Adaptation based MOPSO*). That algorithm utilises a vertical mutation operator that modifies one or several components of the current position along the iterations. This way, the mutation affects a particle *vertically*. That is, it does not only operates on current position of a particle, but it also does on its previous positions.

Summarising, we think that further research should be carried out on the study of these key components (leader selection, archiving, diversity) separately. Most works use more than one mechanism to tackle multi-objective problems. This makes it hard to know which component is actually contributing the most in the effectiveness of the algorithm.

Moreover, most papers surveyed here are for MOPSOs applied to continuous problems. Therefore, another important issue to be addressed is to prove if these elaborated algorithms are equally successful on multi-objective combinatorial problems.

CHAPTER 3

CODEA - An Agent Based Multi-Objective Optimisation Framework

Summary

This chapter introduces CODEA, a COoperative Decentralised Architecture for Multi-Objective Optimisation (MOO). This is a C++ object-oriented framework for the creation of groups of agents that cooperate to tackle complex optimisation problems. Unlike most multi-objective frameworks, CODEA provides explicit mechanisms to enable agents to send and receive information either synchronously and asynchronously. Agents execute tasks in an autonomous fashion, so no entity controls agents' operations. Besides, agents undergo communication phases in which they share information. The interactions among agents can create complex cooperative behaviours, which contribute to tackling complex problems as teamwork.

3.1 Introduction

Heuristic algorithms have become more elaborate in recent years. This means that the time required for the design, debugging and testing of this kind of methods is consider-

able. Source code re-utilisation emerges as a mean to re-use already coded algorithms to solve different problems without starting a new development from scratch [68].

A number of frameworks and libraries have been proposed to facilitate the development of heuristic algorithms in different programming languages. In particular for multi-objective optimisation, some feature-rich multi-objective optimisation frameworks and libraries are discussed in Section 3.2.

In this chapter, we introduce CODEA (COoperative DEcentralised Architecture). This framework is designed to create flexible groups of agents to tackle multi-objective optimisation problems using the paradigm of cooperative problem solving. CODEA consists of a number of classes developed in C++ that accelerate the development of cooperative metaheuristics.

The reminder of this chapter is organised as follows. Section 3.2 provides an overview of existing feature-rich frameworks for multi-objective optimisation. Section 3.3 gives a detailed description of CODEA and its components. This section also introduces the lastest two major versions of this framework: *CODEA v2* (Section 3.3.1) and *CODEA v3* (Section 3.3.5). Section 3.4 presents four contributions in which CODEA has been used. Two of these contributions correspond to collaboration with other researchers, and the other two correspond to minor developments, but still within the scope of this PhD thesis. Finally, some conclusions and ongoing work are provided in Section 3.5.

3.2 Frameworks for Multi-objective Optimisation

This section provides a concise overview of some of the most relevant multi-objective optimisation frameworks. This review focuses on mature and feature-rich frameworks. In [165] a number of Evolutionary Multi-objective Optimisation (EMO) frameworks are classified according to: type of multi-objective optimisation problems they are able to tackle (combinatorial and/or continuous), availability of statistical tools (online and/or offline), availability of hybridisation, availability of parallelisation mechanisms, type of framework (whitebox or backbox), programming language and license (free or commercial). We modify this classification in order to compare mature and feature-rich

state-of-the-art multi-objective optimisation frameworks. Since most recent toolboxes are able to deal with both continuous and combinatorial problems, and most of them are open source, we do not compare these criteria. Instead, we add the following criteria for the comparison: the *starting date* of the project, availability of a *Graphical User Interface* (GUI), availability of *cooperation* and *license agreement*.

This review focuses on multi-objective optimisation frameworks which appear to be still supported. We do not include frameworks which may be related to our research but are either unavailable for downloading (e.g. pALS [15]), or no longer supported (e.g. GAlib [254], Open-BEAGLE [103]). We also exclude toolboxes with limited features for multi-objective optimisation such as Shark [141] or Opt4J [172]. For a general overview of optimisation software class libraries, one can refer to [249].

Table 3.1 summarises the comparison of four optimisation frameworks according to the following criteria. From left to right, the first column gives the name of the framework. The second column states the publishing starting date of the project (a good indicator of the maturity of the framework). The next column states the availability of statistical tools. This criterion can be online (if statistic functions are available to analyse information throughout the optimisation process), offline (if statistics can be run after the optimisation process) or *none* (if no mechanisms are provided for this purpose). The fourth column states to which extent a GUI is provided. GUIs facilitate the use of frameworks by all types of users. We distinguish three possible values for this criteria: basic (if a GUI is provided to set parameters and/or to present some basic graphic information), yes (if the framework comes with a full-featured GUI) and *none* (if no GUI is provided). The next column states if the framework supports cooperation. This criteria can be: *explicit* (if the framework ships mechanisms to perform cooperation), *implicit* (if cooperation can be achieved using non-specialised existing components) or *none* (if no explicit or implicit cooperation mechanisms are available). Columns six and seven state wether or not the framework supports Parallel schemes and their programming language, respectively. Finally, we consider the end user license agreement for each framework. While license agreements are typically not a concern for scientists, they might be important from a business strategy point of view. Due to some terms

in licence agreements, many companies may or may not contemplate the use of these frameworks. In this review, we find three types of licences. GNU General Public License (GPL) [240] is the most widely used free software licence. This licence allows the free copy, modification and distribution of source code. GPL licence forces the distribution of any derived work under the same license terms. GNU Lesser General Public License (LGPL) [102] is a less restrictive licence in which only the code that uses LGPL libraries is forced to be open source. This way, LGPL permits to use the library in proprietary software. Academic Free License (AFL) [190] is similar to GPL, but it requires the acknowledgement of the use in any derived work in publications.

Table 3.1: Classification of some existing frameworks based on [165].

Framework	Started	Statistics	GUI	Cooperation	Parallel	Language	License
jMetal	2006	offline	basic	none	none	Java	LGPL
HeuristicLab	2002	on/offline	yes	none	yes	C#	GPL
ECJ	2001	none	basic	none	yes	Java	AFL
ParadisEO	1999	on/offline	none	implicit	yes	C++	GPL
CODEA v3	2007	on/offline	none	explicit	yes	C++	GPL

In the following subsections, we provide a high-level overview of the four multi-objective optimisation frameworks shown in Table 3.1 focusing on the above-mentioned criteria.

3.2.1 JMetal

JMetal [80] (Metaheuristics Algorithms in Java) is a LGPL Java-based framework for Multi-objective optimisation. JMetal ships the most relevant multi-objective optimisation algorithms and quality indicators. A large set of statistics are available for offline analysis, including the automatic generation of LaTeX tables and pair-wise comparison in Wilcoxon tests. However, there do not seem to be online statistical analysis tools. In terms of parallelisation, JMetal only supports multi-threading when performing several experiments and analysis simultaneously in multi-core CPUs. JMetal comes with a GUI that enables users to set parameters and perform several statistical operations. This toolbox has a 70+ pages manual that provides a detailed overview of the most im-

portant features. Regarding cooperation, JMetal does not provide implicit or explicit mechanisms to create systems of cooperative agents.

Compared to the other three frameworks under review, JMetal ships the largest number of multi-objective algorithms and quality indicators. Like ECJ, JMetal provides out-of-the-box portability across operating systems thanks to its implementation in Java. However, the use of Java makes both ECJ and JMetal slower than ParadisEO due to the overhead produced by the Java Virtual Machine (JVM). JMetal is a good option for those looking for portability, ready to use algorithms and quality indicators, high quality offline statistical analysis. Another point in its favour is the license agreement. JMetal's LGPL is the most appealing licence in this review from a commercial point of view. JMetal's major disadvantage is the lack of support for parallelism. Its documentation could be improved with more tutorials.

3.2.2 ECJ

ECJ [82] is an AFL (Academic Free License) Java-based Evolutionary Computation (EC) research system. This framework is a robust and flexible library for the design of Evolutionary Computation methods. Currently, it implements very few multi-objective optimisation algorithms, but a number of hooks are provided for supporting the development of other multi-objective optimisation methods. ECJ does not come with pre-implemented statistics, but it also provides hooks for users to implement their own. The authors indicate that this toolkit is designed for big projects and its learning curve can be steep [173]. This project was started more than 10 years ago, but it is still quite active. Users have contributed a number of packages including a test-case using Graphical Process Units (GPUs) support. ECJ has an exhaustive manual (200+ pages) that covers every supported feature by the system. Unlike JMetal, ECJ supports multithreaded evaluation and breading. It also supports parallel synchronous and asynchronous Island Model on a grid of computers. Regarding cooperation, no mechanisms seem to be available for this purpose. This toolkit comes with a very basic GUI. This GUI can be used to set parameters, but the authors discourage its use and recommend to operate with the command line.

ECJ is a rich featured Java multi-objective framework alternative for the development of big projects. It is highly flexible and has a large number of features to work with EC methods. The two main disadvantages of ECJ are low performance and complexity. Firstly, it suffers from the same problem as JMetal due to its use of the Java Virtual Machine (JVM). Plus, in ECJ almost all classes and parameters are determined at runtime which creates an extra overhead. Secondly, in the author's opinion, it is not easy to get started with ECJ. More tutorials would be useful. ECJ's strong points are portability and community support.

3.2.3 HeuristicLab

HeuristicLab [252, 251] is a GPL C# graphic framework for the design and prototyping of metaheuristics. HeuristicLab is easy to use and yet extensible thanks to its plug-in based design. More than a toolkit, HeuristicLab is a software in itself. It comes with a large number of algorithms and benchmarking problems. Moreover, a very high featured user interface provides the means to modify pre-implemented problems and algorithms and create new ones. HeuristicLab ships problems such as Vehicle Routing Problem, Knapsack and Classification. New algorithms can be created and existing ones can be extended using the GUI. Functionalities work as plugins: users can add and modify plugins to get new functionalities. HeuristicLab supports parallelisation through one of its components called Hive. This component enables the system to perform experiments in parallel on a computer cluster. This software does not seem to provide any mechanism to create cooperative systems of agents. The documentation of this toolbox is not extensive and no instructions are provided on how to create plugins. However, thanks to its simplicity and intuitive user interface, non-expert users can use this software.

HeuristicLab is an easy to use environment and, unlike the other optimisation frameworks in this review, it does not require programming knowledge to be operated. This software is ideal to teach and prototype optimisation algorithms. HeuristicsLab's major flaws are performance, portability and documentation. The use of C# as programming language is one factor that makes HeuristicLab the slowest optimisation platform in

this review (for a comparison of performance of programming languages one can refer to [162]). Besides, HeuristicLab runs only on Microsoft Windows, as C# is fully supported by this operating system only. Finally, very little documentation is provided. More effort should be put into explaining how to develop additional plugins.

3.2.4 ParadisEO

ParadisEO [25] is a GPL-compatible C++ object-oriented framework for the design of flexible optimisation algorithms. ParadisEO's architecture consists of four interconnected modules: EO, MOEO, MO and PEO (Figure 3.1).

- EO (Evolving Objects) [87, 149] is a C++ LGPL template based evolutionary computation library. EO is a component-based framework that contains a large number of components such as selection, replacement, stopping criteria or command-line interface. These mechanisms are designed to work with each other in order to build up systems to solve problems.
- MO (Metaheuristics Objects) is a module that contains components to implement single-objective metaheuristics such as Local Search or Tabu Search.
- MOEO (Multi-Objective Evolving Objects) is a component that provides a flexible platform for the design of Evolutionary Multi-objective Optimisation (EMO) metaheuristics. The architecture of this module is based on the design of EO.
- PEO (Parallel Evolving Objects) provides a set of classes for the creation of parallel and distributed systems.

ParadisEO is a flexible, fast and robust platform for the development of all types of metaheuristics to tackle both continuous and combinatorial problems. ParadisEO includes a number of classes to perform online and offline statistical analysis. Besides this, users can create new components for statistical analysis and monitors by extending some classes. ParadisEO-PEO is the component responsible for the parallelisation of algorithms. It contains classes for the most common parallel and distributed models

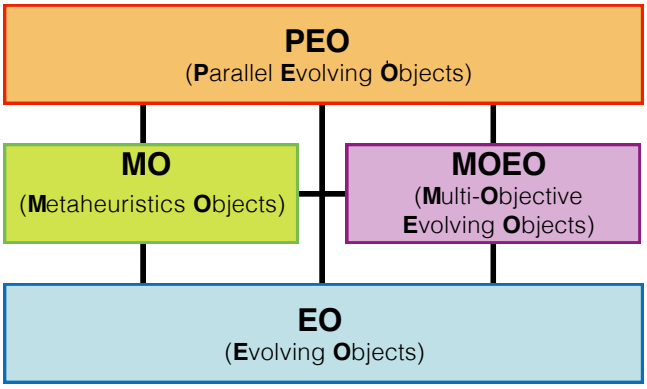


Figure 3.1: ParadisEO's architecture consists of four interconnected component.

and hybridisation mechanisms such as the cooperative island model. Moreover, the latest version of this optimisation platform (ParadisEO 1.3) provides Graphical Processor Unit (GPU) acceleration support. In terms of speed, thanks to its C++ template-based implementation, ParadisEO is the fastest framework in this review. ParadisEO is also portable on Microsoft Windows, Unix and MacOS. Regarding cooperation, ParadisEO does not provide explicit mechanisms for this. Instead, the user needs to extend some base classes in order to provide mechanisms for communication between components. ParadisEO does not come with any GUI by default. Some attempts have been made to provide ParadisEO with a basic GUI. An example is GUIMOO [125], a graphic software for the analysis of results in multi-objective optimisation.

ParadisEO's project website provides a basic API documentation and a large number of tutorials. However, the learning curve can be very steep. A fine-grained structure of components comes at a price of a high level of complexity. The poor API documentation together with the high level of complexity is the major weakness of this optimisation framework. The use of ParadisEO is appropriate when looking for the best balance between flexibility and efficiency.

3.3 CODEA

The development of CODEA (COperative DEcentralised Architecture) started at the University of La Laguna (Spain) by the Group of Intelligent Computing (GCI) in 2007. The core idea of this project was to create a simple, flexible and fast framework to create groups of cooperating agents for tackling complex optimisation problems. To this aim, we designed CODEA using a number of modules with abstract functionalities. We put special emphasis on the design and development of modules for communication tasks, so that it does not require an in-depth knowledge of how the core of this MOO platform works. In this sense, CODEA follows a similar approach as the one used in ParadisEO's architectures (Section 3.2.4). In the first version of CODEA, we used this framework to create a number of synchronous agents, each with the same local search. Agents performed two phases: solving and crossover. In the solving phase, each agent ran a local search and the new resulting solution was sent to other agents. In the crossover phase, each agent combined its own solution with the best solution received from the other agents. This approach was tested using several communication topologies in a number of Travelling Salesman Problem (TSP) instances and the results were presented in [30].

The second version of CODEA (CODEA v2) was started at the University of Nottingham (UK) in 2008. CODEA was rewritten almost from scratch to support the optimisation of multiple objectives, but many structural components were kept. This new version of CODEA included new features like the support for multi-objective optimisation problems, various ranking schemes to discriminate solutions in multi-objective optimisation (aggregation approach, lexicographic ordering and Pareto dominance), parallelisation supported by OpenMP (Open Multi-Processing), etc. CODEA v2 was used to implement a Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) algorithm to tackle the Vehicle Routing Problem with Time Windows (VRPTW). In this implementation, agents had to deal with eight minimisation objectives, from which four were soft constraints. The aim of this research was to investigate whether the implemented MODPSO was capable of evolving swarms from infeasible to feasible regions. The results of this research were presented in [31]. A second investigation involved the

creation of a new ranking scheme called the Dynamic Lexicographic Approach (DLA). This ranking approach consists of comparing solutions according to random lexicographic orderings generated with a certain probability distribution. The performance of the DLA was compared to the common Lexicographic approach and Pareto Dominance and the results were published in [32] and [37]. The next chapter discusses this investigation in depth. Finally, we illustrated the use of CODEA v2 in [33] by applying the above-mentioned MODPSO algorithm to TSP instances with two objectives. In this work, a number of experiments were conducted to test different communication topologies.

In order to pursue our research in multi-objective optimisation, we had to implement a number of new features in CODEA, such as MOO algorithms, archives and quality indicators. However, most of these features have already been implemented in many other MOO frameworks such as those discussed in Section 3.2. Instead of developing all these new features in CODEA v2, we decided to implement our framework as an extension of ParadisEO (Section 3.2.4). ParadisEO was the best alternative as it provides the best balance between flexibility and efficiency. In 2011, CODEA v3 was released as the result of the hybridisation between CODEA v2 and ParadisEO-MOEO, the component to deal with multi-objective problems (Figure 3.1).

Since the contribution of this PhD thesis covers CODEA v2 and v3, the next two subsections will explain the characteristics of both in detail. In order to keep this explanation as simple as possible, these characteristics are explained at a high level. For more technical documentation of the API with UML diagrams, please refer to [35].

3.3.1 CODEA v2

Figure 3.2 shows the basic scheme of CODEA v2. We note the element *system* at the top of the structure, which contains a number of agents and some properties. This class stores the data structure that hosts the agents and orchestrates their operations. It is worth noting that even if *system* holds the group of agents, it does not know the operations that agents perform (*phases*), nor their relationships (*neighbourhood*). The element *system* is merely intended to store general properties like *elapsed time* or up-

time, the number of the current *iteration*, the *best solution* found by the agents, and the *stop criterion* used to stop the agents.

The next element in the figure represents the *agent* as an individual who is able to communicate with other agents and operate on its own. This cell shows the sub-elements: *neighbourhood*, *core* and *phases*. These components give agents special abilities to share information (*neighbourhood*), to hold the information needed to solve problems (*core*), and to carry out their operations (*phase*). An agent also has a number of parameters to control its state such as *isActive* or *id*.

3.3.2 Neighbourhood

The *neighbourhood* component manages the list of population members with whom agents communicate. The component *neighbourhood* is implemented as an interface so users can develop their own rules to establish new neighbourhoods. By default, CODEA v2 allows the use of three types of neighbourhoods: star topology (all to all communication), ring topology (each individual receives and sends from/to two other individuals) and k-random topology (each individual shares its information with *k* random individuals). Although these topologies are static, CODEA v2 does not limit the creation of dynamic systems of cooperation. It is fairly easy to implement dynamic cooperation schemes based on scores or rules. In CODEA v2, the cooperation is based on the interchange of messages. For this purpose, this framework uses special containers which can store not only solutions, but any type of information. A message consists of a number of adjacent cells. Each cell has two compartments: an id describing the type of information and the actual information. This mechanism is so flexible that agents could even send themselves within a message. Each agent has a container to store messages called *mailbox*.

3.3.3 Core

The *core* element is responsible for manipulating the information of the problem, its solution and the solution process. In order to re-utilise code as much as possible, the

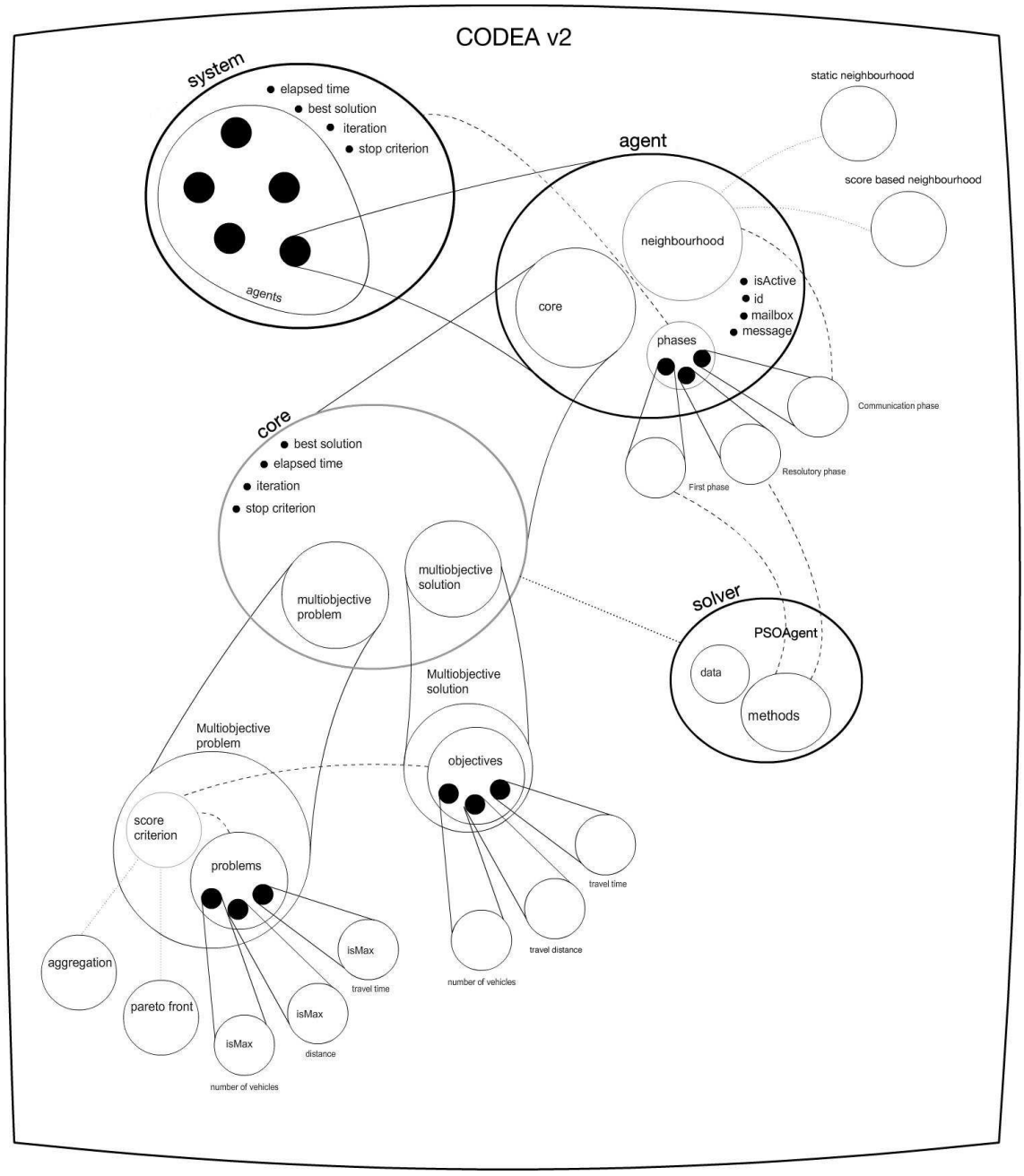


Figure 3.2: CODEA v2 - Diagram of the architecture implementing a Multi-objective Discrete Particle Swarm Optimisation (MODPSO) (see Chapter 8) applied to the VRPTW with three objectives: *number of vehicles*, *travel distance* and *travel time*.

core has three main components: *multiobjective problem*, *multiobjective solution* and *solver* (which in Figure 3.2 is represented by *PSOAgent*). By using this methodology, changing the solution or the problem does not affect the rest of the structure. For example, we might want to change the manner we evaluate solutions, but without modifying the solutions themselves. Or just the opposite, keep the evaluators unchanged, while the solutions have a different encoding. The same is designed to happen with the solver. The way the solver is implemented should not depend on the solution encoding nor on the data structure of the problem. For this purpose, both *Multiobjective problem* and *Multiobjective solution* have an independent design.

Multiobjective problem

The *Multiobjective problem* component has two parts: *score criterion* and *problems* (function evaluators). These function evaluators are related to the problem. In Figure 3.2 there are three evaluators: *number of vehicles*, *travel distance* and *travel time*. The *score criterion* is another interface that enables the user to create criteria to rank solutions in multi-objective scenarios. Some ranking criteria implemented include Pareto dominance, aggregation and lexicographic ordering. The *Multiobjective problem* also holds the data of the problem and the set of evaluation functions.

For each objective that the agents are optimising, there is an associated function to calculate its value. This function is represented in the structure as a simple file that specifies how to assess a certain objective. In the example shown in Figure 3.2, there are three objectives: *number of vehicles*, *travel distance* and *travel time*. Each objective is coded in a different file, so that adding a new objective is as easy as creating a new file describing how to compute that objective. Moreover, this provides the means for enabling and disabling objectives on line. Regarding the score criterion, one may consider that, since the system compares two solutions, this sub-component should be contained within *Multiobjective solution* rather than within *Multiobjective problem*. However, it must be noted that many ranking schemes require to know whether we are maximising or minimising objectives. This characteristic is inherent to the evaluation of objective function and therefore, must be placed within *Multiobjective problem*.

Multiobjective solution

The *Multiobjective solution* component has two basic units: a vector of *objective* values and the subjacent solution. The latter unit simply stores the solution using the encoding provided for the problem and the user preference. The evaluation of this solution in the set of objectives described in *Multiobjective problem* is always saved to the vector of objectives. Thus, the system avoids the re-evaluation of all the objective functions every time we want to compare two solutions.

Solver

The *solver* component handles both *Multiobjective solution* and *Multiobjective problem*. This part of the *core* contains atomic operations creating an abstraction of the behaviour of the agent. For example, in order to develop the Multi-objective Discrete Particle Swarm Optimisation (MODPSO), we designed a class that coordinates the operations without direct relation to the problem or the solution being used. Beneath in the structure, it is where we make the connection to our problem and solution design. Therefore, the core of the solver does not depend on the problem tackled by the system.

3.3.4 Phases

The atomic operations contained in the *core* are orchestrated by the *Resolutory phase* component within the *phases* class in *agent*. This part of the agent acts as an interface for users to provide a standard to implement their own phases. The flow of CODEA starts with the system invoking the agent's phases. The agents then take the control coordinating their operations using the phases. There is not a limited number of phases and they do not have to be synchronous (all agents doing the same task simultaneously). In addition, agents are able to delete, add or modify phases in real time. The use of this feature allows the creation of complex and very dynamic systems. For example, we can create systems of evolving agents who change their phases in real-time.

3.3.5 CODEA v3

CODEA v3 is the framework resulting from the implementation of CODEA v2 as an extension of ParadisEO-MOEO. Thanks to this hybridisation, systems of agents within CODEA have access to all features available in ParadisEO at all levels. This way, CODEA v3 inherits all features of ParadisEO, plus explicit mechanisms inherited from CODEA v2 to work with cooperative schemes as stated in Table 3.1.

Figure 3.3 depicts the architecture of CODEA v3. At the highest level, CODEA preserves its structure so that the *system* remains unchanged with respect to the previous version. The structure of the element *agent* is almost the same. The element *core* is no longer needed as this new version of CODEA uses specialised objects inherited from ParadisEO-MOEO. Components *neighbourhood* and *phases* keep the same structure and behaviour as before. CODEA v3 adds a new layer of abstraction in the element *phases*. This abstraction allows an agent to be both one of the ParadisEO's pre-implemented algorithms and a single individual of a system of agents. In practice, this new feature enables CODEA v3 to work with groups of cooperating heterogeneous multi-objective algorithms with just a few lines of code.

In order to illustrate how this new version of CODEA works, Figure 3.3 shows the implementation of a Multi-objective Discrete Particle Swarm Optimisation (MODPSO) applied to the Vehicle Routing Problem with Time Windows (VRPTW) with five objectives (see Chapter 8).

The MOPDSO depicted in Figure 3.3 contains two phases:

1. The *defaultCommunicationPhase* is responsible for sending the particle's position to all the other particles within the swarm.

2. The *moeoJFOPhaseAlgorithm* holds specific data and operations related to this MODPSO for each particle.

In bold letters we note the components inherited from ParadisEO such as *fitnessAssigment*, *diversityAssigment* and *archive*. The component that stores the actual solution of the VRPTW is *moeoVRP*. This is a problem-dependent object that holds the *routePlan* as

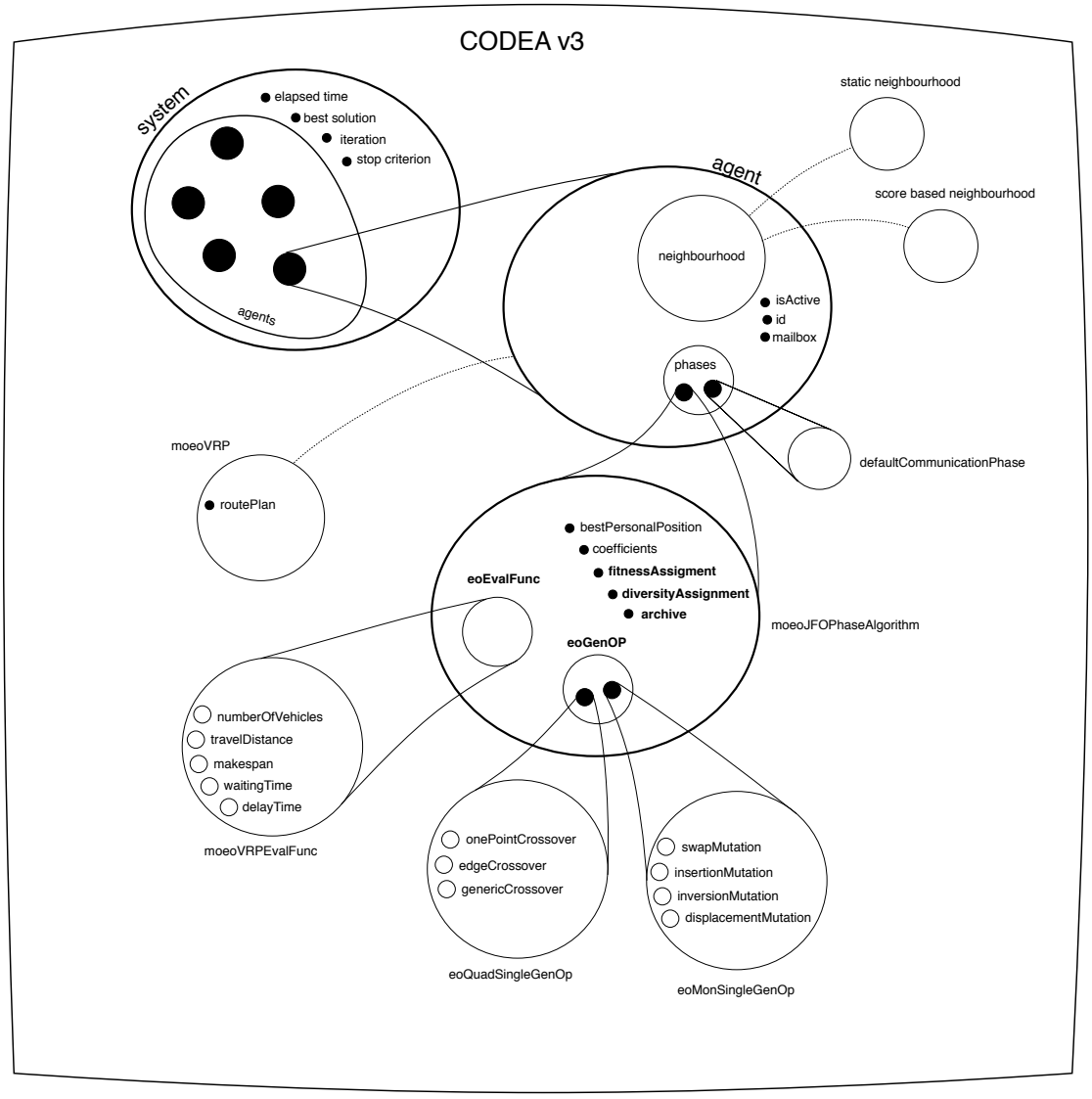


Figure 3.3: CODEA v3 - Diagram of the architecture implementing a MODPSO applied to the VRPTW with five objectives: *number of vehicles*, *travel distance*, *makespan* (or travel time of the longest route), *waiting time* and *delay time*.

the abstraction of the position of a particle. The element *moeoVRP* inherits properties from objects in ParadisEO (objective vector and internal representation) and from the component *agent* (ability to cooperate and perform phases). Using this fine-grained approach, it is fairly easy to add and modify components to extend the system. In order to have access to an abstraction of the operators at the phases level, we had to create the interfaces *eoMonSingleGenOp* and *eoQuadSingleGenOp*. Both interfaces act as containers inherited from ParadisEO's *eoGenOp*. The interfaces *eoMonSingleGenOp* and *eoQuadSingleGenOp* hold the mutation and crossover operators, respectively. In this example, there are four mutation operators: *swapMutation*, *insertionOperator*, *inversionMutation* and *displacementMutation*, and three crossover operators: *onePointCrossover*, *edgeCrossover* and *genericCrossover*. Finally, the element *eoEvalFunc* in *moeoJFOPhaseAlgorithm* contains the interface for the evaluation functions. In the above-mentioned figure there are five evaluators: *numberOfVehicles*, *travelDistance*, *makespan*, *waitingTime* and *delayTime*.

As a result of the hybridisation of CODEA $v2$ and ParadisEO, CODEA $v3$ combines a number of new features. In this way, ParadisEO-MOEO has contributed to CODEA with several sets of new components:

- *Solution representations and operators*. Multi-objective Optimisation Problems (MOP) require a solution representation in the decision space and in the objective space. The solution representation in the objective space is problem-independent. However, the representation in the decision space is related to the tackled problem. ParadisEO provides a number of standard vector-based solution representations, including those composed of bits, of integers and of real-coded values. Furthermore, ParadisEO provides the most common variation operators for these representations.

- *Ready-to-use metaheuristics*. ParadisEO comes with a number of ready-to-use solving methods, such as Tabu Search [114], Simulated Annealing [228], Particle Swarm Optimisation [81] and NSGA-II [64].

- *Fitness Assignment* methods guide the search in multi-objective optimisation prob-

lems towards Pareto optimal solutions. ParadisEO provides a number of fitness assignment schemes, such as achievement scalarising functions, dominance count [272] and dominance depth [116].

- *Diversity Assignment* aims at providing well spread non-dominated solutions over the objective space. ParadisEO provides a number of standard diversity assignment schemes including sharing [117], crowding [64] and a nearest neighbour scheme [274].

- *Statistics*. ParadisEO provides a number of tools to run statistics as the search progresses. It also allows the generation of plots of variables, statistics and/or results in real-time.

- *Parallelisation mechanisms* can speed up the search process significantly. ParadisEO provides a layer of abstraction to use MPI (Message Passing Interface) and OpenMP (Open Multi-Processing). This layer of abstraction eases the process of parallelisation of components within any solving algorithm.

CODEA has also contributed to ParadisEO in a number of ways:

- *Cooperation*. CODEA provides several mechanisms to create groups of cooperative agents. Two essential mechanisms are: *messages* and *topologies*. Agents share information by using messages. Messages are containers of virtually any type of information. Thus, agents can share not only their current solution, but also variables, structures and even themselves (an agent can also be sent within a message). Agents exchange messages according to a certain communication topology. CODEA provides a number of static communication topologies which can be assigned to any agent. Agents do not need to have the same topology, and this topology can be dynamic.

- *Autonomous agents*. In CODEA, agents are *autonomous* and can operate *asynchronously*. No entity controls the way agents work. In addition, thanks to the use of the component *mailbox* (See Figure 3.3), agents do not have to process messages as they receive them.

- *Organic groups*. Agents within CODEA can also perform three special actions in real time: 1) *clonation* which allows agents to duplicate themselves, 2) *self-deletation* which allows agents to remove themselves from the system, and 3) *mutation* which allows agents to change their behaviour (subjacent algorithm).

CODEA $v3$ combines all the features provided by ParadisEO and CODEA $v2$. In fact, CODEA $v3$ takes a step ahead providing mechanisms to create special types of agents. In CODEA $v3$, agents can be ParadisEO algorithms, sub-components of ParadisEO algorithms or both at the same time. For example, we can create a group of agents in which each agent is a particle, each agent is a swarm, or the group is a combination of both. We can also create groups of hybrid agents with other solving methods such as Variable Neighbourhood Search (VNS), all running at the same time. CODEA $v3$ also allows to embed ParadisEO statistical tools in some agents to control diversity of convergence among the whole group.

CODEA v2 and v3 are free GPL-licensed software and are available in [35].

3.4 Other uses of CODEA

This section presents four contributions which have directly or indirectly benefited from CODEA. These contributions are presented in chronological order. For each, we provide a short introduction about the work and results results. We then provide more details about the methodology and the way CODEA contributed to the research. Finally, some important findings of each work are briefly discussed. In the reminder of this chapter we explain each contribution in a subsection as follows:

- Subsection 3.4.1 presents the first work implemented in CODEA, a Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) for the Vehicle Routing Problem with Time Windows (VRPTW).
- Subsection 3.4.2 presents an adaptation of the algorithm in Section 3.4.1 to the Steiner Tree in Graph and Delay Constrained Multicast Routing Problem.

- Section 3.4.3 presents CODEA $v2$, and introduces it as a framework to create groups of cooperative agents to solve multi-objective optimisation (MOO) problems.
- Subsection 3.4.4 presents an adaptation of the algorithm in Section 3.4.1 to the University Course Timetabling Problem (UCTTP).
- Subsection 3.5 draws some conclusions and raises a series of key considerations for other uses of CODEA.

3.4.1 Exploring feasible and infeasible regions in the Vehicle Routing Problem with Time Windows using a Multi-Objective Particle Swarm Optimisation Approach

This project presents an approach to the Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) applied to the Vehicle Routing Problem with Time Windows (VRPTW). The purpose of this investigation was to determine the ability of the proposed MODPSO to evolve very infeasible solutions (large number of constraints violated) to feasible regions. The results of this work were presented in the 2008 Workshop on Nature Inspired Cooperative Strategies for Optimisation (NICSO 2008) [31].

Our MODPSO was based on a single-objective DPSO applied to the Minimum Labelling Steiner Tree Problem [54]. This DPSO algorithm is explained in Chapter 8 - Section 8.2. This algorithm was intentionally simple. All particles within the swarm were initialised with random solutions (route-plans). Besides, only two operators were implemented: 1) an inter-route operator that exchanges pairs of customers within a route-plan (for inertial moves) and 2) an operator that copies an entire sub-route from one to another route-plan and then removes duplicates (for all other moves). Besides, our algorithm performed a simple local search after each move to encourage local exploration. In order to promote the move of the swarm towards feasible areas, we forced the minimisation of hard constraint violations in the objective function. In total, the algorithm dealt with the minimisation of 8 objectives (4 of which were hard constraints turned into soft constraints). These minimisation objectives were: 1) number of vehi-

cles, 2) travel distance, 3) travel time, 4) waiting time, and the soft constraints: 1) time window violation, 2) number of time window violations, 3) capacity violation and 4) number of capacity violations. Pareto dominance was used in order to select the leader of the swarm: local-best (lbest) and global-best (gbest).

The proposed MODPSO was tested on 6 instance from the Solomon's dataset. This dataset is explained in-depth in Chapter 5. Results showed that this basic MOPSO was able to evolve very poor quality infeasible solutions (randomly initialised) to feasible ones which are also reasonably close to the known optima.

3.4.2 Particle Swarm Optimisation for the Steiner Tree in Graph and Delay Constrained Multicast Routing Problems

In this work, we collaborated in the implementation of the first MODPSO proposed to tackle both the Steiner Tree in Graph and the Delay Constrained Multicast Routing Problems. CODEA was not used for this implementation. Instead, we developed a DPSO (called JPSOMR) with all the features that worked for VRPTW (Section 3.4.1) in a simulator to test these problems. This work has been accepted in the *Journal of Heuristics*, and to appear in 2012.

Given an undirected graph $G = (V, E)$, where V is a set of nodes and E is a set of links which interconnects the nodes in V. Each link $e_{ij} \in E$ has a weight associated that represents the cost to go from node i to node j. The Steiner Tree Problem (STP) consists of finding the least-cost tree $T \in G$, $T \subseteq E$ to go from a source node $s \in V$ to a number of destination nodes R, such that $V = R \bigcup \{s\}$ and $s \notin R$. Multicast Routing Problems concerns the search of optimal routing trees to go from the source node s to the destination nodes in R while meeting all QoS (Quality of Service) requirements. The two most common QoS requirements are: delay and cost. The delay is denoted by the sum of total delays from the source node s to all destination nodes in R. The cost is calculated as the sum of the weights associated to all links to go from the source node s to all destination nodes in R.

JPSOMR initialises the swarm with randomly generated trees. A path replacement op-

erator was implemented to recombine particles which move towards one another. This operator consists of the replacement of a path in the *follower* according to the cheapest path found in the *attractor*. Another operator was introduced to promote the local exploration of each particle. This operator randomly removes a *superpath* in the current solution structure and reconnects the resulting two sub-trees by using a random link. A local search was also implemented in order to improve the solutions found by the swarm in each generation. This local search was powered by a simple operator that removes a non-destination node and creates a new spanning tree using the Prim's spanning tree algorithm. The local search used two criteria to accept solutions: first improvement (the first solution that improves the current solution is accepted) and best improvement or greedy (the best neighbouring solution is accepted).

A number of experiments were carried out to find the optimal size for the population, as well as the best strategy to accept new solutions (first improvement / best improvement). Using this configuration, our JPSOMR was compared to an extensive number of algorithms using four different sets of instances. In the first set, we compared the performance of JPSOMR to the one of a GRASP algorithm tailored for STP (GRASP-CST). The tests were carried out on small instances and showed that JPSOMR got better results minimising the tree cost. However, our algorithm seemed to be worse than GRASP-ST in terms of speed. For this reason, the second set of experiments was executed on the same instances, but limiting the execution time to 60 seconds. With this configuration, GRASP-CST got slightly better results. JPSOMR was also compared to another DPSO (DPSO-ST) tailored for STP. In this third set of experiments, with larger and harder instances, we compared the performance of JPSOMR, GRASP-CST and DPSO-ST. According to the results, DPSO-ST is the best performing algorithm, followed by JPSOMR and GRASP-CST. In the last set of experiments, a number of algorithms were compared to JPSOMR in random networks. This set of algorithms included Heuristics, GA-based and TS-based algorithms, Path Relinking, VNS and GRASP-CST. In these tests, JPSOMR produced the best average tree cost.

3.4.3 CODEA - An Agent Based Multi-Objective Optimisation Framework

This paper introduces CODEA as a framework to create flexible groups of agents to tackle multi-objective optimisation problems using the paradigm of cooperation. This work provides an overview of how the core components in CODEA $v2$ work. Furthermore, we explain how to implement the MODPSO presented in Section 3.4.1 and the Travelling Salesman Problem (TSP) with two objectives. In order to demonstrate the communication capabilities of CODEA, we test our MODPSO applied to TSP with three communication topologies: 1) start topology (all agents send and receive information (solutions) of all other agents), 2) ring topology (each agent sends information to two neighbouring agents), and 3) k-random topology (each agent sends information to k random number of agents). This work was presented at the 2010 Spanish Congress on Metaheuristics, Evolutionary and Bio-inspired Algorithms (MAEB 2010) [33].

Given a graph $G = (V, E)$, where V is a set of nodes and E is a set of links which interconnects the nodes in V. Each link $e_{ij} \in E$ has a weight associated that represents the cost (travel distance) to go from node i to node j, denoted as $d[v_i, v_j]$. The Travelling Salesman Problem (TSP) consists of finding the permutation (travel-plan) $P = [v_0, v_1,\ldots,v_{N-1}]$, $v \in V$, such that, $Length[P] = d[v_{N-1}, v_0] + \sum_{i=1}^{N} d[v_{i-1}, v_i]$ is minimum. This paper goes through the key steps to implement our MODPSO and TSP with two objectives in CODEA. The implementation process is split in two sections, one for the problem (TSP) and another for the solving method (MODPSO). The problem section consists of four classes: TSPDataProblem (container of data for the TSP instance), TSPOperator Lib (contains a mutation operator, a crossover operator and a local search), TSPSolution (contains the data structure of the travel-plan) and matrixSumObjective (generic evaluator for TSP objective functions). Regarding the TSPOperator class, we implemented: 1) a mutation operator which swaps pairs of randomly selected customers *square root* times the size of the travel-plan, 2) a crossover operator which creates an offspring (travel-plan) out of the interchange of a random section of the parents, and 3) a local search which explores the neighbourhood of each travel-plan by swapping pairs of customers iteratively. The solving method section consists of two classes: DPSOResolutorPhase (models the behaviour of the DPSO agent) and DPSOAgent (in-

terfaces DPSOResolutorPhase and TSPOperator).

In order to test this implementation, we evaluate the performance of the three above-mentioned communication topologies. These tests are carried out on 20 bi-objective TSP instances, half of them with 50 customers, and the other half with 100. We use the S-metric [275] for the assessment of the results. This quality metric computes the hyper-volume covered by the approximation set of non-dominated solutions obtained by using each topology and a reference point. According to this metric, the k random topology is the best strategy to communicate information. However, the ring topology offered the best compromise between speed and performance.

3.4.4 Developing Asynchronous Cooperative Multi-agent Search

This research is another collaborative work. In this case CODEA was applied to the University Course Timetabling Problem (UCTTP) [214]. Two novel asynchronous co-operative search approaches were developed in CODEA. These two approaches were based on the DPSO presented in Section 3.4.1. The first approach is an Asynchronous Cooperative Multi-heuristic (ACMH), and the second is an Asynchronous Cooperative Multi-hyper-heuristic (ACMHH). Both approaches were fairly similar, the only difference being that agents in ACMHH could choose from three low-level heuristics. This work is discussed in the PhD thesis by Joe Obit [187].

The UCTTP consists of assigning a number of events $E = \{e_1, e_2, \ldots, e_n\}$ to be scheduled in a set of 45 time-slots $T = \{t_1, t_2, \ldots, t_{45}\}$ (9 time-slots a day, 5 days a week). There is a set of m available rooms $R = \{r_1, r_2, \ldots, r_m\}$ in which the events take place, a set of k students $S = \{s_1, s_2, \ldots, s_k\}$ who attend the events, and a set of l features $F = \{f_1, f_2, \ldots, f_l\}$ which are provided by rooms and required by events. Besides, there are number of hard constraints and soft constraints. A solution is considered feasible if it does not violate any hard constraint. The goal of this problem is to find the feasible solution with the least number of soft constraint violations.

ACMH and ACMHH algorithms used an initialisation heuristic to create a feasible solution. Three low-level heuristics were introduced to improve the quality of agents

positions: 1) a heuristic that selects one event randomly and assigns it to a randomly selected feasible pair time-slot/room, 2) a heuristic that selects two events randomly and swaps their time-slots and room forcing feasibility, and 3) a heuristic that selects three events randomly and exchanges their time-slots and room randomly forcing feasibility. Furthermore, two acceptance criteria were implemented to enable agents to accept new candidate solutions: 1) a non-linear great deluge, and 2) a simulated annealing.

A number of simulations were carried out on a standard UCTTP dataset. According to the results, the two algorithms proposed found better results outperforming all other existing approaches, both in the literature and previously discussed by Obit [187]. For small instances and in all runs, both algorithms found the optimal solutions. For medium instances and in comparison to other algorithm in the literature discussed by the author, ACMH and ACMHH algorithms produced better results in four instances which reached reduction of penalties up to 70%. In large instances there was also a reduction of penalties by 6%. Summarising, ACMH and ACMHH algorithms found new best solutions for 6 out of the 11 problem instances in the chosen dataset.

3.5 Conclusions

In this chapter we introduce CODEA, a COoperative DEcentralised Architecture for the development of cooperative agent systems. We also discuss four other feature-rich frameworks for multi-objective optimisation: *JMetal*, *ECJ*, *HeuristicLab* and *ParadisEO*. None of these frameworks seem to provide explicit mechanisms to create cooperative systems of agents to tackle multi-objective optimisation problems. We present CODEA as a multi-objective optimisation framework to fill this gap. We provide a high-level overview of the main features of the last two versions of CODEA: *CODEA v2* and *CODEA v3*. CODEA v2 is an evolution of the first version that provides a number of features to deal with multi-objective optimisation problems. CODEA v3 is the result of the hybridisation of CODEA v2 and ParadisEO-MOEO. Instead of implementing more multi-objective algorithms on CODEA, we built CODEA v2 on ParadisEO. This extension enables ParadisEO to work with cooperative agent systems, and allows CODEA

v3 to use ParadisEO's objects at all levels. The product of this hybridisation brings up a number of new ready-to-use features which are not present in state-of-the-art multi-objective optimisation frameworks, such as cooperation and organic groups.

This chapter also presents four research works related to CODEA $v2$. These works concern four different combinatorial optimisation problems: 1) Vehicle Routing Problem with Time Windows (VRPTW), 2) Travelling Salesman Problem (TSP), 3) Steiner Tree Problem (STP) and Multicast Routing Problem, and 4) University Course Time Tabling Problem (UCTTP). These problems have been tackled with a MODPSO-based algorithm. Our MODSPO has been adapted to each problem domain by specifying three problem-dependent components: 1) a set of operators, 2) evaluation function(s) and 3) a solution data structure. The only work that has not been implemented in CODEA is the one concerning the Steiner Tree Problem and Multicast Routing Problem (Section 3.4.2). The other three research projects have been directly benefited from CODEA. This has shown the potential of CODEA for applying agent-based solving methodologies to other problem domains.

The following chapters discuss a number of studies in which CODEA has been involved. The next chapter presents the Dynamic Lexicographic Approach (DLA). DLA is a multi-objective ranking approach to discriminate solutions using random lexicographic orderings based on certain probability distributions.

Currently, we are focusing our efforts on the development of new algorithms and the generation of documentation of the API. Further work should include the incorporation a number of tutorials explaining basic uses of CODEA. Once the documentation is ready, we plan to submit CODEA to ParadisEO's project webpage.

CHAPTER 4

Dynamic Lexicographic Approach for Multi-objective Optimisation

Summary

There is a variety of methods for ranking objectives in multi-objective optimisation and some are difficult to define because they require information a priori (e.g. establishing weights in a weighted approach or setting the ordering in a lexicographic approach) [52]. In many-objective optimisation problems, those methods may exhibit poor diversification and intensification performance. We propose the Dynamic Lexicographic Approach (DLA). In this ranking method, the priorities are not fixed, but they change throughout the search process. As a result, the search process is less liable to get stuck in local optima and therefore, DLA offers wider exploration ability in the objective space.

DLA is one of the novel features introduced in CODEA v2. This ranking scheme was implemented in order to study alternative ranking methods in multi-objective optimisation. This chapter compares the performance of DLA to that of Pareto dominance and lexicographic ordering according to the hypervolume [271]. These methods are tested on a Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) algorithm tackling the Vehicle Routing Problem with Time Windows.

4.1 Introduction and motivation

Multi-objective Optimisation Problems (MOPs) have a number of objectives that are usually in conflict, so improving one objective leads to worsen another. In particular, *many-objective optimisation problems* involve the optimisation of four or more objectives, presenting a considerable challenge for some solution methods. Most research in this area has focused on the study of MOPs with two or three objectives. This is due to the assumption that the scalability of multi-objective methods to *many* objectives would be straightforward. Recently, several papers have investigated this issue. Khare et al. [152] investigated the scalability of a number of Multi-Objective Evolutionary Algorithms (MOEA) with respect to 2 to 8 objectives. Results were assessed according to three criteria: 1) ability to converge to Pareto front, 2) diversity in the obtained non-dominated solution set, and 3) running time. The authors concluded that the results obtained with 2 or 3 objectives cannot be generalised to a larger number of objectives. Hughes [138] compared the performance of NSGA-II [65] to that of multiple single objective optimisers in both MOPs and many objectives optimisation problems. Results indicated that NSGA-II looses efficiency as the number of objectives increases. Similar results were found by Wagner et al. [253]. That study concluded that a Pareto-based approach cannot succeed in dealing with many-objective problem instances.

Pareto dominance (Section 2.3.1) uses a strict ranking scheme that sometimes fails to discriminate between solutions, as it only accepts improvements in all objectives at the same time. Other methods like the lexicographic approach (Section 2.3.3) impose a static behaviour, as objectives are ranked according to a fixed relative importance.

Two main alternatives have been proposed to deal with the scalability problem in many objectives optimisation problems [48]. One is the relaxation of the form of Pareto optimality, so that it is possible to accept solutions which worsen some objectives in certain quantity, if others witness an improvement [224]. Another one considers the reduction of objectives of the original MOP [23]. However, due to the difficulty involved in reducing the dimensionality of the MOP, the first alternative is more popular in the literature [171].

Dynamic Lexicographic Approach (DLA) is an alternative ranking approach for *many-objective optimisation*. DLA presents a relaxation, not of the form of Pareto dominance, but of the form of the lexicographic approach. DLA offers an intuitive approach to establish a dynamic ranking among objectives. Rather than establishing a fixed priority among the objectives, the decision maker establishes a preference. This preference is then used with a probability mass function (*pmf*) to generate a vector of priorities that changes dynamically throughout the search process.

This chapter is organised as follows. The algorithm for the Dynamic Lexicographic Approach is detailed and exemplified in Section 4.2. We describe our experiments in Section 4.3 and discuss results in Section 4.4. Finally, our contribution and proposed further research are summarised in Section 6.9.

4.2 Dynamic Lexicographic Approach

The Dynamic Lexicographic Approach (DLA) offers an intuitive mechanism to establish a dynamic ranking among objectives. DLA allows decision makers to establish preferences among objectives rather than fixed priorities as in the lexicographic approach. Decision makers also define a probability mass function (*pmf*) which associates a probability to each preference. These probabilities are used to create different priority vectors which are later used by a standard lexicographic ranking technique. DLA does not rule out any preference. Thus preferences with a low probability still have a chance to appear in the first position of the vector of priorities. Additionally, the continuous change of priorities makes it possible to avoid premature convergence as the exploration is broader.

Figure 4.1 shows the pseudo-code of the algorithm that generates the vector of priorities. First, this algorithm initialises the vector of priorities v with an empty vector (Figure 4.1 - line 1). This vector will hold the final ordering that will be used by the lexicographic ranking approach. A temporary vector P is then assigned with the output of the procedure *generateProbVector(pmf, N)*, where pmf is the *probability mass function* and N the number of objectives to operate (Figure 4.1 - line 2). The actual generation

of the final vector of priorities v is carried out using roulette-wheel selection on the vector P. The roulette-wheel selection procedure interprets the probability vector P as a segment spanned in $[0, 1]$. This segment has N sub-segments, so that the size of each sub-segment corresponds to each probability value in P. The roulette-wheel selection mechanism is performed in four steps which are repeated N times (Figure 4.1 - line 3). The first step is to generate a random number r with uniform distribution in $[0, 1]$ (Figure 4.1 - line 5). This random number is then passed to the procedure *selectPriority(r, P)* (Figure 4.1 - line 6), which selects the preference p associated to the sub-segment in which the random number r falls. In the third step, p is pushed back in the vector of priorities v (Figure 4.1 - line 7). The fourth step is to re-scale P (Figure 4.1 - line 8). The re-scaling procedure removes p from P and re-normalise P by dividing the remaining probability values by their sum.

The procedure *generateProbVector(pmf, N)* takes two steps: 1) it calculates the probability values evaluating pmf for all preferences in $\{1, ..., N\}$, and 2) it normalises these probabilities before returning the resulting vector. The normalisation process consists of dividing each probability value by the sum of all probabilities. Thus, the sum of probability values in P equals 1.

```
1 v = [ ]
2 P = generateProbVector(pmf, N)
3 for x = 1 to N
4 {
5    r = rand()                  // "r" is a random number in U[0,1]
6    p = selectPriority(r, P)
7    v.pushBack(p)
8    re-scale(P, p)
9 }
```

Figure 4.1: Pseudo-code of the algorithm that generates the lexicographic sequence

To clarify how DLA works, we provide an example for $N = 4$ objectives. Let us assume that objective i is assigned preference i, that is, $pref(o_i) = i$ for $i = 1, \ldots, N$, where N is the number of objectives. Suppose the decision maker provides the function $p(i) = 0.8exp(-0.4 * (i - 1))$. Firstly, we evaluate this function for $i = 1, \ldots, N$. Since $N = 4$, we calculate the probabilities as $p(1) = 0.8, p(2) = 0.54, p(3) = 0.36$ and $p(4) =$

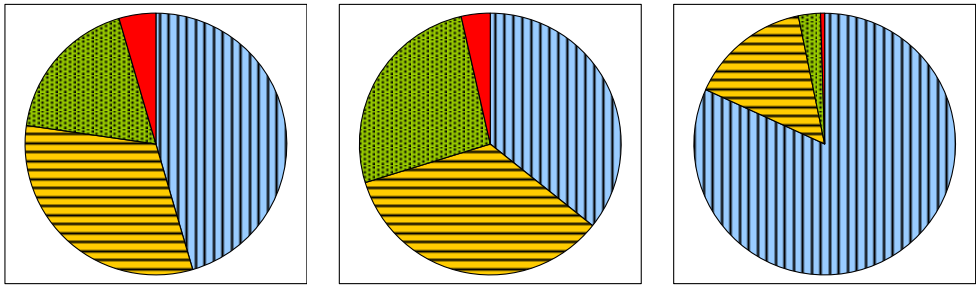


Figure 4.2: Distribution of probabilities according to three probability mass functions for $N = 4$ objectives. The resulting distribution of a *linear* function is depicted on the left - $p(i) = -3(i-1) + 10$, a *quadratic* function in the middle - $p(i) = -(i-1)^2 + 10$, and an *exponential* function on the right - $p(i) = e^{-1.7(i-1)}$, for $i = 1, \ldots, N$

0.24. Secondly, these probabilities are normalised as $p'(i) = p(i) / \sum_{k=1}^{N} p(k)$, obtaining the values $0.41, 0.28, 0.19, 0.12$. In a third step, we split the segment $[0, 1]$ into sub-segments, such that the length of each sub-segment is equal to each of the the above-mentioned probability values. Therefore, we obtain $P = [0, 0.41, 0.69, 0.88, 1]$ (Figure 4.1 - line 2), in which each preference has a sub-interval assigned. The first preference has interval $[0, 0.41)$, the second $[0.41, 0.69)$, the third $[0.69, 0.88)$ and the fourth $[0.88, 1]$. The algorithm goes through the above steps only once. Regarding the generation of priority vectors, roulette-wheel selection is applied $N = 4$ times using this segment (Figure 4.1 - line 3). First, a random number r is generated with uniform distribution in $[0, 1]$ (Figure 4.1 - line 5). Lets suppose that $r = 0.70$. As r falls in the sub-interval $[0.69, 0.88)$, the third objective ($p = 3$) will be selected (Figure 4.1 - line 6) and pushed back in the vector of priority $v = [3]$ (Figure 4.1 - line 7). After this operation, we remove the selected sub-interval and the segment is re-scaled (Figure 4.1 - line 8), so we obtain $P = [0, 0.51, 0.85, 1]$. The loop then starts over and a new random number is generated. When this loop is over, v contains the priority vector that can be used in a lexicographic approach to discriminate solutions.

The probability mass function (*pmf*) plays a fundamental role in the performance of the DLA. Depending on the shape of this function, different probability values are assigned to each objective producing different lexicographic orderings (Figure 4.2). Therefore, assuming that the decision maker establishes decreasing preferences (as in the example above), there are three main types of functions: linear, quadratic and exponen-

tial. Linear functions assign probabilities with a constant step among them. Quadratic functions assign a similar probability to those objectives with the highest preferences and zero or close to zero to those with the lowest. Finally, an exponential function assigns high and distinct probabilities to objectives with high preference and assigns low but non-zero probability to those with low preference. Furthermore, decision makers can fine tune the shape of *pmfs* by setting *coefficients*. Figure 4.2 presents the distribution of probabilities according to three *pmfs* for $N = 4$ objectives. The linear *pmf* $p(i) = -3(i-1) + 10$, $(i = 1, \ldots, N)$ uses: 1) -3 as slope, and 2) 10 as the *y-intercept* of the line. Similarly, the quadratic *pmf* $p(i) = -(i-1)^2 + 10$, $(i = 1, \ldots, N)$ uses: 1) -1 to set the direction of the curve, and 2) 10 as the *y-intercept* of the curve. Finally, the exponential *pmf* $p(i) = e^{-1.7(i-1)}$, $(i = 1, \ldots, N)$ uses -1.7 to set the curvature.

4.3 Experiments

In this section, we describe the settings used in our experiments. Our efforts focus on comparing different ranking approaches. For this purpose, we implemented a canonical Discrete Particle Swarm Optimisation (DPSO) inspired by [54]. This algorithm was intentionally simple. Particles could only perform one out of four types of moves at each generation. These four moves included one inertial move (only the moving particle was involved), and three follower-attractor moves (two particles involved). While in the inertial move only the moving particle was involved in a mutation operation, the follower-attractor moves involved the crossover operation between the moving particle (follower) and one of the three particles/positions (attractor): 1) best personal position b_i, 2) the best position achieved by the swarm so far g, and 3) the best position in the neighbourhood of the moving particle at the current generation g_i.

All particles within the swarm were initialised with random solutions (route-plans). Besides, only two operators were implemented: 1) an inter-route operator that exchanges pairs of customers within a route-plan (for inertial moves), and 2) an operator that copies an entire sub-route from one to another route-plan and then removes duplicates (for all other moves).

The probability of all attractors were set to 0.25. In our simulations, the swarm was formed by 50 particles evolving for 2000 iterations. This algorithm was applied to the Vehicle Routing Problem with Time Windows (VRPTW) using the Solomon's dataset [75]. These instances are divided in three classes: *C1XX* (customers positioned in clusters), *R1XX* (customers randomly spread) and *RC1XX* (some customers forming clusters and others randomly positioned). Regarding the DPSO implementation, two operators are used to move the particles within the swarm. A crossover operator is used to move particles towards other particles' locations. This operator copies a random route from an attractor to the moving particle. A mutation operator is used to encourage the local exploration within route-plans. In the experiments, this operator exchanges customers from one route to another within the route-plan in the solution of the moving particle.

In order to assess the performance of each ranking approach, a number of minimisation objectives were considered: *Number of vehicles* (Z_{nv}), *Travel Time* (Z_{tt}) or elapsed time of the route-plan, *Waiting Time* (Z_{wt}) or sum of all time the drivers need to wait in case of an early arrival, *Travel Distance* (Z_{td}) or length of the whole route-plan, *Time Window Violation* (Z_{twv}) or sum of lateness of all arrivals, *Number of Time Window Violations* (Z_{ntwv}) or number of customers not served within the appropriate time, *Capacity Violation* (Z_{cv}) or amount of exceeding capacity on vehicles and *Number of Capacity Violations* (Z_{ncv}) or number of vehicles whose capacity is being exceeded. Reducing violations are considered as objectives in this study. In this way, we entitle the decision maker to decide on the convenience of serving customers out of their time windows or exceeding the capacity of some vehicles.

Regarding the *coefficients* for the ranking schemes, DLA is presented in two versions for selecting leaders and updating best particle's positions (e.g. the best personal position b_i and the best position achieved by the swarm so far g).

The first version of DLA uses a greedier approach to establish the probability for each preference using Equation 4.3.1. The first coefficient (0.9) increases its curvature and the second moves it up by 0.05. We set these coefficients according to some preliminary

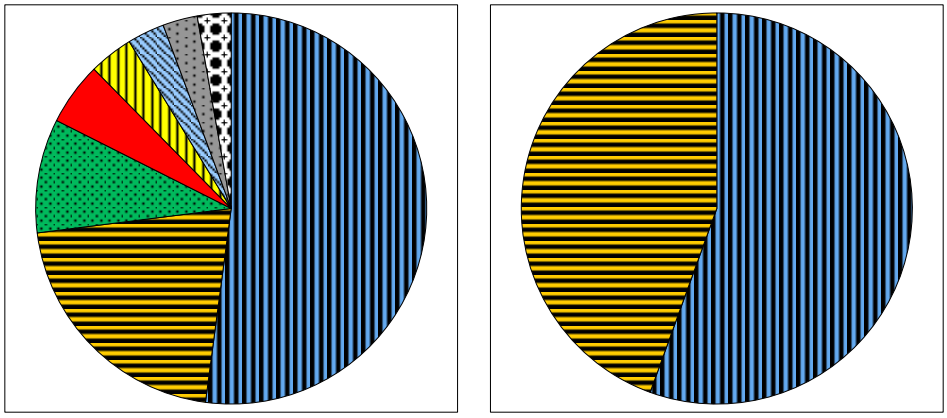


Figure 4.3: Distribution of probabilities used by DLA and DLA2 for $N = 8$ objectives. The resulting distribution according to the probability mass functions: 1) $p(x) = 0.9 * e^{-x} + 0.05$ is depicted on the left, and 2) $p(x) = 0.6 * cos(0.7x + 0.1) + 0.3$ with $x = \{0, 1\}$ is depicted on the right.

tests (Figure 4.3 - on the left).

$$p(x) = 0.9 * e^{-x} + 0.05 \tag{4.3.1}$$

The second version of the DLA ($DLA2$) splits the ranking process into two phases (Equation 4.3.2). If the current number of iterations is less or equal than a given k, a probability mass function is used to encourage intensification. Otherwise, a different probability mass function is employed to encourage diversification. To this aim, the first phase only takes into account the first two highest preferences using the *pmf* $p(x) = 0.6 * cos(0.7x + 0.1) + 0.3$, with $x = \{0, 1\}$. For values including $x = 0$ and $x = 1$, this function (equivalent to a quadratic expression in the given range) assigns high and similar probabilities to the two objectives with the highest preference (Figure 4.3 - on the right). The coefficients were set, as in the previous *pmf*, using preliminary computational experiments. This intensification phase lasts k iterations. In our experiments k is set to 500 which corresponds to 25% of the total number of iterations that the algorithm runs. The diversification phase runs for the remaining iterations using the *pmf* $p(x) = 0.9 * exp(-x) + 0.05$, working with the whole set of preferences (Figure 4.3 - on the left).

$$p(x) = \begin{cases} 0.6 * cos(0.7x + 0.1) + 0.3, x = \{0, 1\} & k \leq 500 \\ 0.9 * e^{-x} + 0.05 & k > 500 \end{cases} \quad (4.3.2)$$

Summarising, the success of DLA2 strives to provide a good compromise between intensification and diversification. For this purpose, Equation 4.3.2 splits the ranking process in two phases. In the first phase, as shown in Figure 4.3 - on the right, the two objectives with highest preference have probabilities 56% and 44%, respectively. Thus, the search focuses on the optimisation of these two objectives only. This speeds up the convergence, but it might lead to local optima. The second phase changes the distribution of probabilities to avoid this side effect. This diversification phase, as shown in Figure 4.3 - on the left, assigns the descending probabilities 52%, 21%, 9%, 5%, 4%, 3%, 3% and 3% to each preference from the highest to the lowest.

In a preliminary study, we tested a number of different combinations of objectives using Pareto dominance and Lexicographic ordering. This study compared the average hypervolume (Section 2.3.4) values obtained by using each combination. We ran experiments with Pareto dominance involving 2 to 8 objectives. We found the best results working with pairs of objectives. With three or more objectives, Pareto dominance produced the premature convergence of the swarm. Furthermore, these experiments also showed that the best setting for Pareto dominance was to discriminate solutions using (Z_{td}, Z_{twv}). For the lexicographic approach, the study involved finding the best sequence (ordering) of objectives. We found that the best sequence of objectives was (Z_{ntwv}, Z_{td}, Z_{wt}, Z_{tt}, Z_{twv}, Z_{cv},Z_{ncv}, Z_{nv}).

With respect to the DLA and DLA2, they both used the same sequence for preferences as the lexicographic for priorities. But, for DLA2 this sequence was used after 500 generations as explained above.

4.4 Results

In order to analyse the performance of the proposed DLA variants, we present the results in two modes, qualitative and quantitative. Firstly, we depict the approximated

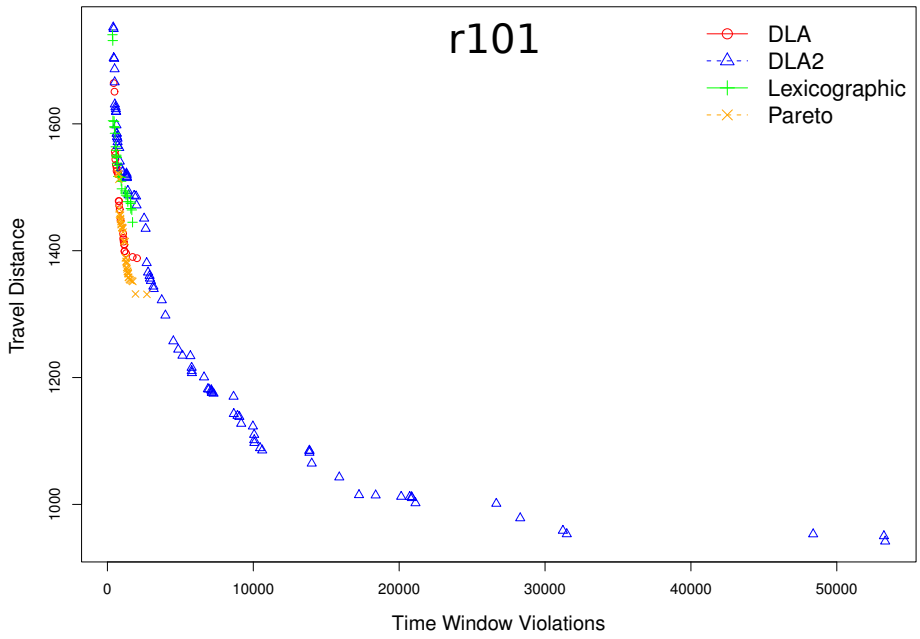


Figure 4.4: Approximation sets from different ranking techniques on the Solomon's instance *R*101. *Time Window Violations* (Z_{twv}) vs *Travel Distance* (Z_{td}).

non-dominated sets obtained by each strategy using two objectives. Secondly, we show a table containing the normalised average *hypervolume* values for each instance family and technique. Figures 4.4 and 4.5 show the approximated non-dominated sets obtained when using Pareto dominance, lexicographic ordering, DLA and DLA2 on instances *R*101, *RC*101 respectively. The results are presented using two pairs of objectives. We only show the most meaningful combinations of pairs of objectives: 1) *Time Window Violations* (Z_{twv}) vs *Travel Distance* (Z_{td}) (for both the plots and the hypervolume), and 2) *Travel Time* (Z_{tt}) vs *Travel Distance* (Z_{td}) (for the hypervolume). Thus, we do not consider those pair-wise relationships in which the hypervolume was zero. Note that we use pair-wise comparisons as this was the best scenario for Pareto dominance. In order to provide a fair comparison, we filtered out the non-dominated solutions with respect to only the objectives under study. For example in the comparison (Z_{twv}) vs (Z_{td}), the non-dominated solutions obtained by using the other methods (i.e. the lexicographic ordering, DLA and DLA2) are classified with respect to this pair of objectives only.

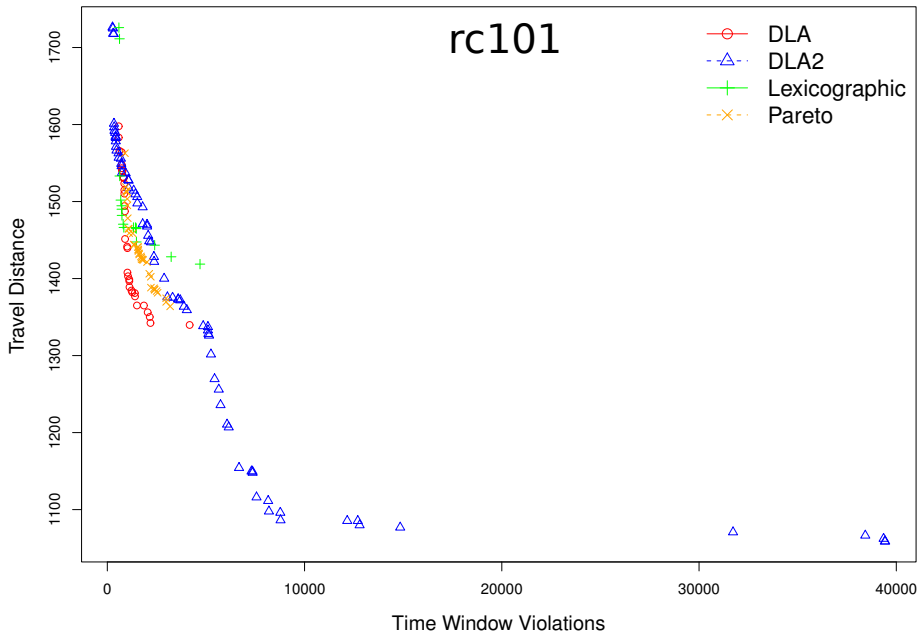


Figure 4.5: Approximation sets from different ranking techniques on the Solomon's instance *RC*101. *Time Window Violations* (Z_{twv}) vs *Travel Distance* (Z_{td}).

4.4.1 Qualitative analysis

Time window violations (Z_{twv}) vs Travel distance (Z_{td})

With respect to the Z_{twv} vs Z_{td} comparison, DLA2 is clearly superior in both intensification and diversification. The approximation sets obtained by $DLA2$ on $R101$ (Figure 4.4) and $RC101$ (Figure 4.5) have solutions with closer values to the origin, revealing better intensification behaviour. Additionally, these solutions, compared to those of other ranking methods, seem to be more spread along both axes, showing better diversification. Moreover, DLA2 seems to get better results as the difficulty of the instances increases.

In Solomon's instances, time windows are much more restrictive on instance sets $R1XX$ and $RC1XX$. Moreover, the geographical location of customers in these two sets of instances make the problem grow in complexity. This complexity is due to the fact that optimising the distance does not guarantee to obtain a good set of solutions. This improvement can be seen on the results for instances $R101$ and $RC101$ where the difference in performance between DLA2 and the other ranking methods is much more noticeable. For this comparison, DLA gets a slightly better performance than that of

lexicographic approach. This is because DLA uses a greedy function to assign probabilities to each preference. Therefore, the priority vectors were generated within a short distance with the one used for the Lexicographic ordering. Pareto shows a reasonable performance in instances $R101$ and $RC101$.

4.4.2 Quantitative analysis

Table 4.1 shows the performance of each ranking approach on each instance, calculated according to the hypervolume. This metric computes the size of the space limited by the solutions in an approximation set and a reference point. The larger the value of the hypervolume, the higher the quality of the approximation set. We calculated the hypervolume using the same pairs of comparisons used for the previous graphs. So, these results extend the information conveyed by the plots. All values were normalised in relation to 1 according to the highest hyper-volume value obtained in each case. Table 4.1 is divided in three sections. The first section identifies the instances used. The second and third sections show the hypervolume value comparing Z_{twv} against Z_{td} and Z_{tt} against Z_{td} respectively.

For the first comparison, *Time Window Violations* (Z_{twv}) vs *Travel Distance* (Z_{td}), DLA2 gets the best hypervolume value for all instances. On average, it obtains an improvement of 60% over lexicographic ordering which is the second best ranking method almost in a tie with the DLA. Pareto dominance gives the worst performance with an average of 0.18 across all instances. In the second comparison, *Travel Time* (Z_{tt}) vs *Travel Distance* (Z_{td}), DLA2 is not the best in only one instance: $C108$. However, it is very close to that value with a distance of only 0.5%. In general and across all instances, DLA2 presents an improvement of about 25% over the lexicographic ordering, which is again the second best ranking technique. Very close to the lexicographic approach, the DLA gets better results in some instances and worse in others, but on average the former outperforms the latter by about 5%. Pareto dominance comes last with an overall performance of 0.60.

Table 4.2 provides average values across all instance in Solomon's subsets $C1XX$, $R1XX$ and $RC1XX$. This table has the same structure as the one described before. It is di-

Table 4.1: Performance of different ranking approaches on Solomon's instances according to the hypervolume quality measure. The hypervolume is calculated over two comparisons: *Time Window Violations* (Z_{twv}) Vs *Travel Distance* (Z_{td}) - on the left, and *Travel Time* (Z_{tt}) Vs *Travel Distance* (Z_{td}) - on the right. Best values are in bold face.

	Z_{twv} **vs** Z_{td}				Z_{tt} **vs** Z_{td}			
Instance	Pareto	Lex	DLA	DLA2	Pareto	Lex	DLA	DLA2
*C*101	0.1170	0.6128	0.7190	**1.0000**	0.5467	0.8473	0.9052	**1.0000**
*C*102	0.2167	0.7936	0.6162	**1.0000**	0.4729	0.9512	0.8054	**1.0000**
*C*103	0.1407	0.7988	0.3852	**1.0000**	0.3671	0.8991	0.4910	**1.0000**
*C*104	0.2813	0.7111	0.2247	**1.0000**	0.5204	0.8294	0.3501	**1.0000**
*C*105	0.1707	0.7388	0.7779	**1.0000**	0.4015	0.9004	0.9090	**1.0000**
*C*106	0.1712	0.6759	0.4818	**1.0000**	0.4235	0.8482	0.7004	**1.0000**
*C*107	0.6065	0.5958	0.3778	**1.0000**	0.8407	0.8403	0.6738	**1.0000**
*C*108	0.0580	0.6313	0.6128	**1.0000**	0.4982	**1.0000**	0.9826	0.9947
*C*109	0.0793	0.6271	0.5875	**1.0000**	0.4670	0.8838	0.8511	**1.0000**
*R*101	0.1212	0.0571	0.1182	**1.0000**	0.6385	0.5017	0.5714	**1.0000**
*R*102	0.1304	0.1306	0.1413	**1.0000**	0.7226	0.6339	0.6064	**1.0000**
*R*103	0.3395	0.2387	0.2276	**1.0000**	0.8931	0.7139	0.6804	**1.0000**
*R*104	0.1953	0.2579	0.2530	**1.0000**	0.7578	0.7443	0.7946	**1.0000**
*R*105	0.2265	0.2108	0.2843	**1.0000**	0.8286	0.7686	0.8226	**1.0000**
*R*106	0.2636	0.1880	0.1813	**1.0000**	0.8074	0.6511	0.6525	**1.0000**
*R*107	0.1708	0.2964	0.2179	**1.0000**	0.6766	0.6749	0.5988	**1.0000**
*R*108	0.2866	0.3597	0.3544	**1.0000**	0.7296	0.7672	0.7510	**1.0000**
*R*109	0.3666	0.3186	0.3125	**1.0000**	0.8853	0.8185	0.7724	**1.0000**
*R*110	0.1885	0.3749	0.3069	**1.0000**	0.5713	0.7450	0.6579	**1.0000**
*R*111	0.1361	0.4660	0.4376	**1.0000**	0.4760	0.8457	0.8244	**1.0000**
*R*112	0.1468	0.4163	0.4043	**1.0000**	0.5984	0.7920	0.7976	**1.0000**
*RC*101	0.1399	0.1529	0.2562	**1.0000**	0.6493	0.5765	0.6892	**1.0000**
*RC*102	0.1186	0.1331	0.1085	**1.0000**	0.5671	0.5582	0.4900	**1.0000**
*RC*103	0.0787	0.2687	0.2389	**1.0000**	0.5615	0.7196	0.6696	**1.0000**
*RC*104	0.0795	0.3235	0.3014	**1.0000**	0.5211	0.7029	0.7289	**1.0000**
*RC*105	0.1235	0.1844	0.1729	**1.0000**	0.5581	0.6048	0.5779	**1.0000**
*RC*106	0.1915	0.1900	0.2660	**1.0000**	0.5853	0.5534	0.6164	**1.0000**
*RC*107	0.0912	0.2747	0.2789	**1.0000**	0.6664	0.7554	0.7444	**1.0000**
*RC*108	0.0903	0.4350	0.5918	**1.0000**	0.4303	0.7250	0.8049	**1.0000**

Table 4.2: Performance of different ranking approaches on Solomon's instances according to the hypervolume quality measure calculated over two comparisons: *Time Window Violations* (Z_{twv}) vs *Travel Distance* (Z_{td}) - on the left, and *Travel Time* (Z_{tt}) vs *Travel Distance* (Z_{td}) - on the right. Average values and their respective standard deviations are shown for each subset of Solomon's instances ($C1XX$, $R1XX$ and $RC1XX$).

	Z_{twv} **vs** Z_{td}				Z_{tt} **vs** Z_{td}			
Instance set	Pareto	Lex	DLA	DLA2	Pareto	Lex	DLA	DLA2
$C1XX$	.20(.17)	.69(.08)	.53(.18)	1(.0)	.50(.14)	.89(.06)	.74(.21)	1(.0)
$R1XX$	.21(.08)	.28(.12)	.27(.10)	1(.0)	.72(.13)	.72(.09)	.71(.09)	1(.0)
$RC1XX$	.11(.04)	.25(.10)	.2(.14)	1(.0)	.57(.07)	.65(.88)	.67(.10)	1(.0)

vided in three sections: the first states the instance subset, and the second and third show the average hypervolume value comparing Z_{twv} against Z_{td} and Z_{tt} against Z_{td} respectively. We state the standard deviation in parenthesis.

The Frieman test was used to analyse the global differences in the hypervolume values obtained for each method in both comparisons. There are significant differences at 0.01 significance level. In order to find out where these differences are, we carried out a number of pair-wise comparisons using the four ranking approaches with the Wilcoxon test. According to this test, there is a significant difference between DLA2 and lexicographic ordering at 0.01 significance level. The normalised values for the statistic were 4.70 and 4.68, respectively. So, we can safely say that DLA2 is superior to lexicographic ordering in this scenario. Similarly, we compared DLA and lexicographic ordering obtaining the two-way p-values 0.1236 and 0.1285, respectively. This reveals that there is not significant difference between these two ranking approaches at 0.10 significance level. Finally, DLA and Pareto dominance were compared obtaining the two way p-values 0.0007 and 0.0434, respectively. Thus, their results are significantly different at 0.05 significance level.

4.5 Conclusions

Standard ranking schemes such as Pareto dominance have been proven to have scalability problems in dealing with many-objective optimisation problems [152, 138, 253]. Two main alternatives have been proposed to overcome this issue. The one that has

drawn more attention from researchers consists of relaxing the form of Pareto optimality (see [89, 71, 224]).

This chapter presents the study of a novel approach to rank solutions in many-objective optimisation problems. This approach, called Dynamic Lexicographic Approach (DLA), uses a relaxed form of the Lexicographic ordering. DLA does not fix the priorities, but they change throughout the search process by using a probability mass function (*pmf*). As a result, the search process is less liable to get stuck in local optima and therefore, DLA offers a wider exploration in objective spaces with high dimensionality. DLA2 is variant of the DLA that uses double *pmf*. This double *pmf* combines a phase of intensification and a phase of diversification.

A number of experiments were conducted on a canonical Discrete Particle Swarm Optimisation (DPSO) applied to the Vehicle Routing Problem with Time Windows (VRPTW) with 8 objectives. The analysis of results is carried out using two pairs of objectives as this is the best scenario for Pareto dominance. Our simulations show that under this scenario, DLA is a valuable technique to discriminate solutions. In particular DLA2 exhibits much better performance than lexicographic ordering, Pareto dominance and DLA.

According to the results, the best performing ranking scheme is $DLA2$, and the worst is Pareto dominance. Results also revealed that there is not significant difference between the performance of DLA and Lexicographic ordering. This might be due to DLA does not focus on the convergence of the search, but it starts diversifying too soon.

The poor performance of Pareto dominance in this study also corroborates the findings of other researchers regarding the scalability problems of this technique in many-objective optimisation problems [152, 138, 253].

DLA is an alternative ranking technique which is easy to implement and intuitive for the decision maker. Future research should include:

- Investigate the effectiveness of this approach in other many-objective optimisation problems and algorithms. The results that we have obtained in our simulations might be biased by the problem, by the solving technique, or by both.

- Investigate the effectiveness of this approach in comparison to that of other ranking techniques such as ϵ-dominance [161].
- Investigate the adaptability of the DLA with respect to the probability mass function (*pmf*), including setting the parameter k to switch between intensification and diversification.

In order to pursue these lines of research, we plan to implement DLA in CODEA $v3$. This will allows us to compare the performance of this technique to that of other ranking schemes inherited from ParadisEO. We will also extend these experiments to other MOPs such as the multi-objective Travelling Salesman Problem (TSP).

In this study we use the Solomon's dataset for the assessment of three ranking methods. However, as we briefly discussed in Section **??**, this instance-set was not designed for multi-objective optimisation. In the next chapter we carry out an in-depth study on the characteristics and multi-objective suitability of the 100 customers Solomon's instances. The purpose of the next three chapters is to deepen the findings of other research works [105, 238] on the multi-objective suitability of this dataset.

CHAPTER 5

Overview of Solomon's dataset

Summary

A large number of test datasets have been proposed for the Vehicle Routing Problem (VRP) and its variants. One important variant of the VRP takes into account the time slots in which customers want to be served: VRP with Time Windows (VRPTW). For this problem, several datasets have been introduced in order to assess the performance of routing algorithms. One of the most famous datasets for the VRPTW is the 100 customers Solomon's dataset. Results in this set of benchmark problems is the most common way to compare heuristics [20].

In the previous chapter, we used this dataset to test the performance of different multi-objective ranking schemes. However, we found that this dataset might not be suitable for the assessment of the VRPTW with multiple objectives. This chapter discusses the most important characteristics of this dataset. Based on these characteristics, we study the suitability of this dataset for investigating the multi-objective aspect of the VRPTW.

5.1 Introduction

A number of test datasets have been proposed for the VRPTW. Some benchmark problems and their characteristics are:

- **Breedam's instances** (Breedam, 1994 [75]) are organised in two sets: T1 and T2,

each with 60 instances. Each instance has 100 customers. Each customer has a fixed demand (10 units) and an unlimited number of vehicles is available to serve them, each with capacity 100. In set T1, all customers have the same time window $[0, 60]$. In T2, all customers have two time windows $[0, 30]$ and $[50, 80]$. For both sets T1 and T2, the time window for the depot is $[0, 960]$.

- **Russell's instances** (Russell et al. 1995 [218]) consist of two pairs of real-world instances with data taken from a fast food delivery company in the USA. The first pair of instances $\{D249, E249\}$ have 249 customers, with vehicle capacities of 50 and 100, respectively. For both of these instances, the travel time was calculated by a linear function that allows velocity to vary as a function of distance. The second pair of instances $\{D417, E417\}$ have 417 customers. Instance $D417$ has wider time windows than $E417$. For both of these instances, it is assumed that a unit of time is equal to a unit of distance.

- **Homberger's instances** (Homberger, 1999 [75]) are an extension of Solomon's instances. For each Solomon's subset, new instances are provided with 200, 400, 600, 800 and 1000 customers. These instances have the same specifications as Solomon's except that customers are located in different positions.

- **Cordeau's instances** (Cordeau, 2003 [75]) are also based on Solomon's instances. This dataset is divided in three subsets according to the position of customers: clustered $CXXX$, random $RXXX$ and mixed $RCXXX$. Each subset contains two smaller sets, that is, there are six subsets in total: $C1XX$, $C2XX$, $R1XX$, $R2XX$, $RC1XX$ and $RC2XX$. Customers in each of these subsets share the same data with the exception of the time windows. All instances have 100 customers with individual demand equal to 1 in all cases. Customers must be served by one of a limited set of specific vehicles which varies according to the instance. Vehicles capacities also take different discrete values depending on the problem instance $\{200, 700, 1000\}$.

- **Solomon's instances** (Solomon, 1987 [233]) is perhaps the most widely used VRPTW dataset in the literature. It consists of 56 instances in which 100 customers must

be served. According to the geographical distribution of customers, the dataset is divided into three subsets: $CXXX$ (customers are grouped in clusters) 17 instances, $RXXX$ (customers are uniformly distributed) 23 instances, and $RCXXX$ (some customers are located in clusters while others are uniformly distributed) 16 instances. Each of these three subsets consists of two subsets. The clustered instances are in subsets: $C1XX$ and $C2XX$. The subset $C1XX$ has a different layout for customer locations and narrower time windows than subset $C2XX$. For both $RXXX$ and $RCXXX$, their two subsets share the same layout, but customers in the first subset have narrower time windows like in $CXXX$. The service time for customers is 90 time units for subsets $CXXX$ and $RCXXX$, and 10 time units for subset $RXXX$. The demand varies depending on the customer and instance, but it takes a discrete values in $\{10, 20, 30, 40\}$. More information about Solomon's instances, link to download and optimal solutions can be found at [75].

In the next sections, we analyse the main characteristics of this dataset. In section 5.2, we give an overview of the structure of the dataset files. Section 5.3 discusses the layout of the customers' locations. Section 5.4 explores the shape of the time windows and service times of customers across all instances. Section 5.5 is focused on the study of the demands and the vehicles capacities. Section 5.6 discusses the suitability of this dataset for VRPTW according to the characteristics under review. Finally, the complexity of this dataset is discussed in Section 5.7.

5.2 Structure of the data files

All the Solomon's dataset files share the same structure (Figure 6.1). The first line states the name of the instance $\{C|R|RC\}XXX$. The second line is always a blank space. The third and the fourth lines contain the keywords 'VEHICLE' and 'NUMBER CAPACITY' respectively. The following line specifies the maximum number of vehicles available and the maximum capacity of each of those vehicles. The sixth line is another blank space. The next line contains the keyword 'CUSTOMER'. The eighth line contains the keywords: 'CUST.NO.', 'XCOORD.', 'YCOORD.', 'DEMAND', 'READY TIME', 'DUE

DATE' and 'SERVICE TIME'. The next line is blank. The tenth line specify the details of the depot according to the keywords stated in the eighth line. The 'CUST.NO.', 'DEMAND', 'READY TIME' and 'SERVICE TIME' of the depot are always 0. From the eleventh line and onwards, we find the details of the customers according to the keywords stated in the eighth line.

```
1  <name of the instance>
2  <blank>
3  <keyword: VEHICLE>
4  <keyword: NUMBER> <keyword: CAPACITY>
5  <number of available vehicles> <capacity of each vehicle>
6  <blank>
7  <keyword: CUSTOMER>
8  <keywords for each column>
9  <blank>
10 <id_{0}> <X coord_{0}> <Y coord_{0}> <demand_{0}> <a_{0}> <b_{0}> <service time_{0}>
11 <id_{1}> <X coord_{1}> <Y coord_{1}> <demand_{1}> <a_{1}> <b_{1}> <service time_{1}>
12  ...
13 <id_{n}> <X coord_{n}> <Y coord_{n}> <demand_{n}> <a_{n}> <b_{n}> <service time_{n}>
```

Figure 5.1: Structure of Solomon's datafiles.

5.3 Geographical distribution of customers

Solomon's instances are divided in three subsets according to the geographical location of the customers: $CXXX$ (clustered), $RXXX$ (uniformly distributed) and $RCXXX$ (mixed). Figure 5.2 shows the position of the customers located in clusters. On the top of this figure, we see the location of customers in instances within subsets $C1XX$. This sub-set shows 10 clear customer cluster patterns. The optimal solution for each of these instances have 10 routes.

Figure 5.2 also shows the location pattern for customers in the instance subset $C2XX$. In this pattern, clusters are not so clear, as some customers are mid-way among different groups. For both subsets positions (x, y) range in $x : [0, 100]$ and $y : [0, 90]$.

Figure 5.3 shows the location of randomly spread customers in subset $RXXX$. Both subsets $R1XX$ and $R2XX$ have the same layout and only time windows are different. Positions (x, y) are in the range $x : [0, 70]$ and $y : [0, 80]$.

Figure 5.4 depicts the layout of customers in the subset $RCXXX$. This subset presents customers in cluster and randomly spread. Like in subset $RXXX$, customers in $RC1XX$

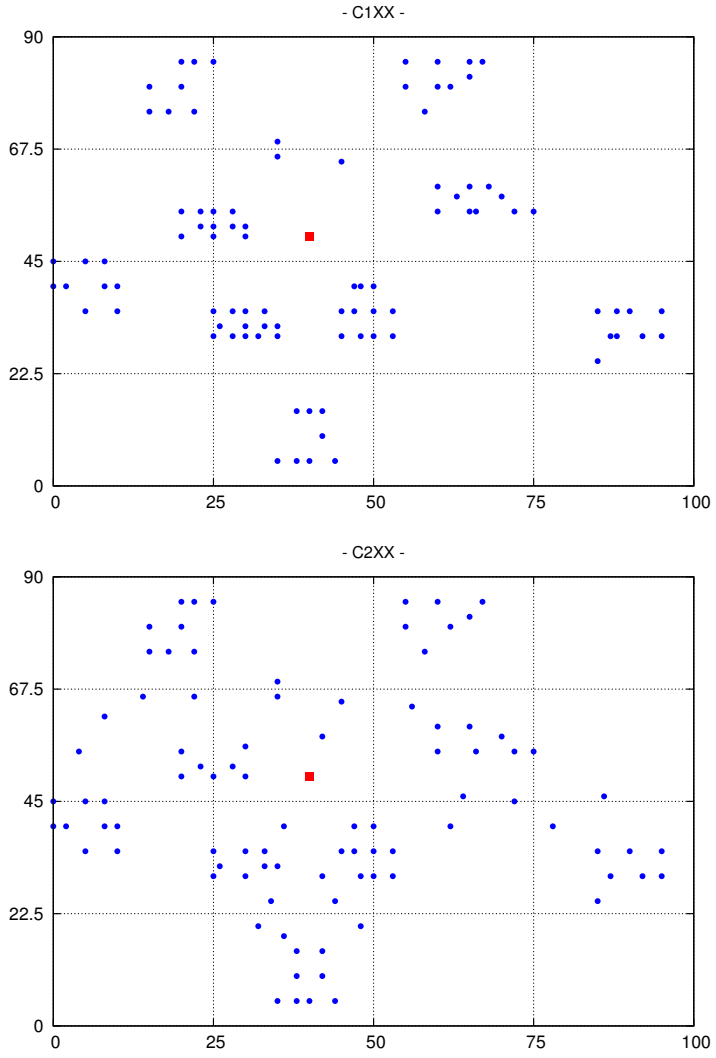


Figure 5.2: Geographic representation of Solomon's clustered datasets - C1XX (above) and C2XX (bellow) both with 100 customers. The red square represents the depot.

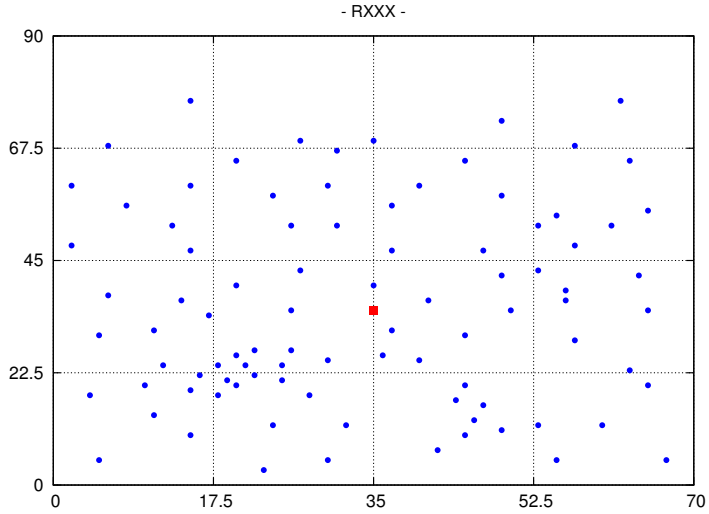


Figure 5.3: Geographic representation of Solomon's datasets - RXXX with 100 customers. The red square represents the depot.

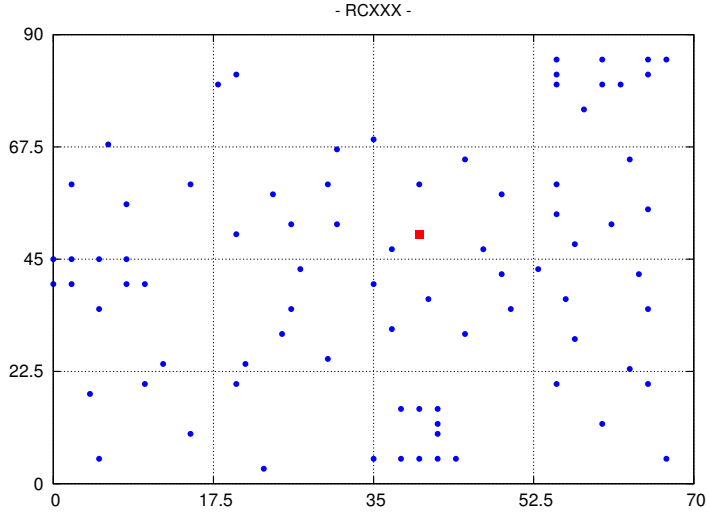


Figure 5.4: Geographic representation of Solomon's datasets - RCXXX with 100 customers. The red square represents the depot.

and $RC2XX$ have the customers located the same the positions. These positions (x, y) are in the range for $x : [0, 100]$ and $y : [0, 90]$.

Solomon's instances cover several possible realistic scenarios with respect to the location and distribution of customers: clustered (C), randomly spread (R) and a combination of both (RC). However, two unrealistic features in these instances are:

- The *travel distance* (or just *distance*) between each pair of customers are not given. Euclidean distances are often used to calculate these distances. Given two customers c_i located at (x_i, y_i), and c_j located at (x_j, y_j), the distance between c_i and c_j is calculated with l_2 norm (Equation 5.3.1).

$$l_p = \sqrt[p]{|x_i - x_j|^p + |y_i - y_j|^p} \tag{5.3.1}$$

  This assumes that: 1) there is a way to go from c_i to c_j in a straight line, and 2) the distance from c_i to c_j is the same as from c_j to c_i, that is, $d_{ij} = d_{ji}$. These two assumptions are unrealistic. Alternatives could be to: 1) use another way to measure distances (e.g. the city block distance), and 2) multiply d_{ij}, d_{ji} or both by random numbers greater than 1, respectively. The city block distance (also known as Manhattan distance) corresponds to the l_1 norm (Equation 5.3.1). This is similar to the way one moves in a city, and is therefore more realistic. The l_p norms with p between 1 and 2 have been shown to be more realistic than the Euclidean distance. For these values of p, l_p produces length values halfway the Manhattan distance (the distance in x plus the distance in y) and the Euclidean distance (the distance in straight line).

  In addition, if we multiply any of the distances or both by random number greater than 0, then we can get $d_{ij} \neq d_{ji}$. It is worth mentioning that this multiplication step must consider the triangular inequality. The triangular inequality states that for any three customers c_i, c_j and c_k, the sum of the distances corresponding to any two pair of customers must be greater than or equal to the distance between the remaining pair of customers, that is, $d_{ij} + d_{jk} \geq d_{ik}$.

- Solomon's datasets have all customers located within an area that forms an almost perfect squares. Customers are located in 1) $x : [0, 100]$ and $y : [0, 90]$ in $CXXX$ dataset, and in 2) $x : [0, 70]$ and $y : [0, 90]$ in $RXXX$ and $RCXXX$ datasets. Besides, the depot is located near the centre of the square in all instances. A more realistic scenario should consider not only different layouts in the distribution of customers but also different shapes of the enclosing geographic areas. For example, customers could be spread in diagonal and the depot could be located at $(0, 0)$. This could represent a more challenging scenario regarding the customers location.

5.4 Time Windows and Service Times

Time windows are periods of time in which the delivery vehicles must serve the customers. Service times indicate the duration of the actual delivery, once the vehicle has arrived at a customer's location. In this section, we discuss the main characteristics of time windows and service times in the Solomon's instances.

Solomon's dataset does not provide the travel times. Instead, it is assumed that a unit of distance is equal to a unit of time. This assumption is unrealistic due to many factors, such as traffic jams and low-speed roads. A way to overcome this problem is to generate a travel time matrix by using the previously calculated travel distance matrix. Thus, the travel time between each pair of customers c_i and c_j, denoted as t_{ij}, could be equal to $t_{ij} = d_{ij} * r$ (where d_{ij} is the travel distance from c_i to c_j, and r is a random number greater than 0). A similar procedure was carried out by Russell et at. [218]. The authors calculated the travel time by using a linear function that allows the velocity to vary in function of distance. The new generated travel time matrix must also satisfy the triangular inequality explained in Section 5.3. We believe that having travel distance and travel time matrices not equal between them but also non-symmetric for each pair of customers would present a more challenging and realistic scenario for the assessment of VRPTW solving techniques.

Regarding service times, all customers in the subset $CXXX$ have 90 units of time. While

customers in subsets $RXXX$ and $RCXXX$ have 10 units of time.

We show the representation of the time windows for each customer in a number of figures where the x-axis denotes the customer id and the y-axis represents the length of the time window. The first customer is always the depot, with the largest time window. Figures A.1, A.2 and A.3 (see Appendix) depict the time windows for three pairs of instances with 100 customers in Solomon's dataset. Solomon's instances subsets $C1XX$, $R1XX$ and $RC1XX$ have short scheduling horizons, $C1XX$ ranging in $(0, 1300)$ (see Solomon's C101 in Figure A.1 on the left) while $R1XX$ and $RC1XX$ ranging in $(0, 250)$ (see Solomon's R101 and RC101 in Figures A.2 and A.3 on the left). Subsets $C2XX$, $R2XX$ and $RC2XX$ have longer scheduling horizons, $C2XX$ ranging in $(0, 3400)$ (see Solomon's C201 in Figure A.1 on the right) while $R2XX$ and $RC2XX$ ranging in $(0, 1000)$ (see Solomon's R201 and RC201 in Figure A.3 on the right).

We calculated a number of statistics to analyse the time windows across all instances. We computed means and standard deviations for the *opening times*, *closing times*, *time window amplitudes* and *time window centres*. However, we did not find a pattern in these features. Even in the same subset, there is a high variability across time windows from one instance to another.

We observe that in general, time windows are narrower in instances in subsets $C1XX$, $R1XX$ and $RC1XX$ than those in $C2XX$, $R2XX$ and $RC2XX$. Instances with narrow time windows have fewer feasible solutions and waiting times can be high. This is because time windows are so restrictive that feasible solutions can be only archived if customers are visited in a certain sequence. The extreme case is that of customers with such narrow time slots that only one sequence leads to a feasible solution. The opposite occurs in instances with wide time windows. These instances have customers with more flexible time windows, and therefore there is a larger number of feasible solutions. We find three types of time windows across all instances in Solomon's dataset: 1) instances with only narrow time windows, 2) instances with only wide time windows, and 3) instances with a combination of narrow and wide time windows. Some instances with only narrow time windows are: $C101$, $C201$, $R101$, $R201$, $RC101$ and $RC201$. In these instances, the feasible solution space is so restricted that the problem

can be often reduced to a scheduling problem. Some instances with only wide time windows are: $C109$, $C208$, $R109$, $R211$, $RC108$ and $RC208$. Instances with wide time windows can be often tackled by using Capacitated VRP solution methods. Other instances that combine narrow and wide time windows include: $C103$, $C104$, $C203$, $C204$, $R103$, $R104$, $RC203$, $RC204$. These instances are interesting as most customers have time-slots with the same length as the depot. This way, they can be scheduled on any position of the service sequence.

The information regarding the time windows of customers provides valuable knowledge. According to the type of instance (narrow/wide/mixed time windows), different techniques could be applied to produce better results. This information can be used to develop search methods that not only take into account the travel distance, but also the setting of time windows in the problem instance.

5.5 Demands and Vehicles Capacities

In Solomon's dataset, each customers demand takes a discrete value within $\{10, 20, 30, 40\}$. All instances have a maximum number of available vehicles available equal to 25. Table 5.1 summarises the value of demands for Solomon's subsets: $C1XX$, $C2XX$, $R1XX$, $R2XX$, $RC1XX$ and $RC2XX$. The first column shows the name of the subset. The second column specifies the *Total Demand* or sum of demands across all customers in the same instance. The third column states the *Vehicle Capacity*. The fourth column shows the *Ratio* between *Total Demand* and *Vehicle Capacity*. This ratio represents the minimum number of vehicles needed to serve all customers if we do not take into consideration their time windows. The maximum capacity of the delivery vehicles is 200 for instances in subsets $C1XX$, $R1XX$ and $RC1XX$; 1000 for $R2XX$ and $RC2XX$; and 700 for instances in subset $C2XX$. Summing up customers' demands of individual instances, we find a total demand of 1080 in $C1XX$ and $C2XX$, 1458 in $R1XX$ and $R2XX$, and 1724 in $RC1XX$ and $RC2XX$. The ratio between the total demand and the vehicle capacity provides some good approximations to the optimum number of vehicles in the solution for each instance. For example, the average optimum number of vehicles

Table 5.1: For subsets *C1XX*, *C2XX*, *R1XX*, *R2XX*, *RC1XX* and *RC2XX*, this table shows the *maximum capacity* of each vehicle, *total demand* or sum of demands of all customers and *ratio* between the *Total Demand* and *Vehicle Capacity*.

Subset	Total Demand	Vehicle Capacity	Ratio
C1XX	1080	200	9.05
C2XX	1080	700	2.58
R1XX	1458	200	7.29
R2XX	1458	1000	1.46
RC1XX	1724	200	8.62
RC2XX	1724	1000	1.74

in instances in subset *C1XX* is 10, and the ratio is 9.05. The same occurs in *C2XX*, in which the average optimal value is 3 and the ratio is 2.58. We find larger differences between the ratio and the actual optimal number of vehicles for instances in subsets *R1XX* and *RC1XX*, with average optimal values 13 and 12, while ratios ratios 7.29 and 8.62, respectively. Mid-way gap values are found in instances in subsets *R2XX* and *RC2XX*, with average optimal values of 3 vehicles in both, and ratios of 1.46 and 1.74, respectively.

5.6 Multi-objective Suitability

We define the *multi-objective suitability* of a dataset for the VRPTW based on the works proposed by Purshouse and Fleming [204], Wegman [257] and Li et al. [163]. Given two datasets: D_1 and D_2, we say that D_1 is more *multi-objective suitable* compared to D_2, if the correlation values between pairs of objectives obtained by a standard Evolutionary Multi-Objective Algorithm (EMOA) are closer to -1 or 1 for D_1 than for D_2. In this case, D_1 would present a more realistic and challenging scenario, and the multi-objective assessment using this dataset would be therefore more reliable.

In the literature, the vast majority of algorithms proposed to tackle the multi-objective variant of VRPTW have been assessed using Solomon's dataset. However, this dataset was not designed to test multi-objective algorithms and its suitability has been recently

questioned by Garcia-Najera and Bullinaria [105] and Tan et al. [238]. Aspects that might make this dataset inappropriate for multi-objective assessment are:

- Equality between distance and time matrices. Since Solomon's dataset only provides the position of the customers (x, y), Euclidean distance must be used to calculate travel distances and travel times. This way, for instances with wide time windows, the only difference between the total travel time and the total travel distance is the sum of service times to the travel time. For instances with narrow time windows, the difference relies on the interaction between the waiting time and travel time. Other objectives related to the travel time, such as *makespan* (or travel time of the longest route), might be also affected for the same reasons.

- Symmetry in distance and time matrices. In Solomon's benchmark problems, it is assumed that both the travel time and travel distance from a customer A to another customer B is the same no matter in which order they appear. This might also affect the difficulty of the problem because changing the order of a pair of customers (which might be beneficial to improve another objective - (e.g. waiting time)) does not affect the travel distance. This may occur if, for example, the customers in the route (c_1, c_2, c_3) are visited in the opposite order (c_3, c_2, c_1). The travel distance of both sequences are equal, yet the value of other objective (e.g. waiting time) might be different.

5.7 Difficulty

Referring to difficulty as the time it takes to achieve optimal solutions, Solomon's dataset does not seem to present a high-complexity in modern computation. On one hand, CPLEX 11.1 [139] finds optimal solutions for some Solomon's instance in less than an hour. On the other hand, using state-of-the-art algorithms like MACS-VRPTW, optimal solutions are achieved in less than half an hour [104].

5.8 Conclusions

This chapter provides an analysis of the most important characteristics of the Solomon's dataset. We have observed a number of unrealistic features regarding: 1) the position of customers and depot, 2) the travel distances, and 3) the travel times. The first assumption concerns the location of customers and depot. Customers are located in almost a perfect square in which the depot is always close to its centre. Assumptions two and three are related to the way travel distances and travel times are calculated. Solomon's dataset does not provide a matrix for travel distances and travel times. Instead, the Euclidean distance measure is often used to calculate both using the customers' positions (geographical coordinates). This assumes that: 1) there is a way to go from one to another customer in straight line, and 2) travel distances and travel times between a pair of customers are equal regardless of the moving direction. These assumptions hardly occur in reality. Alternatives could be to: 1) create new instances with different geographic distributions of customers, and 2) create non-equal and non-symmetric travel distance and travel time matrices.

According to the characteristics observed in Solomon's dataset and based on the study by Purshouse and Fleming [204], we think this dataset might not be suitable for the assessment of multi-objective algorithms. This hypothesis is based on 1) the lack of differences between travel distances and travel times from one to another customer, and 2) the assumption that travel distances and travel times between pairs of customers are symmetric.

The next chapter proposes a novel dataset for the assessment of Multi-Objective Vehicle Routing Problem with Time Windows (MOVRPTW). Unlike Solomon's dataset, MOVRPTW has different travel distances and travel times, and they are non-symmetric. The next chapter analyses the characteristics of this dataset following a similar structure to the one in this chapter.

CHAPTER 6

A Dataset for the Multi-Objective Vehicle Routing Problem with Time Windows

Summary

This chapter introduces a new set of benchmark instances: MOVRPTW dataset. This project has two aims:

- It provides a public dataset for the assessment of multi-objective algorithms for the Vehicle Routing Problem with Time Windows (VRPTW).
- It presents a more challenging scenario in VRP(TW) benchmarks to motivate further improvement in the development of solving methods.

This dataset is based on real information provided by a distribution company. The structure of the test instances is similar to the one of Solomon's but the MOVRPTW instances provide travel distance and travel time matrices. This chapter gives an overview of the main characteristics of the MOVRPTW dataset using the same organisation as the previous chapter.

6.1 Introduction

The previous chapter provides an overview of datasets proposed for the assessment of the Vehicle Routing Problem with Time Windows (VRPTW). In particular, we analysed some important characteristics of the Solomon's dataset. This dataset presents a number of unrealistic features due to:

- Travel distance and travel time matrices are not provided. The positions of the customers are used to calculate both. As a result, it is assumed that:
  - There is a straight path that connects any pair of customers.
  - A unit of time is equal to a unit of distance.
  - Travel distances and travel times are the same regardless of the moving direction.
- Customers are place within the boundaries of a square, and the depot is very close to the centre.

This chapter presents a new set of problem instances which do not present the above-mentioned issues. This new dataset is based on data from a distribution company in Tenerife, Spain. The company delivers food products and serves more than 1000 customers overall, with around 150 customers being served each day. Realistic data for the travel distance and travel time between each pair of customers was obtained using Google Maps database. Travel distance and travel time matrices are distinct and non-symmetric, hence representing a realistic trade-off between travel distance and travel time. For example, for pairs of customers located within an urban area, travel time is high compared to the corresponding distance, reflecting the fact that travelling in urban areas is more time consuming than travelling in rural areas. Moreover, higher differences are found comparing travel times and travel distances due to difficult orography in the island. Time windows specifications were generated according to some information provided by the company. While demand specifications were established using a number of parameters in order to present different scenarios. We believe that

the features of this dataset make a better challenge for the assessment of multi-objective algorithms.

The remaining of this chapter contains the following sections. Section 6.2 explains the structure of the different files in each instance. Section 6.3 shows the details of customers' locations layouts. Time windows characterisation are stated in Section 6.4. Section 6.5 explains the demands characterisations. The multi-objective suitability of this dataset is discussed in Section 6.7. Finally, a brief in-sight on the difficulty of this dataset is provided in 6.8.

6.2 Structure of data files

Each instance in the MOVRPTW dataset has three associated files:

- `(instance name)DistanceMatrix.dat` contains the travel distances between all pairs of customers.
- `(instance name)TimeMatrix.dat` contains the travel times between all pairs of customers.
- `(instance name)Specs.dat` contains other information of the instance: max size of the fleet, capacity of the vehicles, location of the customers, etc. This file has the same structure as the Solomon's dataset files (Figure 6.1). The first line contains the name. The second line is always a blank space. The third and the fourth lines contain the keywords 'VEHICLE' and 'NUMBER CAPACITY' respectively. The following line specifies the maximum number of vehicles available and the maximum capacity of each of those vehicles. From 10^{th} line onwards, the main characteristics of customers are specified in several columns. The first line is always reserved for the depot. The first column states the id of the customers (0 for the depot). The second and the third columns specify the location of customers ($latitude, longitude$). The fourth column denotes the demand for each customer. The fifth and sixth state the time windows and the last column the service times.

```
<name of the instance>
<blank>
<keyword: VEHICLE>
<keyword: NUMBER> <keyword: CAPACITY>
<number of available vehicles> <capacity of each vehicle>
<blank>
<keyword: CUSTOMER>
<keywords for each column>
<blank>
<id_{0}> <X coord_{0}> <Y coord_{0}> <demand_{0}> <a_{0}> <b_{0}> <service time_{0}>
<id_{1}> <X coord_{1}> <Y coord_{1}> <demand_{1}> <a_{1}> <b_{1}> <service time_{1}>
 ...
<id_{n}> <X coord_{n}> <Y coord_{n}> <demand_{n}> <a_{n}> <b_{n}> <service time_{n}>
```

Figure 6.1: Structure of MOVRPTW data files.

6.3 Geographical distribution of customers

Recall that Solomon's instances are divided into three subsets according to the geographical location of customers: CXXX (clustered), RXXX (uniformly distributed) and RCXXX (a combination of clusters and uniformly distributed). The MOVRPTW instances have customers located in clusters and randomly spread (equivalent to Solomon's RCXXX).

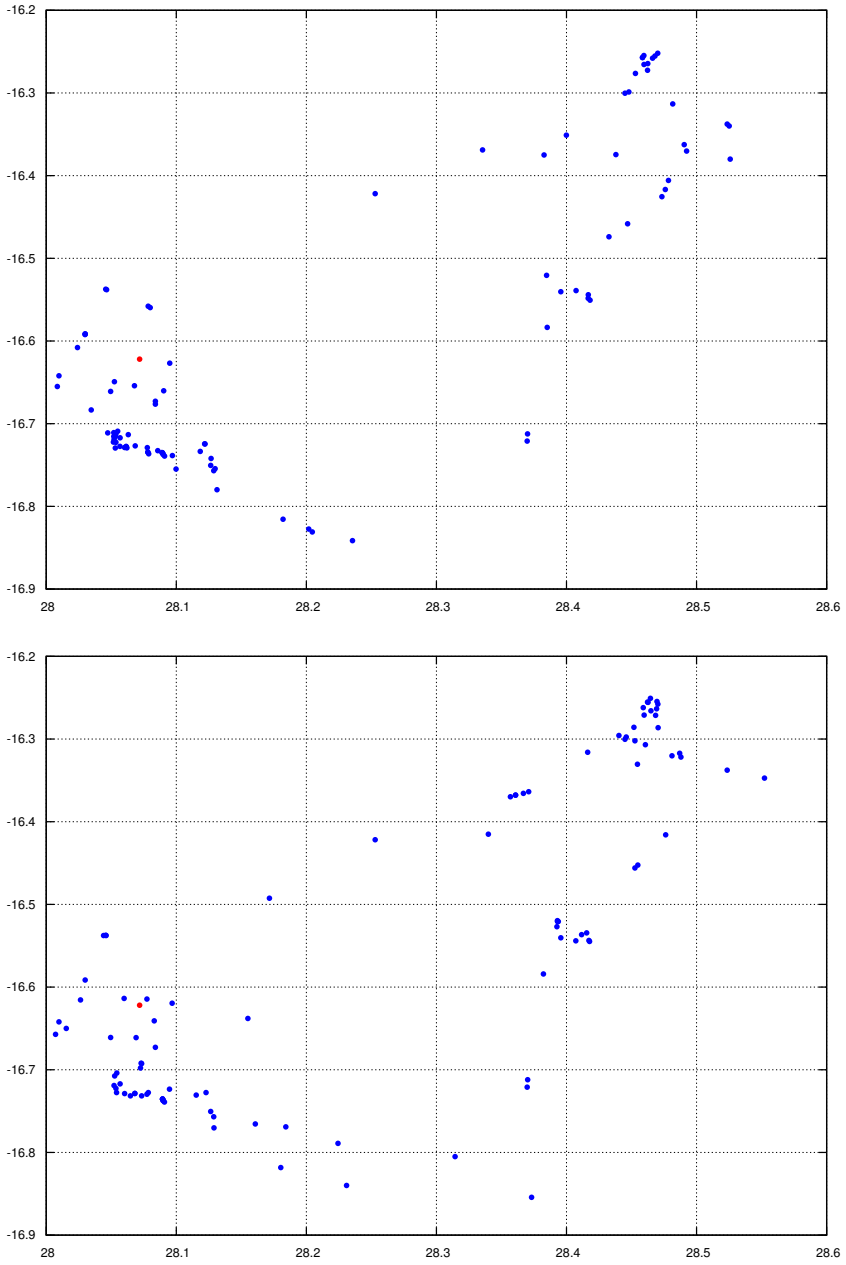
Solomon's dataset has customers located in four different layouts: $\{C1XX, C2XX, RXXX, RCXXX\}$. The MOVRPTW dataset has customers located in two different layouts (see Figure 6.2). From a pool of 1000 customers, we have randomly selected 100 for each layout, using two randomly chosen seeds: 0 and 10 in a custom dataset generator [34].

Some important features of the MOVRPTW dataset regarding the location of the customers are:

- Both layouts have two well-defined clusters and some customers randomly spread. These two clusters correspond to (1) customers located in the capital (upper right corner) and (2) customers located in touristic areas (lower left corner). In these two areas, travel distances and travel times are different due to congestions and low-speed roads.

- Customers are not located within a square geographic area, but they are unevenly located within the rectangle given by $latitudes : [28, 28.6]$ and $longitudes : [-16.9, -16.2]$. Besides the depot is not located at the centre of the layout, but at

Figure 6.2: Layout of the costumers' locations. Distribution of costumers using the layout seed 0 (above) and using the layout seed 10 (bellow). Blue dots indicate the customers while the red dot is the depot. Showing lines of latitude (Y-axis) and longitude (X-axis).



the position given by latitude / longitude: $(-16.78, 28.07)$.

- Travel distances have been obtained using Google Maps database. These correspond to the real road distances which, due to the complex orography of Tenerife, are not equal to the length given by the Euclidean distance. Furthermore, the distances were calculated in both moving directions, so it is often the case that the distance from a customer c_i to a customer c_j is not the same as from c_j to c_i, as these values depend on factors, such as average transit vehicle speeds.

The above-mentioned features present a more realistic scenario to that of the Solomon's dataset. These features create a complex scenario which is more suitable for the assessment of multi-objective algorithms.

6.4 Characterisation of Time Windows

The time window specifications for each customer have been designed to imitate what the delivery company faces everyday. Customers have similar time windows profiles as their commercial activities involve the same opening times. Examples of customers are: restaurants, cafes, hotels, retailers, etc. The depot operates 8 hours (time window) a day. In the first time windows profile (Profile 1 - Fig. 6.3), all customers are available all day (8 hours = 480 minutes). In the following three profiles, we distinguish three types of customers, early customers (those who want to be served in the morning), midday customers (those who want to be served at midday) and late customers (those who want to be served the latest). In order to cover the whole day with these three types of customers, we create time windows with a length of: 8 hours (480 minutes) divided into 3 types of customers = 160 minutes/type of customer, as seen in Profile 2 - Fig. 6.3. In this profile, early customers will be served in the time window $[0, 160]$ minutes. Midday customers will be served in the time window $[160, 320]$ minutes. Late customers will be served in the time window $[320, 480]$ minutes. For the third and fourth profiles, we follow a similar approach. However, we decrease the length of each time window by 30 and 60 minutes respectively. Thus, in the third profile (Profile 4 - fig. 6.3), a time window has a length of 130 minutes. So, the opening hours will

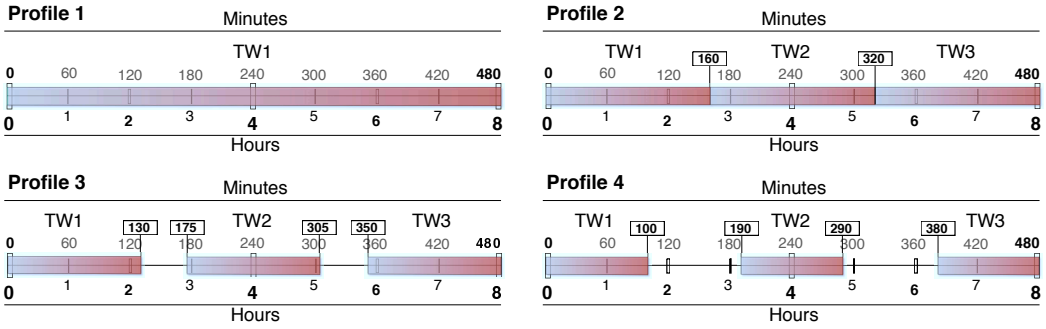


Figure 6.3: Four of the time windows profiles. This figure shows the opening times and closing times for each profile.

be: early customers $[0, 130]$, midday customers $[175, 305]$ and late customers $[350, 480]$. Time windows of midday customers are symmetric with respect to the midday. For example, if the length of a time window is 100, it will grow 50 to the left and 50 to the right respect to the central point 240 (4th hour). The fourth profile (Profile 4 - Fig. 6.3) has time windows with length of 100 minutes. Therefore, the time windows will be: early customers $[0, 100]$, midday customers $[190, 290]$, and late customers $[380, 480]$. In a fifth time windows profile (Profile 5), customers are given one of the 10 time windows types in the previous profiles (see Fig. 6.3). That is, Profile 5 will contain the only time window type of the Profile 1, three time window types of Profile 2, 3 and 4.

We suggest 5 time windows profiles for the dataset. Each profile, except the first one, has three classes of time windows. Given a time window profile, a customer has the same probability of having any of the time windows within that profile.

These time windows profiles are designed: 1) to cover a wide range of scenarios as in Solomon's dataset, and 2) to present a realistic scenario based on the information provided by the company.

Regarding the travel times, Google Maps database was used to calculate the time that it takes to go from any customer to another. These travel times are based on reliable data provided by GPS device manufacturers. As with the travel distances, it is often the case that the travel time from a customer c_i to a customer c_j is not the same as from c_j to c_i. The travel time between a pair of customers depends on many factors, such as traffic and road speed limits, and they might not be the same for both moving directions.

6.5 Characterisation of Demands

Let p_i be the demand of customer i and Q the capacity for each vehicle. This capacity of each vehicle Q is bounded by $\underline{D} = \max_i\{p_i\} \leq Q \leq \overline{D} = \sum_{i=1}^{n} p_i$, where n is the number of customers. The lower bound is the maximum demand $\underline{D}$ among the customers. And the upper bound is the sum of all customers' demands $\overline{D}$. The closer Q is to its lower bound, the more constraint each vehicle will be with respect to the total demand. Conversely, the larger is Q, the more spare capacity each vehicle will have.

Parameter δ (delta) is used to modulate the slack margin of an instance, $Q = \underline{D} + (\delta/100)(\overline{D} - \underline{D})$, where $\delta \in (0, 100]$. If δ takes values close to 0, the capacity of the vehicle Q will be very limited. On the other hand, for large values of δ, the vehicles will have a capacity Q close to the total demand.

Regarding the fleet size, the maximum number of vehicles needed is equal to the total customers' demand divided by the vehicle capacity. However, since $Q \geq \underline{D}$, an upper bound for the size of the fleet m would be: $m \leq \overline{D}/\underline{D}$.

6.6 Dataset Settings

Based on the guidelines mentioned in this section, we have created our benchmark dataset using the following combinations:

- Number of Customers: 100. MOVRPTW consists of instances with 100 customers, similar to those of Breedam's, Cordeau's and Solomon's datasets (see Section 5.1).
- Time Windows: Profiles $\{1, 2, 3, 4, 5\}$, within each profile, the same probability was assigned to each time window type. For example, a probability of $1/3$ was assigned to each time window type in Profiles $\{2, 3, 4\}$. While a probability of $1/10$ was used in Profile 5 for each type, since it has 10 time window types. In order to create various problem scenarios, we assigned the same probability to each time window type.
- Customer Demand: three values $\{10, 20, 30\}$, each with probability $1/3$ and three

types of $\delta = \{60, 20, 5\}$. These values are not based on real data. We established these values to have three standard scenarios in which the slack margin is high ($\delta = 60$), normal-low ($\delta = 20$), and very tight ($\delta = 5$).

- Service Times: three values $\{10, 20, 30\}$ minutes, each with probability $1/3$. These values are based on real data provided by the company. The service time depends on a number of factor such as the customer activity (hotel, cafe, etc), its location and the time of the day. We assigned the same probability to all service time in order to create general scenarios.

- Seeds: two groups of instances were created with two different sets of seeds $\{0, 0, 0, 0\}$ and $\{10, 7, 5, 1\}$. These seeds were randomly chosen.

Summing up, a total of 30 MOVRPTW instances were generated (1 size * 5 time windows profiles * 3 deltas * 2 groups of seeds). This dataset and the configuration files can be download from [34].

6.7 Multi-objective Suitability

As in the previous chapter, we consider that given two datasets: D_1 and D_2, D_1 is more *multi-objective suitable* compared to D_2, if the correlation values between pairs of objectives obtained by a standard Evolutionary Multi-Objective Algorithm (EMOA) are closer to -1 or 1 for D_1 than for D_2.

Aspects that might make this dataset appropriate for multi-objective assessment are:

- No equality between travel distance and travel time matrices. Information about travel distances and travel times have been obtained from a reliable source (Google Maps database). This is important because conflicting pair wise relationships may arise in comparisons involving travel distance and objectives related to travel time (e.g. makespan − or travel time of the longest route). For example, consider a route in which travel distance is minimum and the makespan is maximum because the delivery vehicle has to go through urban areas.

- No symmetry in travel distance and travel time matrices. In this dataset, the travel time and travel distance between a pair of customers A and B are not necessarily the same in both directions. This might increase the difficulty of the problem in several objectives, as changing a sequence of customers require the re-calculation of the whole sub-route. For example, consider the route (c_1, c_2, c_3) are visited in the opposite order (c_3, c_2, c_1). The travel distance of both sequences are equal, yet the value of other objective (e.g. waiting time) might be different.

6.8 Difficulty

We refer to difficulty as the time that it takes to achieve optimal solutions. We ran a number of experiments on CPLEX 11.1 [139] towards the optimisation of travel time or travel distance. CPLEX failed to produce optimal solutions in all instances in a modern desktop PC (Core2Duo, 4GB). In all tests, CPLEX created over-sized search trees exceeding the memory capacity of the machine. This is because these instances have wide time windows which makes the feasible region larger.

due to the large feasible spaces as a consequence of the large time windows.

Some preliminary tests have been carried out with a Multi-Objective Particle Swarm Optimisation and the Non-Dominated Genetic Algorithm-II (NSGA-II). Chapter 8 discusses the results obtained by both multi-objective optimisers in this new dataset.

6.9 Conclusions

This chapter presents the MOVPTW dataset, a new set of benchmark problems for the Vehicle Routing Problem with Time Windows (VRPTW). This dataset is designed to overcome unrealistic features found in Solomon's dataset. Unrealistic features in Solomon's dataset are related to: 1) the location of the customers, and 2) the absence of travel distances and travel times matrices.

The proposed MOVRPTW dataset is based on real data from a distribution company. Some customers are located in two regional clusters and other in the vicinity of such

clusters. The depot is not located at the centre of the layout (as many existing test instances), but it is in the west-south region of the map. We calculated travel distances and travel times matrices using Google Maps. These matrices are non-equal and non-symmetric. Furthermore, based on realistic data we designed five time windows profiles from which specific time windows can be assigned to customers. The capacity of the vehicles was characterised by using a slack margin parameter (δ). We used this parameter to create instances in three different scenarios: 1) high slack margin ($\delta = 60$), 2) normal-low slack margin ($\delta = 20$), and 3) very tight slack margin ($\delta = 5$). Summarising, the MOVRPTW dataset consists of 30 instances: 5 time windows profiles, 3 values for δ and 2 groups of seeds. According to these criteria, we think that MOVRPTW dataset presents a more realistic scenario than the one by Solomon's dataset.

We also think that the MOVRPTW dataset might be suitable for the assessment of multi-objective algorithms. This hypothesis is based on: 1) the differences between travel distances and travel times from one to another customer, and 2) the non-symmetry between travel distances and travel times between pairs of customers.

The next chapter provides an in-depth study on the multi-objective suitability of both MOVRPTW and Solomon's dataset based on the work by Purshouse and Fleming [204]. According to the authors, relationships can be of harmony (if the optimisation of one objective leads to the improvement of the other), conflict (if the optimisation of one objective leads to the worsening of the other) and independence (if the optimisation of one objective does not affect the other). This way, we study the conflict, harmony and independence of five common objectives among non-dominated solutions obtained by NSGA-II in both datasets. This study provides the means to understand the relationship between these five objectives, and to discern which benchmark dataset is better for the assessment of multi-objective algorithms.

CHAPTER 7

Investigating the Multi-objective Suitability of VRPTW datasets

Summary

Chapter 5 provides an overview of the characteristics of the Solomon's dataset for the Vehicle Routing Problem with Time Windows (VRPTW). Preliminary observations seemed to indicate that these test instances might not be useful for multi-objective benchmarking. The insights of this study led us to generate some problem instances using data from a real-world distribution company. Chapter 6 presented the characteristics of this new dataset. Features such as 'distinct and non-symmetric travel distance and travel time matrices' support the hypothesis that this new dataset might be better suited for the multi-objective assessment of the VRPTW.

This chapter presents an experimental study that compares the multi-objective suitability of both datasets based on the work by Purshouse and Fleming [204]. Experiments based on this work show that our dataset has stronger multi-objective features. This chapter focuses on achieving a better understanding of the multi-objective nature of the VRPTW. In particular, we study the conflicting relationships between 5 objectives: number of vehicles, total travel distance, makespan, total waiting time, and total delay time.

7.1 Introduction

This chapter presents a study based on the work by Purshouse and Fleming [204]. They indicate that three main relationships may occur between pairs of objectives: *conflict*, *harmony* or *independence*. If there is a dependence between the objectives, they can be: 1) *conflicting* (if it is not possible to improve one without worsening the other), or 2) *harmonious* (the improvement in one witnesses an enhancement in the other). Conversely, if the optimisation of one objective does not affect the other, the relationship is of *independence*. When conducting multi-objective optimisation benchmarking, the most important relationships are that of dependence. Multi-objective problems with independent relationships among their objectives can be addressed by decomposing the problem into sub-problems [204]. However, dependent relationships present a real challenge to multi-objective algorithms. In general, the more conflicting objectives exist in a given problem, the more suitable this problem is for benchmarking this type of algorithm. Purshouse and Fleming also state the importance of keeping harmonious objectives in the optimisation process because, for example, this might provide additional knowledge to the decision maker.

The work by Purshouse and Fleming also provides an overview of qualitative and quantitative methods for the analysis in multivariate studies. They mention two important qualitative methods: 1) Parallel coordinates plot (P-plots) and 2) Scatterplot (S-plot). P-plots are a common way to analyse multivariate data. In order to show vectors with n criteria, n parallel axes are drawn vertically and equally-spaced. A vector of criteria is represented by a polyline with vertices on the axes. Thus, if lines connecting a pair of axes cross, there is a conflict between the pair of criteria associated to those axes. Conversely, those lines fail to cross if there is harmony. It is possible to visually estimate the magnitude of conflict or harmony by observing the number of crossing lines (conflict) or parallel lines (harmony). Some quantitative methods have been proposed to calculate the magnitude of conflict and harmony based on P-plots. Wegman [257] and Li et al. [163] related the P-plots notion of crossing lines to the concept of correlation. This way, the magnitude of conflict or harmony between a pair of objectives is related to their correlation value. If the correlation value is close to -1, then there will

be many crossing lines and the conflict with be high. Conversely, if the correlation values is close to 1, then there will be many parallel lines, and the harmony will be high. If the correlation takes values close to 0, P-plots will show very little interaction between the pair of objectives, which can be interpreted as an independence relationship.

The use of Scatterplots is another method to visualise correlation. In order to show vectors with n criteria, we generate a matrix with n columns and n rows: M_{nxn}. Each element of this matrix represents the trade-off surface between a pair of objectives. For example, $M_{2,3}$ will depict the trade-off surface between the second and the third criteria. Purshouse and Fleming [204] and Wegman [257] pointed out that, on occasions, it is difficult to extract information using S-plots. However, Li et al. [163] conducted two user studies comparing P-plots and S-plots. In these studies 25 participants had to judge the degree of correlation in S-plots and P-plots under different conditions. Results showed that the participants tended to under-estimate correlation in P-plots. The authors concluded that S-plots are better tools than P-plots for visual correlation analysis.

The purpose of this Chapter is twofold: firstly, to analyse the pair-wise relationships that appear throughout the optimisation process, and secondly, to compare the pair-wise relationships that appear in final non-dominated sets. The first part of our study focuses on the analysis of the pair-wise relationships that appear as the search progresses. This part of the study uses correlation to investigate how the optimisation of one objective affect the others. The second part of the study focuses on the understanding of the pair-wise relationships that appear in final non-dominated solution archives obtained by an Evolutionary Multi-objective Optimisation Algorithm (EMOA). We use S-plots and correlation to calculate the magnitude of the pair-wise relationships in the non-dominated sets found with the Solomon's and MOVRPTW datasets. For both studies we investigate the following five minimisation objectives: *number of vehicles* (denoted as Z1) needed to serve all customers, *total travel distance* (Z2), *makespan* (Z3) or travel time of the longest route (from/to depot), *total waiting time* (Z4) of the delivery vehicles, and *total delay time* (Z5) or sum of tardiness for all deliveries. Note that this notation is not the same than that of Chapter 4 because in this study we investigate other

objectives. Furthermore, in this research we investigate only pair-wise relationships, as relationships involving more objectives can be inferred from combined pair-wise relationships [163, 204].

The reminder of this chapter is organised as follows. Section 8.4 describes the experiments settings. Section 7.3 presents the discussion of the results. Conclusions and further work is stated in Section 7.4.

7.2 Experimental Design

The two aims of this Chapter are explored in two studies. The first study uses a Variable Neighbourhood Search (VNS) to investigate which pair-wise relationships appear throughout the optimisation process across all instances in both datasets. We set the VNS to optimise one of the five objectives each time, while monitoring the evolution of the others. We chose to use VNS for this task as it systematically changes only one solution at a time, so that we can track how the objectives change from one to another solution. The second study uses NSGA-II [65] to find out which pair-wise relationships appear in the approximation sets obtained with each dataset. In this case, we chose NSGA-II because we wanted to obtain not one, but a set of non-dominated solutions.

7.2.1 VNS settings

We encode route-plans as sequences of customers identifiers (customers ids). Each sequence represents a route within the route-plan. The first and last element of all sequences is 0, meaning the depart and return from/to the depot. For example, the route-plan (0 2 1 4 0 5 3 6 0 9 8 0) has three routes. The first route is (0 2 1 4 0), the second route is (0 5 3 6 0) and the third route is (0 9 8 0). In the first route, the vehicle departs from the depot 0, then visits customer 2, followed by customers 1 and 4 before returning to the depot 0. The other two sequences in the encoding have similar interpretations.

We designed the VNS algorithm to explore neighbouring solutions by swapping pairs

of customers with distance k within a route-plan. This distance k corresponds to the number of elements in between two other elements. For example, the distance k between the customers with ids 2 and 5 in the route-plan (0 2 1 4 0 5 3 6 0 9 8 0) is 4.

Initially, a route-plan is randomly generated as a list (sequence) of customers ids. VNS swaps pairs of customer ids from $k = 1$ (adjacent customers) to $k = L - 1$ (customers in the extremes of the route-plan), where L is the number of elements (length) of the route-plan. If a better solution is found for a given k, the process starts over by resetting k to 1. This process finishes when no improvement occurs from $k = 1$ to $k = L - 1$. As mentioned before, the improvement is tested according to the fitness value of only one objective.

Figure 7.1 shows how VNS operates when using the route-plan (0 2 1 4 0 5 3 6 0 9 8 0). It first swaps customers with distance k equal to 1. That is, it will swap the pair of customers in positions $i = 1$ and $j = 2$. Next, VNS will swap the pair of customers in positions $i = 2$ and $j = 3$. This will continue until i and j take values 9 and 10, respectively. The process will then start over with k equal to 2. VNS will therefore swap the pair of customers in positions $i = 1$ and $j = 3$. The whole process will finish when k is equal to its maximum value 9. If a better solution was found from $k = 1$ to $k = 9$, the process will start over using this new solution as starting point. If no solution was found, the process will be over.

In the experiments, VNS was run 20 times for each objective and instance of both datasets with the same seeds.

7.2.2 NSGA-II settings

For the NSGA-II development we used the implementation of an Evolutionary Algorithm (EA) for the VRPTW [158] as a starting point. This implementation is based on the optimisation framework ParadisEO-MOEO [25]. We extended this implementation to support multiple objectives, process our dataset and use the NSGA-II.

In this implementation we used a different encoding to that of the VNS. The encoding of an individual for the NSGA-II is a list of routes (a list of lists). Each element in a

```
  i j
0 2 1 4 0 5 3 6 0 9 8 0    i: 1, j: 2 [k = 1]

0 1 2 4 0 5 3 6 0 9 8 0    swap i: 2, j: 3
0 2 4 1 0 5 3 6 0 9 8 0    swap i: 3, j: 4
0 2 1 0 4 5 3 6 0 9 8 0    swap i: 4, j: 5
...
0 2 1 4 0 5 3 6 0 8 9 0    swap i: 9, j: 10

  i   j
0 2 1 4 0 5 3 6 0 9 8 0    i: 1, j: 3 [k = 2]

0 4 1 2 0 5 3 6 0 9 8 0    swap i: 2, j: 4
0 2 0 4 1 5 3 6 0 9 8 0    swap i: 3, j: 5
...
0 2 1 4 0 5 3 6 8 9 0 0    swap i: 3, j: 5

...

  i               j
0 2 1 4 0 5 3 6 0 9 8 0    i: 1, j: 10 [k = 9]
0 8 1 4 0 5 3 6 0 9 2 0    swap i: 1, j: 10 [k = 9]
```

Figure 7.1: Example of VNS running using the route-plan (0 2 1 4 0 5 3 6 0 9 8 0)

list represents a customer. The position of a customer within a list specifies the turn in which he/she will be served.

The population is initialised using a constructive method that aims at satisfying first the customers farthest from the depot. After the initialisation process, all individuals are evaluated. The fitness assignment procedure of NSGA-II is called non-dominated sorting criterion [65]. It consists of dividing the population into non-dominated fronts. This way, the fitness of an individual depends on the depth of its front.

Once the individuals are evaluated, a sub-group is selected for crossover. This process recombines two parents (solutions) with certain probability γ, creating one or two offsprings (new solutions). In this implementation, NSGA-II has 3 standard crossover operators: (1) One-point crossover, (2) Edge crossover and (3) Generic crossover [21].

In the One-point crossover, a random number of consecutive customers are copied from one parent to another, removing duplicates. The Edge crossover consists of (1) constructing new intermediate solutions by joining edges from both route-plans (parents) and (2) merging sub-tours creating feasible solutions. In the Generic crossover, an entire route is copied from one route-plan to another, removing duplicates.

In order to promote diversity within the population, the offspring solutions undergo a mutation operation with certain probability ν. NSGA-II was run using four standard

mutation operators (Swap, Insertion, Inversion and Displacement) [21].

The Swap mutation interchanges the position of two customers within a route-plan. The Insertion mutation consists of moving a random customer to a new position within the route-plan. In the Inversion mutation, customers in a portion of the route-plan are reversed. The Displacement mutation is a generalisation of the insertion mutation in which a number of consecutive customers are moved.

In order to re-use the genetic operators of the previous implementation [158], route-plans were forced to be feasible in terms of vehicle capacity constraints. This process was carried out by splitting routes in which the vehicle capacity was exceeded.

In the experiments, NSGA-II evolved a population of 50 individuals for 10000 generations. We compared Solomon's 100 customer dataset against our dataset with 100 customers. The algorithm is applied to each dataset 20 times (repetitions) with the same parameters and seeds. This new implementation is open source and is available at [34].

7.3 Discussion of Results

The results of the experiments stated in Section 7.2 will be discussed in two sections. In the first subsection, we will analyse the causal relationships of pair of objectives throughout the optimisation process carried out by the VNS (Sect. 7.2.1). In these experiments, we study how the optimisation of a single objective affects the others as the search progresses. In the second subsection, we will focus on the pair-wise relationships among objectives in the non-dominated sets found by the NSGA-II (Sect. 7.2.2) on each dataset.

7.3.1 Correlation Between Objectives throughout the Optimisation Process

As said before, our aim with the VNS was to study the relationships between pairs of objectives throughout the optimisation process. It should be therefore noted that this study does not compare the Solomon's and MOVRPTW datasets.

Table 7.1: Type of pair-wise casual relationships across Solomon's and MOVRPTW datasets. In the second row, conflict relationship is denoted by $\ominus$, harmony is denoted by $\oplus$, and independence is denoted by NA or $approx.0$. In the third row, $\sim$ indicates that causal relationships have the same sign in both directions, while $\neq$ is used to denote the opposite case. Causal relationships with NA or ~ 0 are denoted with $-$.

	[Z1,Z2]	[Z1,Z3]	[Z1,Z4]	[Z1,Z5]	[Z2,Z3]	[Z2,Z4]	[Z2,Z5]	[Z3,Z4]	[Z3,Z5]	[Z4,Z5]
Sign:	$\oplus/\oplus$	$\ominus/NA$	$\oplus/\oplus$	$\ominus/NA$	$\ominus/\oplus$	$\oplus/\ominus,\sim 0$	$\oplus/\ominus,\sim 0$	$\oplus/\ominus$	$\oplus/\oplus$	$\ominus/\oplus$
Rel:	$\sim$	$-$	$\sim$	$-$	$\neq$	$-$	$-$	$\neq$	$\sim$	$\neq$

Table 7.2: Average $\mathcal{C}$ value obtained by VNS for all instances in Solomon's subsets $\{CXXX, RXXX, RCXXX\}$ and MOVRPTW subsets $\{s0, s10\}$ (20 runs). Relationships from Z1→Z2 to Z3→Z2.

Instance	Z1→Z2	Z1→Z3	Z1→Z4	Z1→Z5	Z2→Z1	Z2→Z3	Z2→Z4	Z2→Z5	Z3→Z1	Z3→Z2
C1XX	0.88	-0.77	0.97	-0.88	0.94	-0.61	0.89	-0.70	-0.54	0.91
C2XX	0.76	-0.78	0.98	-0.87	0.92	-0.37	0.89	-0.47	NA	0.83
R1XX	0.88	-0.70	0.93	-0.88	0.95	-0.04	0.69	-0.44	-0.53	0.95
R2XX	0.65	-0.79	0.97	-0.85	0.91	0.22	0.81	-0.24	NA	0.88
RC1XX	0.88	-0.76	0.93	-0.88	0.95	-0.04	0.67	-0.34	-0.61	0.96
RC2XX	0.66	-0.72	0.98	-0.86	0.92	0.05	0.84	-0.32	NA	0.89
s0	0.82	-0.75	0.99	-0.88	0.92	-0.75	0.92	-0.75	-0.71	0.57
s10	0.83	-0.74	0.99	-0.88	0.93	-0.73	0.91	-0.73	-0.73	0.43

We define $\mathcal{C}(Z_x, Z_y)$ as the correlation between the objectives Z_x and Z_y when the optimisation process is carried out only on Z_x. It should be therefore noted that $\mathcal{C}(Z_x, Z_y)$ = $\mathcal{C}(Z_y, Z_x)$ is not necessarily true, as the optimisation is carried out on different objectives. Since this function represents the way the optimisation of one objective affects the other, in this section we will refer to $\mathcal{C}(Z_x, Z_y)$ or $\mathcal{C}(Z_y, Z_x)$ as causal relationships between a pair of objectives. We denote this causal relationship using an arrow, so $\mathcal{C}(Z_x, Z_y)$ is the same as $Z_x \rightarrow Z_y$.

We present a pair of tables which correspond to each type of Solomon's 100 customers dataset $\{CXXX, RXXX, RCXXX\}$ and MOVRPTW dataset $\{s0, s10\}$ (Tables 7.2 and 7.3). Each table shows in its first column the name of the subset. Each of the remaining columns represent the average value of the *causal relationship* or (just *relationship*) across all instances. As said before, we work with five objectives: $Z1$ is the *number of vehicles*, $Z2$ is the *travel distance*, $Z3$ is the *makespan*, $Z4$ is the *waiting time* and $Z5$ is the *delay time*. For example, the column Z1→Z2 shows the average $\mathcal{C}$ values between the *number*

Table 7.3: Average $\mathcal{C}$ value obtained by VNS for all instances in Solomon's subsets $\{CXXX, RXXX, RCXXX\}$ and MOVRPTW subsets $\{s0, s10\}$ (20 runs). Relationships from Z3→Z4 to Z5→Z4.

Instance	Z3→Z4	Z3→Z5	Z4→Z1	Z4→Z2	Z4→Z3	Z4→Z5	Z5→Z1	Z5→Z2	Z5→Z3	Z5→Z4
*C*1*XX*	0.67	0.98	0.91	0.27	-0.59	-0.72	NA	0.68	0.97	0.79
*C*2*XX*	0.41	0.97	0.92	-0.43	-0.59	-0.57	NA	0.50	0.94	0.57
*R*1*XX*	0.64	0.98	0.87	0.25	-0.50	-0.68	NA	0.87	0.98	0.89
*R*2*XX*	0.54	0.83	0.93	-0.45	-0.48	-0.51	0.85	0.42	0.96	0.61
*RC*1*XX*	0.52	0.98	0.89	0.34	-0.63	-0.71	NA	0.88	0.98	0.85
*RC*2*XX*	0.53	0.98	0.92	-0.30	-0.56	-0.62	NA	0.46	0.96	0.53
*s*0	-0.39	0.94	0.94	-0.20	-0.67	-0.39	-0.64	-0.08	0.84	0.02
*s*10	-0.21	0.97	0.96	0.07	-0.53	-0.31	-0.68	-0.16	0.88	0.31

of vehicles and the *travel distance*, when optimising the *number of vehicles*. Correlation values vary from -1 (conflict) to 1 (harmony). Entries NA in the table indicate that one objective or both have the same value across all solutions during the optimisation process. Correlation values close to zero ($\sim$ 0) or (NA) indicate an independence relationship. In these experiments, all values are averaged over the 20 runs and over all instances within the same subset.

Since we are working with five objectives, we have 20 possible relationship pairs. In these results, we find three types of pair-wise relationships.

- Zx $\sim$ Zy (1^{st} type): if the casual relationship has the same sign in both ways. That is, Zx $\rightarrow$ Zy and Zy $\rightarrow$ Zx are both positive or both negative. The pair-wise relationships of this type found in our experiments are always positive. In this case, as pointed out by Purshouse and Fleming [204], one of the two objectives is redundant and can therefore be removed. However, there might be reasons to keep both: either because of Decision Maker (DM) preference, of for completeness (given that the inclusion of both does not, necessarily, hinder the search process [204]).

- Zx $\neq$ Zy (2^{nd} type): if the casual relationship has different sign in each way. That is, Zx $\rightarrow$ Zy is positive and Zy $\rightarrow$ Zx is negative, or viceversa. In this pair-wise relationship type it is important to identify which relationship pair is positive, so that we can remove the criterion on the right hand side of the harmonious

relationship. For example, results indicate that Z2 $\neq$ Z3, because Z2 $\rightarrow$ Z3 is negative, and Z3 $\rightarrow$ Z2 is positive. Since Z3 $\rightarrow$ Z2 is positive, Z2 can be treated as for positive Zx $\sim$ Zy.

- Zx $-$ Zy (3^{rd} type): if one of the casual relationships does not hold. That is, Zx $\rightarrow$ Zy or Zy $\rightarrow$ Zx have average $\mathcal{C}$ values close to 0 or NA (null values). In this pair-wise relationship type it is important to analyse the sign of the non-null pair-wise relationship. For positive pair-wise relationships we can remove the criteria on the right hand side of the relationship pair. For negative pair-wise relationships we cannot separate the criteria on the left hand side of the relationship pair, because this criterion is not independent.

An example of the first type of relationships is Z1 $\sim$ Z2. In Z1 $\rightarrow$ Z2 (Table 7.2 $-$ column 2) , since all vehicles have to return to the depot, increasing the number of delivery vehicles will lead to an increase of the travel distance. Between the two objectives we find high and positive average $\mathcal{C}$ values (> 0.85) in subsets {C1XX, R1XX, RC1XX}, ($>$ 0.80) in subsets {s0.dX.twX, s10.dX.twX}, and (approx. 0.70) in subsets {C2XX, R2XX, RC2XX}. The opposite casual relationship Z2 $\rightarrow$ Z1 (*travel distance* $\rightarrow$ *number of vehicles*) (Table 7.2 $-$ column 6) produces the same behaviour. The total travel distance is reduced when the number of vehicles is minimised. High and positive average $\mathcal{C}$ values (> 0.90) are found across all instances in both Solomon's and MOVRPTW datasets.

Another example of this type of relationships is Z1 $\sim$ Z4 (*number of vehicles* Vs *waiting time*). Reducing the number of vehicles might also reduce the waiting time. For example, given a route-plan in which a given vehicle serves exactly one customer, the waiting time is maximum. In the results (Table 7.2 $-$ column 4) , we find a high and positive average $\mathcal{C}$ (> 0.90) across all instances in both datasets. For the opposite casual relationship is Z4$\rightarrow$Z1 (*waiting time* $\rightarrow$ *number of vehicles*) (Table 7.3 $-$ column 4) the effect is not so clear. According to the results, reducing the waiting time also improves the use of the fleet. We find high and positive average $\mathcal{C}$ values (approx. 0.90) across all instances in both datasets. This might be due to the way VNS works. VNS swaps customers within the sequence in order to create new route-plans. In this case, VNS guides the search towards the optimisation of the *waiting time*. In those routes

with a few customers with late opening times, the waiting time will be high. Therefore, moving these customers to other routes might result in an improvement of the waiting time, as well as in a better the use of the fleet.

We find the second type of relationships comparing Z2 $\neq$ Z5 (*travel distance* Vs *delay time*). In Z2 $\rightarrow$ Z5 (Table 7.2 $-$ column 9) , we get negative average $\mathcal{C}$ values (approx. -0.4) across Solomon's instances, and (< -0.70) across MOVRPTW instances. The opposite relationship is Z5 $\rightarrow$ Z2. For this relationship, the average $\mathcal{C}$ values for Solomon's dataset are positive (> 0.60), and near 0 for the MOVRPTW dataset. The difference between these results lies in the design of both datasets. In Solomon's instances, the layout of the customers and their time windows were carefully designed. In MOVRPTW instances, the customers are randomly spread and their time windows were randomly selected. Thus, in MOVRPTW instances, decreasing the *travel distance* will lead to an increase of the *delay time* (Z2 $\rightarrow$ Z5) due to customers in same areas might have different time slots. The results in the opposite relationship (Z5 $\rightarrow$ Z2) (Table 7.3 $-$ column 9) are explained using the same rationale, arriving earlier to the customers does not affect the *travel distance* because customers with similar time slots are not necessarily in the same areas.

Number of vehicles Vs *makespan* (Z1 $-$ Z3) is an example of the third type of relationships. On one hand, we have Z1 $\rightarrow$ Z3 (*number of vehicles* $\rightarrow$ *makespan*) (Table 7.2 $-$ column 3) . As we increase the number of vehicles, the makespan should decrease. In the extreme case of using as many vehicles as customers to be served, the makespan will be minimum. A high and negative average $\mathcal{C}$ value (approx. -0.75) is found throughout all instances in both datasets. The opposite relationship is Z3 $\rightarrow$ Z1 (*makespan* $\rightarrow$ *number of vehicles*) (Table 7.2 $-$ column 10) . In this relationship, due to the effect of time windows, minimising the makespan might not have a direct impact on the number of vehicles used to serve the customers. Our results show that in most instances the number of vehicles is not affected by the minimisation of the makespan. In all instances with NA, the number of vehicles did not change throughout the optimisation process. However, we find low and negative average $\mathcal{C}$ values (approx. -0.50) in instances {C107, C108, R101, R102, R104, R108, R109, R111, RC101, RC108} in Solomon's dataset

and {s0.d0.tw0, s0.d1.tw0, s0.d2.tw4, s10.d1.tw0, s10.d1.tw1, s10.d1.tw2, s10.d1.tw3}. These values might be due to the effect of the waiting time. One possible scenario is that of route-plan with many routes so that: 1) the first customer of one of these routes has a late opening time, and 2) customers within the route are far from each other. This scenario has a large number of vehicles and the makespan is high.

The type of casual pair-wise relationships are summarised in Table 7.1. The first row shows all possible pair-wise relationships across the 20 causal relationships. The second row shows all *signs* found in the average $\mathcal{C}$ values of all instances for each pair-wise relationship. The third row indicates the resulting type of pair-wise relationship according to the above mentioned classification. For example, [Z1,Z2] exhibits in both Z1 $\rightarrow$ Z2 and Z2 $\rightarrow$ Z1 harmony ($\oplus/\oplus$ in the second row), thus the pair-wise relationship is of 1^{st} type ($\sim$ in the third row) which means that either Z1 or Z2 could be omitted. However, the omission of one of these objective will usually depend on the DM's preference.

Most pair-wise relationships are of the same type of dependency across all instances. However, in some cases the type of dependency changes between instances. The relationship Z3 $\rightarrow$ Z4 (*makespan* $\rightarrow$ *waiting time*) (Table 7.3 $-$ column 2) (RCXXX)) presents both types of dependency: harmony ($\oplus$) and conflict ($\ominus$). This discrepancy appears between the results in Solomon's and MOVRPTW instances. In Solomon's instances, the average $\mathcal{C}$ value is of 0.54. In MOVRPTW instances, the average $\mathcal{C}$ value is of -0.31. We find independence (*NA*) and conflict $\ominus$ in the relationships Z3 $\rightarrow$ Z1 (*makespan* $\rightarrow$ *number of vehicles*) and Z5 $\rightarrow$ Z1 (*delay time* $\rightarrow$ *number of vehicles*). In Z3 $\rightarrow$ Z1 (Table 7.2 $-$ column 10) (RCXXX)) , *NA* is found in all instances in subsets {*C*2*XX*, *R*2*XX*, *RC*2*XX*}. Probably due to the interaction with time windows, conflict $\ominus$ is found in some instances in subsets {*C*1*XX*, *R*1*XX*, *RC*1*XX*} and in some MOVRPTW instances. In Z5 $\rightarrow$ Z1 (Table 7.3 $-$ column 8) (RCXXX)) , we also find independence and conflict. In this case, conflict $\ominus$ is found in some instances of the MOVRPTW dataset. Interdependence *NA* is found in all instances in Solomon's dataset and most MOVRPTW instances. Other relationships that present independence are Z4 $\rightarrow$ Z2 (*waiting time* $\rightarrow$ *travel time*) and Z5 $\rightarrow$ Z2 (*delay time* $\rightarrow$ *travel distance*). In Z4 $\rightarrow$ Z2 (Table 7.2 $-$ column

8) (RCXXX)) , we find harmony $\oplus$ in all instances in subsets $\{C1XX, R1XX, RC1XX\}$, conflict $\ominus$ $\{C2XX, R2XX, RC2XX\}$, and independence (approx. 0) in all instances of the MOVRPTW dataset. Finally, in Z5 $\rightarrow$ Z2, (Table 7.3 $-$ column 9) (RCXXX)) we find harmony $\oplus$ across all instances in Solomon's dataset and independence (approx. 0) in MOVRTW instances.

7.3.2 Correlation Between Objectives in Pareto Approximation Sets

In the previous section, we studied which causal relationships arise between pairs of objectives as the search progresses. This second section of the experimental analysis aims to address the following three questions:

1. Which pair-wise relationships do appear in final non-dominated solutions sets?
2. Are these relationship pairs consistent with those that occur throughout the optimisation process found by the VNS?
3. Is there any difference between the correlation values obtained with Solomon's dataset and MOVRPTW dataset?

Following the platform for the treatment of large number of criteria by Purshouse and Fleming [204], we treated the results using Scatterplot matrices. Figure 7.2 shows 6 scatterplot matrices related to: three Solomon's instance on the left, and three MOVPTW instances on the right. Each scatterplot matrix shows a section of the trade-off surface between a pair of objectives (below the main diagonal), and the correlation value associated to each pair (above the main diagonal) [257, 163]. The closer the correlation value is to 1 or -1, the larger is the font type, and the stronger is the dependence relationship between the corresponding pair of objectives. A positive correlation value indicates a harmonious relationship. The opposite occurs when the correlation values are negative. In each matrix the objectives $Z1$ to $Z5$ are shown in the main diagonal of the matrix (where $Z1$ is the *number of vehicles*, $Z2$ is the *travel distance*, $Z3$ is the *makespan*, $Z4$ is the *waiting time* and $Z5$ is the *delay time*). Dots across the scatterplot matrices represent non-dominated solutions.

At first sight, the correlation values seem much larger for the three MOVRPTW instances. An example of harmonious behaviour is seen in the instance $s0.d0.tw4$ (top - right matrix Figure 7.2) when comparing $Z1$ vs. $Z4$. Its correlation value of 0.88 indicates that, as we decrease the *number of vehicles* (Z1), the *waiting time* (Z4) for each of them gets shorter, which is logical. In the same instance, an example of conflict arises comparing $Z1$ vs. $Z5$. Here, the correlation value of -0.90 means that by decreasing the number of vehicles $Z1$, the delay time $Z5$ gets longer and vice-versa. In the Solomon's $R101$ (middle - left), the pair-wise dependence relationships between objectives appears to be weaker. For example, $Z1$ seems not to be related to any of the other objective under study. The bi-criterion shape sections of the trade-off surface are plotted below the main diagonal for both instances. It is clearly seen that MOVRPTW instances (on the right) presents more uniform shapes than those of Solomon (on the left).

The scatterplot matrices depicted in Figure 7.2 aim to show the relationships between the shape sections of trade-off surfaces and their correlation values. For this reason we only show three pairs of scatterplot matrices. Similar values are obtained for other instances. From this section onwards we discuss the results based only on the average correlation values. We first present a subsection with results for the Solomon's instances and then another with results for the MOVRPTW instances.

Solomon's Dataset:

Table 7.4 shows the average correlation values obtained by NSGA-II for all Solomon's subsets $\{CXXX, RXXX, RCXXX\}$ and MOVRPTW subsets $\{s0, s10\}$. Each table shows in its first column the name of the subset. Each of the remaining columns shows average the correlation value corresponding to a pair-wise comparison between objectives. For example, Z1-Z2 compares the *number of vehicles* against the *travel distance*. Correlation values vary from -1 (conflict) to 1 (harmony). Those correlations values equal to NA indicate that one or both objectives have the same value across all the non-dominated solutions found. All values are averaged over the 20 runs and over all instances within the subset.

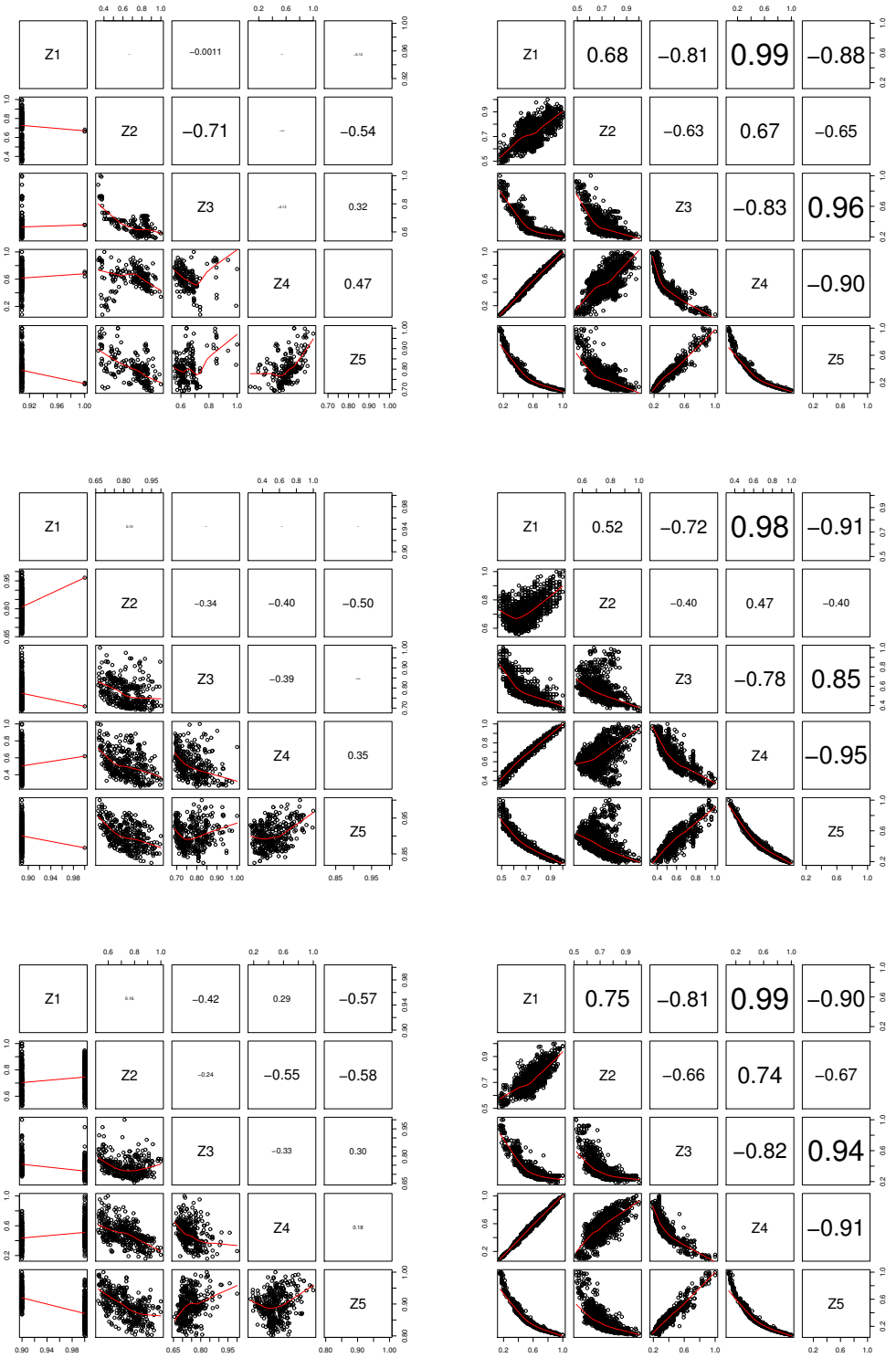


Figure 7.2: Scatterplot matrix for Solomon's $C107$ (top - on the left), $R101$ (middle - on the left) and $RC101$ (bottom - on the left) and for MOVRPTW $s0.d0.tw4$ (top - on the right), $s0.d2.tw4$ (middle - on the right) and $s0.d1.tw3$ (bottom - on the right). In the main diagonal the objectives $Z1$ to $Z5$ are shown (where $Z1$ is the number of vehicles, $Z2$ is the travel distance, $Z3$ is the makespan, $Z4$ is the waiting time and $Z5$ is the delay time). Each scatterplot matrix shows, below the main diagonal, a section of the trade-off surface between a pair of objectives and, above the main diagonal, the correlation value associated to each pair

Table 7.4: Average correlation values obtained by NSGA-II for all instances in for all instances in Solomon's subsets $\{CXXX, RXXX, RCXXX\}$ and MOVRPTW subsets $\{s0, s10\}$ (20 runs).

Instance	Z1-Z2	Z1-Z3	Z1-Z4	Z1-Z5	Z2-Z3	Z2-Z4	Z2-Z5	Z3-Z4	Z3-Z5	Z4-Z5
$C1XX$	0.23	-0.12	0.28	-0.20	-0.62	0.18	-0.27	-0.56	-0.02	0.24
$C2XX$	0.69	-0.82	0.99	-0.88	-0.77	0.65	-0.77	-0.83	0.92	-0.86
$R1XX$	NA	NA	NA	NA	-0.32	-0.48	-0.13	-0.26	-0.03	0.23
$R2XX$	0.72	-0.64	0.73	-0.30	-0.62	0.36	-0.41	-0.73	0.47	-0.29
$RC1XX$	0.09	-0.18	0.18	-0.59	-0.28	-0.47	-0.04	-0.31	0.05	-0.10
$RC2XX$	0.75	-0.69	0.70	-0.59	-0.72	0.30	-0.70	-0.66	0.68	-0.43
$s0$	0.66	-0.80	0.99	-0.90	-0.63	0.65	-0.63	-0.81	0.85	-0.94
$s10$	0.66	-0.78	0.99	-0.89	-0.56	0.64	-0.58	-0.81	0.94	-0.91

Table 7.4 shows in the first two rows the average correlation values across all instances with customers in clusters (i.e. subsets $C1XX$ and $C2XX$). Subset $C1XX$ presents the customers in well defined clusters and their time windows are narrow. Subset $C2XX$ has wider time windows. This might explain why the correlation values are so low in the $C1XX$ subset. Table 7.4 shows in the first row, as Tan et al. [238] and Garcia-Najera and Bullinaria [105] previously stated, that no conflict exists between *number of vehicles* (Z1) and *travel distance* (Z2). However, our results indicate that no dependency holds for any objective against Z1 in almost all the instances in $C1XX$. The largest values for this subset seem to appear in the comparison of *travel time* (Z2) against *makespan* (Z3), with an average correlation value of -0.62. Conversely, in the second row of table 7.4, the results for the subset $C2XX$ seem to indicate clear pair-wise dependency relationships. In this subset, the most conflicting objectives are *number of vehicles* (Z1) against *delay time* (Z5), with an average correlation value of -0.88. And it is worth noting that in this set, the improvement of *makespan* (Z3) is in harmony with the improvement of *delay time* (Z5).

Table 7.4 shows in the second pair of rows the average correlation results for the randomly spread customers $RXXX$. This subset is also divided into two subsets: $R1XX$ and $R2XX$. In this case, both share the same geographical distribution of their customers. However, $R1XX$ has narrower time windows than $R2XX$. The third row of table 7.4 corresponding to $R1XX$ subset, we appreciate a similar behaviour as in $C1XX$. That is, $Z1$ (*number of vehicles*) seems not to have a pair-wise dependence relationship to

any other objective under this study. The only considerable conflict relationship arises in $R103$ (-0.62), $R107$ (-0.58) and $R108$ ($-0,58$) comparing *travel distance* (Z2) against *waiting time* (Z4). The fourth row of table 7.4 shows the results for the $R2XX$ subset. In a similar fashion to the $CXXX$ instances, the subset $R2XX$ presents average correlation values closer to 1 and -1 with respect to $R1XX$. However, the average correlation values for $R2XX$ do not seem to be as large as for $C2XX$. According to these results, the most conflicting relationship is *makespan* (Z3) versus *waiting time* (Z4), with an average correlation value of -0.73. Additionally, it is important to highlight that in the pair-wise comparison of *travel distance* (Z2) versus *waiting time* (Z4), we have a conflicting relationship in the subset $R1XX$ and a harmonious one in $R2XX$. It is also interesting to see that the instance $R208$ does not hold the same pair-wise relationship than the rest of the comparisons involving *delay time* (Z5) in the $R2XX$ group of instances.

The average correlation values for the random-cluster Solomon's instances appear in the third pair of rows in Table 7.4. Subset $RCXXX$ includes two subsets: $RC1XX$ and $RC2XX$. The difference lies in that $RC1XX$ has narrower time windows than $RC2XX$. In the first subset, there seems to be very little interaction between the *number of vehicles* (Z1) and most other objectives. However, unlike the subsets $C1XX$ and $R1XX$, in this subset the *number of vehicles* (Z1) presents a conflict relationship with the *delay time* (Z5). We find a similar situation as in $R2XX$ with comparing *travel distance* (Z2) versus *waiting time* (Z4). The subset $RC1XX$ shows a conflict relationship, while for the $RC2XX$ is harmonious.

MOVRPTW Dataset:

Table 7.4 shows in the last pair of rows the average correlation values for the MOVRPTW subsets $s0$ and $s10$. For $s0$ (Table 7.4 - row 7), the strongest conflicting relationships are found for *number of vehicles* (Z1) versus *waiting time* (Z5) with average correlation value of -0.9, and for *waiting time* (Z4) versus *delay time* (Z5) with average correlation value of -0.93. For $s10$ (Table 7.4 - row 8), the most conflicting relationships are, as in the previous case, *number of vehicles* (Z1) versus *waiting time* (Z5) with -0.89, and *waiting time* (Z4) versus *delay time* (Z5) with -0.91.

At the beginning of this section, we listed three questions: (1) which pair-wise relationships do appear in final non-dominated solutions sets?, (2) are these relationship pairs consistent to those that occur throughout the optimisation process found by the VNS?, and (3) is there any difference between the correlation values obtained with Solomon's dataset and MOVRPTW dataset?

1. Table 7.5 shows a summary of the pair-wise relationships that appear in the final non-dominated solutions sets across all the Solomon's instances in subsets: $(C2XX, R2XX, RC2XX)$ and all instances in the MOVRPTW Dataset. Note that Solomon's subsets: $(C1XX, R1XX$ and $RC1XX)$ (instances with narrow time windows) are not considered as the pair-wise relationships within these sets are weak or not consistent with the others. The pair-wise relationships found in non dominated solutions sets across all instances excluding those with narrow time windows are consistent and logical. For example, Z1-Z5 (*number of vehicles - delay time*) presents a conflicting relationship. It seems logical that improving (reducing) the number of vehicles leads to the worsening of the delay time.

2. There is consistency in the pair-wise relationships found by: 1) VNS throughout the optimisation of one criteria (Table 7.1), and 2) NSGA-II among non-dominated solutions sets (Table 7.5). Table 7.1 shows that [Z1,Z2], [Z1,Z4] and [Z3,Z5] have a bi-directional harmonious pair-wise relationship: $\oplus/\oplus$. Similarly, Table 7.5 shows that Z1-Z2, Z1-Z4 and Z3-Z5 are all harmonious pair-wise relationships. The rest of objective pairs shown in Table 7.1 have more than one pair-wise relationship type. According to Table 7.1 [Z1,Z3] and [Z1,Z5] are $\ominus$/NA and their pair-wise relationship is of the 1^{st} type: $Z1 - Z3$ and $Z1 - Z5$. As previously noted, *for negative pair-wise relationships we cannot separate the criteria on the left hand side of the relationship pair, because this criterion is not independent* (Section 7.3.1). Table 7.5 shows that the pair wise relationship between these pair of objectives is indeed of conflict. Objective pairs [Z2,Z3], [Z2,Z5], [Z3,Z4] and [Z4,Z5] can be analysed using the same rationale (using the 2^{nd} type) (Section 7.3.1). The objective pair [Z2,Z4] is a special case due to the nature of the objectives. There is not a clear pattern for neither average $\mathcal{C}$ values (VNS tables) nor correlation values

Table 7.5: General dependency relationships across the Solomon's instances subsets: $C2XX$, $R2XX$, $RC2XX$ and MOVRPTW Dataset. Conflict relationship is denoted with $\ominus$, while harmony uses $\oplus$.

	Z1-Z2	Z1-Z3	Z1-Z4	Z1-Z5	Z2-Z3	Z2-Z4	Z2-Z5	Z3-Z4	Z3-Z5	Z4-Z5
Relationship	$\oplus$	$\ominus$	$\oplus$	$\ominus$	$\ominus$	$\oplus$	$\ominus$	$\ominus$	$\oplus$	$\ominus$

(NSGA-II tables) across all instances.

3. The proposed MOVRPTW dataset, which is designed based on real-world data and with a multi-objective mindset, presents better dependency relationships using pair-wise comparisons of the objectives under consideration. We have seen that the subset of Solomon's instances containing narrow time windows ($C1XX$, $R1XX$ and $RC1XX$) might not be entirely adequate for the assessment of multi-objective algorithms as the studied pair-wise relationships are weak. Solomon's instances with wider time windows present a sound but still not ideal benchmark scenario for multi-objective VRPTW.

7.4 Conclusions

The main contribution of this chapter is a better understanding of the multi-objective nature of the VRPTW. We conduct a twofold study to analyse which pair-wise relationships appear: 1) throughout the optimisation process, and 2) among non dominated solutions sets.

This investigation is carried out using five common objectives: *number of vehicles* (denoted as Z1) needed to serve all customers, *total travel distance* (Z2), *makespan* (Z3) or travel time of the longest route (from/to depot), *total waiting time* (Z4) of the delivery vehicles, and *total delay time* (Z5) or sum of tardiness for all deliveries.

This first study uses a Variable Neighbourhood Search (VNS) to investigate how the optimisation of one objective affect the others. We find three types of pair-wise relationships: 1) Zx $\sim$ Zy (1^{st} type) when the relationship has the same sign in both ways, 2) Zx $\neq$ Zy (2^{nd} type) when the relationships has a different sign, and 3) Zx $-$ Zy (3^{rd} type) when one of the relationships does not hold. The type of pair-wise relationship

provide information about the possibility of decomposing or reducing the problem at hand in terms of number of objectives. For example, in harmonious pair-wise relationships of the 1^{st} type, it is possible to leave out one of the two criteria because the optimisation in one objective witnesses an improvement in the other. The same occurs for those pair-wise relationships of the 3^{rd} type in which the non-null relationship is harmonious.

The second study uses a Non-dominated Sorting Genetic Algorithm (NSGA-II) to investigate the pair-wise relationships that appear close to the Pareto optimal front. We highlight three findings: 1) pair-wise relationships found in non-dominated solution sets across all instances (excluding Solomon's instances with narrow time windows) are consistent and logical, 2) pair-wise relationships found: a) throughout the optimisation (using VNS), and b) in non-dominated solutions sets (using NSGA-II) are also consistent, and 3) pair-wise relationships found in the MOVRPTW dataset seems to be stronger that those found in the Solomon's dataset.

The third finding above deeds us to think that the Solomon's instances might not be entirely adequate to investigate the multi-objective VRPTW. Our experiments also revealed that there is a much clearer interaction between different objectives in our problem instances. This is potentially a very good thing because the multi-objective nature of this important logistic problem can be better investigated using this realistic dataset.

It is our belief that the magnitude of pair-wise relationships present in Solomon's dataset might be improved by implementing the suggestion discussed in Chapter 5. For example, the Manhattan distance notion could be used to calculate the distance between each pair of customers instead of using the Euclidean distance. The resulting distance could be also multiplied by a noise factor so that travel distance and travel time matrices are non-equal and non-symmetric. This new set-up could be tested using the same methodology to that applied in this study. We opted to create a real-world based dataset in order to provide an as much close-to-real scenario as possible. A scenario in which: travel distances and travel times are obtained from reliable sources, time windows are based on realistic profiles, etcetera.

The next chapter presents a Multi-objective Discrete Particle Swarm Optimisation im-

plemented in CODEA$v3$. This algorithm extends the Discrete Particle Swarm Optimisation proposed by Consoli et al. [54]. This implementation is applied to the Vehicle Routing Problem with Time Windows (VRPTW) and tested with Solomon's and MOVRPTW datasets.

CHAPTER 8

A simplified MODPSO for VRPTW

Summary

Chapter 3 introduced CODEA, a software library to create systems of cooperative agents to tackle combinatorial optimisation problems. Chapters 5 and 6 described and studied two datasets for the assessment of Multi-Objective Vehicle Routing Problems with Time Windows (VRPTW): Solomon's and MOVRPTW datasets. Chapter 7 showed that MOVRPTW test instances are more suitable than those of Solomon for assessing multi-objective aspects of the problem. These studies have provided a solid foundation upon which to build and test a cooperative multi-objective optimiser in CODEA.

This chapter presents a simplified Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) algorithm implemented in CODEA v3. We apply this implementation to the Solomon's and MOVRPTW datasets. The results obtained by this algorithm are compared to those obtained by NSGA-II. The performance of both algorithms is assessed using three quality indicators: hypervolume, coverage and overall non-dominated vector generations. According to experimental results, the proposed MODPSO performs better than NSGA-II.

8.1 Introduction

This chapter proposes a simplified Multi-objective Discrete Particle Swarm Optimisation (MODPSO) to tackle the Vehicle Routing Problem with Time Windows (VRPTW). This algorithm has two main components: 1) a Multi-objective component to update the leaders based on the notion of Dominance Depth Fitness Assignment introduced by Goldberg [116], and 2) a Discrete Particle Swarm Optimisation (DPSO) algorithm based on the work by Consoli et al. [54].

Dominance Depth Fitness Assignment [116] classifies solutions in different classes or fronts according to Pareto non-domination relationships. A solution that belongs to a class is not dominated and does not dominate any other solution in that class. The fitness value assigned to each solution depends on the class they belong to. Thus, solutions in the first class or front are assigned with the highest fitness value, whereas solutions in the last class or front are assigned with the lowest. The Dominance Depth Fitness Assignment is the underlying mechanism in the two versions of the non-dominated sorting algorithm: NSGA [234] and NSGA-II [65]. The non-dominated sorting mechanism has also been included in MOPSO algorithms, yielding to the so-called Non-dominated Sorting Particle Swarm Optimisation (NSPSO). Li [164] presented a NSPSO which appears to be a hybrid between a PSO and NSGA-II. Li's algorithm creates a temporary population of $2N$ individuals (where N is the size of the main population) and non-dominated sorting is employed to select the best N individuals. Furthermore, two mechanisms were used to maintain diversity: niche count [99] and crowding distance [65]. The performance of Li's approach was compared to that of NSGA-II in four test functions the ZDT problems [67] with two objectives. Ongsakul and Saksinchai [188] successfully applied Li's algorithm to the multi-objective strategic bidding problem with two objectives: profit maximisation and risk minimisation. That approach was also compared to NSGA-II in three ZDT problems [67]. However, this comparison was qualitative (based on non-dominated solutions plots) and no many details were provided. Benabid et al. [12] proposed a NSPSO to tackle the multi-objective voltage stability problem. Their algorithm used an external archive to stored non-dominated solutions. Non-dominated sorting was used to: 1) reduce the size of the

external archive, and 2) to select the particle's best personal solution. Results were successfully indicating that NSGA-II and a standard PSO were less effective than NSPSO. Another NSPSO was presented in the works by Feng et al. [91] and Li et al. [269]. Those two similar algorithms used an external population to store non-dominated individuals. Non-dominated sorting was used to reduce the size of that external population and to update the swarm's global leader. The performance of the proposed NSPSO was compared to NSGA-II [65] and SPEA2 [274] in six ZDT problems [67] with two objectives. Results indicated that the proposed NSPSO outperformed the other two algorithms in all instances.

All the above research works have the following three characteristics in common:

- They are validated using continuous problems.
- They do not consider problems with more than three objectives.
- They use specific mechanisms to boost convergence and maintain diversity (external archives / diversity mechanisms).

The literature on NSPSO shows that a signifficant effort has been made to imitate the ideas in NSGA-II. Many authors have incorporated NSGA-II mechanisms, such as non-dominated sorting and crowding distance into their NSPSO algorithms. However, no research have been made towards a better understanding of which specific components of NSGA-II improve the performance of NSPSOs. Besides, most NSPSO approaches have been mainly validated using continuous problems (usually from the ZDT series problem-set [67]), and not taking into consideration more than three objectives.

The research proposed in this chapter focuses on the design of a simplified MODPSO that solely uses the concept of Dominance Depth Fitness Assignment [116]. Thus, our MOPDSO does not employ any extra mechanism to promote convergence or diversity. We validate our algorithm using the problem studied earlier in this thesis, that is, the multi-objective VRPTW with five minimisation objectives: 1) *number of vehicles* needed to serve all customers, 2) *total travel distance*, 3) *makespan* or travel time of the longest route (from/to depot), 4) *total waiting time*, and 5) *total delay time*. Moreover, in a similar fashion to other NSPSO studies, we compare the performance of our MODPSO

algorithm to that of NSGA-II. Results in both Solomon's and MOVRPTW datasets are analysed using three standard quality indicators: hypervolume, coverage and overall non-dominated vector generations.

To the best of our knowledge, NSPSO has not been applied to the VRPTW. However, we can find other MODPSO based approaches to tackle the VRPTW, such as those by Muñoz-Zavala et al. [185] and by Shurog et al. [232]. The former work [185] proposed a MODPSO with two main features: 1) a perturbation operator to keep diversity, and 2) a ring topology to improve the exploration capacity. They also introduced a new data mining technique (sector models) to create customer clusters according to their geographical positions. Experiments were carried out on Solomon's instances with three objectives: 1) travel distance, 2) total waiting time and 3) number of vehicles. The latter work [232] proposed a MODPSO that combines two features: 1) dynamic inertial weights, and 2) an immunity concept. Experiments in that work were validated using Solomon's instances with two objectives: 1) travel time and 2) total delay time.

We noted that these two MODPSO algorithms: 1) do not consider more than three objectives, and 2) use extra-mechanism to promote convergence and/or diversity.

Our MODPSO is an extension of a single-objective DPSO previously proposed by Consoli et al. [54] for the Minimum Labelling Steiner Tree Problem. The DPSO by Consoli et al. was first applied to the VRPTW in [31]. In that implementation the swarm had to deal with eight minimisation objectives: 1) number of vehicles, 2) total travel distance, 3) total travel time, 4) total waiting time, 5) total delay time, 6) number of time window violations, 7) total capacity exceeded and 8) number of vehicles in which their capacity is exceeded. The purpose of that investigation was to study the ability of an oversimplified MODPSO to evolve poor infeasible solutions (large number of constraints violated) to feasible regions. Results indicated that a simple MODPSO algorithm was able to generate feasible solutions, in some cases with objective values close to the global optimum (in terms of 1) number of vehicles, and/or 2) travel distance [75]). We used that study as starting point to design a simplified MOPDSO algorithm capable of outperforming the NSGA-II. The present chapter shows that a MODPSO with two components: 1) Dominance Depth Fitness Assignment, and 2) a simple DSPO, produces

better results than NSGA-II with the above mentioned settings.

The reminder of this chapter is organised as follows. Section 8.2 explains how the Consoli's Discrete Particle Swarm Optimisation works. Section 8.3 introduces our new MODPSO approach. Sections 8.4 and 8.5 state experiment setup and results, respectively. Section 8.6 discusses the performance of this algorithm in terms of computational speed. Finally, conclusions are presented in Section 8.7.

8.2 Discrete Particle Swarm Optimisation (DPSO)

The Jumping Frog Optimisation (JFO) is an approach to the Discrete Particle Swarm Optimisation (DPSO) proposed by Consoli et al. [54]. The metaphor of JFO is that of a group of frogs looking around for food while jumping from one lilypad to another. Frogs compete for food by moving towards the best locations. Thus, if a frog is well-placed, the others will tend to move towards its position. This DPSO drops the concept of speed. Positions are updated using a *follower-attractor* mechanism. A *follower* is a particle (frog) that is about to move to a new position. Conversely, the *attractor* is the particle that the *follower* uses as a reference. This way, the *follower* copies some of the *attractor's* features in order to 'look' similar to its *attractor*. After this operation, the *follower* is in a new location with a new structure. This new structure is the product of combining his previous features with new ones copied from its *attractor*.

Figure 8.1 shows the pseudocode for this approach. First, initial positions (solutions) are assigned to all particles in the swarm. Then, for k generations and for each particle p in the swarm(lines $2-3$), the following operations are repeated. A roulette-wheel selection mechanism (lines $4-5$) is used to select the attractor. There are four possible attractors (lines $6-9$): no-attractor, best neighbour at the current generation `p.g_k`, best particle's previous location `p.b_id`, and best particle's location found by the swarm so far g. Attractors are assigned with certain probability $c_1 \cdots c_4$, so that the sum of probabilities equals 1. If $r \in c_1$ (line 6) then no particle acts as attractor and the follower makes a random move (mutation) with respect to its current position `p.x_id`. The purpose of this *Inertial Move* is to explore the area around the follower's (particle's) po-

sition. If $r \in c_2$ (line 7) then the particle moves towards (crossover) the best positioned particle in its neighbourhood at the current generation `p.g_k`. The aim of this *Cognitive Move* is to influence an attraction force within the swarm. If $r \in c_3$ (line 8) then the attractor is the previous best location of the follower `p.b_id` (crossover). This *Local Move* helps the particle to explore similar features to the ones in `p.b_id`. In case $r \in c_4$ (line 9) then the attractor is the best position `g` found by the swarm so far (crossover). This *Global Move* encourages the re-use parts of the overall best solution `g` on the design of different structures. The result of the operation of mutation or crossover is stored is `n`. Then, a local search is applied to improve the quality of this new position (solution) `n` (line 11). Finally, `n` is evaluated and the values of `p.x_id`, `p.b_id` and `g` are updated if necessary (lines $12 - 13$).

8.3 Proposed MODPSO Algorithm

The Jumping Frog Optimisation algorithm introduced in Section 8.2 is quite simple and flexible, and it can be applied to many combinatorial problems. The original version was applied to the *Minimum Labelling Steiner Tree Problem* [54]. In that application, the authors built the JFO algorithm towards the optimisation of the cost of trees created throughout the search.

This work extends the JFO to the multi-objective scenario by incorporating the notion of Dominance Depth Fitness Assignment introduced by Goldberg [116]. This fitness assignment scheme is used when computing the quality of the new position (solution) `n` (line 12, Figure 8.1). This fitness value is then used to decide wether or not to update `p.b_id` and `g` (line 13, Figure 8.1).

In order to show the power of this DPSO in multi-objective combinatorial problems, no extra mechanisms are incorporated to boost diversity or convergence. Thus, unlike many other NSPSO algorithms, the proposed MODPSO does not use mechanisms, such as niche count [99] and crowding distance [65]. Additionally unlike in our previous work [31], in order to further simplify our MODPSO we do not use any *local search* for intensification purposes (line 11, Figure 8.1).

```
1  initialise(swarm)
2  for (k in generations)
3     for (p in swarm)  {
4        r = rand(0,1);
5        case (r) {
6           r is in c1: n = mutation(p.x_id)           // Inertial Move
7           r is in c2: n = crossover(p.x_id, p.g_k)   // Cognitive Move
8           r is in c3: n = crossover(p.b_id, p.x_id)  // Local Move
9           r is in c4: n = crossover(g, p.x_id)       // Social Move
10       }
11       localSearch(n)
12       computeFitness(n)
13       update(n, p.x_id, p.b_id, g)
14    }
```

Legend:

- `c1 ... c4`: probabilities assigned to each attractor.
- `p.x_id`: current position of the particle.
- `p.b_id`: best position found by the own particle.
- `g_k`: best position in the swarm/neighbourhood in the current generation `k`.
- `g`: best position found by the swarm up to the current iteration `k`.
- `n`: new position of the particle.
- `mutation`: unary operator that randomly modifies features of a particle.
- `crossover`: binary operator that crosses a pair of particles (parents) in order to produce offsprings with a random combination of features.

Figure 8.1: JFO Algorithm Pseudo-code

8.3.1 Solution Representation and Initialisation

In this implementation, the encoding of an individual is a large sequence of customer identifiers (customer ids). This sequence determines the order in which customers are served. The resulting route-plan is obtained by decoding the individual. The decoding process is a deterministic algorithm that splits the large sequence into smaller sequences (routes). The criterion used to split the sequence is the violation of the capacity of the delivery vehicles.

For example, the encoded individual [2 1 7 4 5 3 6 9 8] could result in the route-plan [

2 1][7 4 5 3 6][9 8], after the decoding process. This route-plan has three routes: 1) [2 1], 2) [7 4 5 3 6], and 3) [9 8]. In the first route, the customer with id "2" is served in the first place and the customer with id "1" in the second. The other two routes have similar interpretations.

The population is initialised using a constructive method that aims at satisfying first the customers farthest from the depot. The first customer of each route is selected trying to maximise a function that takes into account the time windows and how far each customer is from the depot. Each route is then created by selecting the best customer (in terms of time windows) and then by adding the cheapest customers (in terms of travel distance) until either time windows or capacity constraints are violated.

8.3.2 Constraints and Objectives

In order to provide a more realistic scenario, we treated the hard constraints derived from the time windows as soft constraints.

Hence, the quality of route-plans are assessed according to five common objectives (see Section 8.1). These objectives are: 1) *number of vehicles* needed to serve all customers, 2) *total travel distance*, 3) *makespan* or travel time of the longest route (from/to depot), 4) *total waiting time*, and 5) *total delay time*.

8.3.3 Operators: Mutation and Crossover

The *crossover* process recombines two parents (solutions) creating one or two offsprings (new solutions). In this implementation, we worked with three standard crossover operators: (1) Generic crossover, (2) Two-point crossover, and (3) Edge crossover. In the One-point crossover (Fig. 8.2), a random number of consecutive customers are copied from one parent to another, removing duplicates. The Edge crossover (Fig. 8.3) consists of (1) constructing new intermediate solutions by joining edges from both route-plans (parents) and (2) merging sub-tours creating feasible solutions. In the Generic crossover (Fig. 8.4), an entire route is copied from one route-plan to another, removing duplicates.

The *mutation* mechanism is the only process that the proposed MODPSO uses to pro-

mote diversity within the population. We used four basic mutation operators: Swap (Fig. 8.5), Insertion (Fig. 8.6), Inversion (Fig. 8.7) and Displacement (Fig. 8.8).

The Swap mutation interchanges the position of two customers within a route-plan. The Insertion mutation consists of moving a random customer to a new position within the route-plan. In the Inversion mutation, customers in a portion of the route-plan are reversed. The Displacement mutation is a generalisation of the insertion mutation which moves not one but a number of consecutive customers.

```
Given two route-plans:
Route-plan #1: [4] [3 2 7] [8 6 1 5]
Route-plan #2: [7 1] [2 6] [3 5 4 8]

1) Select a random Route from a route-plan (e.g. Route-plan #1).
Route-plan #1: [4] [3 2 7] [8 6 1 5]  => Route: [3 2 7]

2) Copy the nodes of the other route-plan (i.e. Route-plan #2) in the new route-plan
without the nodes contained in Route, that is:

Route-plan #2: [7 1] [2 6] [3 5 4 8] => [1] [6] [5 4 8]

3) Insert Route (selected in Step 1) in the new route-plan.
New route-plan: [1] [6] [5 4 8] [3 2 7]
```

Figure 8.2: Generic recombination operator.

```
Given two route-plans:
Route-plan #1: [4] [3 2 7] [8 6 1 5]
Route-plan #2: [7 1] [2 6] [3 5 4 8]

1) Select a random Sequence of nodes from one parent (e.g. Route-plan #1).
Route-plan #1: [4] [3 2 7] [8 6 1 5]  => Sequence: 2 7 8

2) Copy the nodes of the other route-plan (i.e. Route-plan #2) in the new route-plan
without the nodes contained in Sequence, that is:

Route-plan #2: [7 1] [2 6] [3 5 4 8] => [1] [6] [3 5 4]

3) Place the Sequence (selected in Step 1) at a random place in the new route-plan.
New route-plan: [1] [6 2 7 8] [3 5 4]
```

Figure 8.3: Two-point recombination operator.

8.4 Experimental Design

We validate this approach performance using the Solomon's dataset [233] and the MOVRPTW dataset [37].

The development of the NSGA-II was carried out in CODEA *v*3. We used the implementation of an Evolutionary Algorithm (EA) for the VRPTW [158] as a starting point.

```
Given two route-plans:
Route-plan #1: [4] [3 2 7] [8 6 1 5]
Route-plan #2: [7 1] [2 6] [3 5 4 8]

1) Create an edge-list with adjacent nodes to each node in both route-plans:
 Node 1: edges to other nodes: 6 5 7
 Node 2: edges to other nodes: 3 7 6
 Node 3: edges to other nodes: 2 5
 Node 4: edges to other nodes: 5 8
 Node 5: edges to other nodes: 1 3 4
 Node 6: edges to other nodes: 2 1
 Node 7: edges to other nodes: 2 1
 Node 8: edges to other nodes: 6 4

2) Select a random node - Current node (e.g. Current node = 4).

3) Create the new route-plan using the edge-list.
 3a) Examine which nodes can be accessed from the Current node.
     Node 4: edges to other nodes: 5 8
     Candidate nodes are 5 and 8
 3b) Examine the degree of the Candidate nodes:
     Node 5: edges to other nodes: 1 3 4 => degree = 3
     Node 8: edges to other nodes: 6 4 => degree = 2
 3c) Insert the Candidate node with the lowest degree in the new route-plan (i.e. 8).
     (in case of a tie, select a random Candidate node).
 3d) Set Current node to the just inserted node (i.e. Current node = 8).
 3e) Go to Step 3a if all nodes have not been yet inserted in new route-plan.
```

Figure 8.4: Edge recombination operator.

```
Given a route-plan:
Route-plan: [4] [3 2 7] [8 6 1 5]

1) Select two random nodes from the parent:
Route-plan: [4] [3 2 7] [8 6 1 5]  => Selected nodes: 2 and 1

2) Interchange the positions of the selected nodes within the route-plan:
New route-plan: [4] [3 1 7] [8 6 2 5]
```

Figure 8.5: Swap operator.

```
Given a route-plan:
Route-plan: [4] [3 2 7] [8 6 1 5]

1) Select a random node from the parent:
Route-plan: [4] [3 7 7] [8 6 1 5]  => Selected node: 7

2) Insert the selected node in a random position within the route-plan:
New route-plan: [7 4] [3 2] [8 6 2 5]
```

Figure 8.6: Insert operator.

```
Given a route-plan:
Route-plan: [4] [3 2 7] [8 6 1 5]

1) Select a random Sequence of nodes from the parent:
Route-plan: [4] [3 2 7] [8 6 1 5]  => Sequence: 2 7 8

2) Invert the selected sequence:
New route-plan: [4] [3 8 7] [2 6 1 5]
```

Figure 8.7: Inversion operator.

```
Given a route-plan:
Route-plan: [4] [3 2 7] [8 6 1 5]

1) Select a random Sequence of nodes from the parent:
Route-plan: [4] [3 2 7] [8 6 1 5]  => Sequence: 2 7

2) Invert the selected sequence:
New route-plan: [4] [3] [8 2 7 6 1 5]
```

Figure 8.8: Displacement operator.

This implementation is based on the optimisation framework ParadisEO-MOEO [25]. We extended this implementation to support multiple objectives (Section 8.3.2), process our dataset and use the NSGA-II.

The proposed MOPSO used the same probability for all attractors: $c_x = 0.25, x \in \{1, 2, 3, 4\}$. All particles used the star communication topology. Thus each particle knew the location of all the others at any time. No mechanisms were used to promote diversity within the swarm.

In the experiments, NSGA-II and the proposed MODPSO evolved a population of 25 individuals for 1000 generations. Both algorithms run on each instance 20 times (repetitions), with the exact same operators, probabilities, parameters and seeds.

8.5 Discussion of Results

In order compare the performance of our MODPSO against that of NSGA-II, we use three standard metrics: *hypervolume*, *coverage* and *overall non-dominated vectors* (see Section 2.3.4). We calculate the value of these metrics using the approximation sets obtained by the proposed MODPSO and NSGA-II on Solomon's and MOVRPTW datasets. Tables 8.1, 8.2 and 8.3 present these metric results averaged over 20 runs. These values are also averaged over instance categories for the Solomon's instance sets $\{C1, C2, R1, R2, RC1, RC2\}$ and the MOVRPTW instance sets $\{s0, s10\}$. The second and fourth rows of each table show the average values obtained by MODPSO and NSGA-II. The third and fifth row show the number of instances for which the result is significantly better than the other algorithm in brackets. We ran pair-wise comparisons using the Mann-Whitney-Wilcoxon test to determine which algorithm is better for each instance and metric.

For clarity, we discuss the results obtained for each metric in three subsections. Additionally, we provide an overview of the performance of both algorithms in terms of speed in Section 8.6.

8.5.1 Hypervolume

The hypervolume (or S-metric) measures the portion of space delimited by a non-dominated solutions set and a nadir point. This nadir point is a position in the space dominated by all solutions in the non-dominated set. In these experiments, we set this point to the position dominated by all solutions in the approximation sets obtained by both algorithms in each instance and for all runs. Since this metric represents the section of the objective hyper-space covered by each approximation set, the higher this value is, the better the performance of the algorithm (see Section 2.3.4).

Table 8.1: Normalised hypervolume metric values, averaged over instance categories, for solutions obtained with MODPSO and NSGA-II. The number of instances for which the result is significantly better than the other approach is shown in brackets.

Algorithm	C1	C2	R1	R2	RC1	RC2	s0	s10
MODPSO	0.53	0.78	0.47	0.47	0.68	0.54	0.88	0.90
	(5)	(8)	(7)	(4)	(3)	(3)	(12)	(12)
NSGA-II	0.36	0.31	0.33	0.39	0.55	0.39	0.28	0.25
	(0)	(0)	(2)	(0)	(0)	(0)	(0)	(0)

Table 8.1 presents the average hypervolume values obtained by MODPSO and NSGA-II. MODPSO obtains larger average hypervolume values across all categories in both Solomon's and MOVRPTW datasets. The largest differences are found in Solomon's $C2$ and MOVRPTW $s0$ and $s10$, for which MODPSO obtains 2.5, 3.1 and 3.6 times more average hypervolume than NSGA-II. MODPSO achieves significantly better results in 30 of the 56 Solomon's instances and 24 of the 30 MOVRPTW instances. NSGA-II produces significantly better results in only 2 instances of the Solomon's dataset.

According to the average hypervolume values obtained in these experiments, the proposed MODPSO performs better in all Solomon's instances except $C104$, $R204$, $R208$

and *RC*202. In the MOVRPTW dataset, MODPSO performs better in all instances.

Across all instances in Solomon's dataset, MODPSO gets 2.62 times more average hypervolume than NSGA-II. Moreover, the standard deviation in the MODPSO results is almost half of that of the NSGA-II, which reveals its performance to be more stable. In the MOVRPTW dataset, our MODPSO gets 3.25 times more average hypervolume than NSGA-II. However, the standard deviation in the hypervolume values of MODPSO results is only 0.74 times lower than that of NSGA-II.

Overall, we observe that MODPSO consistently produces better results than NSGA-II according to the hypervolume metric. MODPSO is therefore deemed to produce approximation sets which are more well-spread and closer to the optimal set.

8.5.2 Coverage

The coverage metric measures the extent to which one solution set is covered by another solution set. Thus, according to this metric, the algorithm with the best performance is the one whose approximation solution sets obtain the largest coverage (see Section 2.3.4).

Table 8.2: Coverage metric values, averaged over instance categories, for solutions obtained with MODPSO and NSGA-II. The number of instances for which the result is significantly better than the other approach is shown in brackets.

Algorithm	C1	C2	R1	R2	RC1	RC2	s0	s10
MODPSO	0.48	0.47	0.61	0.39	0.61	0.47	0.75	0.83
	(2)	(7)	(2)	(3)	(5)	(6)	(12)	(12)
NSGA-II	0.17	0.07	0.17	0.15	0.25	0.17	0.10	0.08
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

Table 8.2 presents the average coverage values obtained by MODPSO and NSGA-II. The second row shows the extent to which the solution sets obtained by MODPSO cover those of NSGA-II while the third row shows the opposite. It is clearly seen that MODPSO obtains higher average coverage metric values across all categories. MODPSO produces in the worst case average coverage values twice as good as NSGA-II's in all categories. The largest differences are found in Solomon's *C*2 and MOVRPTW

$s0$ and $s10$, for which MODPSO obtains 6.7, 7.5 and 10.4 times more average coverage than NSGA-II. MODPSO achieves significantly better results in 25 of the 56 Solomon's instances and 24 of the 30 MOVRPTW instances, while NSGA-II does not produce any significantly better result in any instance of either dataset.

Across all instances in both datasets, the proposed MODPSO gets higher average coverage values, except in the Solomon's instance: $C107$. On average, in the Solomon's dataset MODPSO gets 4 times more coverage than NSGA-II in the categories $C1$ and $C2$, 3.1 times in the categories $R1$ and $R2$, and 2.5 in the categories $RC1$ and $RC2$. In the MOVRPTW dataset, MOPSO produces up to 8 times more average coverage than NSGA-II.

These results indicate that the proposed MODPSO produce solution sets that dominate those obtained by NSGA-II, and are therefore closer to the optimal set.

8.5.3 Overall Non-Dominated Vector Generation

The Overall Non-dominated Vector Generation (ONDV) quality indicator measures the number of distinct non-dominated solutions obtained by each algorithm. According to this metric, the higher this number is, the better the algorithm. However, by itself ONDV does not provide enough information to fully compare the quality of two non-dominated sets (see Section 2.3.4).

Table 8.3: ONDV metric values, averaged over instance categories, averaged over instance categories, for solutions obtained with MODPSO and NSGA-II. The number of instances for which the result is significantly better than the other approach is shown in brackets.

Algorithm	C1	C2	R1	R2	RC1	RC2	s0	s10
MODPSO	222.04	**897.65**	186.53	**331.40**	**273.38**	**403.78**	**1181.01**	**1073.88**
	(2)	(8)	(1)	(5)	(4)	(3)	(11)	(11)
NSGA-II	**225.09**	555.93	**207.47**	221.87	228.08	326.18	858.60	780.05
	(2)	(0)	(2)	(0)	(0)	(0)	(0)	(0)

Table 8.3 presents the average number of non-dominated solution vectors obtained by MODPSO and NSGA-II. MODPSO produces better results than NSGA-II in all cate-

gories except $C1$ and $R1$. However, these differences are of only 1.01 and 1.11, respectively. The largest differences are found again in Solomon's $C2$ and MOVRPTW $s0$ and $s10$, for which MODPSO obtains 1.6, 1.4 and 1.4 times more non-dominated solutions than NSGA-II. MODPSO achieves significantly better results in 23 of the 56 Solomon's instances and 22 of the 30 MOVRPTW instances, while NSGA-II does not produce any significantly better result in any instance of either dataset.

The proposed MODPSO gets higher average ONDV values in 41 of the 56 Solomon's instances, and in 27 of the 30 MOVRPTW instances. NSGA-II produces significantly better results in only 4 instances of the Solomon's dataset. Across all Solomon's instances, MODPSO produces an average metric value of 340.66, while NSGA-II reaches 277.12. In the MOVRPTW instances, the respective average values for this metric are of 1152.23 for our MODPSO and of 819.32 for the NSGA-II.

Summarising, it is clear that MODPSO produces better results according to these three quality indicators in both datasets. The overall non-dominated solution vectors metric indicates that MODPSO produces more non-dominated solutions that NSGA-II. The coverage metric shows that a large number of these solutions are not covered by NSGA-II. Finally, the hypervolumen suggests that these solutions are better spread and closer to the optimal set than those produced by the NSGA-II.

It is also worth mentioning that the largest differences in all quality metrics are found in the MOVRPTW dataset. A recent study shows that MOVRPTW dataset provides a more realistic and challenging scenario for the assessment of multi-objective optimisers [37]. This suggests that the stronger (more conflicting) the pair-wise relationships are between objectives, the bigger the advantage of MODPSO is likely to be over NSGA-II.

8.6 Speed Performance

The proposed MODPSO is implemented in CODEA $v3$. Our MODPSO suffers from a high CPU overhead, due to the use of the communication and other mechanisms provided by CODEA. The NSGA-II implementation that we used in these experiments, on the contrary, is developed straight on Paradiseo. For this reason a comparison of

computation time between our MODPSO and NSGA-II is difficult. Thus, rather than comparing the running time of both algorithms, we compare the number of objective functions evaluations. For both algorithms, we always get a number of evaluations equal to: *number of individuals* * *number of generations*. For example, for the parameters stated in Section 8.4, we always get $25 * 1000 = 25000$ evaluation in both algorithms. Regarding the number of operations (crossovers and mutations), the proposed MOPSO performs the exact same number of operations as evaluations. Across all instances in both datasets, NSGA-II performs 1.09% less operations with a standard deviation of 0.002.

It is also worth noticing that, unlike the NSGA-II, the proposed MODPSO does not use any mechanism to promote diversity. In terms of speed the performance of both algorithms can therefore be considered equivalent.

8.7 Conclusions

This chapter presents a simplified MODPSO which consists of two components:

- A simple Discrete Particle Swarm Optimisation (DPSO) algorithm [54].
- Dominance Depth Fitness Assignment [116] to update the leaders of the swarm.

The development of this MODPSO was motivated by the number of research works which use complex mechanisms to boost convergence and diversity. Some of these mechanisms include crowding distance, niche count and archiving techniques (see Section 2.5). Furthermore, most MOPSO based approaches deal with continuous problems with few objectives (three or less).

Our MOPDSO only uses the key component of the non-dominated sorting [65], the Dominance Depth Fitness Assignment [116]. This fitness assignment scheme in combination with a simple DPSO [54] seems to be enough to outperform NSGA-II in dealing with the VRPTW with five objectives: 1) *number of vehicles* needed to serve all customers, 2) *total travel distance*, 3) *makespan* or travel time of the longest route (from/to depot), 4) *total waiting time*, and 5) *total delay time*. Results in the MOVRPTW and

Solomon's datasets are assessed by using three quality indicators: 1) hypervolume, 2) coverage, and 3) overall non-dominated vector generations. Moreover, our MODPSO seems to perform much better on the MOVRPTW instances which, as we showed in the previous chapter, are more suitable for the multi-objective assessment than those of Solomon's. Regarding the speed performance, both the proposed MODPSO and NSGA-II perform the same number of evaluations and almost the same number of operations (mutations and crossovers).

All NSPSO algorithms presented in Section 8.1 provide a better performance than that of NSGA-II. However, many of these algorithms include a number of specific mechanisms to boost convergence and diversity. We proposed here a MODPSO that is very simple and still produces much better results than NSGA-II. These results raise a series of research questions: 1) which mechanisms are really contributing to improve the results in NSPSO and other MOPSO algorithms discussed in Section 8.1?, 2) what does happen when these algorithms have to deal with more objectives?, 3) is the performance of NSPSO algorithms better than the performance of NSGA-II in combinatorial search spaces? In order to answer these questions, further research should be done to find which components actually contribute to achieve better results, and how they affect the performance. This could provide a better understanding on which components are suitable to each type of problem. This information would then lead to the design of better techniques for addressing multi-objective optimisation problems.

CHAPTER 9

Conclusions and future work

9.1 Conclusions

This thesis presents an investigation of different aspects of Multi-Objective Optimisation (MOO) and the Vehicle Routing Problem with Time Windows (VRPTW), and proposes four tools for the further development of this area:

1. CODEA − a COoperate DEcentralised Architecture for tackling multi-objective optimisation problems (MOP).
2. DLA − a Dynamic Lexicographic Approach to discriminate solutions with multiple objectives using random lexicographic orderings.
3. MOVRPTW dataset − a realistic and challenging dataset for the Vehicle Routing Problem with Time Windows (VRPTW), and a configurable dataset generator.
4. MODPSO algorithm − a simplified Multi-objective Optimisation Discrete Particle Swarm Optimiser to tackle the VRPTW. The following subsections provide a brief description and results of each of these tools.

9.1.1 CODEA − COoperative DEcentralised Architecture

CODEA is an object-oriented framework for the creation of groups of agents to tackle MOP problems by cooperative search. This cooperation is carried out without any individual controlling the cooperation nor the behaviour of the agents. Each agent works

solely to improve itself and collaborates to improve the performance of the group by sharing information.

A number of feature-rich frameworks have been proposed to tackle MOPs, such as JMetal, ECJ, HeuristicLab and ParadisEO. These frameworks are studied in Chapter 3. This study shows that none of these frameworks provides explicit mechanisms to create cooperative groups of agents. CODEA has been evolving for almost five years to fill this gap. We present CODEA in two versions: CODEA $v2$ and CODEA $v3$. CODEA $v2$ is the first version of CODEA that supports MOPs. CODEA $v3$ is the product of the hybridisation of CODEA $v2$ and ParadisEO-MOEO [166]. This new framework enables the creation of groups of cooperative agents with all the features provided by ParadisEO-MOEO. As a result, CODEA $v3$ presents a number of competitive features with respect to those provided by state-of-the-art MOO frameworks as shown in Table 3.1 in Chapter 3.

Most research in this thesis has been conducted using CODEA $v2$ or CODEA $v3$ as a vehicle to test different multi-objective techniques. However, CODEA has also played an important role in four other studies.

- The first study was the creation of a canonical MODPSO to tackle VRPTW with 8 objectives. Results showed that this basic MODPSO was able to evolve very poor quality infeasible solutions (randomly initialised) to the feasible region.
- In the second study, the canonical MODPSO crafted for VRPTW was adapted to solve the Steiner Tree in Graph and Delay Constrained Multicast Routing Problems. A large number of simulations showed that the resulting algorithms was quite competitive with respect to state-of-the-art algorithms in the literature.
- The third study introduced CODEA $v2$. This work provided an overview of key study of this version of CODEA. As a practical example, three communication topologies (start, ring and k-random) were tested on bi-objective TSP instances. According to the results, the k-random topology was the best strategy to communicate information, and the ring topology offered the best compromise between speed and performance.

- This fourth study adapted the canonical MODPSO crafted for VRPTW to the University Course Timetabling Problem (UCTTP). Two algorithms involving groups of asynchronous agents were created within CODEA. Results in standard UCTTP instances show that the proposed algorithms found better results outperforming all other existing approaches in the literature of the problem.

9.1.2 DLA − Dynamic Lexicographic Approach

DLA is a new ranking approach to discriminate solutions with multiple objectives (Chapter 4). DLA uses random lexicographic orderings based on certain probability distribution set by the decision maker. This ranking scheme does not require to set fix priorities among objectives, but merely a preference. This preference is used with a probability mass function (*pmf*) to generate vector of priorities which changes throughout the search process. This ranking scheme was tested in a simple Discrete Particle Swarm Optimisation (DPSO) to tackle the Vehicle Routing Problem with Time Windows (VRPTW). The experiments were carried out on the Solomon's 100 customers instances [233]. We compared the performance of two variants of DLA to that of Pareto dominance and Lexicographic ordering. One variant of the DLA used one *pmf* (denoted as DLA), and the other used a double *pmf* (denoted as DLA2). DLA2 combined an intensification and diversification phase. For this purpose, DLA2 used a *pmf* which only takes into account the two objectives with the highest preference for the first k number of iterations (intensification), and another *pmf* which takes into account all objectives for the remainder iterations (diversification). The performance was assessed using the hypervolume quality indicator [271]. This indicator was calculated using the non-dominated archives obtained by each ranking scheme using 8 objectives. From the results we draw the following conclusions:

- DLA2 produces much better results than Pareto dominance and Lexicographic ordering across all instances of the Solomon's dataset.
- The success of this new ranking approach calls into question the suitability of standard ranking methods for dealing with certain problems, in particular those

problems with a large number of objectives as pointed out by other authors in [138, 152, 203].

9.1.3 MOVRPTW dataset

We used Solomon's dataset to test the DLA, because the most common way to compare heuristics is by the use of these benchmark problems [20]. However, Solomon's dataset was not designed for the assessment of multi-objective solving techniques. Results in the experiments conducted with DLA indicate that this dataset might not be appropriate for comparing multi-objective algorithms. Chapter 5 provides an overview of the main characteristics of the Solomon's dataset. A special emphasis is drawn on several unrealistic features which might affect its potential multi-objective suitability. Aspects that might make this dataset inappropriate for multi-objective assessment are:

- Equality between distance and time matrices. Solomon's dataset only provides the customers geographic location. Thus, Euclidean distances must be used to calculate both travel distance and travel times.
- Symmetry in distance and time matrices. In Solomon's dataset, it is assumed that for any two pair of customers, both travel time and travel distance between them are the same regardless of who is visited first.

In order to overcome the limitations encountered with Solomon's dataset, we presented MOVRPTW in Chapter 6. MOVRPTW is a novel dataset based on data obtained from a distribution company located in Spain. The company provided us information about the geographic location, time windows profiles and type of demands of more than 1000 customers. We used Google Maps database to calculate travel time and travel distance between each pair of customers. MOVRPTW consists of 30 instances of 100 customers each. These instances have five time windows profiles, three types of demands and three types of service times. As in Chapter 5, we analysed those aspects that might make this dataset more suitable for the assessment of multi-objective algorithms. We draw the following conclusions:

- No equality and no symmetry between travel distances and travel times. For each pair of customers, we calculated travel distances and travel times in both directions using Google Maps database.

- Consideration of low speeds in urban areas. Due to the use of Google Maps database, we have real-based travel distances and travel times. In populated areas in which traffic congestions are frequent, there might be a considerable gap between travel distances and travel times. This gap might be an important factor to create better trade-off surfaces between objectives.

Chapter 7 provided an in-depth study comparing the multi-objective suitability of the Solomon's and MOVRPTW datasets. This comparison was based on the work by Purshouse and Fleming [204]. According to their study, there are three possible relationships between pairs of objectives: 1) conflict (if the improvement in one objective leads to worsen the other), 2) harmony (if the improvement in one objective witnesses an enhancement in the other), and 3) interdependence (if the improvement in one objective does not affect the other). We were interested in the dependence relationships as they present a real challenge to multi-objective algorithms. The work by Purshouse and Fleming provides an overview of qualitative methods for the analysis in multivariate studies. Two important qualitative methods are Parallel plots (P-plots) and Scatter plots (S-plots). Wegman [257] and Li et al. [163] related the notion crossing lines in P-plots with the correlation. We therefore use the correlation to study the relationships that arise between pairs of objectives in non-dominated solution sets. The closer the correlation values between pair of objectives are to 1 or -1, the stronger is their dependence relationship. We calculated correlation values between pairs of five common objectives using non-dominated sets obtained by NSGA-II [65] in both datasets. Results show that the MOVRPTW dataset presents more realistic and challenging multi-objective scenarios compared to the Solomon's dataset. This study also revealed that Solomon's instances with narrow time windows are not adequate for the assessment of multi-objective algorithms. Solomon's instances with wider time windows produced better results but still not sufficient for multi-objective benchmarking purposes.

9.1.4 A simplified MODPSO for the VRPTW

A simplified Multi-Objective Discrete Particle Swarm Optimisation (MODPSO) was introduced in Chapter 8. This MODPSO was the first algorithm implemented in CODEA $v3$: the product of the hybridisation of CODEA $v2$ and ParadisEO-MOEO. Our MODPSO has two main components: 1) a Multi-objective component to update the leaders based on the notion of Dominance Depth Fitness Assignment introduced in [116], and 2) a Discrete Particle Swarm Optimisation (DPSO) algorithm based on the work by Consoli et al. [54]. Our MOPDSO does not employ any extra mechanism to promote convergence or diversity.

This simplified MODPSO was applied to VRPTW with 5 common minimisation objectives. A number of simulations compared the performance of this MODPSO to that of NSGA-II on Solomon's and MOVRPTW datasets. In order to assess the performance of both multi-objective optimisers, we used two standard quality indicators: hypervolume [271] and coverage [273]. Results using these metrics indicated that:

- Hypervolume: MODPSO performs better in all Solomon's instances except C104, R204, R208 and RC202, and in all MOVRPTW instances. On average, MODPSO gets 2.62 and 3.25 times more hypervolume than NSGA-II in the Solomon's and MOVRPTW datasets, respectively.
- Coverage: MODPSO gets higher average coverage values in all instances except in the Solomon's C107, and in all MOVRPTW instances. On average, MODPSO gets 3.2 and up to 8 times more coverage than NSGA-II in Solomon's and MOVRPTW datasets, respectively.

Regarding speed performance, it was difficult to compare MODPSO and NSGA-II, as MODPSO runs on CODEA $v3$ and NSGA-II runs on ParadisEO-MOEO. Thus, the proposed MODPSO suffered from a hight CPU overhead. Rather than comparing the running time of both algorithms, we compared: 1) the number of evaluations of the objective functions, and 2) the number of operations (crossovers and mutations). For both algorithms, and for a fixed number of iterations/generations, we always obtained the

same number of evaluations. Regarding the number of operations, NSGA-II performs 1.09% less operations than the proposed MODPSO across all instances in both dataset.

9.2 Proposed Future work

The present thesis deepens into a number of trending topics related to both Multi-Objective Optimisation (MOO) and the Vehicle Routing Problem with Time Windows (VRPTW). It is our belief that the studies here presented opens new research avenues concerning topics, such as multi-objective ranking methods, real-world based datasets and software frameworks for multi-objectives optimisation.

The work presented in this thesis could be extended on in a number of ways. In this section we summarise a number of research lines.

One important research line that we are willing to explore is the creation of more complex groups of cooperative agents within CODEA. For this purpose, we could use CODEA's features, such us the creation/destruction of agents in real-time, complex communication topologies and the use of rich-content messages.

Some effort should be made into the development of a detailed documentation of the API. Both technical information and practical examples should be provided in order to create a strong foundation for new users. ParadisEO has a number of tutorials that explain how to use this framework with a learn-by-example methodology [142]. This methodology could be applied to CODEA by creating tutorials and providing the source code of all test cases discussed in this thesis.

We plan to implement the Dynamic Lexicographic Approach (DLA) in CODEA $v3$, so that we can compare DLA against other ranking methods (e.g. ϵ-dominance [161]). Furthermore, a number of simulations on the MOVRPTW dataset could be used to test the robustness of this ranking approach. The study of the probability mass functions (*pmf*) used within DLA is also worth an in-depth study. It might be interesting to study how to change k (intensification/exploration switcher) dynamically during the search.

The study regarding the multi-objective suitability of Solomon's and MOVRPTW datasets

could also be extended. Further research could be done into finding which elements contribute to producing higher correlation values between pairs of objectives. Based on these findings, experimental analysis could be carried out in tailored instances created with the MOVRPTW generator. Furthermore, Solomon's dataset could be improved by using the Manhattan distance notion to calculate the distance between pair of customers instead of using the Euclidean distance. The resulting distance could also be multiplied by a noise factor, so that travel distance and travel time matrices are non-equal and non-symmetric.

Finally, there are some ways to extend the MODPSO algorithm proposed in Chapter 8. Some enhancements for this MODPSO include mechanisms such as diversity preserving mechanisms, archives or more efficient communication topologies. Additionally, this algorithm might be applied to other problem domains in order to test its robustness.

References

[1] J. Ai and V. Kachitvichyanukul. A particle swarm optimization for the capacitated vehicle routing problem. *International Journal of Logistics and SCM Systems*, 2(1):50–55, 2007.

[2] J. Ai and V. Kachitvichyanukul. Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, 56(1):380–387, 2009.

[3] K. Altinkemer and B. Gavish. Parallel savings based heuristics for the delivery problem. *Operations Research*, 39(3):456–469, 1991.

[4] G. Alvarenga, G. Mateus, and G. Detomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers and Operations Research*, 34(6):1561–1584, 2007.

[5] J. Alvarez-Benitez, R. Everson, and J. Fieldsend. A MOPSO algorithm based exclusively on pareto dominance concepts. In *Evolutionary Multi-Criterion Optimization*, volume 3410, pages 459–473. Springer Berlin / Heidelberg, 2005.

[6] D. Anghinolfi and M. Paolucci. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 193(1): 73–85, 2009.

[7] R. Baldacci, D. Vigo, and P. Toth. *Exact Solution of the Capacitated Vehicle Routing Problem*, pages 1795–1807. John Wiley & Sons, Inc., 2011.

[8] J. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002.

[9] T. Bartz-Beielstein, P. Limbourg, J. Mehnen, K. Schmitt, K. Parsopoulos, and M. Vrahatis. Particle swarm optimizers for pareto optimization with enhanced archiving techniques. In *Congress on Evolutionary Computation, 2003. (CEC '03)*, volume 3, pages 1780–1787 Vol.3, 2003.

[10] U. Baumgartner, C. Magele, and W. Renhart. Pareto optimality and particle swarm optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, 2004.

[11] J. Beasley. Route first–Cluster second methods for vehicle routing. *Omega*, 11(4): 403–408, 1983.

[12] R. Benabid, M. Boudour, and M. Abido. Optimal location and setting of SVC and TCSC devices using non-dominated sorting particle swarm optimization. *Electric Power Systems Research*, 79(12):1668–1677, 2009.

[13] R. Benayoun, J. Montgolfier, J. Tergny, and O. Laritchev. Linear programming with multiple objective functions: Step method (stem). *Mathematical Programming*, 1(1):366–375, 1971.

[14] J. Berger, M. Barkaoui, and O. Bräysy. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *Information Systems and Operations Research*, 41(2):179–194, 2003.

[15] A. Bernal and H. Castro. pals: An object-oriented framework for developing parallel cooperative metaheuristics. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW 2010)*, pages 1–8, April 2010.

[16] D. Bertsimas and D. Simchi-levi. A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. *Operations Research*, 44:286–304, 1993.

[17] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal Of Operational Research*, 181 (3):1653–1669, 2007.

[18] J. Brans, P. Vincke, and B. Mareschal. How to select and how to rank projects: the PROMETHEE method. *European Journal of Operational Research*, 24:228–238, 1986.

[19] O. Bräysy. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal of Computing*, 15(4):347–368, 2003.

[20] O. Bräysy and M. Gendreau. Tabu search heuristics for the vehicle routing problem with time windows. *Top*, 10(2):211–237, 2002.

[21] 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, 2005.

[22] A. Breedam. *An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle related, customer-related, and time-related constraints*. PhD thesis, University of Antwerp, RUCA, Belgium, 1994.

[23] D. Brockhoff and E. Zitzler. Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In Thomas Runarsson, Hans-Georg Beyer, Edmund Burke, Juan Merelo-Guervós, L. Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 533–542. Springer Berlin / Heidelberg, 2006.

[24] E. Burke and G. Kendall. *Search methodologies - Introductory tutorials in optimization and decision support technologies*. Springer, 2005.

[25] S. Cahon, N. Melab, and E. Talbi. ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics. *Journal of heuristics*, 10:357–380, 2004.

[26] J. Castro-Gutierrez and D. Landa-Silva. Dynamic lexicographic approach for many-objective combinatorial optimisation. *Preparing for submission*, 2012.

[27] J. Castro-Gutierrez and D. Landa-Silva. A simplified multi-objective discrete particle swarm optimisation for the vehicle routing problem with time windows. *Preparing for submission*, 2012.

[28] J. Castro-Gutierrez and D. Landa-Silva. Investigating pair-wise relationships between objectives in the vehicle routing problem with time windows. *Preparing for submission*, 2012.

[29] J. Castro-Gutierrez and D. Landa-Silva. Asynchronous cooperative multi-agent search for the university course time tabling problem. *Preparing for submission*, 2012.

[30] J. Castro-Gutierrez, B. Melian Batista, J. Moreno Perez, M. Moreno Vega, and J. Ramos Bonilla. Codea: An architecture for designing nature-inspired cooperative decentralized heuristics. In *Proceedings of the 2007 Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2007), Series Studies in Computational Intelligence, Vol. 129*, pages 189–198. Springer, 2008.

[31] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimization approach. In N. Krasnogor, B. Melián-Batista, J. Pérez, J. Moreno-Vega, and D. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, volume 236 of *Studies in Computational Intelligence*, pages 103–114. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-03210-3.

[32] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Dynamic lexicographic approach for heuristic multi-objective optimization. In *Proceedings of the Workshop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009)*, pages 153–163, Seville (Spain), 2009.

[33] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Codea - an agent based multi-objective optimization framework. In *VII Congreso Español sobre Metaheuristicas, Algoritmos Evolutivos y Bioinspirados (MAEB 2010)*, pages 319–327, Valencia, Spain, September 2010.

[34] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Pérez. Multi-objective vehicle routing problem with time windows dataset. https://github.com/psxjpc/, 2010.

[35] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Codea - COperative Decentralised Architecture. https://github.com/psxjpc/CODEA, 2011.

[36] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 257–264, 2011.

[37] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Improved dynamic lexicographic ordering for multi-objective optimisation. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 31–40. Springer Berlin / Heidelberg, 2011.

[38] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research*, 33:2972–2990, 2006.

[39] A. Chen, G. Yang, and Z. Wu. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University - Science A*, 7:607–614, 2006. ISSN 1673-565X.

[40] P. Cheng, J. Pan, L. Li, Y. Tang, and C. Huang. A survey of performance assessment for multiobjective optimizers. In *Genetic and Evolutionary Computing (ICGEC), 2010 Fourth International Conference on*, pages 341 –345, 2010.

[41] W. Chiang and R. Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9(4):417–430, 1997.

[42] N. Christofides. The vehicle routing problem. *RAIRO*, 10:55–70, 1976.

[43] N. Christofides, A. Mingozzi, and P. Toth. *The vehicle routing problem*. Combinatorial Optimization. John Wiley and Sons, Chichester, 1979.

[44] R. Claes, T. Holvoet, and D. Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):364–373, 2011.

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

[46] M. Clerc. *New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing*, pages 219–239. Springer, Heidelberg, 2004.

[47] C. Coello. A comprehensive survey of Evolutionary-Based multiobjective optimization techniques. *Knowledge and Information Systems*, 1:269–308, 1998.

[48] C. Coello. Evolutionary multi-objective optimization: Some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, 3(1):18–30, 2009.

[49] C. Coello and M. Lechuga. MOPSO: a proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation, 2002. CEC '02.*, volume 2, pages 1051–1056, 2002.

[50] C. Coello, D. Van-Veldhuizen, and G. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Kluwer academic publishers, 2002.

[51] C. Coello, G. Pulido, and M. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.

[52] C. Coello, G. Lamont, and D. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer US, 2007. ISBN 978-0-387-33254-3.

[53] Y. Collette and P. Siarry. *Multiobjective optimization: principles and case studies*. Springer, 2003.

[54] S. Consoli, J. Moreno-Pérez, K. Darby-Dowman, and N. Mladenović. Discrete particle swarm optimization for the minimum labelling steiner tree problem. *Natural Computing*, 9(1):29–46, 2010.

[55] W. Cook and J. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, Computational and Applied Mathematics Department, Rice University, 1999.

[56] J. Cordeau, G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis. *VRP with Time Windows*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-498-2.

[57] J. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001.

[58] J. Cordeau, G. Laporte, M. Savelsbergh, and D. Vigo. Short-haul routing. Technical report, HEC Technical Report, 2005.

[59] J. Cordeau, G. Laporte, M. Savelsbergh, and D. Vigo. Vehicle routing. In *Handbook in Operation Research and Management Science*, number 14 in Surveys in Operations Research and Management Science, pages 367–428. Elsevier B.V., 2007.

[60] E. Correa, A. Freitas, and C. Johnson. A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 35–42, Seattle, Washington, USA, 2006. ACM.

[61] G. Croes. A method for solving traveling-salesman problem. *Operations Research*, 6:791–812, 1958.

[62] E. Danna and C. Le Pape. *Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows*, chapter 3, pages 30–130. Springer, 2005.

[63] K. Deb. Solving goal programming problems using multi-objective genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.*, volume 1, pages 77–84, 1999.

[64] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast elitist Non-Dominated sorting genetic algorithm for Multi-Objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature*, pages 849–858, 2000.

[65] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

[66] K. Deb, M. Mohan, and S. Mishra. Evaluating the epsilon-dominance based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary computation*, 13(4):501–525, 2005.

[67] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

[68] I. García del Amo, F. García López, M. García Torres, B. Melián Batista, J. Moreno Pérez, and J. Moreno Vega. *From Theory to Implementation: Applying Metaheuristics.*, chapter 11, pages 311–351. Springer, 2006. ISBN 0-387-28260-2.

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

[70] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle-routing problem with time windows. *Operations Research*, 40(2): 342–354, 1992.

[71] F. Di-Pierro. *Many-objective evolutionary algorithms and applications to water resources engineering*. PhD thesis, University of Exeter, 2006.

[72] C. Ding, L. Lu, Y. Liu, and W. Peng. Swarm intelligence optimization and its applications. In G. Shen and X. Huang, editors, *Advanced Research on Electronic Commerce, Web Application, and Communication*, volume 143 of *Communications in Computer and Information Science*, pages 458–464. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20367-1.

[73] J. Doran, S. Franklin, N. Jennings, and T. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12:309–314, 1997.

[74] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics - part B: cybernetics*, 26(1):29–41, 1996.

[75] B. Dorronsoro. VRP web. http://neo.lcc.uma.es/radi-aeb/WebVRP/, 2010.

[76] M. Doumpos and C. Zopounidis. *Multicriteria Decision Aid Classification Methods*, volume 73 of *Applied Optimization*. Springer, 2002. ISBN 978-1-4020-0805-4.

[77] G. Dueck. New optimization heuristics the great deluge algorithm and the record-to-record travel. *Journal of computational physics*, 104:86–92, 1993.

[78] G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90:161–175, 1990.

[79] W. Dullaert and O. Bräysy. Routing relatively few customers per route. *TOP*, 11: 325–336, 2003.

[80] J. Durillo and A. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760 – 771, 2011.

[81] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39 –43, Nagoya, Japan, 1995.

[82] ECLab. A Java-based Evolutionary Computation Research System (ECJ). http://cs.gmu.edu/ eclab/projects/ecj/, 2011.

[83] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR spectrum*, 22(4):425–460, 2000.

[84] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Survey*. Number 52 in International Series in Operations Research and Management Science. Springer, Boston, Kluwer, 2002.

[85] N. El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University (Science)*, pages 123–131, 22.

[86] P. Engrand and X. Mouney. Une méthode originale d'optimisation multi-objectif. Technical Report EDF-DER - 98-NJ-00005, lectricité de France, Direction des études et recherches, Clamart, FRANCE, Clamart, France, 1998.

[87] EO Team. Evolving Objects (EO). http://eodev.sourceforge.net, 2011.

[88] H. Fan. Discrete particle swarm optimization for tsp based on neighborhood. *Journal of Computational Information Systems*, 6(10):3407–3414, 2010.

[89] M. Farina and P. Amato. On the optimal solution definition for many-criteria optimization problems. In *Proceedings of the NAFIPS-FLINT International Conference'2002,*, pages 233–238, Piscataway, 2002. IEEE Service Center.

[90] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44:216–229, 2004.

[91] Y. Feng, B. Zheng, and Z. Li. Exploratory study of sorting particle swarm optimizer for multiobjective design optimization. *Mathematical and Computer Modelling*, 52(11-12):1966–1975, 2010.

[92] J. Ferber. *Multi-Agent Systems: An Introduction to Artificial Intelligence*. Addison-Wesley, 1999.

[93] J. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimisation. In *The UK Workshop on Computational Intelligence*, pages 34–44, 2002.

[94] M. Figliozzi. Vehicle routing problem for emissions minimization. *Journal of the Transportation Research Board*, 2197 / 2010:1–7, 2010.

[95] P. Fishburn. Lexicographic orders, utilities and decision rules: A survey. *Management Sciences*, 20(11):1442–1471, 1974.

[96] M. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[97] M. Fisher and J. Jaikumar. A generalized assigment heuristic for the vehicle routing problem. *Networks*, 11:109–124, 1981.

[98] M. Fisher, K. Jörnsten, and O. Madsen. Vehicle routing with time windows – two optimization algorithms. *Operations Research*, 45:488–492, 1997.

[99] M. Fonseca and P. Fleming. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, pages 416–423. Morgan Kaufmann Publishers, 1993.

[100] M. Fonseca and P. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the 5th international conference on genetic algorithms*, pages 416–423, 1993.

[101] B. Foster and D. Ryan. An integer programming approach to the vehicle scheduling problem. *Operations Research*, 27:367–384, 1976.

[102] Free Software Foundation (FSF). GNU Lesser General Public License (LGPL). http://www.gnu.org/licenses/lgpl.html, 2007.

[103] C. Gagné and M. Parizeau. Open BEAGLE: a c++ framework for your favorite evolutionary algorithm. *SIGEVOlution*, 1:12–15, April 2006.

[104] L. Gambardella, É. Taillard, and G. Agazzi. *MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows*, pages 63–76. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[105] A. Garcia-Najera and J. Bullinaria. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 38(1):287 – 300, 2011.

[106] T. Gaskell. Bases for vehicle fleet scheduling. *Journal of the Operational Research Society*, 18(3):281–295, 1967.

[107] H. Gehring and J. Homberger. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics*, 8:251–276, 2002.

[108] M. Gendreau and C. Tarantilis. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Technical report, CIRRELT, February 2010.

[109] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[110] H. Ghaziri. *Algorithmes connexionistes pour l'optimisation combinatoire*. PhD thesis, Ecole Polytechnique Federale de Lausanne, Switzerland, École Polytechnique Fédérale de Lausanne, Switzerland, 1993.

[111] K. Ghoseiri and S. Ghannadpour. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10(4):1096–1107, 2010.

[112] B. Gillett and L. Miller. A heuristic algorithm for the vehicle routing problem. *Operations Research*, 22:340–349, 1974.

[113] F. Glover. New ejection chain and alternating path methods for traveling salesman problems. *Computer Science and Operations Research*, pages 449–509, 1992.

[114] F. Glover, E. Taillard, and D. De Werra. A user's guide to tabu search. *Annals of operations research*, 41:3–28, 1993.

[115] E. Goldbarg, G. de Souza, and M. Goldbarg. Particle swarm for the traveling salesman problem. In Jens Gottlieb and Günther Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 99–110. Springer Berlin / Heidelberg, 2006.

[116] D. Goldberg. *Genetic algorithms in search, optimisation and machine learning*. Addison Wesley, 1989.

[117] D. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on*

Genetic Algorithms on Genetic algorithms and their application, pages 41–49, Cambridge, Massachusetts, United States, 1987. L. Erlbaum Associates Inc.

[118] B. Golden, A. Assad, and E. Wasil. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In Paolo Toth and Daniele Vigo, editors, *The vehicle routing problem*, pages 245–286. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[119] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research Computer Science Interfaces Series*. Springer, 2010.

[120] C. Grosan, M. Oltean, and D. Dumitrescu. Performance metrics for multiobjective optimization evolutionary algorithms. In *Proceedings of Conference on Applied and Industrial Mathematics (CAIM)*, Oradea, 2003.

[121] G. Gutin and A. Punnen, editors. *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Springer, 2002.

[122] Y. Haimes and A. Hall. Multiobjectives in water resource systems analysis: The surrogate worth trade off method. *Water Resources Research*, 10(4):615–624, 1974.

[123] M. Haimovich and A. Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.

[124] P. Hajela and C. Lin. Genetic search strategies in multicriterion optimal design. *Structural and Multidisciplinary Optimization*, 4(2):99–107, 1992.

[125] U. Hekke, T. Simarik, and T. Vironda. Graphical user interface for multi objective optimization. http://guimoo.gforge.inria.fr/, 2009.

[126] M. Helbig and A. Engelbrecht. Archive management for dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2047 –2054, 2011.

[127] S. Helmreich and H. Keller, editors. *FREIGHTVISION - Sustainable European Freight Transport 2050*. Springer-Verlag Berlin Heidelberg, 2011.

[128] J. Hettenhausen, A. Lewis, and S. Mostaghim. Interactive multi-objective particle swarm optimization with heatmap-visualization-based user interface. *Engineering Optimization*, 42(2):119–139, 2010.

[129] S. Ho, S. Yang, G. Ni, E. Lo, and H. Wong. A particle swarm optimization-based method for multiobjective design optimizations. *Magnetics, IEEE Transactions on*, 41(5):1756–1759, 2005.

[130] M. Hoffmann, M. Mühlenthaler, S. Helwig, and R. Wanka. Discrete particle swarm optimization for TSP: Theoretical results and experimental evaluations. In Abdelhamid Bouchachia, editor, *Adaptive and Intelligent Systems*, volume 6943 of *Lecture Notes in Computer Science*, pages 416–427. Springer Berlin / Heidelberg, 2011.

[131] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Information Systems and Operational Research*, 37(3):297–318, 1999.

[132] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal Of Operational Research*, 162(1):220–238, 2005.

[133] J. Horn, N. Nafpliotis, and D. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the 1st IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, volume 1, pages 82–87, 1994.

[134] X. Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the Evolutionary Computation on 2002. CEC '02.*, pages 1677–1681. IEEE Computer Society, 2002.

[135] X. Hu, R. Eberhart, and Y. Shi. Swarm intelligence for permutation optimization: a case study of n-queens problem. In *Proceedings of the Swarm Intelligence Symposium, 2003. SIS 03.*, pages 243–246, 2003.

[136] X. Hu, Q. Ding, Y. Li, and D. Song. Computational intelligence and security. In Y. Wang, Y. Cheung, and H. Liu, editors, *Lecture Notes in Computer Science*, chapter An Improved Ant Colony System and Its Application, pages 36–45. Springer-Verlag, Berlin, Heidelberg, 2007.

[137] D. Huggins. *Panorama of Transport*. Statistical books. Office for Official Publications of the European Communities, 2009.

[138] E. Hughes. Evolutionary many-objective optimisation: many once or one many? In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 222 –227, 2005.

[139] IBM. Ibm ilog cplex optimizer. http://www.ibm.com/cplex, 2011.

[140] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[141] C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[142] INRIA Lille - Nord Europe. ParadisEO. http://paradiseo.gforge.inria.fr/, 2011.

[143] G. Ioannou, M. Kritikos, and G. Prastacos. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *The Journal of the Operational Research Society*, 52(5):523–537, 2001.

[144] C. Ishida, A. Carvalho, A. Pozo, E. Goldbarg, and M. Goldbarg. *Exploring multi-objective PSO and GRASP-PR for rule induction*, volume 4972 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin / Heidelberg, 2008.

[145] N. Jozefowiez, F. Semet, and E. Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2):293–309, 2008.

[146] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers and Operations Research*, 35(7):2307–2330, 2008.

[147] B. Kallehauge, J. Larsen, and O. Madsen. Lagrangean duality applied to the vehicle routing with time windows. *Computers and Operations Research*, 33:1464–1487, 2006.

[148] R. Keeney. Decision analysis: An overview. *Operations Research*, 30(5):803–838, 1982.

[149] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving objects: a general purpose evolutionary computation library. In *5th International Conference in Evolutionary Algorithms (EA'01)*, pages 231–244, Le Creusot, France, 2001.

[150] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 4, pages 1942–1948, 1995.

[151] J. Kennedy and R. Eberhart. Discrete binary version of the particle swarm algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4104–4108, 1997.

[152] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In Carlos Fonseca, Peter Fleming, Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 72–72. Springer Berlin / Heidelberg, 2003.

[153] J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *Proceedings of the 2002 congress on evolutionary computation (CEC 2002)*, pages 711–716. IEEE press, 2002.

[154] N. Kohl and O. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangean relaxation. *Operations Research*, 45: 395–406, 1997.

[155] N. Kohl, J. Desrosiers, O. Madsen, M Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1): 101–116, 1999.

[156] A. Kolen, A. Rinnooy, and H. Trienekens. Vehicle routing with time windows. *Operations Research*, 35(1):256–273, 1987.

[157] A. Land and A. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[158] A. LaTorre, F. Clautiaux, E. Talbi, and J. Peña. VRP-extended: When confidence and fleet size are also important. In E. Talbi, editor, *Proceedings of the 2nd International Conference on Metaheuristics and Nature Inspired Computing, META 2008*, 2008.

[159] H Lau. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal Of Operational Research*, 148(3):559–569, 2003.

[160] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary Multi-Objective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[161] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[162] M. Lewis. Computer language benchmarks game. http://shootout.alioth.debian.org/u32/performance.php?test=nbody, 2012.

[163] J. Li, J. Martens, and J. Wijk. Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization*, 9(1):13–30, 2008.

[164] X. Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Proceedings of the 2003 genetic and evolutionary computation conference (GECCO 2003), LNCS 2723*, pages 37–48. Springer, 2003.

[165] A. Liefooghe, L. Jourdan, and E. Talbi. A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO. Research Report RR-6906, INRIA, 2009.

[166] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, and E.-G. Talbi. *ParadisEO-MOEO: A software framework for evolutionary multi-objective optimization*, volume 272/2010 of *Studies in Computational Intelligence*, pages 87–117. Springer, 2010.

[167] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[168] S. Lin and B. Kernighan. An effective heuristic algorithm for the Traveling-Salesman problem. *Operations Research*, 21(2):498–516, 1973.

[169] G. Lizárraga, A. Hernández, and S. Botello. A set of test cases for performance measures in multiobjective optimization. In Alexander Gelbukh and Eduardo Morales, editors, *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pages 429–439. Springer, 2008.

[170] LogMan. External Factors and Carbon Footprints. Technical report, LOGMAN, 2010.

[171] A. López, C. Coello, and D. Chakraborty. Objective reduction using a feature selection technique. *Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO 08*, page 673, 2008.

[172] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4j: a modular framework for meta-heuristic optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1723–1730, New York, USA, 2011. ACM.

[173] S. Luke. The ECJ owner's manual. Technical report, Evolutionary Computation Laboratory, George Mason University, 2010.

[174] J. Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175(1):210–223, 2006.

[175] Y. Marinakis, M. Marinaki, and G. Dounias. A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 23(4):463 – 472, 2010. ISSN 0952-1976.

[176] R. Martí, M. Laguna, and F. Glover. Principles of scatter search. *European Journal of Operational Research*, 169:359–372, 2006.

[177] D. Mester and O. Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers and Operations Research*, 32 (6):1593–1614, 2005.

[178] S. MirHassani and N. Abolghasemi. A particle swarm optimization algorithm for open vehicle routing problem. *Expert Systems with Applications*, 38(1):11547–11551, 2011.

[179] L. Mohamed, M. Christie, and V. Demyanov. History matching and uncertainty quantification: Multiobjective particle swarm optimisation approach. In *Society of Petroleum Engineers - 73rd European Association of Geoscientists and Engineers Conference and Exhibition 2011 - Incorporating SPE EUROPEC 2011*, volume 4, pages 2927–2943, 2011.

[180] R. Mole and S. Jameson. A sequential Route-Building algorithm employing a generalised savings criterion. *Operational Research Quarterly (1970-1977)*, 27(2): 503–511, 1976.

[181] J. Moore and R. Chapman. Application of particle swarm to multiobjective optimization. Technical report, Department of Computer Science and Software Engineering, Auburn University., Auburn University, 1999.

[182] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proceedings of the Swarm Intelligence Symposium, 2003. SIS '03.*, pages 26–33, 2003.

[183] S. Mostaghim and J. Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1404–1411, 2004.

[184] S. Mostaghim, H. Trautmann, and O. Mersmann. Preference-based multi-objective particle swarm optimization using desirabilities. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *Parallel Problem*

Solving from Nature – PPSN XI, volume 6239 of *Lecture Notes in Computer Science*, pages 101–110. Springer Berlin / Heidelberg, 2011.

[185] A. Muñoz-Zavala, A. Hernández-Aguirre, and E. Villa-Diharce. Particle evolutionary swarm multi-objective optimization for vehicle routing problem with time windows. In C. Coello, S. Dehuri, and S. Ghosh, editors, *Swarm Intelligence for Multi-objective Problems in Data Mining*, volume 242 of *Studies in Computational Intelligence*, pages 233–257. Springer Berlin / Heidelberg, 2009.

[186] G. Nitschke. Emergent cooperation in robocup: A review. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 512–520. Springer Berlin / Heidelberg, 2006.

[187] J. Obit. *Developing Novel Meta-heuristic, Hyper-heuristic and Cooperative Search for Course Timetabling Problems*. PhD thesis, Computer Science, University of Nottingham, 2010.

[188] W. Ongsakul, A. Saksinchai, and C. Boonchuay. Multi-objective bidding strategy for genco using non-dominated sorting particle swarm optimisation. *International Journal of Applied Decision Sciences*, 4(4):305–323, 2011.

[189] G. Onwubolu and M. Clerc. Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. *International Journal of Production Research*, 42(3):473, 2004.

[190] Open Source Initiative. Academic Free License (AFL). http://www.opensource.org/licenses/afl-3.0.php, 2002.

[191] I. Or. *Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking*. PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1976.

[192] I. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operation Research*, 41(1-4):421–451, 1993.

[193] W. Pang, K. Wang, C. Zhou, and L. Dong. Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In *The Fourth International Conference on Computer and Information Technology, 2004. CIT '04.*, 2004.

[194] K. Parsopoulos and M. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607, Madrid, Spain, 2002. ACM. ISBN 1-58113-445-2.

[195] F. Pereira and J. Tavares, editors. *Bio-inspired Algorithms for the Vehicle Routing Problem*, volume 161 of *Studies in Computational Intelligence*. Springer, 2009.

[196] R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008:1–10, 2008.

[197] P. Pongchairerks and V. Kachitvichyanukul. A non-homogenous particle swarm optimization with multiple social structures. In V. Kachitvichyanukul, U. Purintrapiban, and P. Utayopas, editors, *Simulation and Modeling: Integrating Sciences and Technology for Effective Resource Management*, Proceedings of international conference on simulation and modeling 2005. Asian Institute of Technology, 2005.

[198] J. Potvin and J. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993.

[199] J. Potvin and J. Rousseau. An exchange heuristic for routeing problems with time windows. *The Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

[200] J. Potvin, C. Duhamel, and F. Guertin. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6:345–355, 1996.

[201] J. Potvin, T. Kervahut, B. Garcia, and J. Rousseau. The vehicle routing problem with time windows part i: Tabu search. *INFORMS Journal on Computing*, 8(2): 158–164, 1996.

[202] G. Pulido and C. Coello. Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, volume 3102, 2004.

[203] R. Purshouse and P. Fleming. *Evolutionary Multi-Objective Optimisation: An Exploratory Analysis*, volume 3, pages 2066–2073. IEEE Press, 2003.

[204] R. Purshouse and P. Fleming. Conflict, harmony, and independence: relationships in evolutionary multi-criterion optimisation. In *Proceedings of the 2nd international conference on evolutionary multi-criterion optimization (EMO 2003), LNCS 2632*, pages 16–30. Springer, 2003.

[205] Z. Qing, Q. Limin, L. Yingchun, and Z. Shanjun. An improved particle swarm optimization algorithm for vehicle routing problem with time windows. In *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pages 1386–1390, 2006.

[206] R. Qu, Y. Xu, J. Castro-Gutierrez, and D. Landa-Silva. Particle swarm optimization for the steiner tree in graph and delay-constrained multicast routing problems. *Journal of Heuristics*, pages 1–26, 2012.

[207] C. Raquel and P. Naval. An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 257–264, Washington DC, USA, 2005. ACM.

[208] J. Reardon. Fuzzy logic vs. niched pareto multiobjective genetic algorithm optimization: Part II. a simplified Born-Mayer problem. Technical Report LA-UR-97-3676, Los Alamos National Laboratory, Los Alamos, New Mexico, 1997.

[209] M. Reimann, K. Doerner, and R. Hartl. D-Ants: savings based ants divide and conquer the vehicle routing problem. *Computers and Operations Research*, 31(4): 563–591, 2004.

[210] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3164–3169, 2000.

[211] J. Renaud, F. Boctor-Fayez, and G. Laporte. An improved petal heuristic for the vehicle routing problem. *The Journal of the Operational Research Society*, 47(2):329–336, 1996.

[212] M. Reyes-Sierra and C. Coello. Improving PSO-Based multi-objective optimization using crowding, mutation and e-Dominance. In C. Coello, A. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 505–519. Springer-Verlag, 2005.

[213] M. Reyes-Sierra and C. Coello. A study of techniques to improve the efficiency of a Multi-Objective particle swarm optimizer. In *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51, pages 269–296. Springer Berlin / Heidelberg, 2007.

[214] O. Rossi-Doria, M. Sampels, M. Chiarandini, J. Knowles, M. Manfrin, M. Mastrolilli, L. Paquete, and B. Paechter. A comparison of the performance of different metaheuristics on the timetabling problem. In E. Burke and P. De Causmaecker, editors, *Proceedings of PATAT 2002*, volume 2740, 2002.

[215] L. Rousseau, M. Gendreau, and G. Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8: 43–58, 2002.

[216] B. Roy. Classement et choix en présence de points de vue multiples: La mèthode ELECTRE. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 8:57–75, 1968.

[217] B. Roy and D. Bouyssou. *Aide multicritère à la décision*. Economica, 1993.

[218] R. Russell. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation science*, 29(2):156–166, 1995. doi: 10.1287/trsc.29.2.156.

[219] D. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routing. *Journal of Operational Research Society*, 44:289–296, 1993.

[220] S. Sahoo, S. Kim, B. Kim, B. Kraas, and A. Popov. Routing optimization for waste management. *Interfaces*, 35:24–36, 2005.

[221] M. Sakawa and H. Yano. An interactive fuzzy satisficing method for multiobjective linear fractional programming problems. *Fuzzy Sets and Systems*, 28(2): 129–144, 1988.

[222] M. Salazar-Lechuga and J. Rowe. Particle swarm optimization and fitness sharing to solve multi-objective optimization problems. In *The 2005 IEEE Congress on Evolutionary Computation.*, volume 2, pages 1204–1211, 2005.

[223] L. Santana-Quintero, N. Ramírez-Santiago, and C. Coello. Towards a more efficient Multi-Objective particle swarm optimizer. In *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, pages 76–105. IGI Global, Hershey, USA, 2008. ISBN 978-1-59904-498-9.

[224] D. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 772–787. Springer Berlin / Heidelberg, 2007.

[225] J. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic algorithms and their applications: proceedings of the first international conference on genetic algorithms*, pages 93–100, 1985.

[226] H. Schwefel. *Collective phenomena in evolutionary systems*, volume 2, pages 1025–1033. International Soc. for General System Research, 1987.

[227] B. Secrest. *Traveling Salesman Problem for Surveillance Mission Using Particle Swarm Optimization*. PhD thesis, Air Force Institute of Technology, US, 2001.

[228] Paolo Serafini. Simulated annealing for multiobjective optimization problems. In *Procceedings of the 10th international conference on multiple criteria decision making*, pages 87–96, 1992.

[229] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Computer*, 1520:417–431, 1998.

[230] X. Shi, Y. Zhou, L. Wang, Q. Wang, and Y. Liang. A discrete particle swarm optimization algorithm for travelling salesman problem. In *Computational Methods*, pages 1063–1068. Springer Netherlands, 2006.

[231] X. Shi, Y. Liang, H. Lee, C. Lu, and Q. Wang. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information Processing Letters*, 103(5): 169–176, 2007.

[232] Z. Shurong, D. Pengxin, and Z. Hongwei. The improvement of hybrid particle swarm algorithm and its application. *Advanced Materials Research*, 268 - 270:798–802, July 2011.

[233] M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations research*, 35:254–265, 1987.

[234] N. Srinivas and K. Deb. Multiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1995.

[235] D. Srinivasan and T. Seow. Particle swarm inspired evolutionary algorithm (PSEA) for multiobjective optimization problems. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2292–2297, 2003.

[236] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.

[237] S. Talukdar, L. Baerentzen, A. Gove, and P. De-Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4:295–321, 1998.

[238] K. Tan, Y. Chew, and L. Lee. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34:115–151, 2006. ISSN 0926-6003.

[239] S. Thangiah. Vehicle routing with time windows using genetic algorithms. *Time*, 2:1–24, 1999.

[240] The GNU Project. GNU General Public License (GPL). http://www.gnu.org/copyleft/gpl.html, 2007.

[241] M. Thompson and N. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41:935–946, 1993.

[242] G. Toscano-Pulido, C. Coello, and L. Santana-Quintero. Emopso: A multi-objective particle swarm optimizer with emphasis on efficiency. In *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 272–285. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-70927-5.

[243] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Siam Monographs on Discrete Mathematics and Applications, Philadelphia: Siam, 2001.

[244] E. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens. Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of multi-criteria decision analysis*, 8:221–236, 1999.

[245] G. Nemhauser und L. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience, 1988.

[246] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11(8): 5375 – 5390, 2011.

[247] D. Vigo. A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *European Journal of Operational Research*, 89(1):108–126, 1996.

[248] M. Villalobos-Arias, G. Pulido, and C. Coello. A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 22–29, 2005.

[249] S. Voss and D. Woodruff, editors. *Optimization software class libraries*. Kluwer academic publishers, 2002.

[250] C. Voudouris and E. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999.

[251] S. Wagner. *Heuristic Optimization Software Systems - Modeling of Heuristic Optimization Algorithms in the HeuristicLab Software Environment*. PhD thesis, Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria, 2009.

[252] S. Wagner and M. Affenzeller. Heuristiclab grid - a flexible and extensible environment for parallel heuristic optimization. *Journal of Systems Science*, 30(4): 103–110, 2004.

[253] T. Wagner, N. Beume, and B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 742–756. Springer Berlin / Heidelberg, 2007.

[254] M. Wall. GAlib. http://lancet.mit.edu/ga/, 1999.

[255] K. Wang, L. Huang, C. Zhou, and W. Pang. Particle swarm optimization for traveling salesman problem. In *International Conference on Machine Learning and Cybernetics 3*, pages 1583–1585, 2003.

[256] P. Wark and J. Holt. A repeated matching heuristic for the vehicle routing problem. *The Journal of the Operational Research Society*, 45(10):1156–1167, 1994.

[257] E. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[258] K. Wickramasinghe and X. Li. Integrating user preferences with particle swarms for multi-objective optimization. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 745–752. ACM, 2008.

[259] W. Chen X. Tan, X. Luo and J. Zhang. Ant colony system for optimizing vehicle routing problem with time windows. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 209 –214, 2005.

[260] P. Xuan and V. Lesser. Multi-agent policies: from centralized ones to decentralized ones. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, AAMAS '02, pages 1098–1105, New York, NY, USA, 2002. ACM.

[261] J. Yang, J. Zhou, L. Liu, and Y. Li. A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (MOPSO). *Computers & Mathematics with Applications*, 57(11-12):1995–2000, 2009.

[262] P. Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly (1970-1977)*, 21(2):281–283, 1970.

[263] R. Zebulum, M. Pacheco, and M. Vellasco. A Multi-Objective optimisation methodology applied to the synthesis of Low-Power operational amplifiers. In *In Proceedings of the XIII International Conference in Microelectronics and Packaging*, volume 1, pages 264–271, 1998.

[264] Q. Zhang, T. Zhen, Y. Zhu, W. Zhang, and Z. Ma. A hybrid intelligent algorithm for the vehicle routing with time windows. *2010 International Forum on Information Technology and Applications*, pages 47–54, 2010.

[265] S. Zhao and P. Suganthan. Two-lbests based multi-objective particle swarm optimizer. *Engineering Optimization*, 43(1):1–17, 2011.

[266] Y. Zhao, B. Wu, Y. Ma, W. Wang, and H. Sun. Particle swarm optimization for vehicle routing problem with time windows. In *Materials Science Forum*, volume 471-472, pages 801–805, 2004.

[267] X. Zheng and H. Liu. A hybrid vertical mutation and self-adaptation based MOPSO. *Computers & Mathematics with Applications*, 57(11-12):2030–2038, 2009. ISSN 0898-1221.

[268] W. Zhong, J. Zhang, and W. Chen. A novel discrete particle swarm optimization to solve traveling salesman problem. In *IEEE Congress on Evolutionary Computation (CEC).*, pages 3283 –3287, 2007.

[269] L. Zhongkai, Z. Zhencai, and Huiqin Z. Dsmopso: A distance sorting based multiobjective particle swarm optimization algorithm. In *ICNC'10*, pages 2749–2753, 2010.

[270] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*, pages 832–842. Springer, 2004.

[271] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Proceedings of the 5th parallel problem solving from nature (PPSN V)*, pages 292–301. Springer, 1998.

[272] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on evolutionary computation*, 3(4):257–271, 1999.

[273] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[274] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems (EUROGEN 2001)*, pages 95–100, 2002.

[275] E. Zitzler, D. Brockhoff, and L. Thiele. *The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration*, volume 4403 of *LNCS*, pages 862–876. Springer, 2007.

[276] E. Zitzler, L. Thiele, and J. Bader. Spam: Set preference algorithm for multiobjective optimization. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, pages 847–858, Berlin, Heidelberg, 2008. Springer-Verlag.

APPENDIX A

Solomon's Dataset - Time Windows

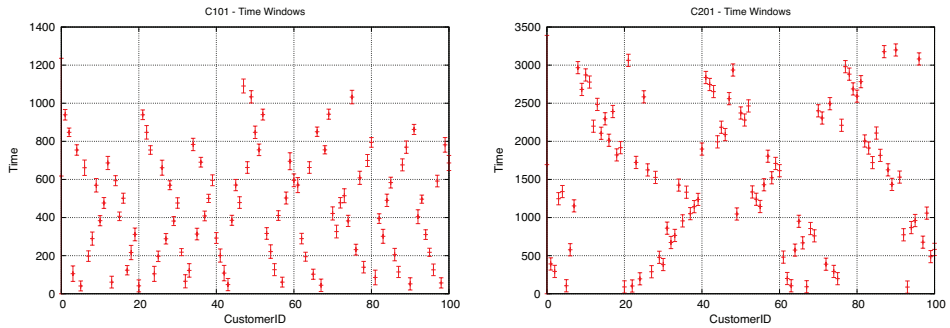


Figure A.1: Time window representation of Solomon's instances: C101 and C201 with 100 customers. The X-axis denotes the ids for both the depot (0) and the customers ($1 - 100$). The Y-axis denotes the time. Customer's time windows are depicted as vertical segments.

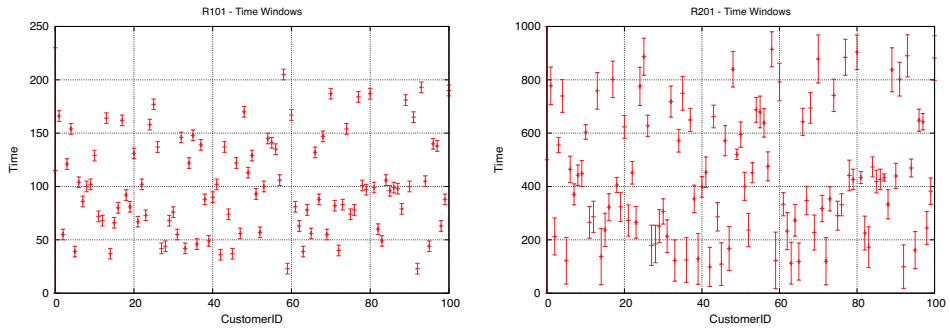


Figure A.2: Time window representation of Solomon's instances: R101 and R201 with 100 customers. The X-axis denotes the ids for both the depot (0) and the customers ($1 - 100$). The Y-axis denotes the time. Customer's time windows are depicted as vertical segments.

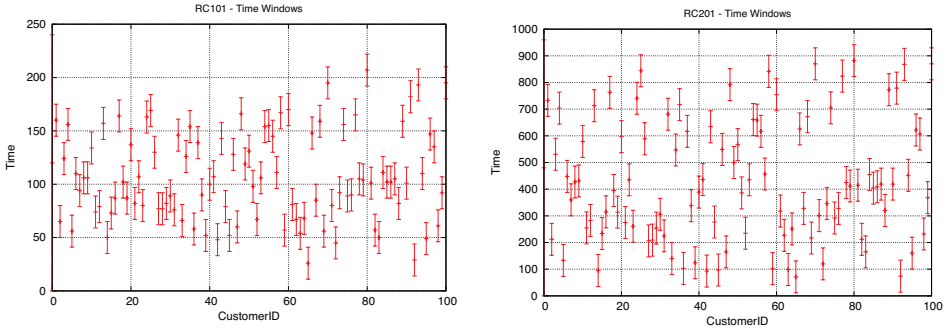


Figure A.3: Time window representation of Solomon's instances: RC101 and RC201 with 100 customers. The X-axis denotes the ids for both the depot (0) and the customers ($1 - 100$). The Y-axis denotes the time. Customer's time windows are depicted as vertical segments.