Thèse
de doctorat
de l'UTT

Elyn Lizeth SOLANO CHARRIS

Optimization Methods for the Robust Vehicle Routing Problem

Spécialité :
Optimisation et Sûreté des Systèmes

2015TROY0026

Année 2015

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE
DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Elyn Lizeth SOLANO CHARRIS

le 15 octobre 2015

Optimization Methods for the Robust Vehicle Routing Problem

JURY

M. A. MOUKRIM	PROFESSEUR DES UNIVERSITES	Président
M. T. F. DE NORONHA	ASSOCIATE PROFESSOR	Examinateur
Mme A. C. DUHAMEL	MAITRE DE CONFERENCES	Directrice de thèse
M. D. FEILLET	PROFESSEUR ENSM SAINT-ETIENNE	Rapporteur
M. C. PRINS	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. K. SÖRENSEN	FULL PROFESSOR	Rapporteur

Acknowledgements

"*To my family and the people I love, specially my parents and sister, thank you for giving me all the support to accomplish this important part of my professional career. Thank you for your company even in the distance, your advices and unconditional love in this travel full of challenges, in which all you have been my principal inspiration. And also to my niece who gives me thousands of reasons to smile*".

I would like to start by acknowledging my advisors, Andréa Santos and Christian Prins for their kindness and intellectual generosity. It has been a pleasure to work with them during these three years in which I have learned not just about vehicle routing, data structures and optimization but also about all the things that have improved my life in many other ways. I will be always grateful to them for the opportunity they gave me for working on this project.

I also thank to Dominique Feillet, Kenneth Sörensen, Thiago Noronha and Aziz Moukrim for accepting to be part of the jury and for their valuable time and suggestions to improve this research.

Likewise, I want to thank to all my colleagues and friends at the UTT, especially to Matthieu Le Berre and Syrine Roufaida for all the conversations and company. I am glad to have you know and to share all this time together. Also my greatest gratitude goes to the people who makes me feel like in family at Troyes, Andrés, Natalia D., Juan Carlos, Natalia L., Guillermo, Yoly, Yoce, Karina, Alex, Carolina, Anthony, Angélica, Gustavo, Marie H., Jorge, Tito.

I want also to thank to all LOSI professors and the administrative team at UTT for their support directly or indirectly in my Ph.D. Especially, thanks Véronique Banse, Bernadette Andre, Pascale Denis, Isabelle Leclercq and Thérèse KAZARIAN for all your help.

I special thanks to Jairo Montoya for his support in this journey, as well as, the support

received by Universidad de La Sabana (Bogotá) for providing the time to start my PhD career.

Finally, this dissertation would not have been possible without the financial support of the Champagne-Ardenne Region under the project METHODI.

Contents

Page

Acknowledgements **iii**

Contents **vii**

List of Figures **ix**

List of Tables **xiii**

List of Algorithms **xv**

Abbreviations **xvii**

1 Introduction **1**

1.1 General Context . . . 1

1.2 Contributions and structure of the manuscript . . . 3

2 Robustness in Optimization **7**

2.1 General definitions in Robust Optimization . . . 7

2.1.1 Robust Optimization criteria . . . 9

2.2 Robust Classification Criteria . . . 11

2.3 Conclusions . . . 13

3 Literature Review **15**

3.1 The Capacitated Vehicle Routing Problem . . . 15

3.1.1 Constructive heuristics . . . 16

3.1.2 Improvement heuristics . . . 17

3.1.3 Exact methods . . . 17

3.1.4 Metaheuristics . . . 18

3.2 The VRP with uncertainties . . . 18

3.2.1 Stochastic vehicle routing problems . . . 19

3.2.2 Robust vehicle routing problems 19

4 The Robust Vehicle Routing Problem with Uncertain Travel Costs 25

4.1 Introduction 25

4.2 Problem definition 27

4.3 Lexicographic min-max robustness criterion 28

4.4 Greedy Heuristics 30

4.4.1 Clarke and Wright heuristic for the RVRP 30

4.4.2 Randomized version 32

4.4.3 Insertion heuristics for the RVRP 34

4.4.4 Pilot versions of Insertion heuristics 34

4.4.5 Computational evaluation 37

4.4.6 Conclusions for constructive heuristics 41

4.5 A genetic algorithm 41

4.5.1 Chromosomes and fitness 43

4.5.2 Initial Population 44

4.5.3 Genetic operators 45

4.5.4 Renewal Procedure and stopping criterion 46

4.5.5 Computational evaluation 46

4.5.6 Conclusions on the GA 50

4.6 Local search-based metaheuristics 50

4.6.1 Local search procedure 50

4.6.2 Greedy randomized adaptive search procedure 53

4.6.3 Two ILS metaheuristics 54

4.6.4 Multi-start ILS with giant tours 55

4.6.5 Computational evaluation 60

4.6.6 Conclusions on the local search-based methods 68

4.6.7 Conclusions 69

5 The Bi-Objective Robust Vehicle Routing Problem with Uncertain Demands and Travel Times 71

5.1 Introduction 71

5.2 Problem definition 72

5.3 Multiobjective Algorithms for the bi-RVRP 74

5.3.1 Shared components of our multiobjective algorithms 74

5.3.2 MOEA and NSGAII for the bi-RVRP 75

5.4 Local Search 77

5.4.1 Acceptance criteria for a move 78
5.4.2 Integrating the local search in our multiobjective heuristics 79
5.5 Computational experiments 80
5.5.1 Implementation and instances 80
5.5.2 Evaluation criteria 81
5.5.3 Parameter tuning 81
5.5.4 Results on instances 82
5.5.5 Statistical tests. 97
5.6 Conclusions 101

6 General Conclusions and Future Research Directions 103

Annexes

A Résumé étendu en français : Méthodes d'optimisation pour le problème de tournées de véhicules robuste 107

Annexes

B Statistical tests for the RVRP 129

Annexes

C Statistical tests for the bi-RVRP 131

Bibliography 149

List of Figures

Page

1.1 The robust decision making framework 2

2.1 Example for the RSP-D. 13

4.1 Examples of solution on normal conditions and perturbated scenarios. 26
4.2 Examples of mergers for a scenario k: (T, U) and $(\overline{T}, \overline{U})$. 31
4.3 Constructive heuristics results for small-size instances. 38
4.4 Calibration of medium-size instances with $g = |N|$. 48
4.5 Calibration of medium-size instances with $g = 2|N|$. 48
4.6 2-opt moves for one trip T and scenario k. 52
4.7 2-opt moves for two trips T and U and scenario k. 52
4.8 Application of the splitting procedure to a giant tour with four customers. . 57

5.1 Examples of solutions for VRP and bi-RVRP. 73
5.2 Example of non-dominated sorting. 76
5.3 Example of the variable descent directions from Murata. 79
5.4 Example of convergence on NSGAII(1) for instance 30-3-30-100-3, $\varrho = 5\%$ on set 1. 85
5.5 Example of convergence on NSGAII(1) for instance 50-5-20-100-5, $\varrho = 25\%$ on set 1. 86
5.6 Example of convergence on NSGAII(1) for instance 50-5-20-100-5, $\epsilon = 50\%$ on set 2. 86
5.7 Impact of the different methods for instance 50-5-20-100-3 with $\varrho = 5\%$ on set 1. . 87
5.8 Impact of the different methods for instance 50-5-20-100-3 with $\varrho = 5\%$ on set 1. . 87
5.9 Impact of ϱ for instance 10-2-20-50-3 on NSGAII(1). 89
5.10 Impact of ϱ for instance 10-2-20-50-5 on NSGAII(1). 90
5.11 Impact of ϱ for instance 20-3-20-100-3 on NSGAII(1). 90
5.12 Impact of ϱ for instance 20-3-20-100-5 on NSGAII(1). 97

A.1 Exemples de solutions pour les VRP et bi-RVRP. 125

List of Tables

Page

2.1 Evaluating solutions for the RSP-D. 14

2.2 Applying the *robust mean score* on the RSP-D. 14

3.1 Vehicle routing problems with uncertainty. 23

4.1 Detailed results for constructive heuristics on small-size instances. 39

4.2 Indicators for constructive heuristics for small-size instances. 40

4.3 Indicators for constructive heuristics for medium-size instances. 41

4.4 Detailed results for constructive heuristics on medium-size instances. 42

4.5 Example of a chromosome representation. 44

4.6 Example of crossover. 45

4.7 Example of mutation. 45

4.8 Detailed results for the GA on small-size instances. 47

4.9 Detailed results for the GA on medium-size instances. 49

4.10 Detailed results for the LS-based methods on small-size instances. 62

4.11 Indicators for the small-size instances – 10 runs with 5,000 calls to the local search. 63

4.12 Detailed results for the LS-based methods on medium-size instances. 64

4.13 Indicators for the medium-size instances – 10 runs with 5,000 calls to the local search. 65

4.14 Indicators for the medium-size instances – Same duration as MS-ILS-GT. 65

4.15 Friedman test for medium-size instances – 5,000 calls to the local search. 66

4.16 Friedman test for medium-size instances – Time Budget 66

4.17 Pairwise test for medium-size instances – 5,000 calls to the local search. 67

4.18 Pairwise test for medium-size instances – Time Budget. 67

4.19 Indicators for 7 CVRP instances – 10 runs with 5,000 calls to the local search. . . 68

4.20 Indicators for the small-size instances. 69

5.1 Summary of results for the VRP (left) and the bi-RVRP (right). 73

5.2 Example of chromosome. 74

5.3 Versions for the evolutionary metaheuristics. 79

5.4 Results of NSGAII(1) with 100 and 200 iterations $\varrho = 5\%$. 82

5.5 Results of NSGAII(1) with 100 and 200 iterations $\varrho = 25\%$. 83

5.6 Results of NSGAII(1) with 100 and 200 iterations $\epsilon = 50\%$. 83

5.7 Results of MOEA(1) with 100 and 200 iterations $\varrho = 5\%$. 84

5.8 Results of MOEA(1) with 100 and 200 iterations $\varrho = 25\%$. 84

5.9 Results of MOEA(1) with 100 and 200 iterations $\epsilon = 50\%$. 85

5.10 Average results for set 1. 88

5.11 Average results for set 2. 89

5.12 Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 5\%$ for set 1. 91

5.13 Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 25\%$ for set 1. 92

5.14 Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 5\%$ for set 1. 93

5.15 Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 25\%$ for set 1. 94

5.16 Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\epsilon = 50\%$ for set 2. 95

5.17 Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\epsilon = 50\%$ for set 2. 96

5.18 Friedman test on Θ, Y_1, Y_2 with $\varrho = 5\%$. 98

5.19 Friedman test on Θ, Y_1, Y_2 with $\varrho = 25\%$. 99

5.20 Friedman test on Θ, Y_1, Y_2 with $\epsilon = 50\%$. 100

5.21 Pairwise test for Y_2 with $\varrho = 25\%$. 101

5.22 Pairwise test for Θ with $\epsilon = 50\%$. 101

5.23 Pairwise test for Y_2 with $\epsilon = 50\%$. 102

A.1 Résumé des résultats pour le VRP (à gauche) et le bi-RVRP (à droite). . . . 125

B.1 Friedman test for the results with LS budget. 129

B.2 Friedman test for the results with time budget. 130

C.1 Friedman test for Θ with $\varrho = 5\%$ Part 1. 131

C.2 Friedman test for Θ with $\varrho = 5\%$ Part 2. 132

C.3 Friedman test for Y_1 with $\varrho = 5\%$ Part 1. 133

C.4 Friedman test for Y_1 with $\varrho = 5\%$ Part 2. 134

C.5 Friedman test for Y_2 with $\varrho = 5\%$ Part 1. 135
C.6 Friedman test for Y_2 with $\varrho = 5\%$ Part 2. 136
C.7 Friedman test for Θ with $\varrho = 25\%$ Part 1. 137
C.8 Friedman test for Θ with $\varrho = 25\%$ Part 2. 138
C.9 Friedman test for Y_1 with $\varrho = 25\%$ Part 1. 139
C.10 Friedman test for Y_1 with $\varrho = 25\%$ Part 2. 140
C.11 Friedman test for Y_2 with $\varrho = 25\%$ Part 1. 141
C.12 Friedman test for Y_2 with $\varrho = 25\%$ Part 2. 142
C.13 Friedman test for Θ with $\epsilon = 50\%$ Part 1. 143
C.14 Friedman test for Θ with $\epsilon = 50\%$ Part 2. 144
C.15 Friedman test for Y_1 with $\epsilon = 50\%$ Part 1. 145
C.16 Friedman test for Y_1 with $\epsilon = 50\%$ Part 2. 146
C.17 Friedman test for Y_2 with $\epsilon = 50\%$ Part 1. 147
C.18 Friedman test for Y_2 with $\epsilon = 50\%$ Part 2. 148

List of Algorithms

Page

Algorithm 2.1 Score evaluation 12

Algorithm 4.1 Lexicographic comparison of two solutions for q scenarios. 29

Algorithm 4.2 Clarke and Wright heuristic for the RVRP – *CW* (ω). 33

Algorithm 4.3 Sequential Best Insertion heuristic for the RVRP – *SBIH* (ω). . . 35

Algorithm 4.4 Paralell Best Insertion heuristic for the RVRP – *PBIH* (ω). . . . 36

Algorithm 4.5 Genetic Algorithm for the RVRP – *GA(ω)* 43

Algorithm 4.6 Local search procedure – *LS* (ω) 51

Algorithm 4.7 Evaluation of a move in *MovesOneTrip* and *MovesTwoTrips* . . . 53

Algorithm 4.8 Greedy Randomized Adaptive Search Procedure – *GRASP* ($\theta, ncalls, \omega$) 54

Algorithm 4.9 Multi-Start ILS – *MS-ILS* ($\theta, niter, ncalls, minswaps, maxswaps, \omega$) 55

Algorithm 4.10 MS-ILS with giant-tours – *MS-ILS-GT* (same header as *MS-ILS*) 58

Algorithm 4.11 Splitting procedure for one giant tour τ – *Split* (τ, ω) 59

Algorithm 5.1 Non-dominated sorting genetic algorithm for the bi-RVRP 78

Abbreviations

bi-RVRP:	Bi-objective Robust Vehicle Routing Problem.
B&C:	Branch-and-Cut Algorithm.
B&B:	Branch-and-Bound Algorithm.
CARP:	Capacitated Arc Routing Problem.
CPLEX:	Linear Optimization Solver.
CVRP:	Capacitated Vehicle Routing Problem.
CW:	Clarke and Wright Heuristic.
DP:	Dynamic Programming.
GA:	Genetic Algorithm.
GAP:	Generalized Assignment Problem.
GLPK:	GNU Linear Programming Kit.
GRASP:	Greedy Randomized Adaptive Search Procedure.
ILS:	Iterated Local Search.
LP:	Linear Programming.
log	Logarithmic Time.
LS:	Local Search.
MDVRP	Multi-Depot Vehicle Routing Problem.
MILP:	Mixed-Integer Linear Programming.
MOEA:	Multiobjective Evolutionary Algorithm.
MS-ILS:	Multi-Start ILS.
MS-ILS-GT:	Multi-Start ILS with Giant Tours.

MTZ:	Miller-Tucker-Zemlin model.
NSGAII:	Non-dominated Sorting Genetic Algorithm.
OX:	Ordered Crossover.
PBI:	Parallel Best Insertion Heuristic.
RCC:	Robust Classifications Criteria.
ROC:	Robust Optimization Criteria.
RCI:	Rounded Capacity Inequalities.
RCW:	Randomized Clarke and Wright Heuristic.
RO:	Robust Optimization.
RVRP:	Robust Vehicle Routing Problem.
SA:	Simulated Annealing.
SBI:	Sequential Best Insertion Heuristic.
SP:	Set Partitioning Formulation.
SVRP:	Stochastic Vehicle Routing Problem.
TS:	Tabu Search.
TSP:	Traveling Salesman Problem.
UB:	Upper Bound.
VRP:	Vehicle Routing Problem.
VRPSD:	Vehicle Routing Problem with Stochastic Demands.
VRPTW:	Vehicle Routing Problem with Time Windows.
2VF:	Two-index Vehicle Flow.

Chapter 1

Introduction

1.1 General Context

Combinatorial optimization is one of the most active fields at the interface of operations research, computer science, and applied mathematics. The largest number of real applications concern communications, network design, scheduling and transportation. In the latter domain, the Vehicle Routing Problem (VRP) is a well-known and hard problem which consists in determining a set of routes used by a fleet of vehicles, based at one depot, to serve a set of clients (Toth & Vigo (1998)). Introduced by Dantzig & Ramser (1959), the VRP holds a central place in distribution management and has become one of the most widely studied problem in combinatorial optimization. General surveys on the VRP can be found in Fisher (1995); Golden *et al.* (2008); Laporte (2009); Toth & Vigo (2014). To handle this problem, one of the main assumptions is that parameters and data are assumed to be deterministic and known in advance. However, in real applications, operations in any traffic network have a fairly high degree of uncertainty which include, among others, arrivals of new clients or cancellations, variable waiting times, and variable travel times due to traffic congestion (Sungur *et al.* (2008)). Therefore, a perturbation of the input data may result in suboptimal or even infeasible solutions (Moghaddam *et al.* (2012)). An important trend for taking into account this perturbation consists in investigating extensions of the VRP with uncertain data, both in terms of theoretical and practical issues.

Several methods are available to model uncertainties in the context of optimization problems, mainly focusing on stochastic programming where uncertain data are modeled as random variables with known probability distributions (Wets (2002); Shapiro *et al.* (2008)). Nevertheless, such an approach is limited to the cases where uncertainties have a stochastic nature (which is not always the case) and when it is possible to identify the probability distribution, which can be difficult to obtain, especially for large-scale problems (Ben-Tal &

Nemirovski (1998)).

Unto now, most studies have dealt with stochastic programming. Pioneer models for stochastic programming were proposed by Beale (1955) and Dantzig (1955). In particular, for the Stochastic Vehicle Routing Problem (SVRP), the most studied problems have been the VRP with stochastic demands and stochastic customers (Bertsimas (1992); Gendreau *et al.* (1995); Secomandi (2001)). In spite of its importance in real applications, especially in big cities, just few works have been done for the VRP with stochastic travel times (Laporte *et al.* (1992); Lambert *et al.* (1993)).

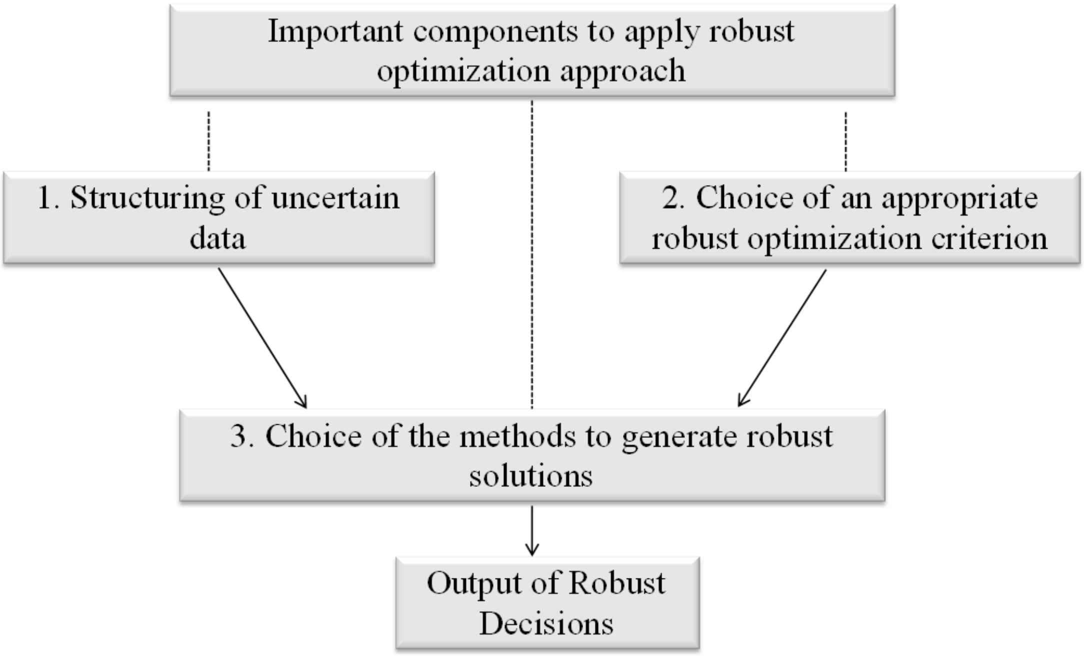

Important components to apply robust optimization approach
1. Structuring of uncertain data
2. Choice of an appropriate robust optimization criterion
3. Choice of the methods to generate robust solutions
Output of Robust Decisions

Figure 1.1: The robust decision making framework

An alternative to stochastic programming is robust optimization, which has been applied in general as a way to self-protect against undesirable impacts induced by vague approximations, incomplete, imprecise, or ambiguous data. In this context, the literature covers a large number of applications such as scheduling (Goren & Sabuncuoglu (2008); Hazir *et al.* (2010)), facility location (Minoux (2010); Baron *et al.* (2011); Alumur *et al.* (2012); Gülpinar *et al.* (2013)), inventory (Bienstock & Özbay (2008); Ben-Tal *et al.* (2009a)), finance (Fabozzi *et al.* (2007); Gülpinar & Rustem (2007); Ç. Pinar (2007); Bertsimas & Pachamanova (2008); Schoettle & Werner (2009)), queuing networks, stochastic systems and game theory (Bertsimas & Doan (2010); Kırkızlar & Andradóttir (2010); Ordóñez & Stier-Moses (2010); Kardes *et al.* (2011)), machine learning and statistics (Xu *et al.* (2010); Ben-Tal *et al.* (2011); Xu *et al.* (2012)), and energy systems (Ribas *et al.* (2010); Bertsimas *et al.* (2011)). The general idea for this approach is to provide robust solutions taking into account several technical challenges such as the evaluation of robust solutions, the adaptation of heuristics methods for large scale routing problems and the assessment of performance and robustness measures. Roughly, three main components need to be addressed: (i) the way uncertain data is modeled (e.g., with the use of scenarios), (ii) the selection of an appropriate robust optimization criterion such as min-max, min-max regret, min-max relative regret, etc., referred here as Robust Optimization Criteria (ROC), and (iii) the choice of a mathematical model and the methods to generate robust solutions (see Figure 1.1, adapted from Kouvelis & Yu (1997)).

This Ph.D thesis is dedicated to robust optimization approaches for the VRP with uncertain data, giving what we call the Robust Vehicle Routing Problem (RVRP). First, uncertainties are handled on traveling times. Then, a second version of the RVRP is considered taking into account both traveling times and demands in a bi-objective version of the RVRP (bi-RVRP). For solving the RVRP and the bi-RVRP different models and methods are proposed to determine robust solutions (less affected by uncertainties) minimizing the worst case. In particular for the RVRP, greedy heuristics, one evolutionary approach, and iterative multistart strategies are adapted to handle uncertainties. Concerning the bi-RVRP different variations for multiobjective evolutionary metaheuristics, coupled with a local search procedure are proposed. Section 1.2 describes the detailed contributions and the structure of the manuscript.

1.2 Contributions and structure of the manuscript

The structure of this document comprises four parts. The first one presents the general definitions and components for Robust Optimization (RO). The second part is devoted to a general review of the literature considered in this work. The third one considers the RVRP with uncertain travel costs, and the last part introduces the bi-RVRP with uncertainty in both demands and travel times. The main contributions classified by chapters are presented as follows:

- Chapter 2 provides the general definitions for RO together with a review of the most common ROC, i.e., min-max, min-max regret, min-max relative regret, lexicographical criterion, α-robustness, bw-robustness and pw-robustness, etc. and their main applications. Then, two Robust Classifications Criteria (RCC) are proposed that can be applied for measuring the robustness of scenarios and solutions. The alternative RCC proposed here could be used independently of other criteria, as an additional support to classical ROC, or still as a tool for intensification in local search strategies. It is important to highlight that the proposed RCC are not optimization criteria, instead they are a measure of quality based on ranking (classification) of solutions over all the scenarios considered.

- Chapter 3 presents a survey on VRP. In particular, a few key-articles are recalled on the Capacitated Vehicle Routing Problem (CVRP) and the methods to solve it, such as constructive and improvement heuristics, exact methods and metaheuristics. A review on the VRPs with uncertainties is also conducted, considering both robust optimization and stochastic programming approaches. As the stochastic vehicle routing problem is not part of this work just some references are presented.

- In Chapter 4, the RVRP with uncertain travel costs addressed via robust optimization is studied. The uncertain travel costs are modeled by discrete scenarios in a complete and directed network (the classical VRP literature mainly considers undirected graphs with symmetric costs), in which each scenario defines one travel time for each arc. The aim is to minimize the worst cost (total route duration) over all scenarios. These scenarios do not correspond to realizations of random variables and are not associated with probabilities of occurrence. This situation reflects for instance transit problems in urban networks and has also application in large scale bio-terrorism emergency. To solve the RVRP a mathematical formulation is introduced. Then, several greedy heuristics are proposed: the Clarke and Wright (CW), Randomized CW (RCW), Parallel Best Insertion (PBI), Sequential Best Insertion (SBI), Pilot Parallel Best Insertion (Pilot PBI) and Pilot Sequential Best Insertion (Pilot SBI). In addition, more sophisticated strategies, such as an evolutionary approach (i.e., Genetic Algorithm (GA)), and local search-based algorithms are adapted to handle the RVRP. The local search-based methods considered are an Iterated Local Search (ILS), a Multi-Start ILS (MS-ILS) and a MS-ILS based on Giant Tours (MS-ILS-GT) converted into feasible routes via a lexicographic splitting procedure. A test bed of 42 instances with up to 100 customers, 20 vehicles and 30 scenarios is generated. The computational experiments are performed for the mathematical formulation and the proposed greedy heuristics and metaheuristics.

- Chapter 5 extends the RVRP to a multiobjective optimization problem, referred as the bi-RVRP with uncertainty on both demands and travel times. Uncertain parameters are addressed in the objective function using the min-max optimization criterion. Uncertain demands per customer are modeled by a set of scenarios representing the deviations from an expected demand. Uncertain travel times are independent from customer demands. Then, possible realizations of travel times are considered to get discrete scenarios for each arc of the transportation network. This problem finds applications in urban transportation, e.g., serving small retail stores. The bi-RVRP is solved through different variations of the Non-dominated Sorting Genetic Algorithm version 2 (NSGAII) and the Multiobjective Evolutionary Algorithm (MOEA). Two set of instances are tested assuming a global perturbation over the total demand of clients and an individual variation on the demand defined in an interval. The set of instances contain 24 instances with up to 50 customers, 5 vehicles, 30 scenarios for travel times and 5 scenarios for demands. Different metrics are used to measure the algorithm performance, the convergence, as well as the diversity on solutions for the different methods.

- Finally, a conclusion closes the thesis by summarizing the contributions and suggesting interesting directions for future research.

Chapter 2

Robustness in Optimization

In this chapter, general concepts on robustness and a review of different Robust Optimization Criteria (ROC) are recalled. Then, two Robust Classifications Criteria (RCC) are proposed that can be applied for measuring the robustness of both scenarios and solutions. They constitute our first contribution to the thesis. The existing applications of robust optimization to vehicle routing problems are presented in Chapter 3.

2.1 General definitions in Robust Optimization

The word “robust" has different meanings in optimization. In practice, the data for an optimization problem can be uncertain, inexact, noised, or likely to change in future. An optimal solution computed using current parameters can be strongly affected by perturbations, becoming suboptimal or even infeasible. What most decision makers call *robust solution* is a solution resisting as much as possible such perturbations.

Robustness can be mainly addressed via stochastic optimization when uncertainty has a probabilistic description. Stochastic programming has been introduced by Dantzig & Ramser (1959) and two methods have been broadly applied to solve stochastic problems: chance-constrained programming and stochastic programming with recourse. In chance-constrained programming, the resulting decision ensures a given probability of complying with constraints, i.e., the confidence level of being feasible. For more information, readers are referred to Charnes & Cooper (1959). In stochastic programming with recourse, some feasibility constraints are relaxed and included in the objective function, assuming that violations induced by random events after the implementation of first stage decisions can be repaired by recourse actions. Good entry points for stochastic optimization are Birge & Louveaux (1997) and Cordeau *et al.* (2007) for an overview, Bianchi *et al.* (2009) about metaheuristics in stochastic combinatorial optimization. Stochastic models are powerful but have two main drawbacks: the underlying probability distributions must be known and the solutions can become infeasible for some realizations of random events.

Robust optimization (RO) is a framework to handle uncertainty while avoiding these drawbacks. This approach is no longer stochastic but rather deterministic and set-based. According to Bertsimas *et al.* (2011), "*instead of trying to protect the solution in some probabilistic sense to stochastic uncertainty, the decision maker seeks a solution that is feasible for any realization of the uncertainty in a given set*". In general, the goal is to optimize the worst case value over all uncertain data.

According to Kouvelis & Yu (1997), important decisions when applying RO are: (i) the way uncertain data is modeled, (ii) the selection of an appropriate Robust Optimization Criteria (ROC) such as min-max, min-max regret, min-max relative regret, α-robustness, bw-robustness, pw-robustness, and (iii) the choice of a mathematical model and the methods to generate robust solutions. Regarding the way to structure uncertain data, the most common representation is done by a convex set, e.g., a polyhedron, a cone, or an ellipsoid (Ben-Tal *et al.* (2009b); Bertsimas *et al.* (2011)), or by an assignment of plausible values to each model parameter, which two common ways of modeling are the interval or discrete scenarios. Concerning a robust model and a robust solution, interesting definitions are introduced by Mulvey *et al.* (1995) and Kouvelis & Yu (1997).

One branch of RO studies mathematical programs for which tractable robust counterparts can be obtained and develops computational tools, see for instance Ben-Tal *et al.* (2009b) for a review. Other authors like Kouvelis & Yu (1997) or Aissi *et al.* (2005) consider combinatorial optimization problems and investigate the algorithmic complexity of their robust versions. For instance, finding a shortest path problem in a graph is a polynomial problem, while the robust version with two scenarios for the arc costs is NP-hard, even in layered networks (Yu & Yang (1998)). In spite of their hardness, NP-hard problems start being solved by means of RO.

NP-hard discrete RO problems can also be solved using classical exact approaches such as branch-and-cut or Dantzig-Wolfe decomposition, see examples in Candia-Véjar *et al.* (2011) and Coco *et al.* (2014b). A significant stream of research develops heuristics with performance guarantees (Aissi *et al.* (2005, 2009); Kasperski *et al.* (2012)). Surprisingly, few metaheuristics have been developed. For instance, Nikulin (2008) designed a simulated annealing algorithm for a robust spanning tree problem.

As robust solutions are often viewed as conservative, Bertsimas & Sim (2003) propose to limit the number of uncertain parameters allowed to deviate from their nominal values to a number Γ, called *budget of uncertainty*. Applied to the cost and constraint coefficients of a linear or mixed integer program, their approach leads to a robust version with a moderate increase in size. In particular, a 0–1 linear program with uncertainties limited to the n cost coefficients can be treated by solving at most $n + 1$ instances of the original problem.

2.1.1 Robust Optimization criteria

Concerning robustness criteria, decisions can be made according to one of the following ROC: min-max, min-max regret, min-max relative regret, lexicographical min-max, α-robustness, bw-robustness and pw-robustness, etc., and their use mainly depend on the application and the optimization goals. Below a review of such criteria and some practice cases are provided based on the work that we conducted with Coco *et al.* (2014a).

The min-max family criteria, i.e., the absolute min-max, the min-max regret and the min-max relative regret, are usually referred as conservative criteria since they aim at minimizing the possible losses whenever the worst case scenario occurs (Kouvelis & Yu (1997)). The min-max criterion has been initially developed for game theory by Von Neumann (1928) for a two-player zero-sum game. The min-max dual problem relies on maximizing the minimum gain, named max-min. Both the min-max and the max-min are used whenever the worst case scenario can imply a major damage. Extending the work of Von Neumann (1928), a non-probabilistic decision-making model based on the worst-case of a decision is proposed by Wald (1939). According to the author definition, the decisions are ranked based on their worst-case outcomes. This strategy discards the worst case scenario among the possible decisions in an optimization process. Actually, the min-max criterion is one of the most studied ROC and has been applied to different problems (Kouvelis & Yu (1997); Ben-Tal & Nemirovski (2002); Kasperski (2008)), such as competitive situations, punctual risky decisions, robust shortest path (Yu & Yang (1998); Karasan *et al.* (2001); Montemanni *et al.* (2004)), and robust minimum spanning tree (Yaman *et al.* (2001); Montemanni & Gambardella (2005); Kasperski & Zieliński (2011)), among others.

Another ROC is the lexicographic min-max introduced by Dresher (1961), which extends the work of Von Neumann known as the nucleolus of a matrix game in game theory. The idea consists in selecting a subset of optimal strategies, based on the optimal solution of min-max, which takes advantage of the opponent player mistakes. In the lexicographic min-max not only the worst case is minimized, as in the classical min-max, but also the second worst case, the third worst, etc. Therefore, the lexicographic min-max refines the concept of solution in the min-max approach, since it selects a unique set of outcomes, but not necessarily a unique solution (if there is more than one solution, all solutions selected have the same distribution).

Extending the work from Dresher (1961), Orgryczak (1997) studied the lexicographic min-max to improve the min-max when uncertain data is modeled with discrete scenarios. The author also showed that the lexicographic min-max complies with both the “Pareto-Optimality Principle” and the “Principle of the Transfers”, whereas the standard min-max may violate both principles. In accordance with the “Pareto-Optimality Principle” the objective function is treated on the same way without preferences and/or specific assumptions, and the “Principle of the Transfers” is commonly recognized as the essential axiom for equity measure (Mandell (1991)). Finally, Orgryczak (1997) showed that solutions obtained by the lexicographic min-max are better than those obtained by the min-max on location problems. The lexicographic min-max has been used on allocation problems (Luss (1999); Wang *et al.* (2004)), location problems (Ogryczak (1999); Narasimhan *et al.* (2005)), and sensor node placement (Abidin & Din (2012)), as well as it has been modeled and integrated to mathematical programming (Kostreva *et al.* (2004); Hooker & Williams (2012)).

An alternative ROC to the min-max using discrete scenarios is presented by Bertsimas & Sim (2003). As described in Section 2.1, the authors proposed a robust integer programming model that allows to control the conservatism degree of a solution by using probabilistic bounds and violation constraints. Besides, Bertsimas & Sim (2003) have proposed an algorithm for robust network flows using their model. The min-max with discrete scenarios has been mostly tested on facility location problems (Averbakh & Bereg (2005)), robust prize-collecting Steiner tree problems (Álvarez-Miranda *et al.* (2013)), robust knapsack problem (Monaci *et al.* (2013)) and robust network loading problem with dynamic routing (Mattia (2013)).

The α-robustness has been proposed by Kalaï *et al.* (2012) to be less conservative. It extends the lexicographic min-max and is applied whenever the uncertain data is modeled using discrete scenarios. On this criterion, a parameter α defines a tolerance threshold for limiting the degree of conservatism of a solution. Both the lexicographic min-max and the α-robustness rank solutions considering the worst scenarios. Concerning the α-robustness, a vector of solutions, called “ideal solutions” is defined. An ideal solution is the best solution obtained for one scenario, and it is used to compute deviations. Given a solution ω, an ideal solution ω^*, and a set of scenarios S, the difference between the cost for ω and ω^* in each scenario $k \in S$ is evaluated and the maximum deviation is kept. For example, on a problem with two scenarios, if the ideal solutions for all scenarios are given in vector $cost(\omega^*) = (10, 8)$ and the cost of a solution ω in vector $cost(\omega) = (15, 10)$, then the maximum deviation between $cost(\omega)$ and $cost(\omega^*)$ is equal to 5. Recently, this criterion has been used to treat uncertain attribute evaluations in outranking methods (Durbach (2014)) which build a list of preferences following a set of predetermined alternatives.

Another ROC called *bw*-robustness is proposed by Roy (2010). The robust optimization is defined in his work as the ability to prevent undesirable impacts when uncertain data are handled. Then, a non-negative value b establishes a goal, in terms of cost, that has to be reached or improved in the highest number of scenarios whereas w sets the value of the worst cost to be guaranteed, regardless the scenarios ($w < b$). The bw-robustness is used only when the scenarios correspond to discrete values and the number of scenarios is large. Likewise for the min-max family criteria, the bw-robustness can be addressed as bw-absolute robustness, bw-absolute deviation and bw-relative deviation. The bw-robustness has been implemented for the robust shortest path problem (Gabrel *et al.* (2011)) and a robust military mission planning problem (Tang *et al.* (2011)), which consists in allocating resources and scheduling tasks during a military mission, in order to protect city borders.

The pw-robustness criterion proposed by Gabrel *et al.* (2013) is an extension of the bw-robustness criterion. A solution is said to be robust whenever it ensures a w value for all scenarios, and if it reaches a value b in a percentage p of scenarios.

2.2 Robust Classification Criteria

Two RCC are suggested here for measuring the quality of solutions, as well as the impact of the scenarios over these solutions: the *robust absolute score* and the *robust mean rank*. The general idea is to consider the proposed RCC for reinforcing local search procedures or for combining these RCC with existing ROC. Our RCC neither look for avoiding the worst case, nor for computing the worst case, the second one, etc., like in the α-robustness, the bw-robustness and the pw-robustness. They introduce new options for the evaluation and classification of solutions and scenarios in accordance with the definition of robustness (less affected by uncertainties). Then, when robustness of solutions is evaluated, a solution is said to be robust if it obtains the best results in a higher number of scenarios. In terms of scenarios, the proposed criteria identify the scenarios which produce a higher number of best solutions.

A possible application is to complement the existing ROC using our RCC, whenever one looks for avoiding the worst case, and after classifies the remaining solutions, i.e., a two-stage evaluation. Considering the application in local search procedures, one may accept a move that slightly degrades a traditional criterion but improves a RCC. Without the RCC, moves that degrade a traditional criterion are rejected and this can lead to a premature convergence of local search.

Starting with the *robust absolute score*, let us first consider a set of feasible solutions Ω to be evaluated and a set of scenarios S. An evaluation of solution $i \in \Omega$ for each scenario $k \in S$ has to be done, defined as the objective value of solution i in k (z_{ik}). Then, the best value b_i^* for each solution i is kept. Whenever a solution is the best for a specific scenario, it is assigned one point to its score ($score_i$). If for a given scenario, two or more solutions draw for best solution, they all count as best and one point is added to their $score_i$. Algorithm 2.1 shows the general structure on the computation of $score_i$. At the end, the number of times a solution has the best value on all scenarios is totalized, giving what we called *robust absolute score*. Regarding a solution i, the higher is its *robust absolute score*, the better the solution is. Considering the set of scenarios S, the *robust absolute score* can also be applied for evaluating each scenario k in Ω by summing the number of times the scenario k has the best value over all known solutions of Ω.

Algorithm 2.1 Score evaluation

```
for each solution $i \in \Omega$
    $b_i^* = \infty$
    //We compute $b_i^*$ for each solution
    for each scenario $k \in S$
        Compute $z_{ik}$
        if $z_{ik} < b_i^*$ then
            $b_i^* = z_{ik}$
        end if
    end for
    //Then we compute the score
    for each scenario $k \in S$
        if $z_{ik} = b_i^*$
            $score_i = score_i + 1$
        end if
    end for
end for
```

The *robust mean rank* is defined as the average rank of a solution over all scenarios. First, ranking for each solution i over the q discrete scenarios ($q \in S$) has to be computed, giving 1 to the best and q to the worst. Then, for a given solution, the z_{ik} with the best value b_i^* receives a ranking value equal to 1, the second best value receives a ranking value equal to 2 and so on. In case of ties, all solutions receives the same ranking value. For each solution and each scenario, the corresponding *robust mean rank* is equal to the mean number of ranking values assigned to each solution. The smaller the *robust mean rank*, the better the corresponding solution. The *robust mean rank* can also be applied for evaluating each scenario k in Ω by the estimation of the average rank for each scenario k over all known solution of Ω.

An example for the Robust Shortest Path with 3 discrete scenarios (RSP-D) is provided in Figure 2.1. Tables 2.1 and 2.2 summarize the corresponding values for some ROC and the proposed RCC. In Table 2.1, each line corresponds to a possible solution for the graph depicted in Figure 2.1. The next columns stand for the cost considering respectively, the first, the second and the third scenario. Then, the remaining columns contain respectively the values for the following criteria: min-max, min-max regret, and *robust absolute score*. Table 2.2 presents the classification for each solution in each scenario according to the *robust mean rank*. The three columns of classification indicate the ranking values assigned for each solution in each scenario. The last column indicates the *robust mean rank* per solution.

For solution 0-1-3-4-6, the *robust absolute score* is set to a value of 2 because this solution is the best for two scenarios out of three. It can be noticed that solution 0-1-4-6 stands for the best one using the mean, the min-max and the min-max regret criterion. On the contrary, the best solutions using the *robust absolute score* and the *robust mean rank* are 0-1-3-4-6 and 0-2-3-5-6 because their cost evaluation is the best for a higher number of scenarios. In fact, for some applications, the robustness may enclose this idea: "robust solutions are those which are good and could be applied in most of the scenarios considered". Clearly, whenever a decision maker wants to avoid the worst case, the RCC criteria cannot be applied alone. But, they could be coupled with a criterion from the min-max family considering the complementary information that is possible to obtain with the proposed RCC. Opening opportunities for sensitivity analysis, can be also done through the RCC.

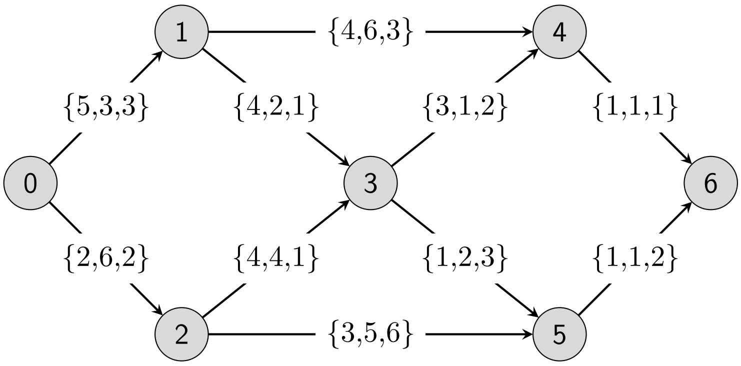


Figure 2.1: Example for the RSP-D.

2.3 Conclusions

In this chapter, general definitions of RO are introduced together with the different approaches to handle uncertainty. A review of the most common ROC including their applications in real context is also provided. Moreover, two RCC are proposed as supporting

	Scenarios			Evaluation			
Solutions	costs $k = 1$	 $k = 2$	 $k = 3$	Mean	min-max	min-max regret	*robust absolute score*
0-1-4-6	10	10	7	**9.00**	**10**	**4**	1
0-1-3-4-6	13	7	7	**9.00**	13	7	**2**
0-1-3-5-6	11	8	9	9.67	11	5	1
0-2-5-6	6	12	10	9.33	12	6	1
0-2-3-4-6	10	12	6	9.33	12	6	1
0-2-3-5-6	8	13	8	9.67	13	7	**2**
robust absolute score per scenario	2	2	**4**				

Table 2.1: Evaluating solutions for the RSP-D.

Solutions	Classification $k = 1$	 $k = 2$	 $k = 3$	*robust mean rank* per solution
0-1-4-6	2	2	1	1.67
0-1-3-4-6	2	1	1	**1.33**
0-1-3-5-6	3	1	2	2.00
0-2-5-6	1	3	2	2.00
0-2-3-4-6	2	3	1	2.00
0-2-3-5-6	1	2	1	**1.33**
robust mean rank per scenario	1.83	2.00	**1.33**	

Table 2.2: Applying the *robust mean score* on the RSP-D.

criteria for the existing ROC, the *robust absolute score* and the *robust mean rank*. In spite of their simplicity, the proposed RCC enclose the idea of better solutions on a higher number of scenarios, as well as they provide an easy way to identify the scenarios which produce a higher number of better solutions. The RCC can be used as independently or be coupled with a ROC as a complementary tool to strengthen local search procedures.

Conferences and Publications

Technical Report

- Coco, A. A., Solano-Charris, E. L., Prins, C., Santos, A. C. & Noronha, T. 2014. Robust optimization criteria: state-of-the-art and new issues. Technical Report UTT-LOSI-14001. ISSN:2266-5064, Université de Technologie de Troyes.

Chapter 3

Literature Review

As a vast literature is devoted to vehicle routing, this section is divided in two parts focusing on essential references and recent works. The first part cites a few key-articles on the Capacitated Vehicle Routing Problem (CVRP) and its solution methods, while the last part reviews the Vehicle Routing Problems (VRPs) with uncertainties.

3.1 The Capacitated Vehicle Routing Problem

The VRP introduced by Dantzig & Ramser (1959) has become a central problem in distribution management. The aim of this NP-hard combinatorial optimization problem is to determine a set of routes with minimum total cost to serve a set of customers, with known demands, using a fleet of identical vehicles based at a depot node. Most authors address the version with capacitated vehicles, or CVRP. Formally, the CVRP is defined on a complete undirected graph $G = (N, E)$ with a set $N = \{0, 1, 2...n\}$ of $n + 1$ vertices (customers), including the depot (0), and a set $E = \{(i, j)|i, j \in N, i \neq j\}$ of edges, where each edge $(i, j) \in E$ has one cost $c_{ij} \in \mathbb{R}$. A demand d_i is associated with each customer $i \in N$ and it cannot be split in several vehicles. A fleet of identical vehicles m is available at the depot, each one with a capacity equal to Q.

Due to its numerous industrial applications, the VRP has led to hundreds of variants with additional attributes such as heterogeneous fleets of vehicles, multiple depots, multi-period horizons, etc. It is also a laboratory-problem to test new ideas in optimization. As quoted by Laporte (2009), "*The study of the VRP has given rise to major developments in the fields of exact algorithms and heuristics. In particular, highly sophisticated mathematical programming approaches and powerful metaheuristics for the VRP have been put forward in recent years*".

The VRP has inspired a rich literature. A taxonomic review published by Eksioglu *et al.*

(2009) found 1494 papers from 1954 to 2006 (included) in bibliographical databases, using only “vehicle routing" as search expression. Several families of heuristics have been proposed along the years for VRPs. Two main classes are: constructive and improvement heuristics, and metaheuristics. Here below a review of such methods are presented.

3.1.1 Constructive heuristics

A large number of heuristics have been introduced between the 1960s and the 1980s to produce feasible solutions. One key characteristic of these heuristics is that they work in a greedy and iterative way. These heuristics are further divided in two categories: constructive and two phase methods.

The most often used constructive heuristic is the Clarke & Wright (1964) method. This method starts from an initial solution s_0 in which each customer is served by a different route, the heuristic looks for merging two routes extremities i and j, maximizing the cost saved s_i, where $s_i = c_{i0} + c_{0j} - c_{ij}$, under the condition that the merged route is feasible. As described in Laporte & Semet (2002), several variants of the saving methods have been proposed, mainly to speed up the computational time. This algorithm is popular due to its conceptual simplicity and its fairly good results. Other class of constructive methods are the sequential and parallel insertion methods. The first group expands one route at a time, see for instance Mole & Jameson (1976). The second group applies a parallel route construction, like in Christofides *et al.* (1979).

In the two phase methods the construction of solutions are decomposed into two separate subproblems, clustering and routing, giving two families, the *cluster-first route-second methods* and the *route-first cluster-second methods*. In *cluster-first route-second methods*, clusters of customers are first created, and then a Traveling Salesman (TSP) subproblem is solved for each cluster. Clusters are created by solving a Generalized Assignment Problem (GAP) for customers from Fisher & Jaikumar (1981). The authors proposed a geometric method based on the partition of the plane into m locations chosen to represent zones with a high customer density. A linear estimation of route costs is used as objective function for the GAP. The priority given to the assignment allows capacity constraints to be better handling highly constrained problems.

An example of *cluster-first route-second* is the sweep heuristic from Gillett & Miller (1974), in which the location of customers is represented in a polar coordinate system with origin at the central depot. A customer is chosen at random and the ray from the origin through the customer is “swept” either clockwise or counterclockwise. Customers are assigned to a given vehicle as they are swept, until the capacity constraint for that vehicle is reached. Then, a new vehicle is selected and the sweep algorithm iterates with assignments now being made to the new vehicle. Interesting surveys for these heuristics can be found in Laporte *et al.* (2000) and Cordeau *et al.* (2002).

In the opposite and less common *route-first cluster-second* approach, a TSP tour is computed and then partitioned into CVRP routes (Beasley (1983)). Interesting surveys for these heuristics were written by Laporte *et al.* (2000) and Cordeau *et al.* (2002). A recent review by Prins *et al.* (2014) covers route-first cluster-second algorithms for a large number of VRPs.

3.1.2 Improvement heuristics

Improvement heuristics, mostly referred as local search, can be employed to reinforce constructive heuristics and as intensification components in metaheuristics. Based on an initial solution ω, a local search heuristic explores a *neighborhood*, generally defined implicity by simple modifications (*moves*) on ω, in order to find an improving solution ω' that replaces ω. Two well-known strategies are employed to search the neighborhood: first improvement and best improvement. The first improvement strategy inspects the neighborhood and stops as soon as an improving solution is detected. In contrast, on the best improvement strategy, the complete neighborhood is explored and the best solution is returned. Many neighborhoods have been defined in the VRP literature, the most common are the *insert* or *relocate* neighborhood, the *swap* or *interchange*, 2-*opt* and 2-*opt**. These moves can be applied either to each route individually (*intra-route moves*) or to each pair of distintic routes (*inter-route moves*). The local search stops when no improving solutions can be found in the neighborhood. For more information and representative examples, the reader is referred to Lin (1965); Or (1976); Thompson & Psaraftis (1993); Breedam (1994); Kindervater & Savelsbergh (1997); Ergun *et al.* (2006), and, for general surveys, to Laporte *et al.* (2000) and Laporte & Semet (2002).

3.1.3 Exact methods

A large number of exact algorithms are available for the CVRP. These are based on integer linear programming (LP), Dynamic Programming (DP), Branch-and-Bound (B&B) or Branch-and-Cut (B&C). The B&C became a dominant approach for CVRP in 1990s and early 2000s (Araque *et al.* (1994); Achuthan *et al.* (2003); Lysgaard *et al.* (2004)) but instances with 50 customers could not be solved to optimality. Then, combination of methods have been developed. Good examples are the works from Fukasawa *et al.* (2006), Baldacci *et al.* (2010), Contardo & Martinelli (2014), Pecin *et al.* (2014). In particular, Fukasawa *et al.* (2006) worked on a combination of B&C and Lagrangian relaxation/column generation called Branch-Cut-and-Price for solving CVRP instances with up to 134 customers. A general approach is introduced by Baldacci *et al.* (2010) for solving both the CVRP and the Vehicle Routing Problem with Time Windows (VRPTW). The exact algorithm is based on set partitioning formulation (SP) of the problem and has tackled instances up to 101 clients. A method is presented by Contardo & Martinelli (2014) for Multi-Depot Vehicle Routing Problem (MDVRP) based on solution of ad-hoc vehicle-flow and SP formulations strengthened with valid inequalities. The SP formulation is solved by column-and-cut generation and validation of the proposed approach is handled on instances with maximum 151 clients. The most recent branch-cut-and-price for CVRP is the one from Pecin *et al.* (2014). This work introduces new cuts and combines several ingredients from previous works, such as route enumeration via dynamic programming and strong branching. All classical instances used for benchmarking exact algorithms are solved, including instances with up to 199 customers. As many real instances are much larger, heuristics are still required to determine good solutions in reasonable running time.

3.1.4 Metaheuristics

Concerning metaheuristics, significant progress has been done over the past two decades. Well known metaheuristics are Tabu Search (TS) which were the leaders in the 90's and Simulated Annealing (SA). Efficient implementations of TS are presented in Osman (1993); Taillard (1993); Gendreau *et al.* (1994); Rochat & Taillard (1995); Xu & James (1996); Barbarosoglu & Ozgur (1999); Cordeau *et al.* (2001); Toth & Vigo (2003), and variants of SA are studied in Tarantilis *et al.* (2002, 2004); Li *et al.* (2005). Another type of metaheuristics are the Population search heuristics which have became very popular, such as Genetic Algorithm (GA). Some successful implementations on VRP are presented in Berger & Barkaoui (2003); Baker & Ayechew (2003); Prins (2004); Mester & Bräysy (2007). The most effective metaheuristic for the CVRP is currently a hybrid GA designed by Vidal *et al.* (2012). Concerning learning mechanisms, good examples are the neural networks presented in Ghaziri (1991); Matsuyama (1991); Schumann & Retzko (1995); Ghaziri (1996). Three successive reviews by Cordeau *et al.* (2005), Gendreau *et al.* (2008) and Vidal *et al.* (2013) illustrate the fast and continuous advances in CVRP metaheuristics.

3.2 The VRP with uncertainties

The two main types of VRPs with uncertainties are the Stochastic Vehicle Routing Problem (SVRP) and the Robust Vehicle Routing Problem (RVRP). As the SVRP is not part of this work only a few references are presented about this approach in Section 3.2.1, while a more complete review for RVRP is introduced in Section 3.2.2.

3.2.1 Stochastic vehicle routing problems

There are many problem parameters of the VRP that in practice have a significant degree of uncertainty. Good entry points on stochastic vehicle routing problems are presented in Gendreau *et al.* (1996); Bertsimas & Simchi-Levi (1996). The most studied cases involve stochastic customers, where a customer needs to be serviced with a given probability (Bertsimas (1988); Waters (1989)); stochastic travel times, in which the service or travel times are modeled by random variables (see, e.g., Laporte *et al.* (1992); Kenyon & Morton (2003); Verweij *et al.* (2003)); and stochastic demands, where customer demands are known as probability distributions (Secomandi (2000); Laporte *et al.* (2002); Christiansen & Lysgaard (2007); Secomandi & Margot (2009); Sun (2014)). A major contribution to the study of Vehicle Routing Problem with Stochastic Demand (VRPSD) comes from Bertsimas (1992). This work illustrates the a priori method with different recourse policies to solve the VRPSD and derives several bounds, asymptotic results and other theoretical properties. Bertsimas & Simchi-Levi (1996) survey the development in VRPSD with emphasis on the insights brought by the algorithms proposed. Besides conventional stochastic programming framework, a Markov decision process for single stage and multistage stochastic models are introduced for the VRPSD in Dror (1993). As already mentioned, probability distributions which accurately model uncertainties must be known to use stochastic optimization approach. A recent and interesting article from Solano-Charris *et al.* (2015) discusses possible laws for stochastic travel times.

3.2.2 Robust vehicle routing problems

In the literature, robust optimization is considered on VRPs mainly to handle uncertainty on time windows, travel times, travel costs or demands, giving the RVRP. Table 3.1 summarizes the works on RVRP with columns corresponding to authors, methods, uncertainty representation, and uncertain data model taking into account the location of uncertainty in the mathematical model.

Regarding the RVRPs, the case of uncertain demands is the most investigated. Sungur *et al.* (2008) were the first to consider this problem. In their work, demand vectors are constructed as a deviation from an expected demand value that belongs to different bounded sets. Their robust formulation handle the uncertain demands in the constraints, following a robust approach introduced by Ben-Tal & Nemirovski (1998). To solve the problem the authors use the open source branch-and-cut-based VRP found in SYMPHONY library. Experiments address three different set of instances, i.e., random, clustered and modified from the literature, in a range from 15 to 100 customers, and show that robust solutions can protect from unmet demand, while incurring a small additional cost compared with the deterministic version.

Moghaddam *et al.* (2012) considered the robust optimization to deal with uncertain distributions of customer demands. A perturbation percentage from the nominal demands is defined over the constraints, while the objective is to minimize the travel distances. A variant of Particle Swarm Optimization is presented and results are compared with the ones obtained by Sungur *et al.* (2008).

The VRP with heterogeneous fleet and uncertain demands is studied by Noorizadegan *et al.* (2012). They provide mathematical formulations for the robust version and for a chance-constrained programming approach, both with uncertainties restricted in the right side of the constraints. A B&C algorithm is considered to add cutting planes that reduce the polyhedral region. Instances with 20 clients using CPLEX are tested and results are analyzed using the extra cost required to achieve a certain level of feasible routes, number of unmet customers and recourse cost.

Another study of RVRP with uncertainties on demands is presented in Gounaris *et al.* (2013). Robust Rounded Capacity Inequalities (RCI) are developed. The models handle customer demands as random variables on the right-hand side of constraints and determine the minimum cost delivery plan that is feasible for all anticipated demands realization. CPLEX is used to solve 90 instances from 15 to 135 customers: its generic cuts are disabled and replaced by RCI cuts. The results show that the best robust formulation, the Two-index Vehicle Flow formulation (2VF), is improved even further if RCI cuts are used: most instances up to 50 nodes are solved to optimality and the average gap for the other instances is below 5%.

The Open VRP with uncertain demands is introduced by Cao *et al.* (2014). A bounded uncertainty set for customer demands is described and transportation costs and unsatisfied demands in the specific bounded uncertainty set are minimized. A differential evolution algorithm is proposed and its performance is analyzed for different strategies, by considering the extra costs and unmet demand.

The most recent work regarding uncertain demands is presented in Gounaris *et al.* (In press). The robust formulation allows uncertain customer demands and the objective is to determine a least cost set of routes that remains feasible for all demand realizations within a prespecified uncertainty set. The authors implemented an adaptive memory programming metaheuristic using two classes of uncertainty sets. Computational experiments on benchmark intances with up to 483 customers and 38 vehicles are retrieved and new best solutions for a total of 123 benchmark instances are found.

As presented in Solano-Charris *et al.* (2014) and as it can be seen in Table 3.1, only some works have dealt with uncertain travel times or travel costs. A robust scenario approach for the vehicle routing problem with uncertain travel times is studied by Han *et al.* (2013). Multiple range forecasts in a set of time intervals are assumed. Uncertainty is handled by limiting the number of uncertain parameters on the constraints allowed to deviate from the nominal values (Bertsimas & Sim (2003)). Then for each realization (scenario) on an interval set of travel time, a robust route is identified, and the minimization of the worst case among all scenarios is applied. A two-stage recourse stochastic programming solved by a B&B algorithm is used. Tests on instances with up to 25 customers are solved assuming normal and severe traffic conditions.

The works from Toklu *et al.* (2013a,b) handle the VRP with uncertain travel costs. The total travel cost is minimized and uncertainty is expressed as intervals. The approach from Bertsimas & Sim (2003) (see Section 2.1) is implemented to control the degree of uncertainty on the model. One ant colony system and a multiple ant colony system are introduced in Toklu *et al.* (2013a,b), respectively, and tested on instances with up to 150 customers with different conservativeness degree configurations.

An approach for the RVRP with uncertain travel times has recently been studied by Solano-Charris *et al.* (2015). The set of arc costs are replaced by a set of discrete scenarios and the main objective is to build a set of routes using the lexicographic min-max criterion from Orgryczak (1997). Thus, the worst cost over all scenarios is minimized but ties are broken using the other scenarios, from the worst to the best. Different methods are adapted, i.e., a Greedy Randomized Adaptive Search Procedure (GRASP), an Iterated Local Search (ILS), a Multi-Start ILS (MS-ILS), and a MS-ILS based on giant tours (MS-ILS-GT). Tests on instances up to 100 clients and 30 scenarios are reported.

In terms of time windows, Agra *et al.* (2013) addressed the uncapacitated VRP raised by maritime transportation. Travel costs and times are asymmetric and depend on the vessel used. The goal is to find a least-cost set of routes respecting time windows and feasibility for all travel times in a given uncertainty polytope. Two new formulations are introduced and solved by ad-hoc decomposition algorithms, making the model more efficient if the “budget of uncertainty" approach of Bertsimas & Sim (2003) is used. Instances with 20 to 50 nodes can be solved in 30 minutes on a 2.5 GHz PC.

Concerning multiple parameters under uncertainty, the work from Ordóñez (2010) considered uncertain demands, travel times, costs and customers. The author outlined different robust models depending on the source of uncertainty, the VRP formulation, and correlation between uncertain coefficients. Uncertainty in travel costs is introduced in the objective function, and uncertain demands and travel time coefficients in the constraints. A convex and bounded uncertainty set is estimated and the problem is solved via the robust optimization approach from Ben-Tal & Nemirovski (1998). Results for small instances are provided and compared with the ones obtained by a chance constrained programming method and models based on stochastic programming with recourse.

Finally, uncertainties on both travel times and demands are found in Lee *et al.* (2012) for a VRP with customer deadlines. The budget uncertainty approach from Bertsimas & Sim (2003) is defined by limiting the sum of deviations on travel times, and the deviation of demands to their nominal values. The robustness of a solution is achieved by looking for a feasible solution for any travel time and demand defined in the uncertainty sets, minimizing the travel time. A set-partitioning formulation is proposed and solved by column generation. The uncertainties are limited to the column generation subproblem, using a robust version of a shortest path algorithm with resource constraints. Instances with 20 to 40 customers are solved to optimality.

Authors	Methods	Uncertainty/Representation			Uncertain data model
		Time windows	Travel cost/Times	Demand	
Solano-Charris *et al.* (2015)	Local Search Based Metaheuristics		Discrete scenarios, Kouvelis & Yu (1997)		Objective function
Gounaris *et al.* (In press)	Adaptive Memory Programming			Budget uncertainty, Bertsimas & Sim (2003)	Constraints
Cao *et al.* (2014)	Differential evolution algorithm			Bounded set, Ben-Tal & Nemirovski (1998)	Constraints/ Objective function
Toklu *et al.* (2013b)	Multiple Ant Colony System		Budget uncertainty, Bertsimas & Sim (2003)		Constraints/ Objective function
Toklu *et al.* (2013a)	Ant Colony System		Budget uncertainty, Bertsimas & Sim (2003)		Constraints/ Objective function
Han *et al.* (2013)	B&C algorithm		Budget uncertainty, Bertsimas & Sim (2003)		Constraints/ Objective function
Gounaris *et al.* (2013)	B&C algorithm			Bounded set, Ben-Tal & Nemirovski (1998)	Constraints
Agra *et al.* (2013)	Cutting plane Technique	Bounded set, Ben-Tal & Nemirovski (1998)			Constraints
Noorizadegan *et al.* (2012)	B&C algorithm			Bounded set Ben-Tal & Nemirovski (1998)	Constraints
Lee *et al.* (2012)	Dantzig-Wolf Decomposition Approach	Budget uncertainty, Bertsimas & Sim (2003)			Demands, travel times, and time windows in constraints/ travel times in objective function
Moghaddam *et al.* (2012)	Particle Swarm Optimization			Interval set	Constraints
Ordóñez (2010)	Analytical		Bounded set, Ben-Tal & Nemirovski (1998)		Demands and travel times in constraints/ travel times in objective function
Sungur *et al.* (2008)	B&C algorithm			Bounded set, Ben-Tal & Nemirovski (1998)	Constraints

Table 3.1: Vehicle routing problems with uncertainty.

Chapter 4

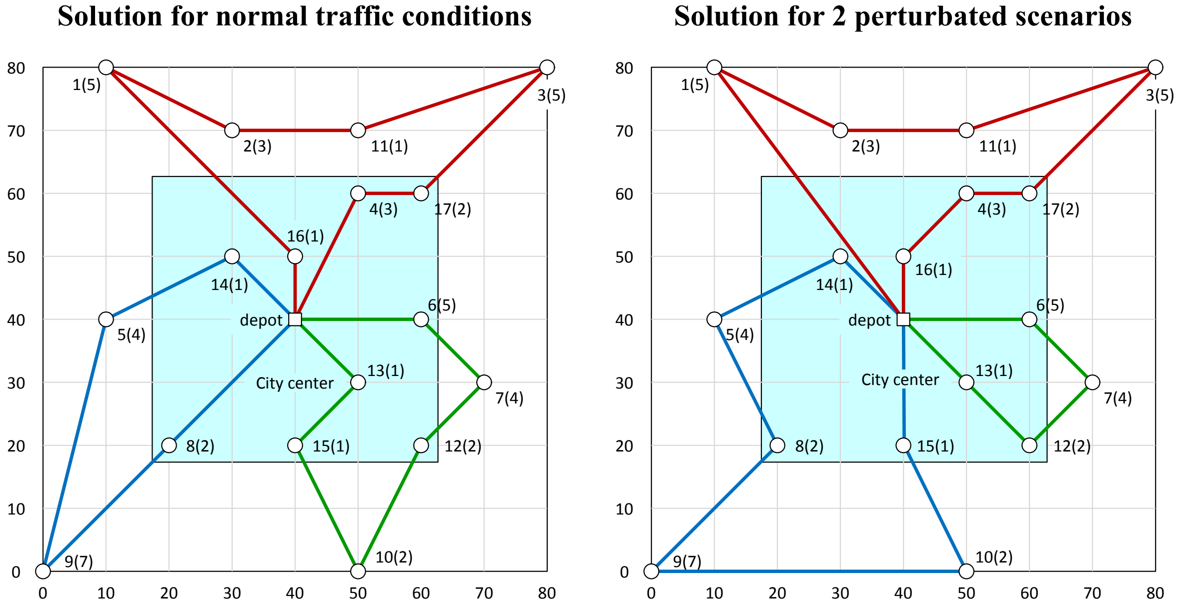
The Robust Vehicle Routing Problem with Uncertain Travel Costs

4.1 Introduction

The RVRP considered here addresses uncertainty in travel costs, modeled by discrete scenarios, in which each scenario defines one travel time for each arc. The aim is to minimize the worst cost (total route duration) over all scenarios, using a lexicographic method to break ties. These scenarios do not correspond to realizations of random variables and are not associated with probabilities of occurrence. Then, a key-issue is the accurate modeling of travel times. Most authors use additive probability distributions, e.g., normal such as Kenyon & Morton (2003) or gamma laws as Taş *et al.* (2013). As travel times often grow quickly from a minimum to a maximum and then slowly decrease with a long tail, a lognormal distribution looks pertinent (Lecluyse *et al.* (2009)). According to Gómez *et al.* (In press), an accurate distribution may not exist and a specific one must be chosen for each problem and even each arc. These authors propose to use phase-type distributions, which can approximate any positive and continuous distribution, while computing exactly their convolutions.

Concerning the realism of the use of discrete scenarios, they can be provided by the traffic control centers implemented in most large cities, with two advantages: they correspond to real data and reflect possible correlations among arc costs. Interesting solutions could be obtained if enough traffic records are considered. The price to pay is a running time multiplied by $O(q \log q)$ for q scenarios, as can be seen in Section 4.4. The confirmation of the accuracy of all models handling uncertain travel times would require large-scale validations on real road networks to see the real cost achieved on the field.

The proposed problem can be found in the city of Troyes, where the municipality sends



Solution for normal traffic conditions

Solution for 2 perturbated scenarios

Each customer i is given with its demand q_i in brackets

Central depot with 3 vehicles of capacity 20

Travel times $c_{i,j}$ are equal to Euclidean distances

Minimum total duration of the routes = 439

Scenario $k = 1$: +20% on $c_{i,j}$ if i, j are in the city center

Scenario $k = 2$: +20% on $c_{i,j}$ if i, j are not in the center

The three routes are affected

The optimal robust solution costs 464 (+5.7%)

Figure 4.1: Examples of solution on normal conditions and perturbated scenarios.

technicians to replace damaged bulbs of public lights. Routes are prepared using average travel times but, due to traffic conditions, the actual duration of the routes is often larger (see Figure 4.1 for an example). The maximum daily working time is not exceeded, but the uncertainties delay the allocation of technicians to other tasks afterwards, like repairs in municipal buildings.

A strong point of the RVRP approach is that can be viewed as a multi-objective problem, each objective being the total cost for one scenario. An optimum for the lexicographic min-max objective is also Pareto-optimal for this multi-objective version (Orgryczak (1997)). As no other solution dominates it, it is appealing for a decision maker.

This chapter is structured as follows: Section 4.2 and 4.3 present the formal definition and the robustness criterion here implemented for the RVRP with uncertain travel times. The proposed greedy heuristics are introduced in Section 4.4. Section 4.5 is dedicated to a population-based metaheuristic, while Section 4.6 describes the different components of the iterative multistart strategies. For all methods here adapted, the computational experiments and the conclusions are provided.

4.2 Problem definition

The RVRP addressed here is motivated by uncertain travel times induced by traffic conditions. As the two directions of a road can be differently affected when traffic is perturbed, a complete and directed graph $G = (N, A)$ is considered, while the CVRP is defined on an undirected network. N denotes the node-set, with one depot (node 0) and n customers with positive demands d_i, while $A = \{(i, j) \mid i, j \in N, i \neq j\}$ stands for the arc-set. A fleet of m identical vehicles with capacity Q is based at the depot.

Instead of being random variables, the uncertain arc costs are modeled by a set of q discrete scenarios $S = \{1, 2, ...q\}$, where each scenario k defines one non-negative cost c_{ij}^k for each arc $(i, j) \in A$. Let Ω define the set of feasible solutions. Like in the CVRP, a solution $\omega \in \Omega$ is a set of routes: each route is done by one vehicle which leaves the depot, serves a subset of customers whose total demand does not exceed Q, and returns to the depot. In particular, for the RVRP the main goal is to build a set of routes considering the minimization of the worst total cost over all scenarios. As the CVRP, known to be NP-hard, corresponds to the RVRP with one scenario ($q = 1$) then, the RVRP is also NP-hard.

$$\min \; z = \delta \tag{4.1}$$

$$\sum_{(i,j)\in A} c_{ij}^k x_{ij} \leq \delta \qquad \forall\, k \in S \tag{4.2}$$

$$\sum_{j\in N} x_{ji} = 1 \qquad \forall\, i \in N\backslash\{0\} \tag{4.3}$$

$$\sum_{j\in N} x_{ij} = 1 \qquad \forall\, i \in N\backslash\{0\} \tag{4.4}$$

$$\sum_{i\in N\backslash\{0\}} x_{0i} \leq m \tag{4.5}$$

$$d_i \leq l_i \leq Q \qquad \forall\, i \in N\backslash\{0\} \tag{4.6}$$

$$l_i - l_j + Q\, x_{ij} \leq Q - d_j \qquad \forall\, (i, j) \in A, i \neq 0, j \neq 0 \tag{4.7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in A \tag{4.8}$$

$$\delta \geq 0 \tag{4.9}$$

The RVRP can be specified by the following min-max MILP given from Equation 4.1 to 4.9 derived from a CVRP model proposed by Kulkarni & Bhave (1985). This compact model avoids vehicle indexes: the routes are defined by binary variables x_{ij}, equal to 1 if and only if arc (i, j) is used in the solution. In (4.2), the total cost of the arcs used for each scenario is bounded above by the variable δ, defined in constraint (4.9). The goal (4.1) is to minimize δ. Constraints (4.3) and (4.4) mean each client is reached by one incoming and one outgoing arc. The maximum fleet size is guaranteed via constraints (4.5). Inequalities (4.6) and (4.7) generalize the model from Miller *et al.* (1960) called MTZ subtour-elimination constraints for the Traveling Salesman Problem (TSP). Assuming that d_i is a collected quantity, l_i is the vehicle load when leaving i. If arc (i,j) is not used, the constraint (4.7) for this arc becomes $l_i - l_j \leq Q - d_j$ and is trivially satisfied: as $l_i \leq Q$ and $l_j \geq d_j$ via (4.6), the left-hand side cannot exceed $Q - d_j$. Consider now a route $(r_1, r_2, \ldots, r_u)$, which means $x_{r_i, r_{i+1}} = 1$ for $i = 1, 2, \ldots, u-1$: constraints (4.6) and (4.7) hold by setting for instance $l_{r_1} = 0$ and $l_{r_i} = l_{r_{i-1}} + d_{r_i}$ for $i = 2, 3, \ldots, u$. Vehicle capacity is also respected since $l_{r_u} \leq Q$ from (4.6). The binary variables are declared in (4.8). The other equations lead to a min-max version.

It is noticed that the following constraints for the CVRP can be used to solve the MILP substantially faster in practice. A simple lower bound for the required number of vehicles is defined by constraint (4.10), where d_{tot} denotes the total demand. Desrochers and Laporte proposed to replace the MTZ constraints (4.7) by the lifted inequalities (4.11), which were incorrectly written in their paper. Here the version corrected by Kara *et al.* (2004) is considered.

$$\sum_{i \in N \setminus \{0\}} x_{0i} \geq \lceil d_{tot}/Q \rceil \tag{4.10}$$

$$l_i - l_j + Q\, x_{ij} + (Q - d_i - d_j)\, x_{ji} \leq Q - d_j \qquad \forall\ (i,j) \in A, i \neq 0, j \neq 0 \tag{4.11}$$

Figure 4.1 shows an example with 16 nodes and 2 scenarios, solved using CPLEX. The arc costs are equal to the Euclidian distances. The CVRP solution cost without uncertainties is 439 (left). On the right, the two scenarios (morning and evening) increase the distances by 20% on same arcs. Our model gives a cost of 464 over the two scenarios, only 5.7% higher than the CVRP solution. The bad idea (yet considered by many people) is to use the trips of the deterministic solution and to recompute their costs using the arc costs of the two scenarios, but in that case the worst cost over the two scenarios is 12% higher. This shows the interest of using a robust optimization approach.

4.3 Lexicographic min-max robustness criterion

As described in Chapter 2 the lexicographic min-max criterion, see Orgryczak (1997), has several interests. First, it gives the same worst cost as the classical min-max approach. Second, the solution offers a better protection against the second worst scenario, the third one, etc. Third, an optimal solution for this criterion is also Pareto-optimal for the multi-objective version of the problem, each objective being the total cost of the routes for one scenario (Orgryczak (1997)). Fourth, a positive effect on the worst cost in local search procedures can be observed: while a search based on the min-max criterion can induce the same decrease in the worst cost and could be blocked as soon as no move improves the worst cost, the decrease of the worst cost can often restart after a few moves which reduce the other costs.

The MILP of Section 4.2 can be solved lexicographically in q iterations. The first iteration solves the min-max version to get the worst scenario π and its cost z^*_π. The worst scenario is the one for which constraint (4.2) is tight. The second iteration solves the min-max version with the set of scenarios $S \setminus \{\pi\}$ and the additional constraint $\delta \leq z^*_\pi$, etc.

In heuristics, lexicographic comparisons must be done explicitly. Let $cost(\omega, k)$ be the cost of solution ω for scenario k, i.e., the total cost of the routes with the arc costs of this scenario. Define $cost(\omega) = (cost(\omega, 1), cost(\omega, 2), \ldots, cost(\omega, q))$ as the vector of costs for all scenarios, and $worst(\omega) = \max\{cost(\omega, k) \mid k \in S\}$ the worst cost. The goal is to determine a solution ω^* minimizing the worst cost over all scenarios, i.e., such that $\omega^* = \arg\min\{worst(\omega) \mid \omega \in \Omega\}$. Then, for each solution ω, a vector $V(\omega)$ with the components of $cost(\omega)$ sorted in decreasing order of costs, is built. If $V(\omega, k)$ is the k-th component of $V(\omega)$, the $worst(\omega) = V(\omega, 1)$. Solution ω is strictly better than solution ω' if and only if $V(\omega)$ is lexicographically smaller than $V(\omega')$, denoted as $V(\omega) < V(\omega')$. The comparison, done like words in a dictionary, is specified in Algorithm 4.1.

All proposed algorithms use lexicographic comparisons to select the best move among a set of candidates. For example, consider a classical operation, cheapest insertion, and one solution ω with $cost(\omega) = (10, 20, 30)$. Its lexicographic vector is $V(\omega) = (30, 20, 10)$, with a worst cost $worst(\omega) = V(\omega, 1) = 30$ reached in scenario 3. If customer i is inserted between nodes u and v in one trip, the cost variation for scenario k is $\Delta_k = c^k_{ui} + c^k_{iv} - c^k_{uv}$. Let Δ denotes the cost variation vector for the q scenarios and compare one insertion $\omega \to \omega'$ with $\Delta' = (15, 16, 4)$, and one $\omega \to \omega''$ with $\Delta'' = (16, 12, 6)$. The resulting solutions are such that $cost(\omega') = (25, 36, 34)$ and $cost(\omega'') = (26, 32, 36)$. They have the same worst cost (36), even if it is now reached by scenario 2 for ω'. The lexicographic vectors are $V(\omega') = (36, 34, 25)$ and $V(\omega'') = (36, 32, 26)$. As $V(\omega'') < V(\omega')$, the second insertion will be preferred.

Algorithm 4.1 Lexicographic comparison of two solutions for q scenarios.

```
function better $(\omega, \omega')$ : boolean
    $k \leftarrow 1$
    while $(k \leq q)$ and $(V(\omega, k) = V(\omega', k))$ do $k \leftarrow k + 1$ end while
    better $\leftarrow (k \leq q)$ and $(V(\omega, k) < V(\omega', k))$
end function
```

In a local search for a minimization problem, an improving move must reduce the cost of the incumbent solution. Here, a move without effect on the worst cost is accepted if it gives a lexicographically better solution. For instance, consider again ω with $cost(\omega) = (10, 20, 30)$ and $V(\omega) = (30, 20, 10)$. A move $\omega \rightarrow \omega'$ with $\Delta' = (2, -3, 2)$ is rejected because its degrades the worst cost. If $\Delta' = (2, -3, -1)$, it is accepted like in a classical local search since the objective function (the worst cost) is improved. Finally, if $\Delta = (9, -1, 0)$, the worst cost is unchanged but the move is accepted since $V(\omega') = (30, 19, 19) < V(\omega) = (30, 20, 10)$.

In the CVRP, most moves can be evaluated in $O(1)$ on the basis of cost variations, e.g., $c^k_{ui} + c^k_{iv} - c^k_{uv}$ in the previous example for node insertion. In the RVRP, the complexity becomes $O(q \log q)$ because of the sorting algorithm required to get the lexicographic vector.

4.4 Greedy Heuristics

Constructive heuristics are required to initialize the proposed metaheuristics. Thus, the well known Clarke & Wright (1964) heuristic (CW) initially designed for the CVRP is adapted and a Randomized Clarke and Wright version (RCW) is also derived. Finally, the insertion heuristics and their pilot versions are implemented. From now on, it is assumed that a RVRP solution ω is encoded as a list of trips, and each trip as a list of customers between two copies of the depot.

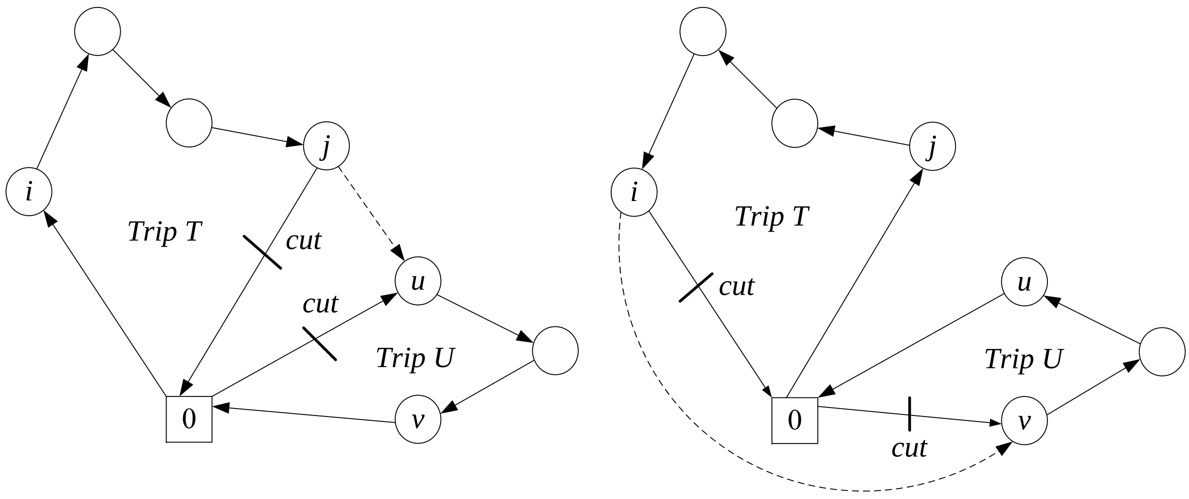
4.4.1 Clarke and Wright heuristic for the RVRP

The original version of CW for the CVRP begins with a trivial solution composed of one dedicated trip for each customer (the “daisy") and then performs mergers (concatenations) of two trips in order to minimize the cost variation at each iteration. For one route with last customer i and another with first customer j, this cost variation $c_{ij} - c_{i0} - c_{0j}$ only depends on i, j and the depot. In parallel, each concatenation saves one vehicle.

An efficient implementation for the CVRP consists in sorting the $n(n-1)/2$ edges of the graph in increasing order of cost variation, giving a list Λ. This step is possible in $O(n^2 \log n)$. Then each edge (i, j) of Λ is tested: if i and j are still at the extremities of two distinct routes whose total load does not exceed Q, then these two routes are merged. These tests cost $O(1)$ and the merger itself $O(n)$. As at most $n - 1$ mergers are executed during the algorithm, the total complexity is dominated by the initial sort in $O(n^2 \log n)$.

For the RVRP, CW is more involved due to the directed networtk and the multiple scenarios, see Algorithm 4.2. There are eight ways to merge two trips T and U: If $\overline{T}$ is the reversal of trip T, then to test, four cases are necessary to evaluate (T, U), $(T, \overline{U})$, $(\overline{T}, U)$ and $(\overline{T}, \overline{U})$, plus four others by exchanging T and U. In the CVRP four cases are enough since for instance (T, U) and $(\overline{U}, \overline{T})$ have the same cost. Moreover, each merger must be evaluated for each scenario and the best merger is the one giving a solution with a minimum lexicographic vector. Finally, sorting the arcs at the beginning brings nothing because in the RVRP the route costs intervene in cost variations.

In lines 1–2, CW builds the initial solution ω (daisy) and pre-computes four types of data: the number of trips, $nbtrips(\omega)$, the load of route T, $load(T)$, the cost of trip T before customer i in scenario k, $b(T,k,i)$, and the cost after i, $a(T,k,i)$. The parameter ω is implicit in the three last symbols, to make the notation lighter. These data are also prepared for the inverted trip $\overline{T}$. Figure 4.2 shows on two cases of mergers how these data can be used.



Concatenation of trips T and U

New trip cost:

$$b(T,k,j)+c^k_{ju}+a(U,k,u)$$

Concatenation of trips T and U but inverted

New trip cost:

$$b(\overline{T},k,i)+c^k_{iv}+a(\overline{U},k,u)$$

Figure 4.2: Examples of mergers for a scenario k: (T,U) and $(\overline{T},\overline{U})$.

Then, each iteration of the *repeat* loop browses all pairs of distinct trips (T,U) and searches for the best pair (lexicographically), even if this leads to a degraded solution. A boolean $found$ is set to true if a feasible merger is detected. For a given pair (T,U), only the first four cases of mergers are evaluated since the pair (U,T) is inspected by another iteration. For each case, CW determines the cost vector W of the resulting solution and sorts it to get the lexicographic vector. When improved, the lexicographic vector of the best merger (W_{best}) is updated and the corresponding pair of routes (T_{best}, U_{best}) is recorded, with indexes multiplied by -1 to remember that a route must be inverted. A detail is not mentioned in the algorithm to reduce its number of lines: the *for* loops which compute W lines 13, 17, 21 and 25 are dropped if $W_k > W_{best}(1)$. This simple condition is sufficient to guarantee that the merger tested is not better than the best one found so far. This test divides by four the running time of CW on average.

In lines 30–35, provided feasible mergers have been detected ($found$ = true), the best merger is executed if it improves upon the incumbent solution or if fleet size is exceeded. In other words, degrading mergers are accepted but only to avoid vehicles in excess. The customers of trip U_{best} are moved at the end of T_{best}, U_{best} is deleted and the pre-computations updated for T_{best} only. The main loop stops when it finds no feasible merger: either a single route remains or all mergers violate vehicle capacities.

Note that more than m vehicles can be used at the end, but this case is not seen on the instances tested which, like all CVRP benchmarks, satisfy the condition $m \geq \lceil \sum_{i=1}^{n} d_i/Q \rceil$. If this occur, the local search in the next section can save vehicles by moving their customers to different trips. Anyway, determining if there exists a solution with at most m vehicles is a bin-packing problem, which is already NP-hard.

The initial daisy and all pre-computations can be done in $O(nq)$. There are at most $n-1$ main iterations, when all routes are merged into one TSP tour. At each iteration, $O(n^2)$ pairs (T, U) are inspected and each pair is evaluated in $O(q \log q)$ due to the sorting algorithm. The complexity to execute the best merger is negligible: the two routes are inverted (if necessary) and concatenated in $O(n)$ while pre-computations are updated in $O(q)$ only for trips T_{best} and $\overline{T}_{best}$. Hence, CW runs in $O(n^3 q \log q)$ for the RVRP, instead of $O(n^2 \log n)$ for the CVRP without uncertainties.

4.4.2 Randomized version

As two of the proposed metaheuristics perform multiple restarts, a randomized version RCW (Randomized Clarke and Wright) is also designed to provide them with initial solutions. Randomness is introduced by perturbing the solution costs evaluated for each possible merger in Algorithm 4.2. A perturbation level θ is fixed and a random number ρ ranging from 0 to θ percent of current solution cost is computed before each merger evaluation. Finally, ρ is added to W_k, $k = 1, 2, \ldots, q$. The random generator is initialized with a fixed seed on input, to get reproducible results, and the algorithm is nested in a main loop doing a given number of iterations $ncalls$, without resetting the random number generator at each iteration. The resulting procedure is invoked by a RCW $(\theta, ncalls, \omega)$ statement.

Algorithm 4.2 Clarke and Wright heuristic for the RVRP – $CW\ (\omega)$.

```
compute the initial solution $\omega$ with one route per customer and set $nbtrips(\omega)$ to $n$
prepare all pre-computed data
**repeat**
    compute the lexicographic vector $V(\omega)$
    set $W_{best}$ to $(\infty, \infty, \ldots, \infty)$ and $found$ to false
    **for** each trip index $T$ of $\omega$ **do**
        $i, j \leftarrow$ first and last customers of $T$
        **for** each trip index $U$ of $\omega$ such that $U \neq T$ and $load(T) + load(U) \leq Q$ **do**
            $found \leftarrow$ true
            $u, v \leftarrow$ first and last customers of $U$
            $ctu \leftarrow cost(T) + cost(U)$ //*Total cost of $T$ and $U$, to shorten formulas below*
            //*evaluate concatenation* $(T, U)$
            **for** $k \leftarrow 1$ **to** $q$ **do** $W_k \leftarrow cost(k) - ctu + b(T, k, j) + c^k_{ju} + a(U, k, u)$ **end for**
            sort $W$ in decreasing cost order
            **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow T$; $U_{best} \leftarrow U$ **end if**
            // *evaluate concatenation* $(T, \overline{U})$
            **for** $k \leftarrow 1$ **to** $q$ **do** $W_k \leftarrow cost(k) - ctu + b(T, k, j) + c^k_{jv} + a(\overline{U}, k, v)$ **end for**
            sort $W$ in decreasing cost order
            **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow T$; $U_{best} \leftarrow -U$ **end if**
            // *evaluate concatenation* $(\overline{T}, U)$
            **for** $k \leftarrow 1$ **to** $q$ **do** $W_k \leftarrow cost(k) - ctu + b(\overline{T}, k, i) + c^k_{iu} + a(U, k, u)$ **end for**
            sort $W$ in decreasing cost order
            **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow -T$; $U_{best} \leftarrow U$ **end if**
            // *evaluate concatenation* $(\overline{T}, \overline{U})$
            **for** $k \leftarrow 1$ **to** $q$ **do** $W_k \leftarrow cost(k) - ctu + b(\overline{T}, k, i) + c^k_{iv} + a(\overline{U}, k, v)$ **end for**
            sort $W$ in decreasing cost order
            **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow -T$; $U_{best} \leftarrow -U$ **end if**
        **end for**
    **end for**
    **if** $found$ and $(W_{best} < V(\omega)$ or $nbtrips(\omega) > m)$ **then**
        **if** $T_{best} < 0$ **then** $T_{best} \leftarrow -T_{best}$; invert trip $T_{best}$ **endif**
        **if** $U_{best} < 0$ **then** $U_{best} \leftarrow -U_{best}$; invert trip $U_{best}$ **endif**
        concatenate the clients of $U_{best}$ at the end of $T_{best}$ then delete trip $U_{best}$
        update pre-computed data for trips $T_{best}$ and $\overline{T}_{best}$
    **end if**
**until** (not $found$)
```

4.4.3 Insertion heuristics for the RVRP

The principle of such heuristics is to perform at each iteration a best insertion among unserviced customers. Thus, the insertion done in each iteration is the one that gives a solution with the smallest cost vector, lexicographically, and ensures vehicle capacities. Algorithms 4.3 and 4.4 presents the structure of the two types of insertion methods, dealing with the Sequential Best Insertion Heuristic (SBIH) and a Parallel version (PBIH), respectively.

SBIH begins with a single trip T, reduced to a loop on the depot. As detailed in lines 9-26, best insertions are performed in the current trip until all customers are serviced or if the insertion of remaining customers violate vehicle capacity. The heuristic stops in the first case or repeats the same process with one new empty trip in the second case. The result for the RVRP is a solution ω. The trips are often not well balanced because the last trip can be small. In the sequel, $next(u)$ denotes the next node after node u in its trip.

PBIH employs all vehicles available and fills m trips in parallel. More precisely, m empty trips are initialized and, at each iteration, the heuristic evaluates all feasible insertions of unrouted customers in these trips (lines-9-26). As demands cannot be split, PBIH sometimes fail to use only m vehicles. In that case, one extra trip is created and the heuristic continues like in SBIH to serve the remaining customers. The final result is a RVRP solution ω. Solutions could be find in $O(n^2q$ log $q)$ by the SBI and for the PBI heuristics in $O(mn^2q$ log $q)$.

4.4.4 Pilot versions of Insertion heuristics

In the pilot method from Duis & Voß (1999), a main heuristic calls an auxiliary heuristic (the pilot heuristic) to guide its decisions. A frequent technique is to reuse the main heuristic as pilot heuristic. We selected this technique to design two pilot methods derived from SBIH and PBIH, called Pilot SBI and Pilot PBI.

Consider for instance the Pilot SBI. SBIH is transformed into a subroutine which can complete a set R of emerging routes SBIH(R) (instead of starting from $R = \emptyset$). Pilot SBI works like the original SBIH: at each iteration it adds to the route being constructed one unrouted customer. The original SBIH selects the customer with cheapest insertion. Pilot SBI also tests each unrouted customer u: u is inserted to minimize the insertion cost but the subroutine SBIH is called to finish $R \cup \{u\}$. The customer which is really inserted is the one giving the smallest final cost. Hence, the SBIH subroutine is a kind of look-ahead procedure which takes into account, to some extent, possible future insertions. The Pilot PBI is similar but calls PBIH as a subroutine to finish partial solutions. Since the pilot

Algorithm 4.3 Sequential Best Insertion heuristic for the RVRP – *SBIH* (ω).

```
initialize a vector $free$ of $n$ booleans to true
prepare an empty solution $\omega$
$nfree \leftarrow n$
repeat
    create a new trip $T$ in $\omega$, as a loop on the depot
    //loop inserting one customer $best_i$ in the current $T$ until it is not possible
    repeat
      set $W_{best}$ to $(\infty, \infty, \ldots, \infty)$ and $found$ to false
      for $i \leftarrow 1$ to $n$ do if $free(i)$ and $(load(T) + d_i \leq Q)$ do
        //loop testing the insertion of $i$ after each node $u$ in $T$
        $u \leftarrow depot$
        repeat
          //compute cost vector if $i$ were inserted after $u$
          for $k \leftarrow 1$ to $q$ do
            $\Delta_k \leftarrow c^k_{ui} + c^k_{i,next(u)} - c^k_{u,next(u)}$
            $cost(\omega', k) \leftarrow cost(\omega, k) + \Delta_k$
          end for
          $W \leftarrow cost(\omega)$ in decreasing cost order
          if $W < W_{best}$ then
            $W_{best} \leftarrow W$
            $best_i \leftarrow i$
            $best_u \leftarrow u$
          end if
          $u \leftarrow next(u)$
        until $u = depot$
      end for
      if $W_{best} < \infty$ then
        insert $best_i$ after node $best_u$ in trip $T$
        $free(best_i) \leftarrow$ false
        $nfree \leftarrow nfree - 1$
      end if
    until $W_{best} = \infty$
until $(nfree = 0)$
```

Algorithm 4.4 Paralell Best Insertion heuristic for the RVRP – *PBIH* (ω).

```
initialize a vector $free$ of $n$ booleans to true
prepare an empty solution $\omega$
$nfree \leftarrow n$
create $m$ empty trips (loop on the depot)
repeat
      $W_{best}$ to $(\infty, \infty, \ldots, \infty)$
      //loop testing all possible insertions for each customer i
      for $i \leftarrow 1$ to $n$ do if $free(i)$ then do
        for each trip index $T$ in $\omega$ such that $(load(T) + d_i \leq Q)$ do
          //loop testing the insertion of i after each node u in the trip
          $u \leftarrow depot$
          repeat
            //compute cost vector if i were inserted after u
            for $k \leftarrow 1$ to $q$ do begin
              $\Delta_k \leftarrow c^k_{ui} + c^k_{i,next(u)} - c^k_{u,next(u)}$
              $cost(\omega', k) \leftarrow cost(\omega, k) + \Delta_k$
            end for
            $W \leftarrow cost(\omega)$ in decreasing cost order
            if $W < W_{best}$ then
              $W_{best} \leftarrow W$
              $best_i \leftarrow i$
              $best_u \leftarrow u$
              $best_T \leftarrow T$
            end if
            $u \leftarrow next(u)$
          until  $u \leftarrow depot$
        end for
      end for
      if $W_{best} < \infty$ then
        insert $best_i$ after node $best_u$ in trip $best_T$
        $free(best_i) \leftarrow$ false
        $nfree \leftarrow nfree - 1$
      end if
      else if $nfreew > 0$ then add one trip reduced to a loop on the depot to $\omega$
until $(nfree = 0)$
```

version of a heuristic consists in calling it as a subroutine at each iteration, the complexity of the non-pilot version is squared, making the pilot approach time-consuming.

4.4.5 Computational evaluation

The algorithms are implemented in the Pascal-like language Delphi and executed on a Dell Precision M6600 portable PC with a 2.2 GHz Intel Core i7, 16 GB of RAM and Windows Professional. The MILP of Section 4.2 is solved with IBM ILOG CPLEX Optimization Studio 12 under default parameters. The following subsections describe the instances used and the results obtained with the adapted greedy heuristics for the RVRP.

4.4.5.1 Implementation and instances

Two sets of instances were randomly generated. The first set, to compare the constructive heuristics with the MILP, contains 18 small-size instances with $n = \{10, 15, 20\}$ customers, $m = \{2, 3\}$ vehicles and $q = \{10, 20, 30\}$ scenarios. The file names have an n–m–q format. All demands d_i and arc costs c_{ij}^k are integers randomly drawn in $[1, 50]$. Vehicle capacity Q is manually selected to load the fleet between 80% and 90% of its capacity. Normally, the number of customers (10 to 20) is a typical limit to solve directly a MILP for the CVRP. For the same size, the RVRP is expected to be harder in practice, due to its multiple scenarios.

The second set gathers 24 medium-size instances with $n = \{50, 100\}$ and $q = \{10, 20\}$. The fleet size is $m = \{5, 10\}$ for $n = 50$ and $m = \{10, 20\}$ for $n = 100$. The two values of m for each problem size correspond to 5 and 10 stops per route on average. The vehicle capacity is $Q = 1,000$ while the demands are random integers drawn to use nearly 90% of fleet capacity, i.e., $d_{tot} \approx 0.9 \times mQ$, d_{tot} denoting the total demand. Each node i has its coordinates x_i and y_i randomly selected in $[0, 1000]$. The arc costs are travel times proportional to the Euclidean distance e_{ij}. For each arc (i, j), e_{ij} is first computed, then the integer cost c_{ij}^k for each scenario k is drawn at random in $[e_{ij}, (1 + \beta/100) \times e_{ij}]$, where $\beta = \{10, 50, 100\}$ is a maximum deviation in percent to the baseline travel time, corresponding to three growing levels of traffic perturbation. For instance, $\beta = 100$ means that travel times are doubled in the worst case. The file name format is the same as in the small-size instances, except that the value of β is added at the end, e.g., 100–20–20–100 for $n = 100$, $m = q = 20$ and $\beta = 100\%$.

4.4.5.2 Results on small-size instances

The time limit for the MILP is set to 4 hours (14,400 seconds). The randomized Clarke and Wright heuristic RCW performs $ncalls = 50$ iterations with a perturbation factor $\theta = 8\%$

to randomize the savings. The detailed results are listed for each instance in Table 4.1 and represented in Figure 4.3. For each instance are mentioned the total demand d_{tot}, the equal capacity Q of vehicles and the trivial lower bound (loads) to $[d_{tot}/Q]$. For the MILP formulation, the solution cost for the relaxed linear program z_{lr}, the best lower bound z_{lb}, the best solution cost z_{ub}, the percentage gap $(z_{ub}/z_{lb} - 1) \times 100$ and the running time in seconds CPU are given. Proven optima are identified in bold while dashes (-) mean the time limit is reached. 10 out of 18 instances are solved to optimality but no instance with $n = 20$ customers and/or $q = 30$ scenarios, except 10–3–30.

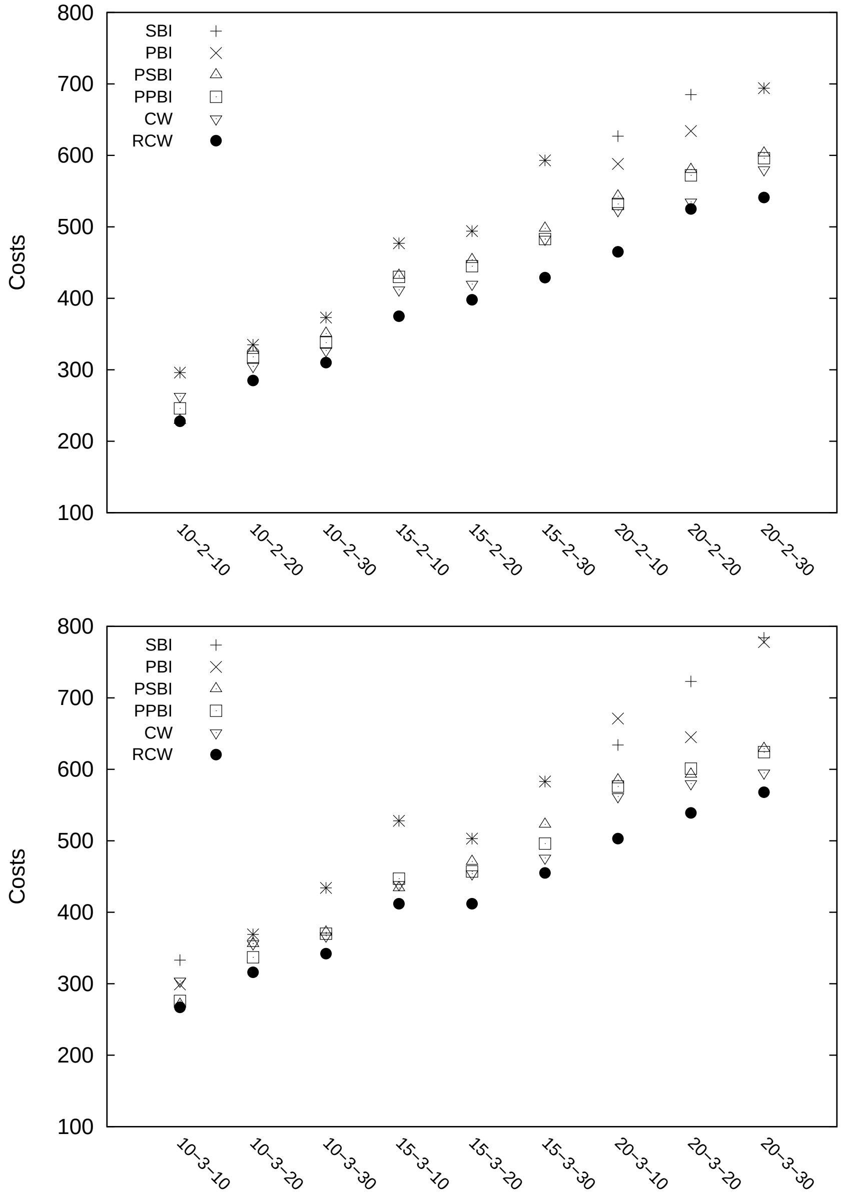


Figure 4.3: Constructive heuristics results for small-size instances.

A summarized version is presented in Table 4.2, using three performance indicators

Instance attributes				CPLEX					Insertion heuristics								Clarke & Wright			
n-m-q	d_{tot}	Q	Loads	z_{lr}	z_{lb}	z_{ub}	*Gap*	CPU	SBI	*Gap*	PBI	*Gap*	Pilot SBI	*Gap*	Pilot PBI	*Gap*	CW	*Gap*	RCW	*Gap*
10-2-10	264	150	1.76	195.5	217	**217**	0.0	8.2	296	36.4	296	36.4	229	5.5	246	13.4	263	21.20	228	5.1
10-2-20	264	175	1.51	250.6	284	**284**	0.0	32.2	335	18.0	335	18.0	327	15.1	318	12.0	305	7.39	285	0.4
10-2-30	264	200	1.32	272.1	301	**301**	0.0	35.8	373	23.9	373	23.9	351	16.6	338	12.3	327	8.64	310	3.0
15-2-10	346	200	1.73	320.2	346	**346**	0.0	313.2	477	37.9	477	37.9	432	24.9	430	24.3	412	19.08	375	8.4
15-2-20	346	230	1.50	339.3	373	**373**	0.0	6,560.00	494	32.4	494	32.4	454	21.7	445	19.3	420	12.60	398	6.7
15-2-30	346	260	1.33	368.7	398	404	1.5	-	593	49.0	593	49.0	498	25.1	483	21.4	483	21.36	429	7.8
20-2-10	441	250	1.76	402.3	419	423	1.0	-	627	49.6	588	40.3	543	29.6	532	27.0	523	24.82	465	11.0
20-2-20	441	300	1.47	443.3	460	470	2.2	-	685	48.9	634	37.8	580	26.1	572	24.3	535	16.30	525	14.1
20-2-30	441	350	1.26	463.7	481	501	4.2	-	694	44.3	694	44.3	603	25.4	596	23.9	580	20.58	541	12.5
10-3-10	264	95	2.78	227.0	255	**255**	0.0	10.40	333	30.6	299	17.3	271	6.3	276	8.2	304	19.22	267	4.7
10-3-20	264	105	2.51	279.5	316	**316**	0.0	24.70	369	16.8	369	16.8	356	12.7	337	6.6	356	12.66	**316**	0.0
10-3-30	264	115	2.30	302.0	337	**337**	0.0	38.90	434	28.8	434	28.8	372	10.4	370	9.8	367	8.90	342	1.5
15-3-10	346	120	2.88	349.1	381	**381**	0.0	2,200.00	528	38.6	528	38.6	434	13.9	447	17.3	439	15.22	412	8.1
15-3-20	346	140	2.47	365.2	399	**399**	0.0	6,871.00	503	26.1	503	26.1	471	18.0	457	14.5	454	13.78	412	3.3
15-3-30	346	160	2.16	394.4	426	433	1.6	-	583	36.9	583	36.9	523	22.8	496	16.4	476	11.74	455	6.8
20-3-10	441	160	2.76	427.3	443	448	1.1	-	634	43.1	671	51.5	585	32.1	576	30.0	562	26.86	503	13.5
20-3-20	441	175	2.52	466.3	481	497	3.3	-	723	50.3	645	34.1	593	23.3	601	24.9	580	20.58	539	12.1
20-3-30	441	190	2.32	489.6	508	528	3.9	-	784	54.3	778	53.1	629	23.8	624	22.8	595	17.13	568	11.8

Table 4.1: Detailed results for constructive heuristics on small-size instances.

averaged on the 18 instances: the minimum percentage gaps to CPLEX lower bound (Min gap *LB*%), the number of CPLEX lower bounds retrieved by each heuristic (*LB hits*) and the average running time per run in seconds.

Indicator	MILP	SBI	PBI	Pilot SBI	Pilot PBI	CW	RCW
Min gap *LB* %	1.05	37.0	34.6	19.6	18.3	16.6	7.3
LB hits	10	0	0	0	0	0	1
Time (s)	9,350.40	<0.001	< 0.001	0.010	0.054	0.005	0.060

Table 4.2: Indicators for constructive heuristics for small-size instances.

CPLEX finds 10 optima out of 18 instances, its average solution gap is close to 1% at the expense of a large running time. The two insertion heuristics are very fast (less than 1 ms on average) but their deviations to CPLEX are disappointing: 37% for SBI and 34.6% for PBI. The pilot heuristics are much better: Pilot SBI reaches 19.6% in 10 ms while Pilot PBI gets 18.3% in 54 ms. Note that the look-ahead mechanism employed in the pilot heuristics is time-consuming. CW is a bit better but 10 times faster, with a gap around 17% in 5 ms. With RCW 7.3% is achieved with RCW with $\theta = 8\%$ and 50 iterations, but in 0.06 seconds. Allocating more iterations increases the running time while bringing only a small improvement. No heuristic is able to retrieve CPLEX optimal solutions, except RCW which finds one.

4.4.5.3 Results for medium-size instances

The medium-size instances are out of reach for the MILP: even in four hours, CPLEX is unable to improve the upper bound obtained by the Clarke and Wright heuristic. This is why the MILP formulation is excluded from the tests on these instances. The parameters used are still $ncalls = 50$ for RCW. The perturbation factor θ randomizing the savings in RCW was 8% for the small instances because too many iterations return identical solutions using smaller values. A 1% factor is enough on medium-size instances.

The results are detailed in Table 4.4 and have the same format as the ones for small instances, except that the reference for the gaps is now the best solution (BS) found for each benchmark problem during the tests. The summarized results are presented in Table 4.3 and contains the average cost (Avg cost), the average gap (Avg gap *BS*) and the average running time per run in seconds.

Indicator	SBI	PBI	Pilot SBI	Pilot PBI	CW	RCW
Avg cost	49,892.7	49,008.5	40,289.1	34,787.7	17,319.5	17,079.38
Avg gap *BS* %	213.81	209.8	151.7	117.3	7.7	6.1
Time (s)	0.002	0.020	2.787	575.64	0.472	28.069

Table 4.3: Indicators for constructive heuristics for medium-size instances.

4.4.6 Conclusions for constructive heuristics

The results for the constructive heuristics demonstrate that in spite of its simplicity, the RVRP is a very hard problem to be solved. On the classical VRP insertion heuristics are typically at 20% above the optimum and CWH at 10%. On the RVRP, their efficiency is strongly degraded. Among the constructive heuristics the most promising methods in terms of running time and average solution gap are the CW with a gap around 17% in 5 ms, and its randomized version achieving 7.3% in 0.060 s. As can be seen, no heuristic is able to retrieve CPLEX optimal solutions, except RCWH which finds one. These results allow to see which are the best heuristics to implement according accuracy and efficiency, and to select them to provide metaheuristics with initial solutions.

4.5 A genetic algorithm

The general framework of the Genetic Algorithm (GA) proposed by Holland (1975) is adapted here to solve the RVRP. The components of our GA are summarized in detail on the following sub-sections. Algorithm 4.5 shows the general structure. The initial population *Pop* with g individuals is generated in line 3, where half of the solutions $g/2$ (rounded up) are obtained by an adaptation of the best insertion heuristic, and the remaining solutions are randomly generated. As this GA was designed before the constructive heuristics of Section 4.4 during the thesis, we have not used the best constructive heuristic to initialize the GA. Solutions are ranked in line 4 according to the min-max criterion in which the solutions are sorted according to the costs (one per scenario k) in decreasing order, giving the min-max lexicographic vector of a solution. At each iteration $iter$ of the GA, an intermediate population Pop' with g new solutions is generated (lines 7-14). To generate a new solution, two parents P_1 and P_2 are selected using the binary tournament selection. This strategy selects two random solutions and the solution with the smallest rank is kept as P_1. The process is repeated to select P_2. Then, a dedicated crossover operator is applied to obtain an offspring ω. Mutation inter-routes and intra-routes are also incorporated. At the end of each iteration, the population contains $2g$ solutions. Only the best g solutions determined by

Instance attributes			Insertion heuristics								Clarke & Wright			
n-*m*-*q*-β	Loads	BS	SBI	CPU	PBI	CPU	Pilot SBI	CPU	Pilot PBI	CPU	CW	CPU	RCW	CPU
50-5-10-10	4.48	8,633	24,318	0.001	26,199	0.002	19,928	0.234	17,404	5.680	8,986	0.062	8,861	3.777
50-5-10-50	4.43	9,936	30,996	0.001	31,826	0.002	26,746	0.252	22,331	6.025	10,687	0.066	10,508	3.370
50-5-10-100	4.47	11,525	38,146	0.001	38,771	0.002	31,511	0.267	24,175	5.465	12,544	0.067	12,236	3.418
50-5-20-10	4.49	7,966	26,383	0.001	27,261	0.005	21,208	0.661	17,568	15.119	8,453	0.158	8,302	9.885
50-5-20-50	4.44	10,014	31,617	0.001	32,841	0.005	23,868	0.588	24,549	13.440	10,609	0.160	10,567	8.886
50-5-20-100	4.54	10,913	36,391	0.001	33,174	0.005	24,196	0.711	24,728	12.931	12,099	0.157	11,497	8.587
50-10-10-10	9.05	10,882	26,339	0.001	28,455	0.003	20,376	0.243	18,879	17.194	11,469	0.059	11,079	3.562
50-10-10-50	8.96	12,849	31,448	0.001	29,696	0.004	26,798	0.217	19,470	20.965	13,861	0.063	13,525	3.298
50-10-10-100	8.97	16,212	44,513	0.000	43,860	0.004	34,199	0.165	30,728	18.668	16,852	0.059	16,852	3.070
50-10-20-10	8.97	11,621	26,761	0.001	27,284	0.009	22,161	0.385	18,319	53.877	12,043	0.147	12,043	9.171
50-10-20-50	8.94	14,337	37,630	0.001	36,498	0.010	28,904	0.489	21,862	51.165	15,089	0.138	14,786	7.783
50-10-20-100	8.99	15,042	39,894	0.001	38,889	0.009	32,733	0.415	26,735	46.300	16,775	0.146	15,777	7.797
100-10-10-10	8.87	12,767	51,879	0.001	52,388	0.014	42,355	3.641	36,292	286.512	13,535	0.527	13,535	32.821
100-10-10-50	8.94	15,082	64,598	0.002	58,604	0.014	48,169	3.182	42,823	276.707	16,863	0.513	16,659	28.492
100-10-10-100	8.92	18,254	83,336	0.002	75,770	0.014	66,606	3.102	58,881	290.115	20,973	0.547	20,769	29.072
100-10-20-10	8.89	13,640	49,034	0.005	49,442	0.037	37,384	7.407	34,974	766.691	14,648	1.173	14,525	81.201
100-10-20-50	9.01	16,124	66,949	0.005	59,617	0.036	50,965	9.000	42,365	688.114	16,938	1.268	16,938	71.862
100-10-20-100	8.93	18,451	72,401	0.004	74,962	0.037	64,136	7.893	56,631	680.131	21,890	1.294	20,971	71.656
100-20-10-10	17.97	22,385	59,203	0.002	59,164	0.028	46,299	2.669	36,789	1048.655	22,925	0.430	22,925	28.627
100-20-10-50	17.88	24,786	71,855	0.001	68,936	0.027	56,098	2.589	51,036	1047.033	26,504	0.481	26,504	27.007
100-20-10-100	17.96	29,187	86,440	0.001	86,657	0.027	68,860	2.282	67,047	1070.001	31,711	0.480	31,711	25.609
100-20-20-10	18.10	19,943	51,430	0.004	55,490	0.063	45,702	7.249	36,218	2446.639	20,582	1.123	20,582	75.298
100-20-20-50	18.11	24,731	71,398	0.004	66,952	0.064	60,285	6.891	46,632	2433.092	27,086	1.097	26,543	65.070
100-20-20-100	17.88	29,583	74,466	0.004	73,467	0.067	67,452	6.350	58,469	2514.749	32,546	1.120	32,210	64.343

Table 4.4: Detailed results for constructive heuristics on medium-size instances.

the ranking procedure are kept for the next iteration (line 17). In order to avoid premature convergence in the population a renewal procedure is implemented after no improvement is found in a fixed number of iterations *niterpr* in lines 19-22. The procedure is repeated until a maximum number of iterations are reached *niter* and the GA returns the ω^*.

4.5.1 Chromosomes and fitness

Each solution in the GA referred as chromosome is encoded as a set of routes, in which the customers appear in the order they are visited by m vehicles. Delimiters (0) are used to separate a route. Table 4.5 illustrates a chromosome for a feasible route with $n = 7$ clients with $m = 3$ vehicles.

Algorithm 4.5 Genetic Algorithm for the RVRP – *GA(ω)*

```
 1: Require $G = (N, A), m, Q, S, d_i \; \forall i \in N$
 2: $iter \leftarrow 1$
 3: Initialize $Pop$ with $g$ solutions
 4: $Pop$=Ranking($Pop$)
 5: repeat stopping criterion is not met
 6:     //Generation of $Pop'$ with $g$ new solutions
 7:     while $\mid Pop \mid < 2g$ do
 8:         Choose $P_1$ and $P_2$ using the binary tournament selection
 9:         $\omega \leftarrow$ Crossover($P_1$, $P_2$)
10:         if $random < \varphi$ do
11:         $\omega \leftarrow$ Mutation($\omega$)
12:         end if
13:         Add $\omega$ to $Pop'$
14:     end while
15:     //Create a new population
16:     $Pop \leftarrow$Ranking($Pop'$)
17:     Reduce $Pop$ to its best $g$ solutions
18:     $iter \leftarrow iter + 1$
19:     if renewal criterion is met after $iter = niterpr$ do
20:         Perform partial renewal($Pop$)
21:         $Pop$ =Ranking($Pop$)
22:     end if
23: until stopping condition
24: Output the solution $\omega^*$
```

Considering the same notation as in Section 4.3 for the evaluation of a solution (chromosome), the cost for each scenario is computed and the maximum cost, $\max\{cost(\omega, k) \mid k \in S\}$, is kept. Then, all individuals in the population are ranked following a minimization, $\arg\min\{worst(\omega) \mid \omega \in \Omega\}$.

0	1	0	3	4	5	0	6	7	0

Table 4.5: Example of a chromosome representation.

4.5.2 Initial Population

The generation of the initial population *Pop* containing g chromosomes combines random solutions, and better solutions computed through an adaptation of the best insertion heuristic which has been shown to be a fast method with reasonable quality on solutions, see Section 4.4.5. The first $g/2$ individuals (rounded up) to enter in *Pop* are built using the best insertion heuristic. Initially, unserviced clients are sorted in decreasing order of demands to reduce the probability of failure. Considering this order of unserviced customers, the classical best insertion heuristic includes in a route, at each iteration, unserviced customers, considering the smallest impact on the cost. However, here, the scenarios have to be taken into account. A simple way is applied and consists in randomly selecting a scenario k to evaluate the generation of each chromosome. Thus, the heuristic builds routes in parallel as follows: a customer is selected in the order of the sorted list and inserted in the first route, the second selected customer is set in the second route, etc. whenever m initial customers (seed customers) has been included per route, the procedure iterates to include the second customer per each route. The procedure expands the current routes by inserting one unserviced customer at a time until no customer can be inserted due to vehicle capacity. Since demands cannot be split, the insertion heuristic can fail to use only m vehicles. Then, extra vehicles can be added to the solution in order to ensure that each customer is visited by a vehicle. This violation on the number of vehicles is not repaired, and it is left to be managed by the GA selection.

The remaining individuals $g/2$ are randomly generated. On the contrary of solutions generated with the best insertion heuristic, routes are built in a sequential way. Thus, unserviced customers are assigned to a vehicle as long as vehicle capacity is respected. Whenever no customer can enter the current route, a new route is created. The procedure stops when all customers are attended by one vehicle.

4.5.3 Genetic operators

Such as genetic operators, crossover and mutation have been applied. For the crossover, the selection of solutions are done following the binary tournament selection proposed by Goldberg (1989). This strategy selects two random solutions. Then, solution with the best min-max cost is kept as P_1. The process is repeated to select P_2.

We designed a crossover operator which selects a sequence of customers from a parent and insert it in the other one. Table 4.6 shows an example of this operator. A sequence of customers from a route in P_1 (gray color) is considered. Then, an offspring (solution) ω is obtained by the cheapest insertion in P_2 of the sequence selected from P_1 (gray color) considering the min-max criterion. Repeated customers are dropped from the offspring ω to ensure that each customer i is visited once.

P_1	0	1	2	0	**3**	**4**	5	0	6	7	0
P_2	0	2	5	0	3	1	4	0	**7**	6	0
ω	0	2	5	0	1	0	7	**3**	**4**	6	0

Table 4.6: Example of crossover.

The mutation operator can be applied to an offspring ω, according to a fixed rate φ. Two different mutations have been developed. The first mutation (M_1) is performed intra-route and randomly swaps two customers from the same route. The second mutation (M_2) applies a swap inter-routes, and it is performed only if the capacity of both vehicles is respected. Table 4.7 presents an example of the two types of mutation.

P_1	0	1	2	0	3	4	5	0	6	7	0
M_1	0	1	2	0	**5**	6	**3**	0	4	7	0
M_2	0	1	2	0	3	**6**	5	0	**4**	7	0

Table 4.7: Example of mutation.

4.5.4 Renewal Procedure and stopping criterion

The renewal procedure is a restarting method for genetic algorithm when the best solution is not improved during a certain number of iterations. Its goal is to restart the evolution process for escaping from local optima. This procedure is applied whenever the best solution found so far does not change after a pre-specified number of iterations $niterpr$. A percentage of the population is replaced by new individuals generated with the best insertion method. Then, Pop is re-sorted. The GA stops after $niter$ iterations.

4.5.5 Computational evaluation

Computational experiments were performed on a Dell Precision 6600M, Intel Core i7-2720QM, 2.2 GHz with 16GB of RAM. The proposed GA was implemented in C++ on Visual Studio Express 2013. The MILP formulation of Section 4.2 is considered in here to see the performance of the proposed GA. As previously mentioned the MILP is solved using IBM ILOG CPLEX Optimization Studio 12 under default parameters. Section 4.5.5.1 and Section 4.5.5.2 describe the instances used and the results obtained with the GA, respectively.

4.5.5.1 Implementation and instances.

The test bed instances are the ones described in Section 4.4.5.1. For small-size instances $n = \{10, 15, 20\}$ customers, $m = \{2, 3\}$ vehicles and $q = \{10, 20, 30\}$ scenarios. And for medium-size instances $n = \{50, 100\}$, $q = \{10, 20\}$, m is up to 20 vehicles, and $\beta = \{10, 50, 100\}$.

4.5.5.2 Results on small-size instances for the GA.

Table 4.8 summarizes the results for the MILP formulation and the GA for the set of small-size instances. The MILP results correspond to the linear relaxation value z_{lr}, the best lower bound z_{lb}, the percentage gap Gap and the running time in seconds CPU. The results for the GA are indicated with the cost solution z, the percentage gap $(z/LB - 1) \times 100$ and the running time in seconds CPU. As previously said in Section 4.4.5.2 the time limit for the MILP is fixed to 4 hours (14,400 seconds). Optimal values for the GA are identified in bold while dashes (-) in the CPU column indicate that the time limited is reached. In fact, the proposed GA could found good solutions within 8 seconds in a small average Gap of 3.14%. Moreover, the proposed GA retrieves 7 optima out of 10 optima values that were met using the MILP.

Instance attributes				CPLEX					GA		
n-*m*-*q*	d_{tot}	Q	Loads	z_{lr}	z_{lb}	z_{ub}	*Gap*	CPU	z	*Gap*	t
10-2-10	264	150	1.76	195.5	217	**217**	0.00	8.20	**217**	0.00	2.76
10-2-20	264	175	1.51	250.6	284	**284**	0.00	32.20	**284**	0.00	4.14
10-2-30	264	200	1.32	272.1	301	**301**	0.00	35.80	**301**	0.00	4.14
15-2-10	346	200	1.73	320.2	346	**346**	0.00	313.20	**346**	0.00	2.76
15-2-20	346	230	1.50	339.3	373	**373**	0.00	6,560.00	374	0.27	6.21
15-2-30	346	260	1.33	368.7	398	404	1.51	-	409	2.76	6.21
20-2-10	441	250	1.76	402.3	419	423	0.95	-	447	6.68	7.59
20-2-20	441	300	1.47	443.3	460	470	2.17	-	493	7.17	4.14
20-2-30	441	350	1.26	463.7	481	501	4.16	-	520	8.11	4.83
10-3-10	264	95	2.78	227.0	255	**255**	0.00	10.40	**255**	0.00	6.90
10-3-20	264	105	2.51	279.5	316	**316**	0.00	24.70	**316**	0.00	4.14
10-3-30	264	115	2.30	302.0	337	**337**	0.00	38.90	**337**	0.00	4.14
15-3-10	346	120	2.88	349.1	381	**381**	0.00	2,200.00	406	6.56	13.10
15-3-20	346	140	2.47	365.2	399	**399**	0.00	6,871.00	411	3.01	6.21
15-3-30	346	160	2.16	394.4	426	433	1.64	-	436	2.35	6.21
20-3-10	441	160	2.76	427.3	443	448	1.13	-	484	9.26	6.21
20-3-20	441	175	2.52	466.3	481	497	3.33	-	520	8.11	6.21
20-3-30	441	190	2.32	489.6	508	528	3.94	-	551	8.46	6.90

Table 4.8: Detailed results for the GA on small-size instances.

Results on medium-size instances for the GA. For the set of medium-size instances, calibration with different configuration of parameters were used to see which one produces the best results from the GA. To select the parameters, a fine-tuning procedure was considered, beginning from a promising configuration using 90% of partial renewal when no improvement is found after 25 iterations, and a φ =10%. Then, this configuration is used on a subset of 4 representative instances varying the number of iterations, and the population size. The number of iterations were stated to 100, 150. The population size was defined according to the number of customers per each instance (Santos *et al.* (2013)). In addition, 5 runs per instance were taken into account using different seeds and the results are summarized in Figure 4.4.

Results for the GA contains the Best Cost solutions (z_{min}) and the Average (z_{avg}) over the 4 configurations tested per instance, considering a population size of $g = |N|$ with 100 and 150 iterations, respectively. Figure 4.5 corresponds to a population size of $g = 2|N|$ tested with 100 and 150 iterations. The results show that the configuration with promising results is the one with $g = 2|N|$ and 150 iterations, which can achieve better cost solutions and good average costs for the min-max criterion. Thus, these parameters were kept for the complete set of medium-size instances, using $g = 2|N|$, 150 iterations, 90% of partial renewal, 25 iterations for the partial renewal and a 10% of rate of mutation.

Table 4.9 summarizes the results for the GA over the set of the medium-size instances. The results correspond to the headings of z_{min}, z_{avg} and CPU$_{avg}$ (Average time) using 5 runs per instance with different seeds. Results show a good performance of the GA which take 118.4 seconds on average to solve instances with $n = 50$, and 573.21 seconds to solve instances with $n = 100$. It is also noticed that whenever n and q increase and the loads of the vehicle are tighted, the complexity of the problem and the computational time to solve the RVRP raise. Considering the impact of the scenarios on the robustness of solutions, it seems that when more scenarios are handled that does not improve the robustness according to the min-max criterion.

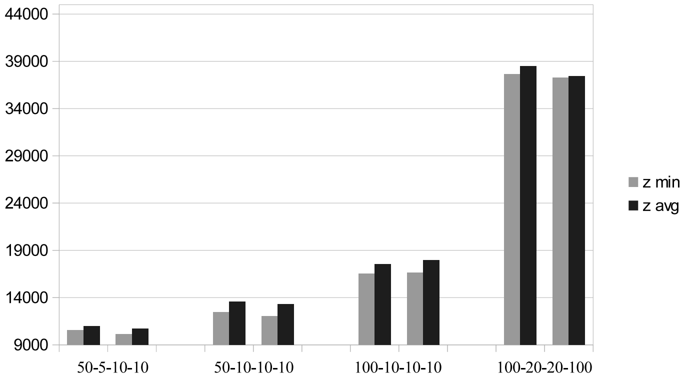


Figure 4.4: Calibration of medium-size instances with $g = |N|$.

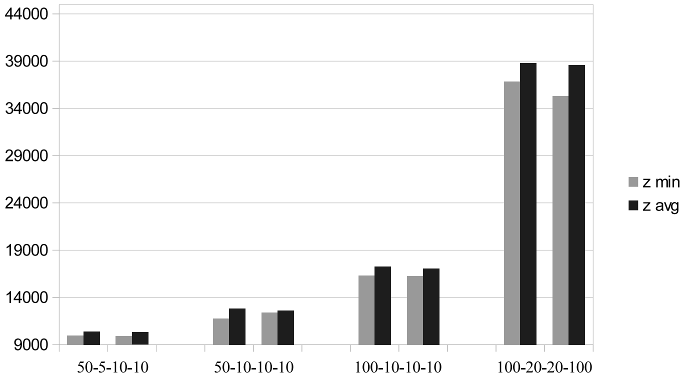


Figure 4.5: Calibration of medium-size instances with $g = 2|N|$.

Instance attributes		GA		
n-m-q-β	Loads	z_{min}	z_{avg}	CPU$_{avg}$
50-5-10-10	4.48	9,797.0	10,049	106.5
50-5-10-50	4.43	12,002.0	12,994	121.1
50-5-10-100	4.47	13,891.0	14,960	122.3
50-5-20-10	4.49	9,302.0	9,740	140.8
50-5-20-50	4.44	11,040.0	12,093	133.7
50-5-20-100	4.54	13,358.0	13,943	125.4
50-10-10-10	9.05	12,394.0	13,073	93.0
50-10-10-50	8.96	15,469.0	16,253	111.9
50-10-10-100	8.97	18,413.0	19,653	102.8
50-10-20-10	8.97	13,060.0	13,293	114.6
50-10-20-50	8.94	15,893.0	16,790	124.1
50-10-20-100	8.99	16,802.0	18,272	124.8
100-10-10-10	8.87	14,813.0	16,532	572.8
100-10-10-50	8.94	19,812.0	20,939	598.9
100-10-10-100	8.92	21,386.0	23,840	649.4
100-10-20-10	8.89	16,980.0	17,738	705.0
100-10-20-50	9.01	19,878.0	21,204	580.7
100-10-20-100	8.93	24,647.0	25,323	548.1
100-20-10-10	17.97	26,596.0	27,418	479.0
100-20-10-50	17.88	30,954.0	31,847	532.6
100-20-10-100	17.96	35,056.0	36,556	552.6
100-20-20-10	18.10	23,553.0	24,592	641.9
100-20-20-50	18.11	28,594.0	29,869	574.3
100-20-20-100	17.88	36,000.0	37,877	443.3

Table 4.9: Detailed results for the GA on medium-size instances.

4.5.6 Conclusions on the GA

In this Section, the GA is adapted to solve the RVRP considering the min-max criterion. Experimental results show that the proposed GA is close to CPLEX in terms of average gap, while being much faster. In fact 7 proven optima out of 10 are retrieved within 8 seconds. Concerning the results and the number of parameters that could be tested for the GA, other metaheuristics are considered mainly to reduce the number of components and parameters, and make more practical the computational experiments. Nonetheless, there is still room for improving the quality of the solutions in terms of min-max cost and computational time.

4.6 Local search-based metaheuristics

This section describes four metaheuristics combining one of the two greedy heuristics (CW or RCW) and a local search procedure LS: one Greedy Randomized Adaptive Search Procedure (GRASP), one Iterated Local Search (ILS), one Multi-Start ILS (MS-ILS) and another MS-ILS alternating between two search spaces, TSP tours and RVRP solutions. These metaheuristic frameworks were selected for their simple general structure and their reduced number of components and parameters. Moreover, assembling the same components into different metaheuristic structures makes easier an appraisal of the respective contributions of each structure.

4.6.1 Local search procedure

The proposed metaheuristics call the local search LS of Algorithm 4.6. Each main iteration (lines 2–17) seeks the best improving move and stops when no such move exists. The inner loops (lines 4–12) browse each trip T, each node i in T, each trip U and each node j in U. The nodes can be customers or the depot at the beginning of the trips. If $T = U$, the procedure *MovesOnOneTrip* evaluates the moves designed for a single trip, otherwise *MovesOnTwoTrips* tests the moves affecting two distinct trips. The best improving move (if any) is applied to the incumbent solution in line 14.

The pre-computations in line 1 are used to speed up move evaluations. They prepare the same data as in the CW heuristic for each route T, each scenario k and each customer i in the route: the load of the route, $load(T)$, the cost to reach customer i, $b(T, k, i)$, and the cost to return to the depot after i, $a(T, k, i)$. Additionally, the cumulated quantity served up to customer i included is pre-computed, $load(T, i)$. The total complexity of these pre-computations is $O(nq)$. After a move, they need to be updated only for the modified trips (line 15).

Algorithm 4.6 Local search procedure – *LS* (ω)

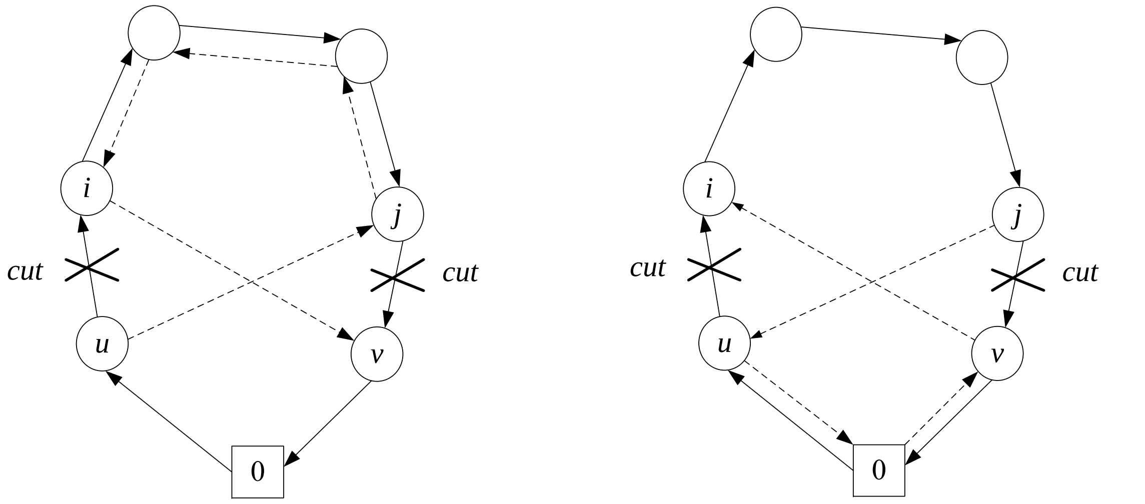
```
do the pre-computations for all trips
repeat
    Initialize the lexicographic vector of the best move: $V_{best} \leftarrow V(\omega)$
    for each trip $T$ of $\omega$ and each node $i$ in $T$ (except the depot at the end) do
        for each trip $U$ of $\omega$ and each node $j$ in $U$ (except the depot at the end) do
            if $U = T$ then
                MovesOnOneTrip $(T, i, j, V_{best}, best\_move)$
            else
                MovesOnTwoTrips $(T, i, U, j, V_{best}, best\_move)$
            end if
        end for
    end for
    if $V_{best} < V(\omega)$ (improving move found) then
        apply best_move to the incumbent solution $\omega$
        update pre-computations for the modified trips (one or two)
    end if
until $V_{best} \geq V(\omega)$ (no improving move is found)
```

MoveOnOneTrip evaluates the following moves on a given trip T and two nodes i, j in T (note that vehicle capacity is always respected):

- relocate node i after node j, if i is a customer,
- relocate i and the next node u (if both are customers) after node j,
- relocate with inversion: same move but i and u are swapped in the reinsertion,
- interchanges: swap two strings with 1 or 2 customers, starting respectively with i and j,
- 2-opt move: reverse the string of customers from i to j included (j must be after i),
- inverted 2-opt: the strings of customers before i and after j are inverted (see Figure 4.6).

The strings in interchanges may have different lengths. Cost variation formulas for relocations and interchanges look like the ones for the CVRP. For instance, if customer i between nodes a and b is relocated between nodes u and v ($u \neq a$), the cost variation for scenario k is computed in $O(1)$ as $c^k_{ab} - c^k_{ai} - c^k_{ib} + c^k_{ui} + c^k_{iv} - c^k_{uv}$. 2-opt moves are more involved because the graph is directed and the cost of a node string changes when reversed. However, the pre-computations allow evaluations in constant time, as can be seen in Figure 4.6. Note that the inverted 2-opt move used here is irrelevant for the CVRP, where arc costs are symmetric.

The moves scanned by *MovesOnTwoTrips* consider two given trips T and U, one node i



2-opt: reversal of subsequence $i \rightarrow j$ in trip T

New trip cost:

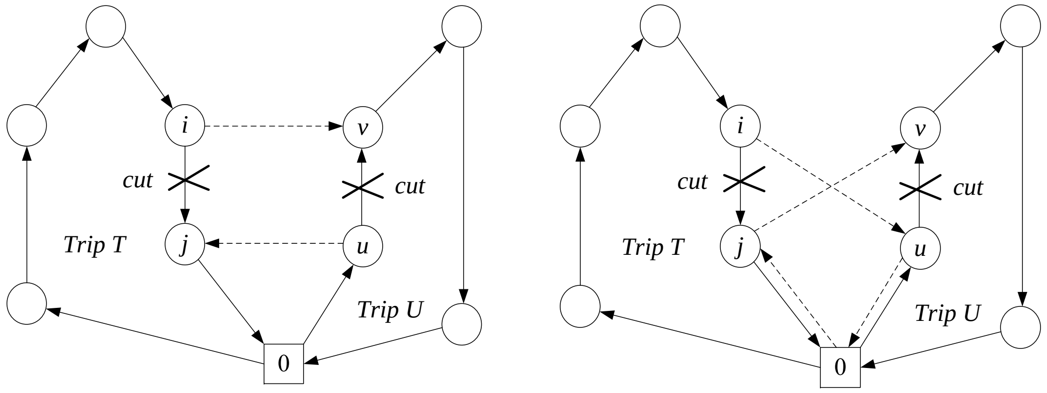
$$b(T,k,u)+c^k_{uj}+b(\overline{T},k,i)$$
$$-b(\overline{T},k,j)+c^k_{iv}+a(T,k,v)$$

Inverted 2-opt: reversal of $0 \rightarrow u$ and $v \rightarrow 0$

New trip cost:

$$b(\overline{T},k,v)+c^k_{vi}+b(T,k,j)$$
$$-b(T,k,i)+c^k_{ju}+a(\overline{T},k,u)$$

Figure 4.6: 2-opt moves for one trip T and scenario k.



2-opt move on two trips – Case A

Feasibility:

$load\ (T,i) + load\ (U) - load\ (U,u) \leq Q$

$load\ (U,u) + load\ (T) - load\ (T,i) \leq Q$

New cost for *T* and *U*:

$$b(T,k,i)\ +c^k_{iv}+b(\overline{U},k,v)$$
$$+\,b(U,k,u)+c^k_{uj}+a(T,k,j)$$

2-opt move on two trips – Case B

Feasibility:

$load(T,i)+load(U,u) \leq Q$

$load(T)-load(T,i)+load(U)-load(U,u) \leq Q$

New cost for *T* and *U*:

$$b(T,k,i)\ +c^k_{iu}+b(U,k,u)$$
$$+\,b(U,k,u)+c^k_{uj}+a(T,k,j)$$

Figure 4.7: 2-opt moves for two trips T and U and scenario k.

in T, and one node j in U. They must check vehicle capacities before computing the cost variations. For instance, $load(T) - d_i + d_j \leq Q$ and $load(U) + d_i - d_j$ must hold if i and j in routes T and U are exchanged. The moves on two trips are similar to the ones on a single trip, except the 2-opt moves depicted in Figure 4.7. No reversal is required for the case on the left, also called 2-opt*. The case on the right involves reversals of two strings of nodes.

The two procedures *MovesOnOneTrip* and *MovesOnTwoTrips* evaluate each move like in Algorithm 4.7. They compute for each scenario k the cost $newcost(k)$ of the solution generated by the move. Then the vector *newcost* is sorted to give a lexicographic vector V_{new}. If this vector improves upon the best cost vector V_{best} found so far, the V_{best} is updated and the associated move, *best_move*, to execute it once all moves have been evaluated.

Each local search iteration browses $O(n^2)$ moves. Thanks to the precomputations, each move can be evaluated in $O(1)$ for each scenario. However, deciding whether a move is improving or not requires a sort of the q scenario costs and a lexicographic comparison, like in the Clarke and Wright heuristic. Hence, a local search doing μ moves runs in $O(\mu n^2 q \log q)$.

Algorithm 4.7 Evaluation of a move in *MovesOneTrip* and *MovesTwoTrips*

```
 1: if the move satisfies capacity constraints then
 2:     for $k \leftarrow 1$ to $q$ do
 3:         compute $newcost(k)$ the solution cost for scenario $k$ if the move were done
 4:         if $newcost(k) > worst(\omega)$ then exit
 5:     end for
 6:     sort $newcost$ in decreasing cost order giving $V_{new}$
 7:     if $V_{new} < V_{best}$ then
 8:         update $V_{best}$ and the kind of move, $best\_move$
 9:     end if
10: end if
```

In line 4 of Algorithm 4.7, if for one scenario the new cost exceeds the worst cost of current solution, the move is non-improving: it can be dropped its evaluation and proceed with the next kind of move. This avoids calling uselessly the sorting algorithm (line 6) and the lexicographic comparison (line 7). During our tests with 10–30 scenarios, this simple trick divides the running times by 15 on average.

4.6.2 Greedy randomized adaptive search procedure

With Simulated Annealing, GRASP is probably the simplest metaheuristic (Feo & Resende (1995)). It consists in sampling the search space using a greedy randomized heuristic and applying local improvement to the obtained solutions. The proposed GRASP for the RVRP performs a fixed number of calls to the local search *LS*, *ncalls*. At the beginning, the CW heuristic described in Section 4.4 and the local search LS are executed to get a provisional best solution ω. This first call to a deterministic heuristic is not a standard feature of GRASP but is justified here by the good results of CW. The other iterations call the randomized Clarke and Wright heuristic RCW, then LS, and update ω if the resulting solution ω' is better. Recall that θ is a percentage defining the degree of randomness in RCW, see Section 4.4.

Algorithm 4.8 Greedy Randomized Adaptive Search Procedure – *GRASP* $(\theta, ncalls, \omega)$

```
initialize the random number generator with a fixed seed for reproducibility
$CW\ (\omega)$
$LS\ (\omega)$
for $calls \leftarrow 2$ to $ncalls$ do
    $RCW\ (\theta, \omega')$
    $LS\ (\omega')$
    if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
end for
```

4.6.3 Two ILS metaheuristics

While GRASP can be viewed as a random sampling of local optima, ILS metaheuristics generate a sequence of local optima with decreasing costs, by applying a perturbation operator and a local search to a copy of the incumbent solution, see Lourenço *et al.* (2010) for a tutorial. Algorithm 4.9 describes a multi-start version (MS-ILS). Its main loop (lines 3-19) launches *niter* successive ILS which return their best solution ω' to update a global best solution ω (line 18). The first ILS starts from the solution computed by the Clarke and Wright heuristic CW, while the next ones are initialized using the randomized version RCW (line 4). The local search *LS* is applied to get the first local optimum (line 5). Then, each ILS performs *ncalls* iterations (lines 7-17). Each iteration takes a copy ω'' of the incumbent solution ω' and applies the perturbation procedure (*Perturb*) and the local search. The search moves to the resulting solution only in case of improvement (lines 11-12).

The perturbation procedure *Perturb* performs a given number (*swaps*) of random exchanges of customers, among the ones respecting vehicle capacities. The perturbation level is adaptive and varies between two bounds *minswaps* and *maxswaps*. At the beginning of each ILS, the variable *swaps* is initialized to *minswaps* (line 6). At each iteration, if the solution generated via perturbation and local search does not outperform the current solution, *swaps* is incremented, but without exceeding *maxswaps*, otherwise it is reset to *minswaps*.

In Section 4.6.5 the MS-ILS and the more classical ILS structure are tested obtained by setting *niter* to 1. The rationale behind MS-ILS is to restart periodically an ILS from diversified solutions instead of losing time in unproductive iterations. MS-ILS is also called GRASP×ILS since it can be viewed as a GRASP where the local search is replaced by an ILS.

Algorithm 4.9 Multi-Start ILS – *MS-ILS* $(\theta, niter, ncalls, minswaps, maxswaps, \omega)$

```
initialize the random number generator with a fixed seed for reproducibility
initialize the components of $V(\omega)$ to $\infty$
for $iter \leftarrow 1$ to $niter$ do
    if $iter = 1$ then $CW\ (\omega')$ else $RCW\ (\theta, \omega')$ endif
    $LS\ (\omega')$
    $swaps \leftarrow minswaps$
    for $calls \leftarrow 2$ to $ncalls$ do
        $\omega'' \leftarrow \omega'$
        $Perturb\ (\omega'', swaps)$
        $LS\ (\omega'')$
        if $V(\omega'') < V(\omega')$ then
            $\omega' \leftarrow \omega''$
            $swaps \leftarrow minswaps$
        else
            $swaps \leftarrow \min(maxswaps, swaps + 1)$
        end if
    end for
    if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
end for
```

4.6.4 Multi-start ILS with giant tours

Beasley (1983) explained how to partition optimally (subject to the sequence) one TSP tour $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$, called *giant tour*, into CVRP routes. The partitioning procedure called *Split* relies on an auxiliary graph $H = (Z, B)$, directed and acyclic. The node-set Z contains one dummy node 0 and one node i per customer. The arc-set B contains one arc

$(i-1,j)$ if the subsequence from τ_i to τ_j (included) can make a feasible trip, i.e., if its total load fits vehicle capacity. This arc is weighted by the trip cost $c_{0,\tau_i} + \sum_{u=1}^{j-1} c_{\tau_u,\tau_{u+1}} + c_{\tau_j,0}$. An optimal splitting of τ corresponds to a minimum-cost path from node 0 to node n in graph H.

Neglected for a long time, this idea has led to efficient memetic algorithms designed for the CVRP (Prins (2004)) and the Capacitated Arc Routing Problem (CARP) (Lacomme *et al.* (2004)). Later Prins *et al.* (2009) showed that even simple constructive heuristics building giant tours and splitting them compete with classical heuristics for the CVRP and the CARP, contrary to a widespread opinion.

In fact, fast splitting procedures can be designed for many vehicle routing problems, although they are sometimes tricky, like for the Generalized VRP studied by Afsar *et al.* (2014). The approach reaches its limits when the routes compete for limited resources, like heterogeneous vehicles: the process becomes an NP-hard resource-constrained shortest path problem. Fortunately, pseudo-polynomial implementations are possible and look fast in practice (Prins (2009)).

The most efficient metaheuristic on 26 VRP variants is currently a hybrid genetic algorithm designed by Vidal *et al.* (2014): although it involves other components which contribute to its efficacy, it is also based on chromosomes encoded as giant tours. The interested reader will find in Prins *et al.* (2014) a survey covering works with algorithms based on giant tours.

Three reasons explain the efficiency of the approach Prins *et al.* (2009): i) the algorithms work on the set of giant tours instead of the much larger set of CVRP solutions; ii) there is no loss of information since each giant tour gives one CVRP solution once splitted and there exists at least one optimal giant tour, i.e., one giving an optimal solution after splitting; and iii) many components of TSP algorithms can be reused, e.g., permutation crossovers.

Neglected for a long time, this idea has been applied in the last decade to powerful metaheuristics for various vehicle routing problems, as shown in a recent survey Prins *et al.* (2014). Then, for the RVRP, the MS-ILS with giant tours (MS-ILS-GT) of Algorithm 4.10 is proposed, which alternates between two search spaces, giant tours and RVRP solutions.

Each iteration of the current ILS works on a pair (ω', τ'), where ω' is an RVRP solution and τ' the giant tour obtained by concatenating its routes (without depot copies) via the *Concat* procedure. A copy τ'' of τ' is perturbed then split to give an RVRP solution ω'', improved by the local search. If ω improves ω', the latter is updated and a new giant tour τ' is deduced using *Concat*, giving a pair (ω', τ') for the next iteration. The perturbation consists in random exchanges of customers, like in ILS and MS-ILS, but applied to giant tours: no capacity check is required.

In this structure, *Split* can be viewed as a large neighborhood operator. Indeed, consider one CVRP solution ω, convert it into a giant tour using *Concat*, then split this tour: the resulting solution is at least as good as ω, since the auxiliary graph still contains the path composed of the arcs modeling the routes of ω and, maybe, better paths. Moreover, *Split* can shift several trip limits, while a move in the local search modifies at most two trips.

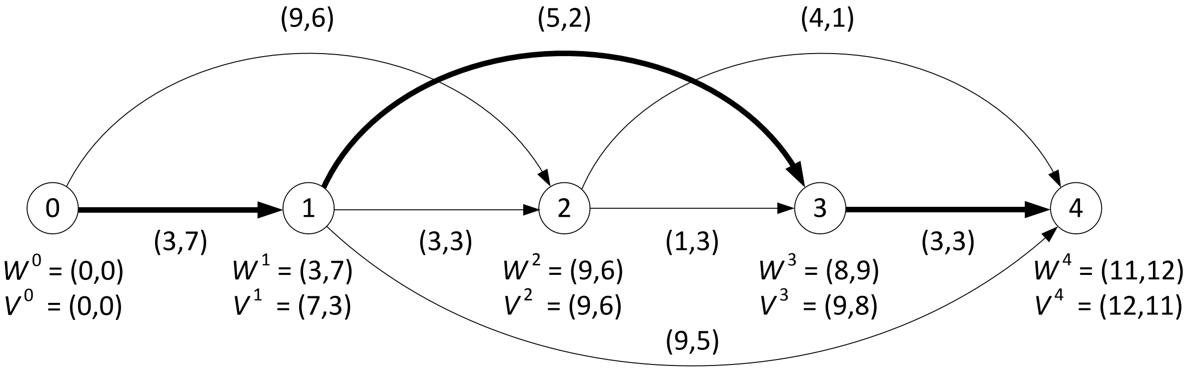


Figure 4.8: Application of the splitting procedure to a giant tour with four customers.

The difficulty here is to design a fast and optimal splitting procedure. The goal is now to split the given giant tour τ to get an RVRP solution ω with minimal lexicographic vector. This can be achieved by the dedicated *Split* procedure of Algorithm 4.11, based on the same auxiliary graph as in the CVRP but computing a shortest path in the lexicographic sense.

Algorithm 4.11 describes a *Split* procedure for the RVRP, based on the same auxiliary graph as in the CVRP but computing a shortest path in the lexicographic sense. The auxiliary graph H is not built explicitly. Each node i has a label (X^i, V^i) composed of two q-vectors. X^i_k is the cost for scenario k of the best path from node 0 to node i. The components of X^0 and V^0 are set to zero (line 1) while those of X^i and V^i, $i \neq 1$, are initialized to a large value (line 2). The two nested loops beginning lines 3 and 5 inspect each feasible route $(0, \tau_i, \tau_{i+1}, \ldots, \tau_j, 0)$ compatible with vehicle capacity and compute its total demand d_{totr} and its cost L_k for each scenario k (lines 6–12). Recall that the subsequence corresponds to arc $(i-1, j)$ in H. A new label $(\overline{X}, \overline{V})$ is built for the path obtained by appending this arc to the best path for node $i-1$. To do this, it is computed $\overline{X} \leftarrow X^{i-1} + L$ (line 14) and sorted $\overline{X}$ in decreasing order of costs to get $\overline{V}$ (line 15). If $\overline{V} < V^j$, a better path to reach node j is necessary to find and the label of j can be updated (lines 17–18). At the end the lexicographic vector of the resulting RVRP solution ω is V^n. The solution itself can be deduced by backtracking on the shortest path, using a simple procedure described for instance in Prins (2004).

Algorithm 4.10 MS-ILS with giant-tours – *MS-ILS-GT* (same header as *MS-ILS*)

```
initialize the random number generator with a fixed seed for reproducibility
initialize the components of $V(\omega)$ to $\infty$
for $iter \leftarrow 1$ to $niter$ do
    if $iter = 1$ then $CW\ (\omega')$ else $RCW\ (\theta, \omega')$ endif
    $LS\ (\omega')$
    $Concat\ (\omega', \tau')$
    $swaps \leftarrow minswaps$
    for $calls \leftarrow 2$ to $ncalls$ do
        $\tau'' \leftarrow \tau'$
        $Perturb\ (\tau'', swaps)$
        $Split\ (\tau'', \omega'')$
        $LS\ (\omega'')$
        if $V(\omega'') < V(\omega')$ then
            $\omega' \leftarrow \omega''$
            $Concat\ (\omega', \tau')$
            $swaps \leftarrow minswaps$
        else
            $swaps \leftarrow \min(maxswaps, swaps + 1)$
        end if
    end for
    if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
end for
```

Figure 4.8 gives an example for a giant tour of 4 customers with demands 3, 6, 2, 1, vehicles with capacity 10, and 2 scenarios. The arcs model feasible trips, e.g., $(0, 3)$ does not exist because a trip with customers $(1, 2, 3)$ would violate vehicle capacity. The values on each arc are the costs for the two scenarios. The computed labels are written under each node. The optimal lexicographic vector $(12, 11)$ corresponds to the shortest path $(0, 1, 3, 4)$ with bold arcs. Hence, the optimal splitting consists in three trips $(0, 1, 0)$, $(0, 2, 3, 0)$ and $(0, 4, 0)$. As *Split* inspects each feasible subsequence $(\tau_i, \tau_{i+1}, \ldots, \tau_j)$, $O(n^2)$ subsequences are tested in the worst case. As $\overline{X}$ have to be sort for each arc, *Split* runs in $O(n^2 q \log q)$. In practice, the algorithm is faster because many subsequences violate vehicle capacity.

Algorithm 4.11 Splitting procedure for one giant tour τ – *Split* (τ, ω)

```
$X^0, V^0 \leftarrow (0, 0, \ldots, 0)$
for $i \leftarrow 1$ to $n$ do $X^i, V^i \leftarrow (\infty, \infty, \ldots, \infty)$ end for
for $i \leftarrow 1$ to $n$ do
    $j \leftarrow i$
    repeat
        if $i = j$ then
            $W \leftarrow f(\tau_i)$
            for $k \leftarrow 1$ to $q$ do $L_k \leftarrow c^k_{0,\tau_i} + c^k_{\tau_i,0}$ end for
        else
            $d_{totr} \leftarrow d_{totr} + f(\tau_j)$
            for $k \leftarrow 1$ to $q$ do $L_k \leftarrow L_k - c^k_{\tau_{j-1},0} + c^k_{\tau_{j-1},\tau_j} + c^k_{\tau_j,0}$ end for
        end if
        if $d_{totr} < Q$ then
            $\overline{X} \leftarrow X^{i-1} + L$
            sort $\overline{X}$ in non-increasing cost order to get $\overline{V}$
            if $\overline{V} < V^j$ then
                $V^j \leftarrow \overline{V}$
                $X^j \leftarrow \overline{X}$
            end if
        end if
    until $(j > n)$ or $(d_{totr} > Q)$
end for
```

4.6.5 Computational evaluation

The local search-based metaheuristics are implemented in the Pascal-like language Delphi and executed on a Dell Precision M6600 portable PC with a 2.2 GHz Intel Core i7, 16 GB of RAM and Windows Professional. The MILP formulation of Section 4.2 is solved in IBM ILOG CPLEX Optimization Studio 12 and is considered to compare the performance of each method. Below the implementation and results for the local search-based metaheuristics are provided.

Implementation and instances The instances are the same described in Section 4.4.5.1 with $n = \{10, 15, 20\}$ customers, $m = \{2, 3\}$ vehicles and $q = \{10, 20, 30\}$ scenarios for the set of the small-size instances. And $n = \{50, 100\}$, $q = \{10, 20\}$, m is up to 20 vehicles, and $\beta = \{10, 50, 100\}$ for medium-size instances.

All randomized algorithms are executed 10 times on each instance. Although they really optimize the lexicographic min-max criterion, our experiments in the two following subsections report only the worst cost, to avoid huge tables of results with detailed cost vectors. As the four metaheuristics include the same local search, two stopping criteria are implemented: a common number of calls to the local search (called "LS budget" and, for the large instances, a fixed running time ("time budget").

Results on small-size instances. As described in the previous sections, for the MILP is given a time limit of 4 hours (14,400 seconds). The randomized Clarke and Wright heuristic RCW performs $ncalls = 50$ iterations with a perturbation factor $\theta = 8\%$ to randomize the savings. The four metaheuristics have the same computing budget of 5,000 local optima, but 10 initial solutions are used in the multi-start versions MS-ILS and MS-ILS-GT. Hence, $ncalls = 5,000$ for GRASP and ILS, while $niter = 10$ and $ncalls = 500$ for MS-ILS and MS-ILS-GT. In the two multi-start metaheuristics, RCW is also called with $\theta = 8\%$ to provide initial solutions. The perturbation level (number of customer exchanges) in the three ILS-based methods varies between $minswaps = 1$ and $maxswaps = 3$. Then, as the CW and the RCW are considered in the proposed metaheuristics results for the two heuristics are also included in the detailed results.

The detailed results are listed for each instance in Table 4.10. For the MIP are given the best lower bound LB, the best solution cost z, the percentage gap $(z/LB - 1) \times 100$ and the running time in seconds CPU. Solutions in bold highlight proven optima while dashes mean the time limit is reached. 10 out of 18 instances are solved to optimality but no instance with $n = 20$ customers and/or $q = 30$ scenarios, except 10–3–30. For the Clarke and Wright heuristic CW, only the solution value z is given because running times are negligible (less than 0.005 s). As the other heuristics are randomized algorithms, the best cost found z_{min}, the average cost z_{avg} and the average time per run in seconds, CPU$_{avg}$ are indicated for 10 runs.

These detailed results are summarized in Table 4.11, using five performance indicators averaged on the 18 instances: the minimum and average percentage gaps to CPLEX lower bound over the 10 runs (these gaps are equal for the MILP and CW, which are tested using a single run), the standard deviation of the 10 gaps (*Std deviation*), the number of CPLEX lower bounds retrieved by each heuristic (*LB hits*) and the average running time per run in seconds.

In here, two variations of the RCW are presented according to $ncalls$. RCW$_1$ corresponds to the solutions listed in Table 4.10 for 50 iterations: the average deviation to LB drops to 7%. In fact, 3 iterations are enough to outperform CW on average. The RCW$_2$ column show the indicators when 5,000 iterations are granted.

As can been seen, the metaheuristics are very close to CPLEX in terms of average gap, while being much faster. All the 10 proven optima obtained by the MILP are retrieved and three upper bounds are even improved (see instances 20–2–10, 20–2–30 and 20–3–30, with costs in boldface in Table 4.10), which explains that the minimum gap is slightly improved for MS-ILS and MS-ILS-GT. Compared to ILS and MS-ILS, MS-ILS-GT lasts 50% more because of its calls to the splitting procedure. GRASP is the slowest metaheuristic, each of its 5,000 calls to RCW needing $O(n^3 q \log q)$ running time.

Summarizing, the tests on small-size instances show that the proposed metaheuristics compete with a direct resolution of the MILP in terms of solution quality and that ILS-based versions slightly outperform GRASP while being 2.5 to 3.5 times faster, using the same number of calls to the local search. However, the results of ILS-based versions are very similar and, compared with MS-ILS, the splitting procedure of MS-ILS-GT brings nothing here. The reason is that the auxiliary graph contains too few feasible paths and only 2 or 3 arcs per path.

Results for medium-size instances. The medium-size instances are out of reach for the MILP. Then, the MILP is excluded from the tests on larger instances. The parameters used are still $ncalls = 50$ for RCW, $ncalls = 5,000$ for ILS, $niter = 10$ and $ncalls = 500$ for MS-ILS and MS-ILS-GT. However, better results using $minswaps = 2$ and $maxswaps = 4$ (instead of 1 and 3) in the perturbations are obtained by the three metaheuristics. The perturbation factor θ randomizing the savings in RCW was 8% for the small-size instances because too many iterations return identical solutions using smaller values. A 1% factor is enough on large instances.

File	MILP				CW	RCW			GRASP			ILS			MS-ILS			MS-ILS-GT		
n-m-q	z_{lb}	z_{ub}	*Gap*	CPU	z	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$
10-2-10	217	**217**	0.00	8.2	263	228	229.9	0.01	**217**	217.0	1.19	**217**	217.0	0.50	**217**	217.0	0.49	**217**	217.0	0.73
10-2-20	284	**284**	0.00	32.2	305	285	294.0	0.01	**284**	284.0	2.00	**284**	284.0	0.91	**284**	284.0	0.91	**284**	284.0	1.16
10-2-30	301	**301**	0.00	35.8	327	310	315.4	0.02	**301**	301.0	2.98	**301**	301.0	1.39	**301**	301.0	1.36	**301**	301.0	1.78
15-2-10	346	**346**	0.00	313.2	412	375	384.6	0.03	**346**	346.0	3.78	**346**	346.1	1.35	**346**	346.0	1.32	**346**	346.0	1.76
15-2-20	373	**373**	0.00	6,560.0	420	398	404.7	0.05	**373**	373.0	6.35	**373**	373.1	2.36	**373**	373.1	2.27	**373**	373.0	3.02
15-2-30	398	404	1.51	–	483	429	439.8	0.07	404	404.2	9.52	404	404.0	3.47	404	404.1	3.36	404	404.0	4.55
20-2-10	419	423	0.95	–	523	465	480.2	0.06	*422	428.1	9.14	*422	424.8	2.89	*422	423.4	2.87	*422	425.3	3.99
20-2-20	460	470	2.17	–	535	525	531.6	0.11	475	481.1	14.93	470	476.0	4.35	470	476.1	4.26	470	476.0	5.83
20-2-30	481	501	4.16	–	580	541	558.5	0.16	501	504.8	22.48	*497	501.1	6.64	*497	500.3	6.70	*497	501.0	9.06
10-3-10	255	**255**	0.00	10.4	304	267	268.8	0.01	**255**	255.0	1.00	**255**	255.0	0.41	**255**	255.0	0.41	**255**	255.0	0.67
10-3-20	316	**316**	0.00	24.7	356	316	324.1	0.01	**316**	316.0	1.78	**316**	316.0	0.72	**316**	316.0	0.71	**316**	316.0	0.96
10-3-30	337	**337**	0.00	38.9	367	342	346.0	0.02	**337**	337.0	2.58	**337**	337.0	1.13	**337**	337.0	1.12	**337**	337.0	1.44
15-3-10	381	**381**	0.00	2,200.0	439	412	429.3	0.02	**381**	384.1	3.35	**381**	384.0	0.79	**381**	381.0	0.80	**381**	381.0	1.48
15-3-20	399	**399**	0.00	6,871.0	454	412	423.3	0.04	**399**	400.6	5.78	**399**	400.2	1.77	**399**	399.4	1.75	**399**	400.0	2.45
15-3-30	426	433	1.64	–	476	455	464.2	0.07	433	433.2	8.72	433	434.6	2.50	433	433.0	2.61	433	433.2	3.77
20-3-10	443	448	1.13	–	562	503	514.5	0.06	448	456.6	8.47	453	456.3	2.05	448	450.2	2.09	448	451.9	3.50
20-3-20	481	497	3.33	–	580	539	555.9	0.11	497	504.0	13.83	501	503.5	3.35	497	502.4	3.36	497	502.6	4.83
20-3-30	508	528	3.94	–	595	568	581.0	0.16	530	533.8	20.94	*523	529.5	5.42	*523	528.3	5.60	*523	529.0	7.56

Table 4.10: Detailed results for the LS-based methods on small-size instances.

Indicator	MILP	CW	RCW$_1$	RCW$_2$	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap *LB* %	1.05	16.56	7.26	3.76	1.12	1.04	0.93	0.93
Avg gap *LB* %	1.05	16.56	9.73	5.27	1.62	1.42	1.22	1.29
Std deviation %	–	–	1.47	0.81	0.30	0.30	0.20	0.20
LB hits	10	0	1	5	10	10	10	10
Time (s)	9350.40	< 0.005	0.06	5.49	7.71	2.33	2.33	3.25

Table 4.11: Indicators for the small-size instances – 10 runs with 5,000 calls to the local search.

The results are detailed in Table 4.12 and summarized in Table 4.13. The two tables have the same format as the ones for small-size instances, except that the reference for the gaps is now the best solution (BS) found for each benchmark problem during the tests.

CW is still interesting for its small running time. The improvement brought by randomization in RCW is less marked than on small instances. The four metaheuristics can now be distinguished. MS-ILS-GT is the best in terms of gaps and number of best solutions found: apparently, using giant tours leads to better solutions on large instances. GRASP, MS-ILS and ILS follow in this order in terms of average gap. The statistical tests (Friedman test and pairwise tests) in the next subsection confirm with a confidence level of 1% that the best and worst methods are MS-ILS-GT and ILS (respectively) but find no statistical difference between GRASP and MS-ILS. The remarks about running times on small instances are still valid: As the number of customers is now larger, the GRASP becomes much slower than the other metaheuristics.

The metaheuristics have been evaluated up to now for 5,000 local optima. As GRASP takes too much time, a natural question is to wonder if this method could achieve the same ranking whether the same running time is allocated to each algorithm. Hence, the running times achieved by MS-ILS-GT are taken in Table 4.12 and the same duration is allocated to GRASP, ILS and MS-ILS. The results are summarized in Table 4.14. Using again the average gaps, the resulting ranking is MS-ILS-GT < MS-ILS < ILS < GRASP, so this time GRASP becomes the least efficient metaheuristic. This strict ordering is also confirmed by the statistical tests, with a confidence level of 1%. Since the four algorithms are assembled from the same basic bricks (randomized heuristic RCW, perturbation procedure, local search), a first conclusion is that ILS-based structures offer a better search efficiency than GRASP for the same execution time.

It can be also seen that ILS, which is the less stable method, can be easily reinforced by doing multiple restarts (MS-ILS). Indeed, restarting periodically the search from a new initial solution is more profitable than prolongating uselessly a single search. Finally, adding

File	BS	CWH		RCWH			ILS			MS-ILS			MS-ILS-GT		
n-m-q-β	z	z	CPU	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$	z_{min}	z_{avg}	CPU$_{avg}$
50-5-10-10	8,633	8,986	0.02	8,708	8,814.9	0.89	8,637	8,637.0	8.77	8,633	8,636.2	9.04	8,633	8,635.0	19.34
50-5-10-50	9,936	10,687	0.02	10,366	10,519.0	0.92	9,936	9,947.9	8.01	9,936	9,948.2	8.95	9,936	9,936.0	21.40
50-5-10-100	11,525	12,544	0.02	11,987	12,268.9	0.95	11,525	11,572.8	9.61	11,525	11,542.6	10.02	11,525	11,525.0	21.83
50-5-20-10	7,966	8,453	0.03	8,218	8,288.9	1.69	7,966	8,089.9	8.79	7,966	7,972.3	9.22	7,966	7,966.0	22.64
50-5-20-50	10,014	10,609	0.03	10,419	10,556.1	1.76	10,014	10,014.0	9.12	10,014	10,015.0	10.24	10,014	10,014.0	24.93
50-5-20-100	10,913	12,099	0.03	11,448	11,607.3	1.73	10,933	10,979.8	9.82	10,913	10,933.5	10.78	10,913	10,923.0	25.97
50-10-10-10	10,882	11,469	0.02	11,194	11,243.0	0.83	10,882	10,885.2	6.86	10,884	10,889.7	7.76	10,882	10,882.0	18.26
50-10-10-50	12,849	13,861	0.02	13,252	13,471.6	0.86	12,849	12,909.9	7.88	12,849	12,859.0	8.30	12,849	12,849.0	19.40
50-10-10-100	16,212	16,852	0.02	16,545	16,821.3	0.85	16,212	16,243.0	7.95	16,212	16,219.8	8.39	16,212	16,212.0	20.04
50-10-20-10	11,576	12,043	0.03	11,901	11,988.9	1.53	11,576	11,679.1	8.27	11,621	11,630.7	8.87	11,576	11,590.6	21.18
50-10-20-50	14,337	15,089	0.03	14,562	14,718.4	1.58	14,346	14,371.0	8.01	14,337	14,349.0	9.07	14,337	14,338.5	21.82
50-10-20-100	15,021	16,775	0.03	15,805	16,067.6	1.62	15,115	15,131.3	10.51	15,036	15,105.7	10.59	15,042	15,076.2	27.81
100-10-10-10	12,767	13,535	0.13	13,328	13,490.1	7.30	12,880	12,966.7	25.30	12,863	12,935.1	28.55	12,767	12,789.7	73.35
100-10-10-50	15,082	16,863	0.13	16,297	16,622.0	7.31	15,104	15,454.1	27.78	15,108	15,345.4	32.21	15,082	15,084.4	81.39
100-10-10-100	18,198	20,973	0.13	20,213	20,621.8	7.39	18,473	18,739.5	29.84	18,358	18,700.3	32.96	18,206	18,280.7	89.97
100-10-20-10	13,614	14,648	0.24	14,220	14,345.4	13.39	13,806	14,004.0	22.79	13,779	13,835.7	31.30	13,614	13,663.1	81.32
100-10-20-50	15,876	16,938	0.26	16,938	16,938.0	14.07	16,204	16,329.9	28.03	16,160	16,273.0	33.39	15,890	15,983.2	88.83
100-10-20-100	18,373	21,890	0.25	20,573	20,786.9	14.05	18,510	19,017.7	32.55	18,703	18,848.6	35.95	18,397	18,457.8	95.79
100-20-10-10	22,234	22,925	0.12	22,925	22,925.0	6.48	22,293	22,433.7	24.78	22,319	22,423.0	28.29	22,252	22,275.1	67.09
100-20-10-50	24,690	26,504	0.12	26,202	26,416.4	6.65	24,895	25,137.0	25.41	24,870	25,077.2	28.32	24,690	24,745.3	72.87
100-20-10-100	29,068	31,711	0.13	31,276	31,597.5	6.92	29,403	29,775.4	27.70	29,251	29,566.1	30.72	29,139	29,243.3	76.33
100-20-20-10	19,942	20,582	0.22	20,526	20,575.6	12.32	19,942	20,008.2	26.43	19,990	20,082.0	32.42	19,942	19,959.4	73.12
100-20-20-50	24,715	27,086	0.23	26,287	26,548.0	12.39	24,972	25,128.2	28.53	24,844	24,992.1	32.77	24,715	24,758.9	78.45
100-20-20-100	29,223	32,546	0.23	31,815	32,186.5	12.45	29,688	29,833.3	29.43	29,602	29,813.4	34.20	29,223	29,467.4	87.48

Table 4.12: Detailed results for the LS-based methods on medium-size instances.

Indicator	CW	RCW	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	7.99	5.02	0.58	0.52	0.45	0.03
Avg gap BS %	7.99	6.22	0.88	1.28	0.93	0.20
Std deviation %	–	0.63	0.19	0.51	0.32	0.13
BS hits	0	0	6	9	9	18
Time (s)	0.10	5.66	593.20	18.63	20.51	51.28

Table 4.13: Indicators for the medium-size instances – 10 runs with 5,000 calls to the local search.

Indicator	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	0.96	0.21	0.41	0.03
Avg gap BS %	1.43	0.93	0.75	0.20
Std deviation %	0.27	0.49	0.21	0.13
BS hits	3	11	10	18
Time (s)	51.3	51.3	51.3	51.3

Table 4.14: Indicators for the medium-size instances – Same duration as MS-ILS-GT.

the cyclic alternation between giant tours and complete RVRP solutions clearly brings an additional improvement. To the best of our knowledge, this is the first time that the contribution of a tour-splitting approach to a metaheuristic structure (here MS-ILS) is quantified.

Statistical tests. The optimality gap of an efficient metaheuristic follows rarely a normal distribution: The algorithm cannot find less than 0% (and this gap can be frequently achieved) then the probability increases rapidly to a maximum and then slowly decreases with a long tail. Conover (1999) recommends non-parametric tests like the Friedman test, which require no assumption about the underlying probability distributions.

Here this test is detailed to compare the $\kappa = 4$ randomized metaheuristics (GRASP, ILS, MS-ILS, MS-ILS-GT) on the test bed of $\varsigma = 24$ large instances, for the average costs of Table 4.12 (5,000 calls to the local search). Let Ψ_{ij} denotes the average cost of 10 runs achieved by algorithm h on instance o. First, compute the ranks Γ_{ho} of the Ψ_{ho}, giving 1 to the best and κ to the worst. Assign average ranks in case of ties, e.g., 2.5 for two heuristics with the same cost and ranks 2 and 3 (See Tables B.1, B.2 on Apendix B). Then calculate the sum of ranks $\mathcal{S}_h = \sum_{h=1}^{\varsigma} \Gamma_{ho}$ for each heuristic o, the mean of these squared sums $\mathcal{G}_2 = \frac{1}{\varsigma} \sum_{o=1}^{\kappa} S_o^2$ and the sum of squared ranks $\mathcal{J}_2 = \sum_{h=1}^{\varsigma} \sum_{o=1}^{\kappa} \Gamma_{ho}^2$. The test statistic is given by Equation

(4.12).

$$\mathcal{T}_2 = \frac{(\varsigma - 1)[\mathcal{G}_2 - \varsigma\kappa(\kappa + 1)^2/4]}{\mathcal{J}_2 - \mathcal{G}_2} \tag{4.12}$$

The null hypothesis (all heuristics have equivalent performances) can be rejected with a confidence level α if $\mathcal{T}_2$ is greater than the $1 - \alpha$ quantile of the $\mathcal{F}$ distribution with $\kappa_1 = \kappa - 1$ and $\kappa_2 = (\kappa - 1)(\varsigma - 1)$ degrees of freedom. The results are given in Table 4.15. As $\mathcal{T}_2 > \mathcal{F}_{0.99,3,69}$ it can be concluded with a confidence level of 1% that the heuristics are not equivalent.

Algorithm	GRASP	ILS	MS-ILS	MS-ILS-GT
Average rank	2.8	3.5	2.7	1.0
Sum of ranks $\mathcal{S}_o$	67.0	84.5	63.5	25.0
Sum of squared ranks	203.5	309.3	177.3	27.0
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$	$\mathcal{F}_{0.99,3,69}$
Value	717.0	678.6	47.1	4.07

Table 4.15: Friedman test for medium-size instances – 5,000 calls to the local search.

Algorithm	GRASP	ILS	MS-ILS	MS-ILS-GT
Average rank	3.9	2.7	2.2	1.2
Sum of ranks $\mathcal{S}_o$	92.5	65.5	53.5	28.5
Sum of squared ranks	359.3	192.3	127.8	36.8
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$	$\mathcal{F}_{0.99,3,69}$
Value	716.0	688.4	73.6	4.07

Table 4.16: Friedman test for medium-size instances – Time Budget

Pairwise tests must be applied to separate the heuristics. Calculate the absolute difference of the sums of ranks of metaheuristics h and o and declare h and o different if this difference exceeds the critical value $\mathcal{CV}$ defined in Equation (4.13), where $t_{1-\frac{\beta}{2}}$ denotes the $1-\frac{\beta}{2}$ quantile of the t distribution with $(\varsigma - 1)(\kappa - 1)$ degrees of freedom. The heuristic with the smallest sum of ranks is the best of the two.

$$|\mathcal{S}_h - \mathcal{S}_o| > \mathcal{CV} = t_{1-\frac{\beta}{2}} \sqrt{\frac{2\varsigma(\mathcal{J}_2 - \mathcal{G}_2)}{(\varsigma - 1)(\kappa - 1)}} \tag{4.13}$$

In this case, the critical value is defined as $\mathcal{CV} = 13.69$ for the 5,000 calls to the local search and the differences in Table 4.17. The significant differences (underlined) indicate that a) MS-ILS-GT dominates the other algorithms, b) ILS is dominated by the others, c) MS-ILS and GRASP are not significantly different.

$\lvert\mathcal{S}_h - \mathcal{S}_o\rvert$	ILS	MS-ILS	MS-ILS-GT
GRASP	17.5	3.5	42.0
ILS		21	59.5
MS-ILS			38.5

Table 4.17: Pairwise test for medium-size instances – 5,000 calls to the local search.

$\lvert\mathcal{S}_h - \mathcal{S}_o\rvert$	ILS	MS-ILS	MS-ILS-GT
GRASP	27.0	39.0	64.0
ILS		12.0	37.0
MS-ILS			25.0

Table 4.18: Pairwise test for medium-size instances – Time Budget.

When the same running time is allocated to all metaheuristics (see Table 4.14) the Friedman test also revealed a statistical difference in the performance of the metaheuristics (see Table 4.16). For this case, the critical value is defined as $\mathcal{CV} = 11.61$. The significant differences (underlined) in Table 4.18 find the following strict ordering, from the best method to the worst: MS-ILS-GT, MS-ILS, ILS, GRASP. MS-ILS-GT is still the best algorithm but, this time, GRASP becomes the worst performer.

Performance on CVRP instances. The algorithms considered in this thesis were not designed for the classical (non-robust) CVRP but their performance can be easily checked by setting the number of scenarios to 1. For this purpose the 14 CMT (Christofides-Mingozzi-Toth) instances (`www.vrp-rep.org`) were selected. They include 7 instances with $n = \{50, 75, 100, 150, 199\}$ and $m = \{5, 7, 8, 10, 12, 17\}$, plus the same instances with an additional maximum route duration. As the RVRP does not consider route duration constraints, the test has been limited to the 7 unrestricted problems (instances 1 to 5, 11 and 12). The results are given in Table 4.19, using the same format as in Table 4.13 (large RVRP instances), the same conditions (5,000 calls to the local search) and the same parameters. The best solutions are here the proven optima recently found by an exact method Pecin *et al.* (2014).

The four metaheuristics still perform well on the CVRP. The best one, MS-ILS-GT, is even able to retrieve 6 out of 7 optima. GRASP displays again the largest gaps and running

Indicator	CW	RCW	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	7.30	4.61	1.24	0.76	0.33	0.10
Avg gap BS %	7.30	5.69	2.45	0.97	1.52	0.35
Std deviation %	–	0.61	0.73	0.13	0.69	0.18
BS hits	0	0	2	3	4	6
Time (s)	0.04	2.29	412.25	30.15	35.17	65.18

Table 4.19: Indicators for 7 CVRP instances – 10 runs with 5,000 calls to the local search.

time. Its longer duration comes from the $O(n^3)$ complexity of the RCW heuristic used to sample the search space. As explained in section 4.4.1, a faster implementation in $O(n^2 \log n)$ is possible for the CVRP, but no changes in the RVRP codes were implemented. Another reason is that each perturbed solution in ILS-based methods is reoptimized in a few moves by the local search, while the independent solutions sampled by RCW in GRASP require more moves on average. Hence, the average time spent per call to the local search is greater in the GRASP.

4.6.6 Conclusions on the local search-based methods

This part of the section has presented a pragmatic approach to compute high-quality solutions for the robust CVRP with uncertain travel costs defined by discrete scenarios. The proposed algorithms are compact and relatively simple since they share the same components and require very few parameters. Their efficiency has been confirmed using small RVRP instances and CVRP instances with known optimal solutions. Moreover, they join the very small family of metaheuristics published for robust vehicle routing problems.

Although multiple scenarios and lexicographic comparisons increase the complexities of initial heuristics and local search moves (compared to the CVRP), the running times remain reasonable. Furthermore, adding an alternation between giant tours and RVRP solutions in MS-ILS brings a statistically significant improvement. The splitting procedure multiplies the running time by 2.5 for a given number of iterations, but as was shown the MS-ILS-GT remains the best even if the same duration is allocated to each algorithm.

Concerning future works, the positive impact of the giant tour approach should be confirmed on other metaheuristics like hybrid genetic algorithms. This task will be probably more difficult because such algorithms involve more components, whose respective contribution must be isolated.

4.6.7 Conclusions

This chapter treated the VRP with uncertain travel times via robust optimization. Uncertainties in travel times are modeled by discrete scenarios. Indeed, most large cities have a traffic control center which measures and records vehicle flows in the streets. The accumulated data can be mined to provide many real scenarios. Here, the RVRP aims at minimizing the total cost of traversed arcs. As far as we know this is the first work that study this problem. For solving the RVRP, a mathematical model is introduced and several greedy heuristics are proposed: the Clarke and Wright (CW), Randomized CW (RCW), Parallel Best Insertion (PBI), Sequential Best Insertion (SBI), Pilot Parallel Best Insertion (Pilot PBI) and Pilot Sequential Best Insertion (Pilot SBI). Besides, a Genetic Algorithm (GA) and different local search-based algorithms are adapted to handle the RVRP. The local search-based methods here developed are an Iterated Local Search (ILS), a Multi-Start ILS (MS-ILS) and a MS-ILS based on Giant Tours (MS-ILS-GT) converted into feasible routes via a lexicographic splitting procedure. The proposed algorithms join the very small family of heuristic and metaheuristics published for robust vehicle routing problems. The good behaviour of all our proposed metaheuristics can be confirmed by comparing its results with the ones of the MILP, which is possible only for small instances (see Table 4.20).

Indicator	MILP	CW	RCW$_1$	RCW$_2$	GA	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap *LB* %	1.05	16.56	7.26	3.76	3.5	1.12	1.04	0.93	0.93
LB hits	10	0	1	5	7	10	10	10	10
Time (s)	9350.40	< 0.005	0.06	5.49	5.71	7.71	2.33	2.33	3.25

Table 4.20: Indicators for the small-size instances.

The experimental results show the RVRP is a hard problem in practice despite of its rather simple statement. Among the proposed constructive heuristics, the CW-based heuristics outperform the others. Regarding our GA and local search-based metaheuristics, the MS-ILS with giant tour has shown to be the most robust method for solving the RVRP.

Conferences and Publications

International journal

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2015. Local Search Based Metaheuristics for the Robust Vehicle Routing Problem with Discrete Scenarios, Applied Soft Computing, 32, 518-531.

Conferences with published proceedings

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. Heuristic approaches for the robust vehicle routing problem. Pages 384-395 of: Combinatorial Optimization, Lecture Notes in Computer Science. Springer International Publishing. Lisbon, Portugal.

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. A robust optimization approach for the vehicle routing problem with uncertain travel cost. Pages 098-103 of: 2014 International Conference on Control, Decision and Information Technologies (CoDIT). Metz, France.

Conferences with abstract proceedings

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. Metaheuristics for the robust vehicle routing problem with discrete scenarios. Working Group on Vehicle Routing and Logistics Optimization (VeRoLog), Oslo, Norway.

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. Heuristics and metaheuristics for the robust vehicle routing problem with uncertain traveling time. Workshop on Optimization under uncertain data (OSI), Belo Horizonte, Brazil.

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. A Genetic Algorithm for the Robust Vehicle Routing Problem with discrete scenarios. International Conference on Metaheuristics and Nature Inspired Computing (META), Marrakech, Morocco.

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2013. An overview on solving robust vehicle routing problem. 14ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), p. 2p, Troyes, France.

Chapter 5

The Bi-Objective Robust Vehicle Routing Problem with Uncertain Demands and Travel Times

5.1 Introduction

The bi-objective Robust Vehicle Routing Problem with uncertainties (bi-RVRP) is studied in this chapter. The uncertainties are handled for both demands and travel times and assumed to be uncorrelated. The demand for each customer is modeled by a discrete set of scenarios representing the deviations from an expected demand. Regarding travel times, traffic records are considered to get discrete scenarios for each arc of the transportation network, like in the RVRP of the previous chapter. The bi-RVRP aims at minimizing the worst total cost of traversed arcs and minimizing the maximum total unmet demand over all scenarios, in the Pareto sense. As can be seen in Section 3.2.2 just a few works have focused on uncertainties in multiple parameters of the VRPs (Lee *et al.* (2012); Ordóñez (2010)) and have seldom used multiobjective optimization techniques (see Table 3.1). This problem may find practical applications in urban transportation, e.g., serving small retail stores which in some cases do not know the exact quantities needed. We assume that the driver discovers the real demand when arriving at each customer. The travel times and demands do not follow any probability distribution, instead representative scenarios for the two uncertain parameters are included. The robustness of feasible solution is affected depending on the demand realizations and the available capacity to serve a client.

To solve the problem, different variations of multiobjective evolutionary metaheuristics are proposed: the Multiobjective Evolutionary Algorithm (MOEA) and the Non-dominated Sorting Genetic Algorithm version 2 (NSGAII). Contrary to the initial versions of these metaheuristics we intensify them using a local search procedure and we evaluate possible locations of this procedure in the structure of each metaheuristic. Different metrics are used to measure the algorithms performance.

This chapter is structured as follows: Section 5.2 presents the formal definition for the bi-RVRP with uncertain demands and travel times. The proposed multiobjective algorithms for the bi-RVRP are introduced in Section 5.3. Section 5.4 is dedicated to a local search procedure and its integration on the multiobjective heuristics while Sections 5.5 and 5.6 describe the computational experiments and the conclusions for the bi-RVRP, respectively.

5.2 Problem definition

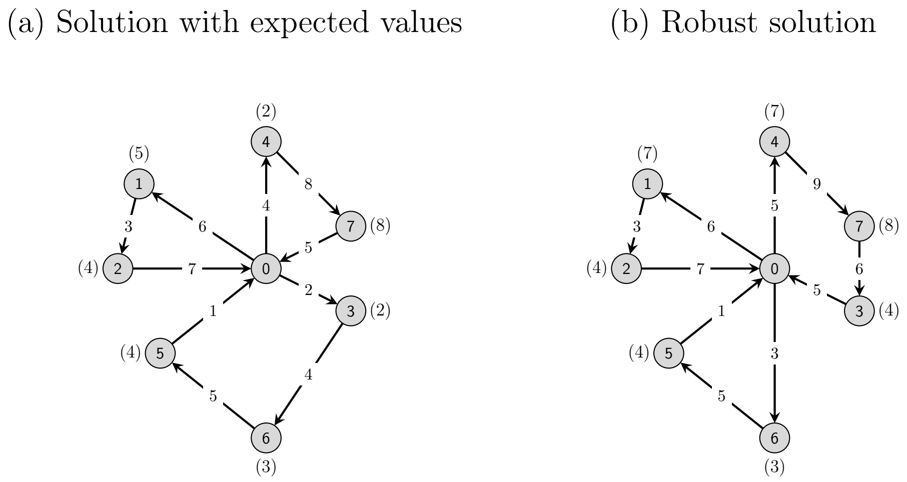
The bi-RVRP can be formally defined on a complete digraph $G = (N, A)$ with a set $N = \{0, 1, 2...n\}$ of $n + 1$ vertices, including the depot (0) and n customers, and a set $A = \{(i, j)|i, j \in N, i \neq j\}$ of arcs. Uncertain data for travel times are modeled as a set of q discrete scenarios $S = \{1, 2, ...q\}$, where each scenario $k \in S$ specifies one cost $c^k_{ij} \in \mathbb{R}$ per each arc $(i, j) \in A$. An expected demand d^*_i is associated with each customer $i \in N$, and its variation is defined as a set of $\hbar$ discrete scenarios $D = \{1, 2, ...\hbar\}$, where each scenario $\ell \in D$ specifies one demand d^ℓ_i for each customer i. Split deliveries are not allowed. A fleet of m identical vehicles is available at the depot, each one with a capacity equal to Q. A solution is a set of vehicle routes starting and ending at the depot, visiting each customer once and respecting vehicle capacities. The bi-RVRP aims at minimizing both the worst total cost of traversed arcs and the maximum total unmet demand, over all scenarios. For a planned route, the following service policy is applied when arriving at a customer i under scenario ℓ: if the total amount already served to previous customers, plus d^ℓ_i, does not exceed vehicle capacity, then the customer is served, otherwise it is not served (i.e., the demand is unmet) and the route proceeds with the next planned customer.

The bi-RVRP is stated in a 2-dimensional space with the worst total cost of traversed arcs z_1 and the worst total unmet demand over all scenarios z_2, defined as follows:

$$z_1 = \min \left(\max_{k \in S} \sum_{(i,j) \in A} c^k_{ij} x_{ij} \right), \quad z_2 = \min \left(\max_{\ell \in D} \sum_{i \in N} d^\ell_i y_i \right)$$

The binary variable x_{ij} is equal to 1 if arc (i, j) is traversed by a vehicle, otherwise $x_{ij} = 0$. y_i is a boolean variable that specifies if a vehicle serves a client $y_i = 1$ or not $y_i = 0$, according to the vehicle capacity Q. The dominance relation for z_1 and z_2 in the Pareto sense is defined for our bi-RVRP as follows, a solution η dominates solution ω if $z_1(\eta) \leq z_1(\omega)$ and

$z_2(\eta) < z_2(\omega)$, or $z_1(\eta) < z_1(\omega)$ and $z_2(\eta) \leq z_2(\omega)$. A solution is non-dominated if no other solution dominates it. The Pareto-optimal front is given by the set of non-dominated robust solutions. The bi-RVRP is illustrated by a simple example in Figure 5.1. The problem considers a single depot (0) and 7 clients to be served by 3 vehicles with capacity equal to 10. Figure 5.1 shows the distances between customers (indicated in the arrows) in normal traffic conditions in Figure 5.1a and with perturbed travel times in Figure 5.1b, together with the demand per client (in brackets).


(a) Solution with expected values

(b) Robust solution

Figure 5.1: Examples of solutions for VRP and bi-RVRP.

Route	Cost (z_1)	Un. Demand (z_2)
0-1-2-0	16	n.a.
0-4-7-0	17	n.a.
0-3-6-5-0	12	n.a.
Total	45	n.a.

Route	Cost (z_1)	Un. Demand (z_2)
0-1-2-0	16	4
0-4-7-3-0	25	12
0-6-5-0	9	0
Total	50	16

Table 5.1: Summary of results for the VRP (left) and the bi-RVRP (right).

The summary of results can be found in Table 5.1. The set of routes for a solution constructed in normal conditions uses the expected value for demands and travel times while dealing with uncertainties, the perturbed scenarios for travel times and demands are handled and the total cost z_1 and unmet demand z_2 are calculated. In Figure 5.1b, one perturbed scenario is considered. The solution is composed of three routes with sequences 0-1-2-0, 0-4-7-3-0, 0-6-5-0, respectively. Then, over the perturbed demands for route 0-1-2-0 the vehicle cannot satisfy demand for client 2 dealing with 4 units of unmet demand and a total cost of 16. Route 0-4-7-3-0 can satisfy only the demand of client 4, leaving with

12 units of unmet demand and a total cost of 25. Finally, for sequence 0-6-5-0 all clients are served and the total cost for the route is 9. The same evaluation is done for the VRP (Figure 5.1a), but in this case a total cost of 45 is found and all clients are attended. For each solution we totalize the cost and the unmet demand obtaining a total cost of 45 and 0 unmet demand in the VRP, and a total cost of 50 with 16 units of the unmet demand in the RVRP.

5.3 Multiobjective Algorithms for the bi-RVRP

Many multiobjective genetic algorithms (MOGAs) are available in the literature. Thus, the selection of the best multiobjective algorithm is not evident for solving our bi-RVRP. Here, we consider the standard version of MOEA and NSGAII, mainly for their flexible structure and successful applications to multiobjective combinatorial optimization problems (Jozefowiez *et al.* (2009); Santos *et al.* (2013); Lacomme *et al.* (2006)). The following subsections describe in detail the shared components and the adaptation of each method.

5.3.1 Shared components of our multiobjective algorithms

5.3.1.1 Chromosome and Evaluation

The chromosome encoding designed for the RVRP is reused: each solution for the bi-RVRP is encoded as a set of routes, in which the customers appear in the order they are visited (See Table 5.2). Delimiters are used to separate a route (grey color). Of course, the evaluation of the chromosomes is now different to handle the two objectives. The ranking procedure for the bi-RVRP initially evaluates for each chromosome its cost and its unmet demand over all scenarios. Let $cost(\omega, k)$ be the cost of solution ω for scenario k and $unmet(\omega, \ell)$ the unmet demand of solution ω for scenario ℓ, where k stands for the index scenario on travel times and ℓ the index scenario for the demands. Then, the maximum total cost, $\max\{cost(\omega, k) \mid k \in S\}$ and the maximum unmet demand, $\max\{unmet(\omega, \ell) \mid \ell \in D\}$ are computed and stored with the chromosome. Considering these computed values for each individual, a second evaluation is performed to classify all individuals in the population, following a minimization.

0	1	2	0	4	7	3	0	6	5	0

Table 5.2: Example of chromosome.

5.3.1.2 Initial Population

The generation of the initial population *Pop* composed of g chromosomes is based on a random generation of solutions and a best insertion heuristic. The first $g/2$ individuals (rounded up) to enter *Pop* are built using a best insertion heuristic and the remaining individuals are randomly generated. In the heuristic, a set of routes is created by selecting unserviced customers and inserting them in one of the partial routes already created. The heuristic starts by selecting m seed customer (one per vehicle) to obtain m initial routes. Then, unserviced clients are sorted in decreasing order of the expected demands and they are inserted one by one in this order, in the current route, considering the smallest impact on cost and respecting vehicles capacity. As Split deliveries are not allowed the insertion heuristic can fail to use only m vehicles. Thus, extra vehicles can be added to the solution in order to ensure that each customer is visited by a vehicle. This violation on the number of vehicles is not repaired, and is left to be managed by our multiobjective algorithms. For the bi-RVRP, a random scenario k is selected to evaluate the best insertion according to travel time. In the solutions randomly generated, unserviced customers are assigned sequentially to a first route in a random order. When no other customer can be added, a second route is created, and so on. The procedure stops when all customers are attended by one vehicle. As a result for the random generation and the best insertion heuristic, solutions with zero units of unmet demand are obtained.

5.3.1.3 Crossover Operator

The crossover from Section 4.5.3 is implemented in the proposed multiobjective genetic algorithms. Initially selection of parents P_1 and P_2 are done following the binary tournament selection. For that, two randomly selected solutions are compared and the solution with the smallest rank is kept as P_1. The process is repeated to get P_2. We recall that the main idea of the crossover operator is to select a sequence of customers from P_1 and to insert it to the best insertion in P_2. Multiple visits to customers are removed from the offspring ω to ensure that each customer i is visited once.

5.3.2 MOEA and NSGAII for the bi-RVRP

The standard MOEA and NSGAII have been adapted to solve the bi-RVRP. For the sake of clarity, the standard MOEA developed here relies on our evolutionary genetic algorithm from Section 4.5 using a ranking operator to classify solutions in fronts at each iteration of the algorithm. Half of the initial population for MOEA is randomly generated and half is created by means of the best insertion heuristic, as already mentioned. Solutions are then classified using the ranking operator. The population is evolved using the binary tournament selection and the crossover operator. NSGAII is a more recent multiobjective method, which is detailed below.

5.3.2.1 NSGAII

The general procedure of the NSGAII from Deb *et al.* (2002) is based on a fast non-dominated sorting and a crowding distance assignment. The non-dominated sorting determines the rank value for each solution in the population, whereas the crowding distance measures the density for each solution in the neighborhood. In the sorting, all solutions are compared with each other according to the Pareto dominance. Hence, solutions are classified into fronts according to their non-domination level, beginning by the non-dominated set which constitutes level 1 or front 1. Then, solutions that belong to the non-dominated front 2 are distinguished, and so on, until the partition of the population is completed (see Figure 5.2).

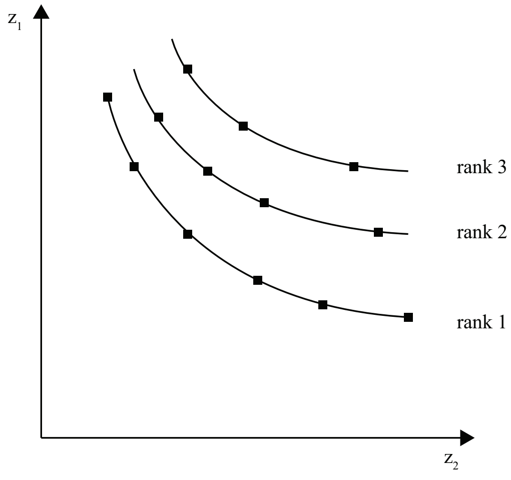


Figure 5.2: Example of non-dominated sorting.

Concerning the crowding distance, solutions from each front are sorted according to each objective component in ascending order. For each objective, the first and last solutions obtain an infinite distance value (∞). The rest of solutions receive a value equal to the absolute difference of the objective function between two adjacent individuals. This process is repeated for each objective function. Then, the crowding distance is equivalent to the sum of the solution distance corresponding to each objective function. Solutions with a higher crowding distance are preferred.

The general structure of the NSGAII is sketched in Algorithm 5.1. At each iteration of the algorithm, g initial solutions are generated. The solutions are sorted and set in successive fronts of non-dominated solutions. Thus, the ranking (line 3) and the crowding distance (line

4) per each solution is computed. Considering our two objectives for the bi-RVRP and a front Π of with a set of solutions, the crowding distance is computed as follows:

$$Crowding(\Pi(\psi)) = \frac{z_1(\Pi(\psi)+1) - z_1(\Pi(\psi)-1)}{z_1^{max} - z_1^{min}} + \frac{z_2(\Pi(\psi)-1) - z_2(\Pi(\psi)+1)}{z_2^{max} - z_2^{min}} \tag{5.1}$$

Where $\Pi(\psi)+1$ and $\Pi(\psi)-1$ are respectively the successor and the predecessor values for the ψth solution in the sorted front. For the extreme points in all fronts, the *crowding*$(\Pi(\psi))$ $= \infty$. The goal is to favor the extreme points on the selection process.

In order to generate a new solution, two parents P_1 and P_2 are selected considering the binary tournament selection (line 8). For that, two random solutions are compared in the Pareto sense ($z_1(\eta) \leq z_1(\omega)$ and $z_2(\eta) < z_2(\omega)$, or $z_1(\eta) < z_1(\omega)$ and $z_2(\eta) \leq z_2(\omega)$). Then, solution with the smallest rank is kept as P_1. The process is repeated to select P_2. If two solutions belong to the same front, the NSGAII selects the one with the highest crowding distance.

Then, a crossover operator is applied to obtain an offspring ω (line 9). The new offspring ω is added to the population if there is not a solution with the same values of the two objective functions (line 10). At the end of each iteration, the population contains $2g$ solutions. After a non-dominated ranking on the $2g$ solutions, only the best g solutions are kept for the next iteration (line 15). The procedure is repeated until a maximum number of iterations is satisfied.

5.4 Local Search

The general definition of the standard MOEA and the NSGAII do not include local search procedures. Nevertheless, a local search procedure is added to improve the results and to accelerate the convergence. The cost and the unmet demand of a solution ω are computed for each scenario if the move were performed. A move is accepted according to different criteria explained in the sequel and a first improvement strategy is applied. The whole local search stops when all neighborhoods are explored without finding any improvement. *Relocation*, *Interchanges*, and *2-opt* moves are developed for the bi-RVRP.

-*Relocation* moves one or two contiguous customers to a different position.

-*Interchange* exchanges two chains which may have 1 or 2 customers each. The lengths of the two swapped chains can be different.

-*2-opt* inverts the sequence of customers from i to j (j must be after i). A modification over *2-opt* intra-route is also applied to consider the asymmetric costs of the transportation

Algorithm 5.1 Non-dominated sorting genetic algorithm for the bi-RVRP

```
Require $G = (N, A), m, Q, S, d_i^* \ \forall i \in N, D$
  Initialize $Pop$ with $g$
  $Pop$ ←Non-Dominated Ranking($Pop$)
  Get crowding distance($Pop$)
repeat
    //Generation of $g$ new solutions
    while $| Pop |< 2g$ do
        Choose $P_1$ and $P_2$
        $\omega \leftarrow$ Crossover($P_1$, $P_2$)
        Add $\omega$ to $Pop$ if $\omega$ is not a clone
    end while
    //Create a new population
    $Pop$ ←Non-Dominated Ranking($Pop$)
    Get crowding distance($Pop$)
    Reduce $Pop$ to its best $g$ solutions
until stopping condition
```

network. Thus, a subsequence i to j is inverted and the cost for the inverted subsequence j to i is computed before reinserting in T.

5.4.1 Acceptance criteria for a move

Our local search procedure involves one of the following criteria to decide if a move $\omega \rightarrow \omega'$ improves the incumbent solution.

LS1: Pareto dominance. Solution ω' dominates solution ω if ($z_1(\omega') \leq z_1(\omega)$ and $z_2(\omega') < z_2(\omega)$, or $z_1(\omega') < z_1(\omega)$ and $z_2(\omega') \leq z_2(\omega)$).

LS2: Convex combination. A move to be accepted must improve the function $weight \cdot [z_1(\omega') - z_1(\omega)] + (1 - weight) \cdot [z_2(\omega') - z_2(\omega)]$ with $weight \in [0, 1]$. Then, the pseudo-weight $weight$ must be computed for a given solution ω. For that, we consider the technique proposed by Murata *et al.* (2003), where $weight$ depends on the position of ω in its front. The estimation of the $weight$ value is showed in Equation 5.2 in which z_1^{min}, z_1^{max}, z_2^{min} and z_2^{max} are respectively the minimum and maximum values of each criterion.

$$weight = \frac{\frac{z_1(\omega) - z_1^{min}}{z_1^{max} - z_1^{min}}}{\frac{z_1(\omega) - z_1^{min}}{z_1^{max} - z_1^{min}} + \frac{z_2(\omega) - z_2^{min}}{z_2^{max} - z_2^{min}}} \tag{5.2}$$

As can be seen in Figure 5.3, this formula yields diverging directions which preserve the spacing of solutions.

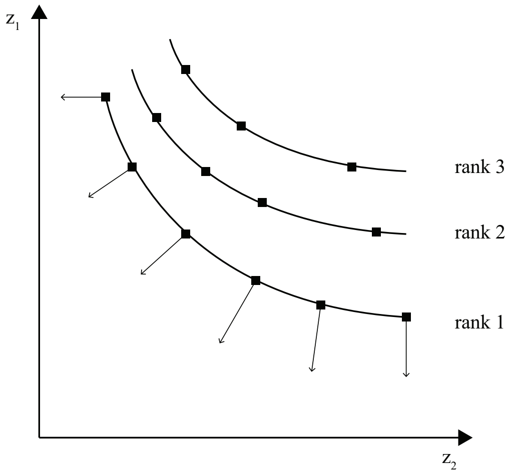


Figure 5.3: Example of the variable descent directions from Murata.

5.4.2 Integrating the local search in our multiobjective heuristics

The local search can be introduced at different locations in the multiobjective evolutionary metaheuristics. Table 5.3 summarizes the different versions tested. LS1 is applied on new solutions (produced by crossover) with a given probability φ. LS2 with Murata's technique is applied to each solution in the first front. MOEA(1) and NSGAII(1) are the reference versions, without local search. The versions number 1 call LS1 only, the ones with number 2 use LS2 only, while the versions number 4 combine both procedure.

Method	Description	Acceptance criteria
MOEA(1)	without local search	n.a.
MOEA(2)	local search on new solutions	LS1
MOEA(3)	local search on first front	LS2
MOEA(4)	local search on new solutions+local search on first front	LS1+LS2
NSGAII(1)	without local search	n.a.
NSGAII(2)	local search on new solutions	LS1
NSGAII(3)	local search on first front	LS2
NSGAII(4)	local search on new solutions+local search on first front	LS1+LS2

Table 5.3: Versions for the evolutionary metaheuristics.

5.5 Computational experiments

The computational experiments were performed on a HP ZBook, Intel Core i7-4800MQ, 2.7 GHz with 16GB of RAM. The proposed metaheuristic was developed in C++ using Visual Studio Professional 2013. The following subsections describe the instances used and the results.

5.5.1 Implementation and instances

The computational experiments are based on two sets of random-instances, called set 1 and set 2, in order to see the impact that have the application of different representations of uncertain demands on the bi-RVRP. Set 1 specifies a perturbation over the total demand of clients d_{tot}, while set 2 considers for each client an individual variation defined in an interval. The two sets contain 24 instances with $n = \{30, 40, 50\}$, $m = \{3, 4, 5\}$, $q = \{20, 30\}$ and $\hbar = \{3, 5\}$ and they are generated as follows. The customers are first randomly placed according to an uniform distribution in a main square, i.e., the coordinates x_i and y_i of each customer i are randomly selected in $[0, 1000]$. We ensure that $\mid x_i - x_j \mid\geq 1$ and $\mid y_i - y_j \mid\geq 1$ to avoid customers with the same location. All node coordinates are recorded with the instance, as real numbers. Concerning the depot node 0, the main square is divided into 3×3 smaller squares and this node is randomly placed in the central square, i.e., its coordinates x_0 and y_0 are drawn in the interval $[333.33, 666.67]$. For each scenario on travel time k and each arc (i, j), the arc cost is randomly drawn in $[dist_{ij}, \beta \times dist_{ij}]$, where $dist_{ij}$ denotes the Euclidean distance and $\beta = \{10, 50, 100\}$ is a maximum deviation factor to this baseline distance. The arc costs are rounded up to the closest integer.

The nominal or ideal values of the demands are generated assuming a vehicle capacity Q. The average demand per customer is first computed as $\bar{d}^* = Q/m$ and then the demand d^*_i of each customer i is randomly drawn as an integer in $[1, 2\bar{d}^*]$. These demands are finally adjusted to get a total demand d_{tot} such that $d_{tot} \cong 90\% \times mQ$, i.e., 90% of the total capacity available. To do so, a customer is randomly selected and its demand is diminished (if $d_{tot} > 90\% \times mQ$) or augmented (if $d_{tot} < 90\% \times mQ$) by a random integer between 1 and 5 (included). This process is repeated until 10% of spare capacity is obtained to avoid infeasible instances. The file names recall the values n–m–q–β–$\hbar$.

In particular, for set 1, discrete scenarios on demands are generated according to a global perturbation ϱ of the total demand d_{tot} in percent, where $\varrho = \{5\%, 25\%\}$. Thus, individual variation for demand on each client i is randomly selected and limited by a global perturbation defined as $\sum\limits_{i\in N} d_i = \varrho\ d_{tot}$, where $d_{tot} = \sum\limits_{i\in N} d^*_i$.

For set 2 uncertainties on demands are defined between a lower and an upper values

$[d_i^-, d_i^+]$, respectively, per each customer i. Here, the interval variation is fixed according to a range of [-50%,100%] around d_i^* and the number of clients allowed to change from their expected value of the demand is limited by a percentage $\epsilon = 50\%$.

5.5.2 Evaluation criteria

The performance of the proposed metaheuristics is measured through four metrics, the number of solutions in the Pareto front (Θ), the distribution of solutions in the Pareto front called spacing (Y_1), the hypervolume (Y_2) and the running time (CPU) in seconds. The spacing proposed by Schott (1995) estimates the relative distance between two consecutive solutions obtained in the non-dominated set, as follows:

$$Y_1 = \sqrt{\frac{1}{|\Theta|}\sum_{i=1}^{|\Theta|}(\gamma_i - \bar{\gamma})^2} \tag{5.3}$$

Where γ_i minimizes the sum of the absolute difference in the objective function values between the $i - th$ solution and any other solution in the non-dominated set, while $\bar{\gamma}$ is the mean value of the above distance measure. When solutions are almost uniformly spaced, the corresponding distance measure is small. Thus, a heuristic having a smaller spacing (Y_1) is better.

The hypervolume introduced by Zitzler *et al.* (2003) calculates the volume covered by members of Θ. Mathematically, for each solution $\omega \in \Theta$, a hypercube vol_i is constructed with a reference point. The reference point is defined by the maximal value of z_1 and z_2 in the initial population. Thereafter, a union of all hypercubes is found and its hypervolume (Y_2) is calculated. An heuristic with a large value of Y_2 is desirable.

$$Y_2 = volume\left(\bigcup_{i=1}^{|\Theta|} vol_i\right) \tag{5.4}$$

The spacing and the hypervolume can be affected by the magnitude order of the objectives, thus it is recommended to compute them using normalized objective functions in the interval [0, 1].

5.5.3 Parameter tuning

For the set of instances, calibration with different configuration of parameters was used to see which one produces the best results of the metaheuristics. To select the parameters, a fine-tuning procedure is used, beginning from a promising configuration using 100 and 200 iterations. The population size was defined according to the number of customers in each instance (Santos *et al.* (2013)). The rate of local search is defined in $\varphi = 10\%$ when the non-systematic local search (LS1) is applied. The results are provided in Tables 5.4-5.9.

	Instance	$g = \|N\|$				$g = 2\|N\|$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	2	0.000	1.000	41	3	0.592	0.996	81
	30-3-30-100-5	5	0.213	0.988	44	2	0.000	0.994	89
	40-4-20-50-3	2	0.000	0.994	83	8	0.158	0.987	166
	40-4-30-100-5	11	0.088	0.975	94	5	0.100	0.996	184
	50-5-20-50-3	10	0.082	0.990	158	6	0.187	0.996	318
	50-5-30-100-5	6	0.187	0.993	172	9	0.092	0.969	343
$niter = 200$	30-3-20-50-3	2	0.000	1.000	77	3	0.592	0.996	151
	30-3-30-100-5	2	0.000	0.998	81	2	0.000	0.994	168
	40-4-20-50-3	2	0.000	0.994	160	7	0.200	0.982	317
	40-4-30-100-5	9	0.100	0.977	180	5	0.096	0.996	353
	50-5-20-50-3	7	0.292	0.996	305	3	0.113	0.997	607
	50-5-30-100-5	8	0.117	0.994	330	8	0.118	0.977	629

Table 5.4: Results of NSGAII(1) with 100 and 200 iterations $\varrho = 5\%$.

Preliminary results showed that in general the best compromise between solution quality and running time is obtained with $g = |N|$ and 200 iterations achieving lower values for Y_1 and prevailing in average high values for Θ and Y_2. According to these results, the same configuration of parameters has been considered on the different variations of NSGAII and MOEA in order to have a fair comparison between the proposed methods. Figures 5.4, 5.5 and 5.6 present an example of convergence for NSGAII(1) on three based-instances, instance 30-3-30-100-3 with $\varrho = 5\%$, instance 50-4-30-50-5 with $\varrho = 25\%$ for the set 1 and instance 50-5-20-100-5 with $\epsilon = 50\%$ for the set 2, showing best Pareto front after 50, 100, 200 iterations for a population size of $g = |N|$.

5.5.4 Results on instances

The detailed results for each set of instances are listed in Tables 5.12-5.17. Each line on the detailed results corresponds to an instance then, metrics Θ, Y_1, Y_2 and CPU are shown for MOEA and NSGAII. These detailed results are summarized in Tables 5.10 and 5.11, using average values of the metrics defined by headers Θ_{avg}, Y_{1avg}, Y_{2avg} and CPUavg.

	Instance	$g = \lvert N \rvert$				$g = 2\lvert N \rvert$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	2	0.000	1.000	42	14	0.048	0.977	83
	30-3-30-100-5	6	0.237	0.983	45	8	0.104	0.989	90
	40-4-20-50-3	10	0.087	0.985	88	7	0.346	0.993	172
	40-4-30-100-5	16	0.041	0.942	96	10	0.057	0.983	189
	50-5-20-50-3	12	0.077	0.968	164	11	0.078	0.967	323
	50-5-30-100-5	9	0.043	0.984	174	12	0.116	0.983	336
$niter = 200$	30-3-20-50-3	2	0.000	1.000	76	16	0.035	0.978	159
	30-3-30-100-5	7	0.290	0.984	79	8	0.110	0.989	173
	40-4-20-50-3	10	0.100	0.987	160	10	0.153	0.988	338
	40-4-30-100-5	12	0.057	0.953	173	4	0.155	0.988	366
	50-5-20-50-3	11	0.068	0.980	310	17	0.045	0.977	638
	50-5-30-100-5	8	0.150	0.986	330	11	0.073	0.982	655

Table 5.5: Results of NSGAII(1) with 100 and 200 iterations $\varrho = 25\%$.

	Instance	$g = \lvert N \rvert$				$g = 2\lvert N \rvert$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	15	0.055	0.955	38	4	0.278	0.975	80
	30-3-30-100-5	4	0.396	0.973	44	8	0.179	0.996	86
	40-4-20-50-3	7	0.107	0.993	83	5	0.234	0.997	173
	40-4-30-100-5	12	0.079	0.983	89	4	0.168	0.996	194
	50-5-20-50-3	9	0.081	0.967	160	15	0.071	0.978	330
	50-5-30-100-5	7	0.063	0.981	176	11	0.129	0.983	344
$niter = 200$	30-3-20-50-3	11	0.102	0.968	75	4	0.294	0.976	147
	30-3-30-100-5	6	0.206	0.974	84	7	0.208	0.996	159
	40-4-20-50-3	8	0.116	0.987	161	9	0.230	0.993	322
	40-4-30-100-5	9	0.126	0.990	176	8	0.186	0.991	352
	50-5-20-50-3	8	0.118	0.977	317	10	0.110	0.988	602
	50-5-30-100-5	9	0.078	0.981	332	14	0.050	0.980	665

Table 5.6: Results of NSGAII(1) with 100 and 200 iterations $\epsilon = 50\%$.

	Instance	$g = \lvert N \rvert$				$g = 2\lvert N \rvert$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	1	0.000	1.000	42	1	0.000	1.000	82
	30-3-30-100-5	2	0.000	0.971	47	1	0.000	1.000	87
	40-4-20-50-3	3	0.278	0.970	87	2	0.000	0.997	168
	40-4-30-100-5	3	0.080	0.986	95	3	0.075	0.993	180
	50-5-20-50-3	4	0.192	0.998	164	3	0.371	0.999	326
	50-5-30-100-5	1	0.000	1.000	178	2	0.000	1.000	343
$niter = 200$	30-3-20-50-3	1	0.000	1.000	75	1	0.000	1.000	150
	30-3-30-100-5	2	0.000	0.971	80	1	0.000	1.000	163
	40-4-20-50-3	3	0.003	0.973	157	2	0.000	0.995	316
	40-4-30-100-5	3	0.387	0.999	173	2	0.000	1.000	343
	50-5-20-50-3	3	0.173	0.997	299	3	0.184	0.998	593
	50-5-30-100-5	1	0.000	1.000	331	2	0.000	1.000	654

Table 5.7: Results of MOEA(1) with 100 and 200 iterations $\varrho = 5\%$.

	Instance	$g = \lvert N \rvert$				$g = 2\lvert N \rvert$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	5	0.264	0.992	39	5	0.207	0.999	82
	30-3-30-100-5	1	0.000	1.000	44	6	0.136	0.993	86
	40-4-20-50-3	6	0.112	0.991	83	4	0.113	0.996	174
	40-4-30-100-5	8	0.166	0.986	96	9	0.089	0.990	190
	50-5-20-50-3	9	0.087	0.969	159	5	0.180	0.997	317
	50-5-30-100-5	5	0.243	0.992	172	9	0.062	0.990	347
$niter = 200$	30-3-20-50-3	2	0.000	0.995	77	6	0.330	0.997	159
	30-3-30-100-5	2	0.000	0.998	84	6	0.128	0.994	164
	40-4-20-50-3	8	0.107	0.991	170	6	0.227	0.998	340
	40-4-30-100-5	9	0.098	0.972	183	8	0.116	0.990	364
	50-5-20-50-3	9	0.079	0.973	313	10	0.128	0.995	616
	50-5-30-100-5	10	0.110	0.989	337	14	0.063	0.990	655

Table 5.8: Results of MOEA(1) with 100 and 200 iterations $\varrho = 25\%$.

	Instance	$g = \|N\|$				$g = 2\|N\|$			
	n–m–p–β–$\hbar$	Θ	Y_1	Y_2	CPU	Θ	Y_1	Y_2	CPU
$niter = 100$	30-3-20-50-3	4	0.089	0.993	41	7	0.148	0.992	81
	30-3-30-100-5	5	0.208	0.999	43	3	0.142	0.999	88
	40-4-20-50-3	10	0.048	0.987	84	2	0.000	0.998	171
	40-4-30-100-5	8	0.132	0.980	94	10	0.135	0.987	184
	50-5-20-50-3	15	0.118	0.982	156	16	0.051	0.964	319
	50-5-30-100-5	13	0.119	0.984	179	7	0.273	0.999	354
$niter = 200$	30-3-20-50-3	4	0.101	0.993	76	6	0.152	0.992	152
	30-3-30-100-5	5	0.208	0.999	81	3	0.142	0.999	165
	40-4-20-50-3	12	0.066	0.987	157	2	0.000	1.000	337
	40-4-30-100-5	7	0.189	0.984	176	14	0.108	0.988	359
	50-5-20-50-3	9	0.047	0.984	300	12	0.091	0.973	636
	50-5-30-100-5	11	0.121	0.981	341	13	0.156	0.997	714

Table 5.9: Results of MOEA(1) with 100 and 200 iterations $\epsilon = 50\%$.

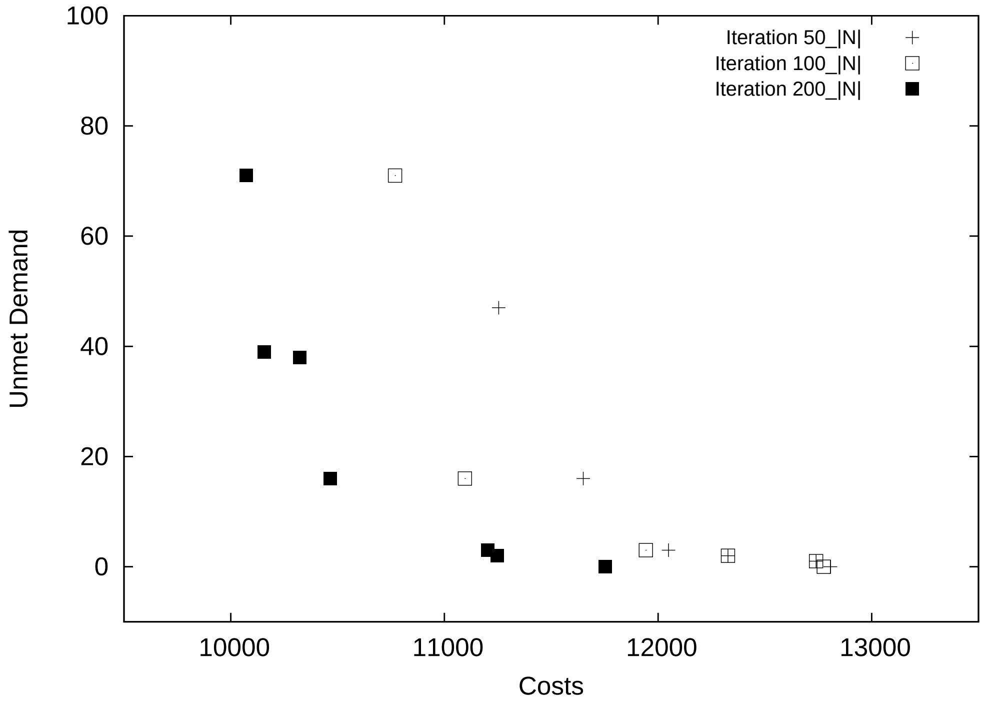


Figure 5.4: Example of convergence on NSGAII(1) for instance 30-3-30-100-3, $\varrho = 5\%$ on set 1.

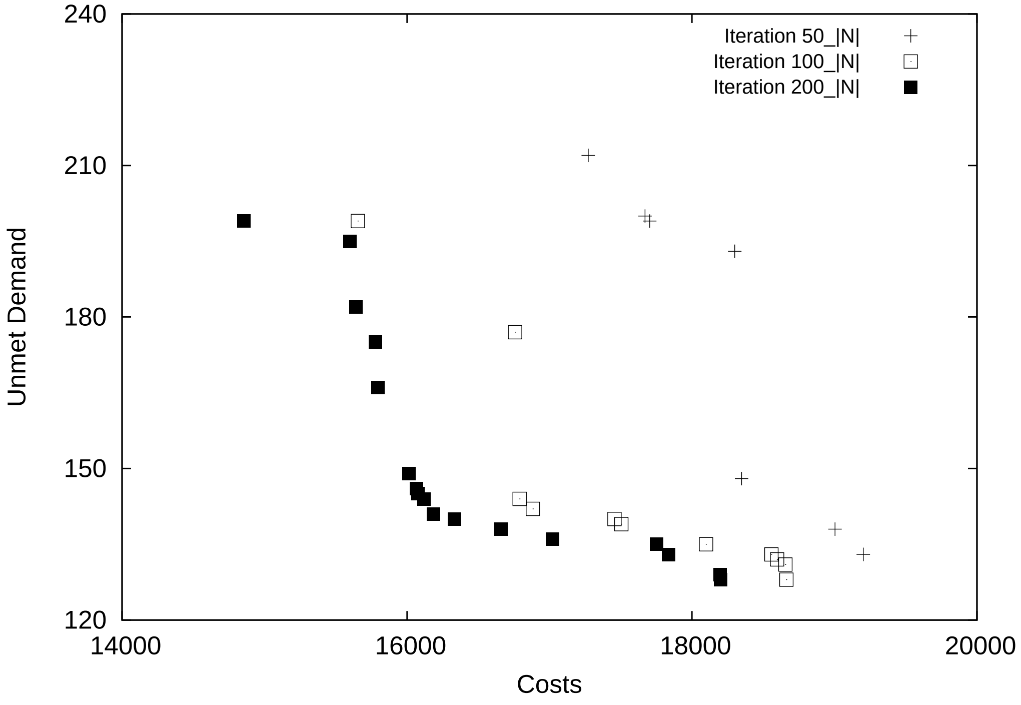


Figure 5.5: Example of convergence on NSGAII(1) for instance 50-5-20-100-5, $\varrho = 25\%$ on set 1.

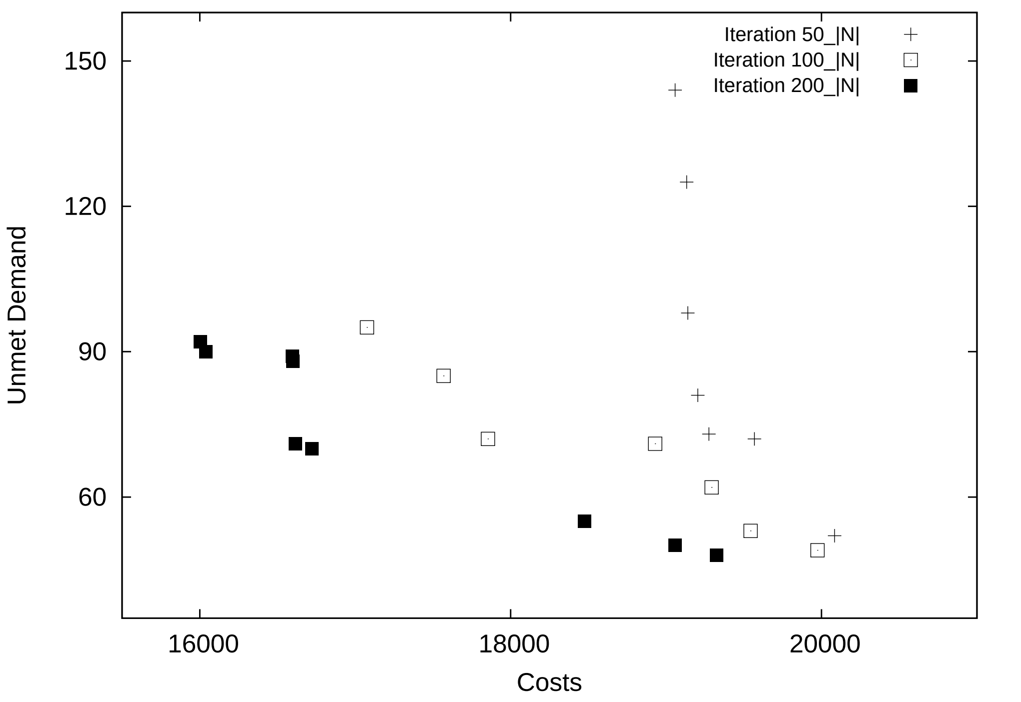


Figure 5.6: Example of convergence on NSGAII(1) for instance 50-5-20-100-5, $\epsilon = 50\%$ on set 2.

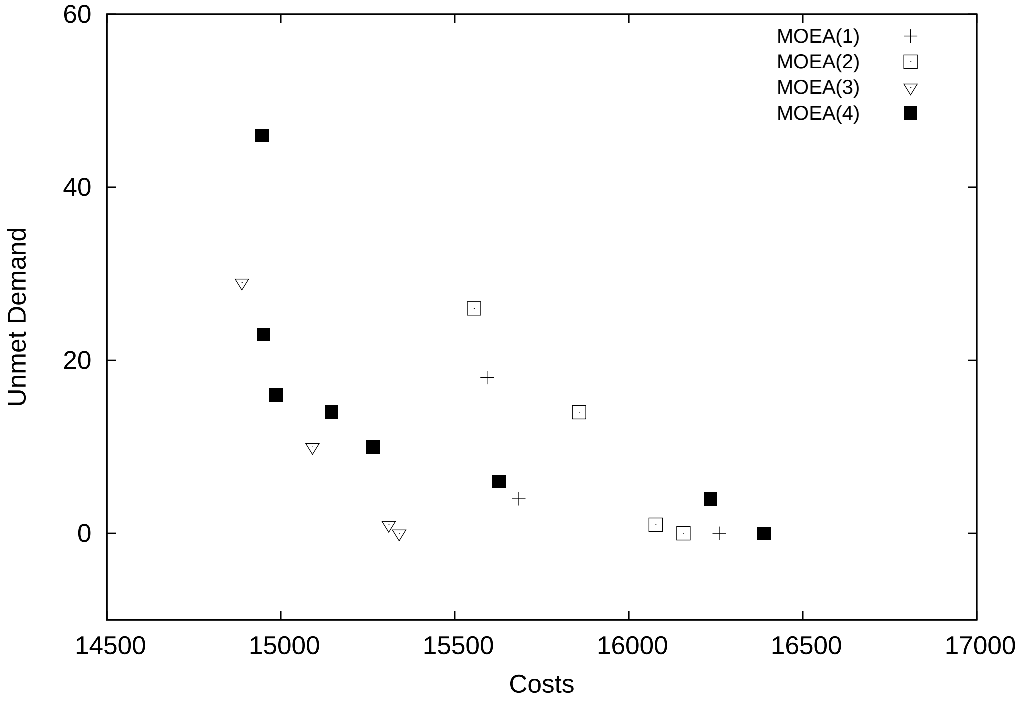


Figure 5.7: Impact of the different methods for instance 50-5-20-100-3 with $\varrho = 5\%$ on set 1.

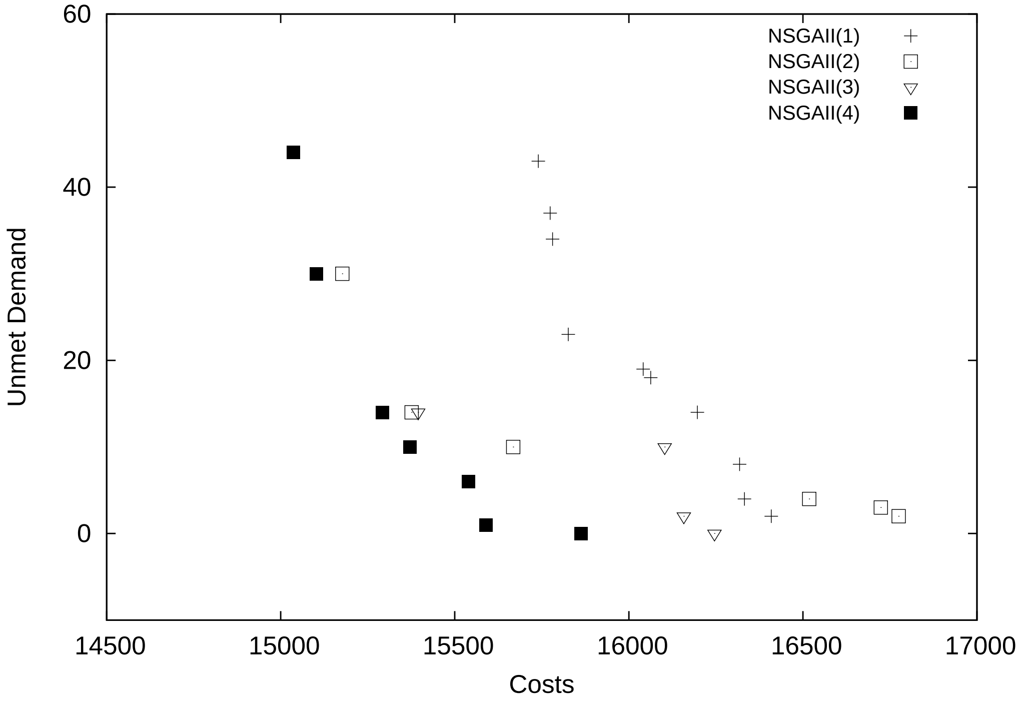


Figure 5.8: Impact of the different methods for instance 50-5-20-100-3 with $\varrho = 5\%$ on set 1.

As can been seen in Table 5.10, 5.11, and supported in graphic examples presented in Figure 5.7 and 5.8, all versions with local search procedure are better than versions without it. In terms of Θ, generally NSGAII found more solutions in the non-dominated Pareto front, offering to the decision maker a set of solutions with a good compromise between the worst total cost and the total unmet demand. Regarding the metric Y_2, both methods have a good coverage in the objective space by members from the non-dominated fronts. In general, NSGAII has got better results for the metric Y_1 than MOEA over its different variations which induce to a better distribution on the non-dominated pareto fronts. It is important to mention that the position of the solutions in the search space has a direct impact on the metric Y_1. Even when the metrics Y_2 and Θ improve their values, this does not guarantee better values for Y_1. As was expected the running time (CPU) differs on MOEA and NSGAII because of their specific operators to classify the population. In accordance with the different variations of the proposed methods, MOEA(4) and NSGAII(4) require more computational time due to the way the local search is integrated (local search on new solutions+local search on the first front).

Method	$\beta = 5\%$				$\beta = 25\%$			
	Θ_{avg}	Y_{1avg}	Y_{2avg}	CPU$_{avg}$	Θ_{avg}	Y_{1avg}	Y_{2avg}	CPU$_{avg}$
MOEA(1)	5.2	0.18	0.99	190.33	7.7	0.15	0.99	193.25
MOEA(2)	4.7	0.17	0.99	209.04	7.2	0.12	0.99	206.50
MOEA(3)	4.8	0.22	0.99	213.75	7.0	0.15	0.99	211.50
MOEA(4)	4.8	0.16	0.99	217.75	5.8	0.15	0.99	215.50
NSGAII(1)	5.6	0.15	0.99	193.33	8.3	0.10	0.98	196.00
NSGAII(2)	5.1	0.16	0.99	212.04	8.1	0.18	0.99	211.50
NSGAII(3)	4.5	0.20	0.99	215.75	6.9	0.18	0.99	213.50
NSGAII(4)	4.5	0.15	0.99	220.75	6.5	0.16	0.99	218.50

Table 5.10: Average results for set 1.

In order to see the impact of the number of scenarios on the solutions, the evaluation of the bi-RVRP with 10 and 20 clients for set A using $\beta = 5\%$ and 25% is considered and compared with the deterministic VRP, which means, nominal values for demands and travel times. Figures 5.9, 5.10, 5.11 and 5.12 show the results obtained with NSGAII(1). One can notice the price to pay for a robust solution against the deterministic case and the impact of the perturbation level β on final solutions. As expected, even when we can achieve the same costs as the VRP, we can face an increasing value for the total unmet demand that varies according to the percentage β and the number of scenarios.

Method	$\epsilon = 50\%$			
	Θ_{avg}	Y_{1avg}	Y_{2avg}	CPU$_{avg}$
MOEA(1)	9.7	0.12	0.98	188.92
MOEA(2)	8.8	0.17	0.98	220.79
MOEA(3)	7.3	0.19	0.99	209.42
MOEA(4)	8.1	0.16	0.99	232.79
NSGAII(1)	9.7	0.12	0.98	190.92
NSGAII(2)	6.6	0.15	0.99	224.79
NSGAII(3)	6.6	0.19	0.99	222.79
NSGAII(4)	7.8	0.13	0.99	240.75

Table 5.11: Average results for set 2.

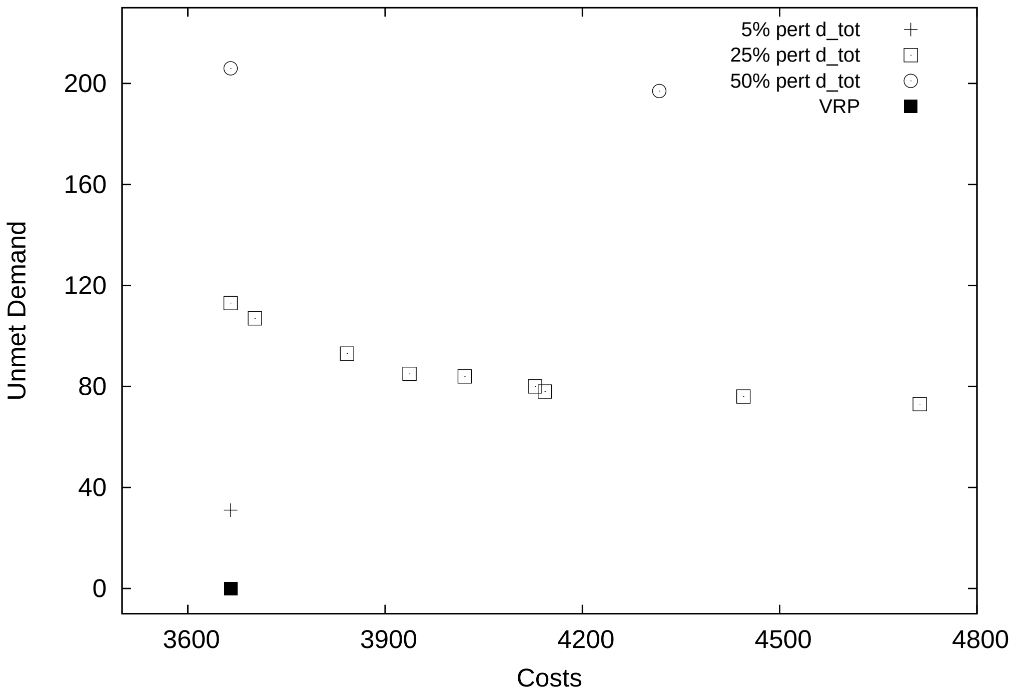


Figure 5.9: Impact of ϱ for instance 10-2-20-50-3 on NSGAII(1).

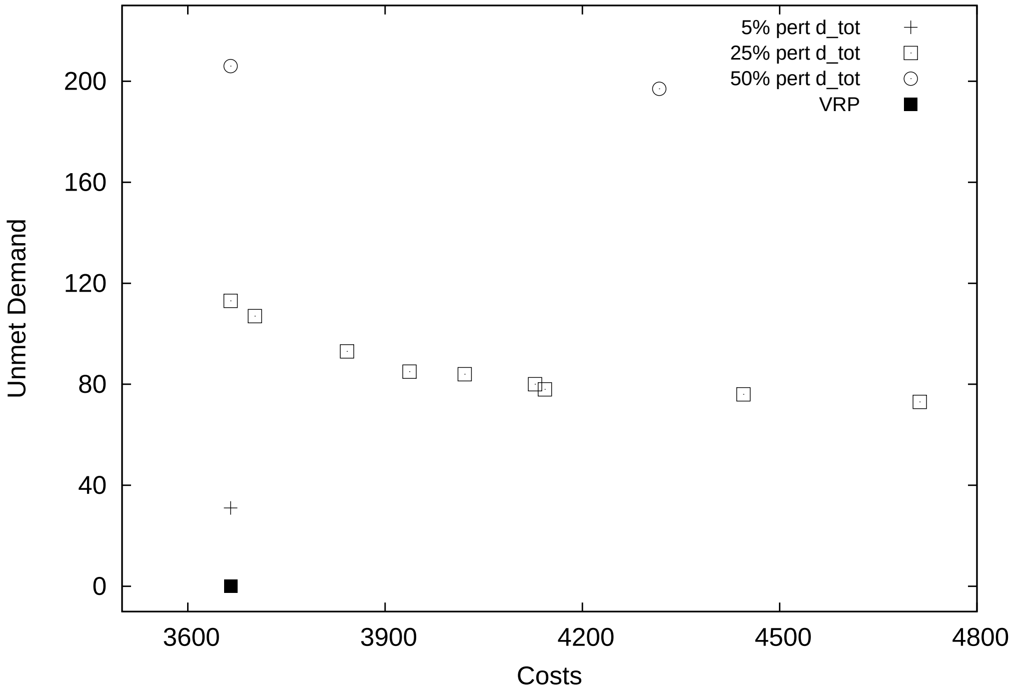


Figure 5.10: Impact of ϱ for instance 10-2-20-50-5 on NSGAII(1).

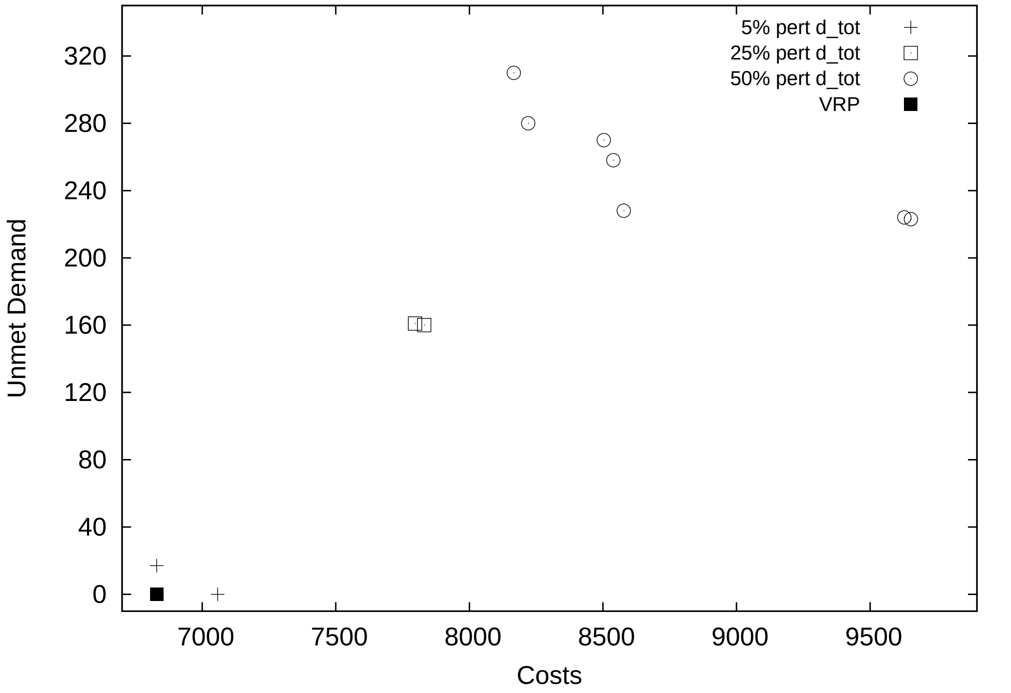


Figure 5.11: Impact of ϱ for instance 20-3-20-100-3 on NSGAII(1).

Instance	NSGAII(1)				NSGAII(2)				MOEA(1)				MOEA(2)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	2	0.00	1.00	73	2	0.00	1.00	78	4	0.26	0.99	70	3	0.17	1.00	75
30-3-20-50-5	7	0.16	0.99	72	4	0.27	0.99	79	4	0.35	1.00	69	4	0.00	0.99	76
30-3-20-100-3	4	0.04	1.00	72	2	0.00	0.99	76	3	0.48	0.99	69	2	0.00	1.00	73
30-3-20-100-5	1	0.00	1.00	85	4	0.20	1.00	88	3	0.32	0.99	82	5	0.25	0.99	85
30-3-30-50-3	2	0.00	1.00	82	1	0.00	1.00	84	1	0.00	1.00	79	2	0.00	1.00	81
30-3-30-50-5	4	0.11	0.99	83	3	0.01	1.00	86	1	0.00	1.00	80	3	0.03	1.00	83
30-3-30-100-3	4	0.48	1.00	110	7	0.17	0.98	111	2	0.00	0.99	107	2	0.00	0.99	108
30-3-30-100-5	2	0.00	1.00	85	5	0.14	0.99	87	2	0.00	1.00	82	5	0.38	1.00	84
40-4-20-50-3	2	0.00	0.99	161	11	0.08	0.99	168	8	0.12	0.98	158	6	0.10	0.99	165
40-4-20-50-5	14	0.08	0.98	164	9	0.19	0.97	169	5	0.29	0.99	161	6	0.23	0.97	166
40-4-20-100-3	8	0.10	0.98	183	6	0.04	0.99	172	7	0.25	1.00	180	5	0.21	0.99	169
40-4-20-100-5	5	0.22	1.00	172	7	0.09	0.99	174	6	0.40	0.99	169	6	0.13	0.99	171
40-4-30-50-3	9	0.10	0.99	179	4	0.02	1.00	190	10	0.09	0.99	176	6	0.24	1.00	187
40-4-30-50-5	7	0.12	0.99	176	7	0.04	1.00	239	10	0.09	0.98	173	8	0.09	1.00	236
40-4-30-100-3	2	0.00	1.00	171	3	0.38	0.97	227	4	0.24	1.00	168	6	0.07	0.98	224
40-4-30-100-5	9	0.10	0.98	180	4	0.41	0.97	229	6	0.34	0.96	177	9	0.11	0.95	226
50-5-20-50-3	7	0.29	1.00	303	6	0.02	0.99	333	6	0.17	0.99	300	7	0.20	0.99	330
50-5-20-50-5	7	0.26	0.99	317	5	0.23	1.00	347	9	0.09	0.98	314	5	0.09	0.99	344
50-5-20-100-3	11	0.19	0.99	319	7	0.24	0.99	349	3	0.08	1.00	316	4	0.37	1.00	346
50-5-20-100-5	5	0.32	1.00	313	4	0.47	0.97	343	9	0.06	0.97	310	4	0.12	0.99	340
50-5-30-50-3	7	0.12	0.99	331	6	0.21	0.98	361	6	0.08	0.99	328	5	0.25	0.99	358
50-5-30-50-5	4	0.45	0.99	335	6	0.30	0.99	365	6	0.22	1.00	332	6	0.23	1.00	362
50-5-30-100-3	4	0.36	0.99	334	5	0.25	1.00	364	5	0.26	0.99	331	3	0.75	0.99	361
50-5-30-100-5	8	0.12	0.99	340	4	0.07	0.98	370	5	0.10	1.00	337	1	0.00	1.00	367
Average	5.6	0.15	0.99	193.33	5.1	0.16	0.99	212.04	5.2	0.18	0.99	190.33	4.7	0.17	0.99	209.04

Table 5.12: Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 5\%$ for set 1.

Instance	NSGAII(1)				NSGAII(2)				MOEA(1)				MOEA(2)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	2	0.00	1.00	101	5	0.19	1.00	108	2	0.00	1.00	98	5	0.17	0.99	102
30-3-20-50-5	10	0.07	0.97	103	2	0.00	1.00	104	1	0.00	1.00	100	2	0.00	1.00	103
30-3-20-100-3	7	0.07	0.99	97	8	0.12	0.98	98	13	0.23	0.98	94	5	0.17	0.99	98
30-3-20-100-5	2	0.00	1.00	97	7	0.11	0.98	99	4	0.39	0.99	94	8	0.16	0.99	98
30-3-30-50-3	10	0.12	0.98	99	4	0.20	1.00	126	5	0.31	0.99	96	6	0.32	0.99	106
30-3-30-50-5	8	0.32	0.95	85	10	0.14	0.99	116	7	0.18	0.96	82	5	0.09	0.99	110
30-3-30-100-3	8	0.07	0.98	108	5	0.29	0.98	112	11	0.12	0.99	105	7	0.17	0.98	107
30-3-30-100-5	7	0.29	0.98	84	12	0.04	0.97	111	3	0.30	1.00	81	4	0.05	0.99	110
40-4-20-50-3	10	0.10	0.99	166	10	0.08	0.99	216	7	0.24	0.99	163	7	0.17	1.00	214
40-4-20-50-5	7	0.15	0.99	167	6	0.24	1.00	214	6	0.05	0.99	164	2	0.00	1.00	212
40-4-20-100-3	8	0.05	0.99	167	7	0.24	0.99	217	7	0.19	0.98	164	11	0.07	0.97	214
40-4-20-100-5	7	0.12	0.98	168	4	0.40	1.00	214	7	0.10	0.99	165	4	0.36	1.00	182
40-4-30-50-3	9	0.05	0.98	183	6	0.40	0.99	224	6	0.15	0.99	180	8	0.16	0.97	181
40-4-30-50-5	1	0.00	1.00	178	3	0.40	1.00	181	1	0.00	1.00	175	2	0.00	1.00	180
40-4-30-100-3	12	0.07	0.97	175	6	0.20	0.99	180	8	0.13	1.00	172	8	0.12	0.99	182
40-4-30-100-5	12	0.06	0.95	178	12	0.07	0.95	180	7	0.23	0.98	175	11	0.09	0.96	182
50-5-20-50-3	11	0.07	0.98	309	13	0.09	0.98	311	11	0.20	0.99	308	8	0.12	0.97	309
50-5-20-50-5	13	0.04	0.95	308	6	0.28	1.00	311	12	0.20	0.98	306	14	0.05	0.98	314
50-5-20-100-3	9	0.10	0.98	307	11	0.13	0.98	314	12	0.15	1.00	304	11	0.04	1.00	310
50-5-20-100-5	13	0.11	0.99	307	17	0.07	0.98	311	12	0.09	0.98	304	10	0.05	0.99	308
50-5-30-50-3	9	0.08	0.98	332	11	0.09	0.99	335	15	0.14	0.98	329	9	0.10	0.98	333
50-5-30-50-5	10	0.04	0.97	328	7	0.20	0.99	327	6	0.02	0.99	325	5	0.17	1.00	339
50-5-30-100-3	7	0.16	0.99	326	13	0.15	0.99	332	10	0.09	0.96	325	11	0.05	0.98	329
50-5-30-100-5	8	0.15	0.99	331	10	0.15	0.99	335	12	0.09	0.99	329	9	0.10	0.99	333
Average	8.3	0.10	0.98	196.00	8.1	0.18	0.99	211.50	7.7	0.15	0.99	193.25	7.2	0.12	0.99	206.50

Table 5.13: Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 25\%$ for set 1.

Instance	NSGAII(3)				NSGAII(4)				MOEA(3)				MOEA(4)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	1	0.00	1.00	81	1	0.00	1.00	86	2	0.00	1.00	79	1	0.00	1.00	83
30-3-20-50-5	2	0.00	1.00	82	5	0.27	0.98	87	5	0.38	0.99	80	4	0.47	1.00	84
30-3-20-100-3	4	0.21	0.99	79	2	0.00	1.00	84	2	0.00	1.00	77	2	0.00	1.00	81
30-3-20-100-5	4	0.22	1.00	89	2	0.00	1.00	94	4	0.40	0.99	87	1	0.00	1.00	91
30-3-30-50-3	2	0.00	1.00	87	1	0.00	1.00	92	1	0.00	1.00	85	2	0.00	1.00	89
30-3-30-50-5	1	0.00	1.00	89	1	0.00	1.00	94	1	0.00	1.00	87	3	0.38	0.99	91
30-3-30-100-3	2	0.00	1.00	118	5	0.55	1.00	123	5	0.53	1.00	116	6	0.01	1.00	120
30-3-30-100-5	3	0.18	0.99	89	3	0.30	0.99	94	4	0.19	1.00	87	2	0.00	0.99	91
40-4-20-50-3	2	0.00	0.99	169	5	0.18	0.98	174	5	0.08	0.99	167	7	0.16	0.99	171
40-4-20-50-5	6	0.30	0.99	171	8	0.21	0.97	176	4	0.52	0.99	169	6	0.14	0.99	173
40-4-20-100-3	6	0.14	1.00	173	7	0.13	0.99	178	3	0.20	1.00	171	5	0.08	1.00	175
40-4-20-100-5	7	0.22	0.99	174	6	0.20	0.99	179	11	0.09	0.99	172	5	0.29	0.99	176
40-4-30-50-3	6	0.19	0.99	194	4	0.02	1.00	199	12	0.11	0.99	192	8	0.16	0.99	196
40-4-30-50-5	8	0.30	0.98	243	6	0.11	0.97	248	13	0.07	0.98	241	6	0.09	0.99	245
40-4-30-100-3	5	0.37	1.00	232	1	0.00	1.00	237	4	0.31	0.99	230	3	0.20	1.00	234
40-4-30-100-5	8	0.18	0.97	235	8	0.10	0.94	240	4	0.18	1.00	233	10	0.16	0.98	237
50-5-20-50-3	4	0.43	1.00	335	5	0.27	0.99	340	3	0.08	1.00	333	2	0.00	1.00	337
50-5-20-50-5	8	0.09	0.99	352	5	0.21	1.00	357	4	0.35	1.00	350	5	0.14	0.99	354
50-5-20-100-3	4	0.36	1.00	358	7	0.10	1.00	363	4	0.44	1.00	356	8	0.12	1.00	360
50-5-20-100-5	4	0.35	1.00	346	6	0.18	0.99	351	6	0.44	0.96	344	7	0.32	1.00	348
50-5-30-50-3	7	0.24	0.98	368	5	0.30	0.98	373	5	0.05	0.99	366	5	0.16	1.00	370
50-5-30-50-5	4	0.38	1.00	371	3	0.20	1.00	376	9	0.15	0.98	369	8	0.16	0.99	373
50-5-30-100-3	3	0.47	1.00	369	6	0.06	0.99	374	4	0.55	1.00	367	4	0.43	1.00	371
50-5-30-100-5	7	0.11	0.98	374	6	0.11	0.99	379	1	0.10	0.99	372	5	0.33	0.98	376
Average	4.5	0.20	0.99	215.75	4.5	0.15	0.99	220.75	4.8	0.22	0.99	213.75	4.8	0.16	0.99	217.75

Table 5.14: Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 5\%$ for set 1.

Instance	NSGAII(3)				NSGAII(4)				MOEA(3)				MOEA(4)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	5	0.18	1.00	110	5	0.34	1.00	115	3	0.46	1.00	108	6	0.15	1.00	112
30-3-20-50-5	5	0.15	1.00	106	1	0.00	1.00	111	5	0.08	1.00	104	3	0.14	1.00	108
30-3-20-100-3	4	0.38	0.98	100	6	0.18	1.00	105	4	0.33	0.99	98	2	0.00	1.00	102
30-3-20-100-5	6	0.26	0.99	101	5	0.07	1.00	106	6	0.22	0.99	99	6	0.07	0.99	103
30-3-30-50-3	7	0.10	0.99	128	5	0.29	1.00	133	6	0.11	0.99	126	4	0.13	1.00	130
30-3-30-50-5	3	0.26	1.00	118	8	0.19	0.98	123	3	0.05	1.00	116	2	0.00	1.00	120
30-3-30-100-3	10	0.08	0.98	114	6	0.18	0.99	119	4	0.01	0.99	112	5	0.17	0.99	116
30-3-30-100-5	8	0.12	1.00	113	5	0.20	1.00	118	5	0.13	0.99	111	5	0.23	0.99	115
40-4-20-50-3	4	0.43	1.00	218	4	0.25	0.99	223	9	0.15	0.98	216	8	0.20	0.99	220
40-4-20-50-5	8	0.09	1.00	216	6	0.35	1.00	221	9	0.14	1.00	214	7	0.16	1.00	218
40-4-20-100-3	9	0.12	0.99	219	7	0.13	0.99	224	5	0.34	1.00	217	9	0.12	1.00	221
40-4-20-100-5	5	0.24	1.00	216	3	0.32	1.00	221	4	0.28	1.00	214	4	0.38	1.00	218
40-4-30-50-3	5	0.22	0.99	226	8	0.08	0.99	231	8	0.13	0.99	224	6	0.23	0.99	228
40-4-30-50-5	2	0.00	1.00	183	1	0.00	1.00	188	1	0.00	1.00	181	1	0.00	1.00	185
40-4-30-100-3	9	0.19	0.98	182	7	0.19	0.98	187	9	0.04	0.98	180	5	0.24	1.00	184
40-4-30-100-5	10	0.10	0.97	182	8	0.20	0.97	187	10	0.10	0.94	180	10	0.08	0.97	184
50-5-20-50-3	15	0.07	0.97	313	7	0.10	0.99	318	13	0.14	0.99	311	6	0.29	0.99	315
50-5-20-50-5	10	0.12	0.97	313	10	0.04	0.99	318	9	0.10	1.00	311	10	0.03	0.99	315
50-5-20-100-3	4	0.12	1.00	316	8	0.06	0.99	321	9	0.13	0.99	314	5	0.25	1.00	318
50-5-20-100-5	4	0.57	1.00	313	8	0.16	0.99	318	9	0.14	0.99	311	10	0.12	0.99	315
50-5-30-50-3	7	0.07	1.00	337	11	0.09	0.98	342	10	0.20	0.99	335	5	0.21	1.00	339
50-5-30-50-5	8	0.11	1.00	329	6	0.28	1.00	334	9	0.09	0.98	327	10	0.19	1.00	331
50-5-30-100-3	9	0.13	1.00	334	8	0.13	0.99	339	6	0.20	0.99	332	6	0.16	1.00	336
50-5-30-100-5	8	0.22	0.99	337	13	0.10	0.97	342	11	0.14	0.97	335	5	0.09	0.99	339
Average	6.9	0.18	0.99	213.50	6.5	0.16	0.99	218.50	7.0	0.15	0.99	211.50	5.8	0.15	0.99	215.50

Table 5.15: Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\varrho = 25\%$ for set 1.

Instance	NSGAII(1)				NSGAII(2)				MOEA(1)				MOEA(2)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	11	0.10	0.97	76	2	0.00	0.99	102	4	0.10	0.99	74	8	0.18	0.99	98
30-3-20-50-5	9	0.17	0.98	78	10	0.06	0.97	103	5	0.18	1.00	76	6	0.16	0.98	99
30-3-20-100-3	7	0.09	0.99	75	4	0.38	0.99	99	5	0.29	0.99	73	6	0.10	0.97	95
30-3-20-100-5	11	0.17	0.97	73	5	0.15	0.98	100	7	0.23	0.98	71	9	0.18	0.97	96
30-3-30-50-3	8	0.14	0.97	83	8	0.09	0.98	109	6	0.17	0.99	81	5	0.08	0.99	105
30-3-30-50-5	7	0.15	0.99	83	10	0.15	0.99	114	11	0.11	0.98	81	7	0.13	0.98	110
30-3-30-100-3	8	0.07	0.99	86	7	0.15	1.00	115	8	0.14	0.98	84	7	0.22	0.99	111
30-3-30-100-5	6	0.21	0.97	83	5	0.27	0.99	115	5	0.21	1.00	81	3	0.70	0.96	111
40-4-20-50-3	8	0.12	0.99	160	5	0.19	0.99	219	12	0.07	0.99	158	6	0.18	1.00	215
40-4-20-50-5	14	0.08	0.98	166	12	0.10	0.98	223	13	0.06	0.99	164	19	0.06	0.97	219
40-4-20-100-3	10	0.10	0.99	166	3	0.31	1.00	230	12	0.07	0.99	164	10	0.17	0.99	226
40-4-20-100-5	8	0.13	0.99	166	4	0.29	0.99	228	14	0.16	0.99	164	4	0.21	0.99	224
40-4-30-50-3	14	0.05	0.99	185	5	0.12	0.99	253	16	0.12	0.97	183	11	0.15	0.98	249
40-4-30-50-5	12	0.06	0.97	185	5	0.12	0.99	252	17	0.05	0.96	183	8	0.19	0.97	248
40-4-30-100-3	6	0.25	0.98	173	5	0.19	1.00	236	10	0.07	0.99	171	5	0.18	0.99	232
40-4-30-100-5	9	0.13	0.99	179	4	0.26	0.99	242	7	0.19	0.98	177	9	0.07	0.96	238
50-5-20-50-3	8	0.12	0.98	306	4	0.04	0.96	411	9	0.05	0.98	304	13	0.12	0.97	407
50-5-20-50-5	6	0.18	1.00	310	4	0.07	0.98	349	11	0.11	0.99	308	14	0.04	0.99	345
50-5-20-100-3	15	0.04	0.99	307	10	0.06	0.98	306	8	0.12	0.97	305	8	0.19	0.98	302
50-5-20-100-5	12	0.17	0.99	311	11	0.03	0.98	304	12	0.06	0.96	309	11	0.14	0.99	300
50-5-30-50-3	9	0.19	0.99	331	10	0.10	0.98	325	19	0.07	0.98	329	17	0.04	0.98	321
50-5-30-50-5	16	0.11	0.97	333	9	0.11	1.00	319	7	0.12	0.99	331	8	0.27	0.99	315
50-5-30-100-3	10	0.11	0.97	330	7	0.20	0.99	317	4	0.04	0.99	328	8	0.20	0.98	313
50-5-30-100-5	9	0.08	0.98	337	9	0.16	0.99	324	11	0.12	0.98	335	9	0.07	0.98	320
Average	9.7	0.12	0.98	190.92	6.6	0.15	0.99	224.8	9.7	0.12	0.98	188.92	8.8	0.17	0.98	220.79

Table 5.16: Results for NSGAII(1), NSGAII(2), MOEA(1) and MOEA(2) with $\hbar = 3$, $\hbar = 5$ and $\epsilon = 50\%$ for set 2.

Instance	NSGAII(3)				NSGAII(4)				MOEA(3)				MOEA(4)			
n–m–p–β–$\hbar$	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*	Θ	S	H	*CPU*
30-3-20-50-3	5	0.08	0.98	100	4	0.18	0.99	108	4	0.18	0.98	81	3	0.49	1.00	104
30-3-20-50-5	4	0.19	1.00	101	6	0.15	0.97	107	4	0.35	0.98	80	5	0.12	0.99	107
30-3-20-100-3	3	0.04	0.99	97	4	0.08	0.99	102	3	0.68	0.99	78	7	0.11	0.98	104
30-3-20-100-5	4	0.36	0.99	98	5	0.33	0.99	104	6	0.08	0.99	81	5	0.19	0.98	107
30-3-30-50-3	4	0.37	0.99	107	5	0.23	0.99	116	3	0.04	0.99	87	4	0.28	0.99	113
30-3-30-50-5	5	0.29	0.99	112	4	0.26	1.00	117	4	0.10	1.00	94	5	0.14	1.00	121
30-3-30-100-3	5	0.35	0.99	113	7	0.15	0.99	123	7	0.06	1.00	96	4	0.66	0.99	117
30-3-30-100-5	5	0.21	1.00	113	6	0.17	0.98	120	6	0.28	0.99	92	14	0.10	0.98	126
40-4-20-50-3	6	0.18	0.99	217	9	0.15	0.99	224	8	0.11	0.99	178	4	0.38	0.99	223
40-4-20-50-5	4	0.24	1.00	221	12	0.07	0.97	234	7	0.21	1.00	183	10	0.13	0.98	238
40-4-20-100-3	8	0.13	0.99	228	8	0.08	0.99	231	5	0.29	0.99	178	6	0.09	1.00	228
40-4-20-100-5	5	0.12	1.00	226	11	0.07	1.00	233	6	0.12	1.00	180	8	0.05	1.00	231
40-4-30-50-3	9	0.25	1.00	251	12	0.03	0.99	261	11	0.12	0.99	252	15	0.04	0.99	272
40-4-30-50-5	5	0.19	0.99	250	8	0.10	0.99	259	6	0.24	1.00	263	13	0.08	0.99	268
40-4-30-100-3	4	0.23	1.00	234	9	0.10	0.98	242	3	0.60	1.00	241	2	0.00	1.00	236
40-4-30-100-5	6	0.25	0.98	240	9	0.08	0.97	255	8	0.16	0.99	247	6	0.20	0.98	227
50-5-20-50-3	8	0.17	0.99	409	13	0.05	0.97	432	7	0.08	0.99	406	16	0.21	0.97	336
50-5-20-50-5	12	0.08	0.98	347	10	0.08	0.99	403	9	0.16	0.99	344	11	0.08	0.99	336
50-5-20-100-3	5	0.24	1.00	304	4	0.29	1.00	322	10	0.05	0.97	301	6	0.09	0.99	322
50-5-20-100-5	11	0.06	0.98	302	12	0.11	0.99	328	12	0.22	0.99	299	10	0.12	1.00	330
50-5-30-50-3	9	0.22	0.99	323	7	0.11	1.00	355	12	0.06	1.00	320	8	0.05	0.99	360
50-5-30-50-5	11	0.04	0.98	317	6	0.28	1.00	358	21	0.04	0.97	314	11	0.12	0.99	363
50-5-30-100-3	6	0.12	0.99	315	1	0.00	1.00	369	6	0.18	0.99	312	6	0.13	1.00	358
50-5-30-100-5	15	0.06	0.98	322	14	0.03	0.99	375	7	0.24	1.00	319	15	0.04	0.98	360
Average	6.6	0.19	0.99	222.79	7.8	0.13	0.99	240.75	7.3	0.19	0.99	209.42	8.1	0.16	0.99	232.79

Table 5.17: Results for NSGAII(3), NSGAII(4), MOEA(3) and MOEA(4) with $\hbar = 3$, $\hbar = 5$ and $\epsilon = 50\%$ for set 2.

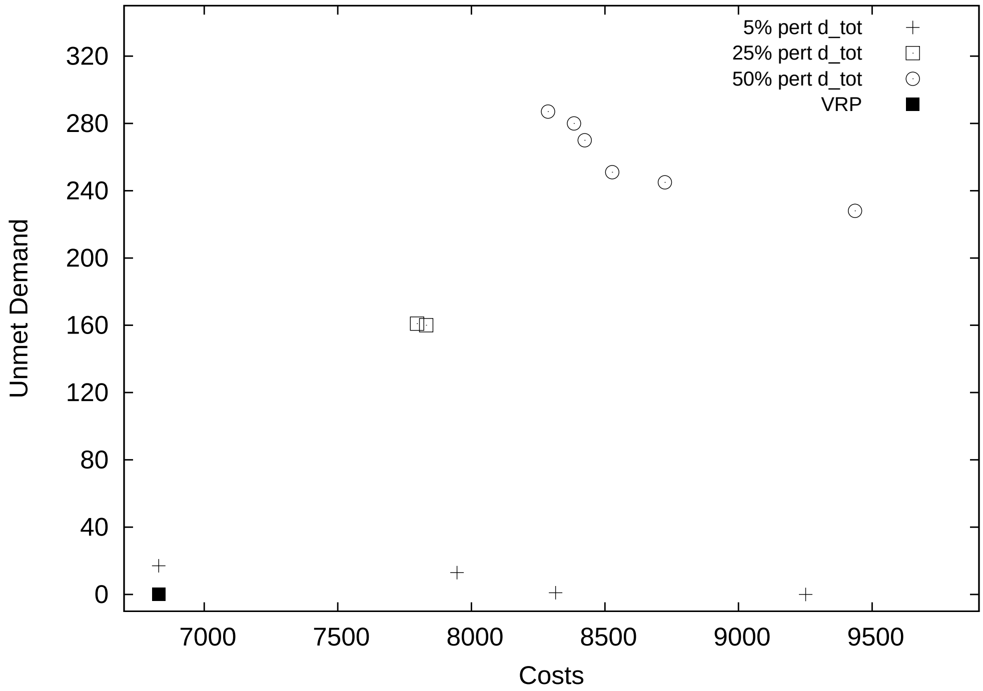


Figure 5.12: Impact of ϱ for instance 20-3-20-100-5 on NSGAII(1).

5.5.5 Statistical tests.

The Friedman test is considered to evaluate the performance of each method according to Θ, Y_1 and Y_2. The time is not considered in this test due to its evident difference on the proposed versions for MOEA and NSGAII.

The test compares the $\kappa = 8$ metaheuristics (NSGAII(1), NSGAII(2),NSGAII(3), NSGAII(4), MOEA(1), MOEA(2), MOEA(3), MOEA(4)) on the test bed of $\varsigma = 24$ instances, considering the results for $\varrho = 5\%$, $\varrho = 25\%$ and $\epsilon = 50\%$ over the three perfomance indicators Θ, Y_1, Y_2. The procedure is the same described in Section 4.6.5. Let Ψ_{ij} denotes the perfomance indicator values achieved by algorithm h on instance o. First, compute the ranks Γ_{ho} of the Ψ_{ho}. Then calculate the sum of ranks $\mathcal{S}_h = \sum_{h=1}^{\varsigma} \Gamma_{ho}$ for each metaheuristic o, the mean of these squared sums $\mathcal{G}_2 = \frac{1}{\varsigma} \sum_{o=1}^{\kappa} S_o^2$ and the sum of squared ranks $\mathcal{J}_2 = \sum_{h=1}^{\varsigma} \sum_{o=1}^{\kappa} \Gamma_{ho}^2$. The test statistic is given by Equation (4.12) in Section 4.6.5.

The null hypothesis assumed all metaheuristics have equivalent performances and it can be rejected with a confidence level α if $\mathcal{T}_2$ is greater than the $1 - \alpha$ quantile of the $\mathcal{F}$ distribution with $\kappa_1 = \kappa - 1$ and $\kappa_2 = (\kappa - 1)(\varsigma - 1)$ degrees of freedom, giving $\mathcal{F}_{0.99,7,161} = 2.64$. Results are detailed in Tables C.1-C.18 on Appendix C and summarized in Tables 5.18-5.20. In the cases where $\mathcal{T}_2 > \mathcal{F}_{0.99,7,161}$ it can be concluded with a confidence level of 1% that the heuristics are not equivalent.

Algorithm	NSGAII	NSGAII	NSGAII	NSGAII	MOEA	MOEA	MOEA	MOEA
	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
Average rank (Θ)	3.8	4.3	4.8	5.1	4.2	4.3	4.8	4.8
Sum of ranks $\mathcal{S}_o$ (Θ)	91.5	102.0	114.5	121.5	101.5	104.0	114.5	114.5
Sum of squared ranks (Θ)	513.3	523.0	653.8	715.8	529.3	563.5	650.3	650.3
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Θ)	4,799.0	3,916.1	0.7					
Average rank (Y_1)	4.3	4.0	5.2	4.1	4.5	4.7	4.9	4.3
Sum of ranks $\mathcal{S}_o$ (Y_1)	103.5	97.0	124.0	99.0	107.5	113.0	117.5	102.5
Sum of squared ranks (Y_1)	554.8	497.5	734.0	518.0	609.8	647.5	704.8	538.8
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_1)	4,805.0	3,914.0	0.7					
Average rank (Y_2)	5.1	4.9	4.0	4.9	4.9	4.5	4.1	3.6
Sum of ranks $\mathcal{S}_o$ (Y_2)	123.5	116.5	96.5	117.5	116.5	108.5	99.0	86.0
Sum of squared ranks (Y_2)	739.8	634.3	451.3	674.8	634.3	567.8	491.0	371.0
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_2)	4,564.0	3,936.9	1.8					

Table 5.18: Friedman test on Θ, Y_1, Y_2 with $\varrho = 5\%$.

In order to separate the metaheuristics when they show statistical differences in their performance a pairwise test is applied. For that the absolute difference of the sums of ranks of metaheuristics h and o are calculated. Two metaheuristics are considered differents if their difference exceed the critical value $\mathcal{CV}$ defined in Equation (4.13) from Section 4.6.5. Results are detailed in Tables 5.21, 5.22 and 5.23.

The critical value is defined as $\mathcal{CV}_1 = 31.7$ for Y_2 with $\varrho = 25\%$, $\mathcal{CV}_2 = 35.8$ for Θ with $\epsilon = 50\%$ and $\mathcal{CV}_3 = 34.0$ for Y_2 with $\epsilon = 50\%$. The significant differences (underlined) indicate that a metaheuristic dominates another one. The metaheuristic with the smallest sum of ranks is the best of the two.

Considering these results, it may notice that integrating a local search procedure in multiobjective evolutionary metaheuristics affects their performance on the bi-RVRP. For the analysis of the proposed metaheuristics, just the first 4th positions in the ordering given by the Pairwise test are given, beginning with the best method, followed by the second one and so on. In terms of Y_2 with $\varrho = 25\%$, the ordering given is: MOEA(4), NSGAII(3), NSGAII(4), MOEA(3). Regarding Θ with $\epsilon = 50\%$, the ordering obtained is: NSGAII(1), MOEA(1), MOEA(2), NSGAII(4). Finally, for Θ with $\epsilon = 50\%$, the ordering obtained is:

Algorithm	NSGAII (1)	NSGAII (2)	NSGAII (3)	NSGAII (4)	MOEA (1)	MOEA (2)	MOEA (3)	MOEA (4)
Average rank (Θ)	3.4	3.8	4.4	5.4	4.5	4.3	4.7	5.6
Sum of ranks $\mathcal{S}_o$ (Θ)	81.5	90.0	105.5	128.5	109.0	102.0	113.5	134.0
Sum of squared ranks (Θ)	387.8	469.5	585.3	762.8	529.3	563.5	650.3	650.3
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Θ)	4,844.0	3,979.5	2.4					
Average rank (Y_1)	3.1	5.1	4.9	4.9	4.6	3.7	4.8	4.9
Sum of ranks $\mathcal{S}_o$ (Y_1)	75.5	122.0	116.5	118.5	115.5	89.0	114.0	117.0
Sum of squared ranks (Y_1)	340.3	753.0	688.3	670.8	650.8	405.5	645.5	702.0
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_1)	4,856.0	3,968.2	2.1					
Average rank (Y_2)	6.1	4.9	3.9	3.9	4.8	5.0	4.4	3.1
Sum of ranks $\mathcal{S}_o$ (Y_2)	145.5	117.5	93.0	93.5	115.5	119.0	106.5	73.5
Sum of squared ranks (Y_2)	963.8	652.3	446.5	423.8	658.3	669.5	554.3	257.8
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_2)	4,626.0	4,025.6	5.3					

Table 5.19: Friedman test on Θ, Y_1, Y_2 with $\varrho = 25\%$.

Algorithm	NSGAII	NSGAII	NSGAII	NSGAII	MOEA	MOEA	MOEA	MOEA
	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
Average rank (Θ)	2.8	5.5	5.8	4.6	3.2	3.9	5.3	4.9
Sum of ranks $\mathcal{S}_o$ (Θ)	68.0	132.0	138.0	110.0	76.5	94.0	127.0	118.5
Sum of squared ranks (Θ)	257.5	844.0	866.0	604.5	338.3	458.0	766.0	719.3
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Θ)	4,853.5	4,085.5	5.9					
Average rank (Y_1)	4.2	4.6	5.5	3.9	3.8	4.7	4.9	4.4
Sum of ranks $\mathcal{S}_o$ (Y_1)	101.0	111.0	131.5	93.0	91.5	113.5	117.0	105.5
Sum of squared ranks (Y_1)	498.0	649.0	845.3	474.5	429.3	648.3	744.5	592.3
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_1)	4,881.0	3,939.0	1.2					
Average rank (Y_2)	5.3	4.6	3.8	4.0	5.2	5.8	3.4	3.8
Sum of ranks $\mathcal{S}_o$ (Y_2)	128.0	109.5	92.0	97.0	125.5	138.5	82.5	91.0
Sum of squared ranks (Y_2)	771.5	597.8	443.5	487.0	747.8	874.8	371.8	409.0
Indicator	$\mathcal{J}_2$	$\mathcal{G}_2$	$\mathcal{T}_2$					
Value (Y_2)	4,703.0	4,011.1	4.1					

Table 5.20: Friedman test on Θ, Y_1, Y_2 with $\epsilon = 50\%$.

MOEA(3), MOEA(4), NSGAII(3), NSGAII(4).

As can be noticed, a good compromise on results for Θ, Y_1 and Y_2 on set 1 and 2 can be achieved with NSGAII(4), which consistently diminishes the worst cost and the maximum unmet demand along the non-dominated Pareto front.

|Ri - Rj|	NSGAII(2)	NSGAII(3)	NSGAII(4)	MOEA(1)	MOEA(2)	MOEA(3)	MOEA(4)
NSGAII(1)	28.0	52.5	52.0	30.0	26.5	39.0	72.0
NSGAII(2)		24.5	24.0	2.0	1.5	11.0	44.0
NSGAII(3)			0.50	22.5	26.0	13.5	19.5
NSGAII(4)				22.0	25.5	13.0	20.0
MOEA(1)					3.5	9.0	42.0
MOEA(2)						12.5	45.5
MOEA(3)							33.0

Table 5.21: Pairwise test for Y_2 with $\varrho = 25\%$.

|Ri - Rj|	NSGAII(2)	NSGAII(3)	NSGAII(4)	MOEA(1)	MOEA(2)	MOEA(3)	MOEA(4)
NSGAII(1)	64.0	70.0	42.0	8.5	26.0	59.0	50.5
NSGAII(2)		6.0	22.0	55.5	38.0	5.0	13.5
NSGAII(3)			28.0	61.5	44.0	11.0	19.5
NSGAII(4)				33.5	16.0	17.0	8.5
MOEA(1)					17.5	50.5	42.0
MOEA(2)						33.0	24.5
MOEA(3)							8.5

Table 5.22: Pairwise test for Θ with $\epsilon = 50\%$.

5.6 Conclusions

In this chapter the bi-RVRP with uncertainty in both demands and travel times is studied by means of robust optimization. Here, the bi-RVRP aims at minimizing the worst total cost of traversed arcs and minimizing the maximum total unmet demand over all scenarios. To the best of our knowledge, this version of the bi-RVRP has never been studied before and finds practical applications in urban transportation. MOEA and NSGAII are adapted to solve the bi-RVRP together with a local search. Different variations for both metaheuristics were considered on the location of the proposed local procedure in the MOEA and NSGAII structure. Different analysis are done including the impact that have the number of scenarios

|Ri - Rj|	NSGAII(2)	NSGAII(3)	NSGAII(4)	MOEA(1)	MOEA(2)	MOEA(3)	MOEA(4)
NSGAII(1)	18.5	36.0	31.0	2.5	10.5	45.5	37.0
NSGAII(2)		17.5	12.5	16.0	29.0	27.0	18.5
NSGAII(3)			5.0	33.5	46.5	9.5	1.0
NSGAII(4)				28.5	41.5	14.5	6.0
MOEA(1)					13.0	43.0	34.5
MOEA(2)						56.0	47.5
MOEA(3)							8.5

Table 5.23: Pairwise test for Y_2 with $\epsilon = 50\%$.

on the worst total cost and the total unmet demand, the impact of the local search on the MOEA and the NSGAII, and the convergence as well as the diversity for the different methods. Results show that our proposed methods have a good behaviour on all the performance metrics used. Although in the cases were the deterministic version of the VRP gives solutions with no unmet demands our proposed method can find solutions that support the variations that might arise in the future with a good comprise between the cost and unmet demand. As future work, there is a room for including other representations for the uncertainties and to design exact approaches handling the bi-RVRP. An interesting option would be to satisfy as much as possible the demand of a customer when it cannot be completely satisfy, and to accept split deliveries.

Conferences and Publications

International Journal

- Solano-Charris, E. L., Prins, C., & Santos, A. C. (Under revision). Solving the Bi-Objective Robust Vehicle Routing Problem with Uncertain Costs and Demands.

Conferences without published proceedings

- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2015. The Bi-Objective Robust Vehicle Routing Problem with Uncertain Demands and Travel Times. Metaheuristics International Conference (MIC), Agadir, Morocco.
- Solano-Charris, E. L., Prins, C., & Santos, A. C. 2015. Bi-objective heuristics for solving the robust vehicle routing problem with uncertain travel times and demands. European Conference on Operational Research (EURO), Glasgow, Schotland.

Chapter 6

General Conclusions and Future Research Directions

The Vehicle Routing Problem (VRP) is a well-known and hard combinatorial optimization problem with a large number of real applications. In this thesis we have studied extensions of the VRP with uncertain data. Several methods are available to model uncertainties, mainly focuses on stochastic programming where uncertain data are modeled as random variables with known probability distributions and robust optimization, which is no longer stochastic but rather deterministic and set-based. This thesis is devoted to robust optimization approaches for VRPs with uncertain data, giving what we call the Robust Vehicle Routing Problem (RVRP). For the RVRP two main problems have been studied, a RVRP with uncertain travel times/costs and a bi-objective version of the RVRP (bi-RVRP) with uncertain travel times and demands. Besides, two robust classification criteria (RCC) for the evaluation of solutions and scenarios have been introduced mainly as a way for reinforcing the local search procedures or for combining these criteria with the existing robust optimization criteria.

The RVRP with uncertain travel times is presented in Chapter 4. Uncertain travel times are modeled by discrete scenarios in a complete and directed network, in which each scenario defines one travel time for each arc in a given hour. The aim is to minimize the total route duration over all scenarios. For the solution of the RVRP a mathematical model and several methods have been developed. The proposed methods include deterministic and randomized versions of constructive heuristics, i.e., the Clarke and Wright (CW), Randomized CW (RCW), Parallel Best Insertion (PBI), Sequential Best Insertion (SBI), Pilot Parallel Best Insertion (Pilot PBI) and Pilot Sequential Best Insertion (Pilot SBI). Furthermore, a Genetic Algorithm (GA) and four local search-based metaheuristics have been proposed, i.e., one GRASP, one ILS, one multi-start ILS (MS-ILS), and a MS-ILS with giant tours.

Computational experiments on 18 small-size instances and 24 medium-size instances randomly generated up to 100 clients and 30 scenarios are provided. The experimental results show the hardness of the RVRP in spite of its rather simple statement. Among constructive heuristics the most promising methods in terms of running time and average solution gap are the CW and its randomized version. Regarding the GA, 7 out of 10 optima values found by the CPLEX are retrieved on the small random instances. On local search-based methods, their efficiency has been demonstrated using the small-size instances and several CVRP instances with known optimal solution. These metaheuristics found very close average gaps in comparison with the results given by CPLEX. Moreover, all the proven optima are retrieved and three upper bounds are improved. Nonetheless, alternation between giant tours and RVRP solutions in MS-ILS brings a statistically significant improvement considering a fixed number of calls to the local search and a time budget.

In Chapter 5 the RVRP is extended to a multiobjective optimization version. Uncertainties are addressed in the objective function, minimizing the total cost of traversed arcs and the total unmet demand over all scenarios. Independent uncertain sets are modeled by discrete scenarios or intervals representing possible deviations from their expected values. For solving the bi-RVRP, a Non-dominated Sorting Genetic Algorithm version 2 (NSGAII) and a Multiobjective Evolutionary Algorithm (MOEA) embedded with a dedicated local search procedure (non included in the classical version of these metaheuristics) are proposed.

Several location of the local search and two acceptance criteria for a move have been tested on two set of 24 random instances assuming a global perturbation over the total demand of clients and an individual variation on the demand defined in an interval. The sets of instances contain 24 instances up to 50 customers, 5 vehicles, 30 scenarios for travel times and 5 scenarios for demands.

Analyses on the impact of the number of scenarios and the local search procedure have been done. Concerning the impact of the number of scenarios on the solutions, the evaluation of the bi-RVRP is compared with the CVRP with 10 and 20 clients unveiled the price to pay for a robust solution. Although, in the cases where the CVRP gives solutions with no unmet demands, our proposed method can find solutions that support the variations that might arise in the future with a good comprise between the cost and unmet demand. Different metrics are also used to measure the efficiency, the convergence as well as the diversity of solutions for these metaheuristics, i.e., the number of solutions in the first front, the hypervolume and the spacing. Results show that all versions for our NSGAII and MOEA with the local search procedure consistently diminish the worst cost and the maximum unmet demand along the no-dominated Pareto front, and have a good compromise on the multiobjective performance measures.

As future directions for the thesis, application for the RCC could be done over local search-based metaheuristics. On the RVRP and the bi-RVRP, there is a room for including other representations of uncertainties and for improving some of the methods introduced in this work. The positive impact of the MS-ILS-GT would be interesting to test on other metaheuristics like hybrid genetic algorithms, although it may increase the difficulty on the RVRP due to the several components that are involved.

A current drawback of our metaheuristics for a robust VRP with q scenarios is a running time multiplied by $q \log q$, compared with the deterministic version. This can lead to excessive running times if the decision maker wishes to employ many scenarios (more than 50 for instance). Using ad-hoc data structures or neighborhood reduction techniques, a substantial reduction of running time (on average) could be expected.

The development of exact algorithms is promising but seems a challenging task. Currently we are working on a mathematical model for the bi-RVRP to handle the specific characteristics of our problem.

It could be also interesting to extend the methods here proposed for handling other optimization criteria such as min-max regret and the min-max relative regret. Adaptation for large scale RVRP and bi-RVRP are also contemplated. Finally, uncertainties on other parameters of the CVRP could be addressed e.g., time windows, and the comparison of the results provided by the robust optimization approach against other techniques for handling uncertainties like stochastic programming.

Appendix A

Résumé étendu en français : Méthodes d'optimisation pour le problème de tournées de véhicules robuste

Chapitre 1 - Introduction Génerale

L'optimisation combinatoire est un des domaines les plus actifs en recherche opérationnelle, informatique et mathématiques appliquées. La majorité des applications réelles incluent les télécommunications, la conception de réseaux, l'ordonnancement et le transport. Parmi les problèmes de transport, le Problème de Tournées de Véhicules (VRP) est un problème complexe où on veut déterminer un ensemble de routes desservies par une flotte de véhicules, basés dans un dépôt, pour servir un ensemble de clients (Toth & Vigo (1998)). Introduit par Dantzig & Ramser (1959), le VRP occupe une place majeure dans le management de la distribution et est devenu un des problèmes les plus étudiés en optimisation combinatoire. Des articles de synthèse sur le VRP ont été écrit par Fisher (1995); Golden *et al.* (2008); Laporte (2009); Toth & Vigo (2014). Pour résoudre ce problème, une des principales hypothèses est que les paramètres et les données sont supposés être déterministes et connus d'avance. Cependant, dans les applications réelles, les opérations dans un réseau de transport quelconque contiennent un assez haut degré d'incertitude, qui comprend, entre autres, l'arrivée de nouveaux clients ou l'annulation des services, des temps d'attente variables, et des temps de transport perturbés par la congestion du trafic (Sungur *et al.* (2008)). Par conséquent, une perturbation dans les données d'entrée peut engendrer des solutions non optimales et parfois infaisables (Moghaddam *et al.* (2012)). Une importante tendance pour considérer cette perturbation consiste à étudier des extensions du VRP avec données incertaines, en termes de problèmes aussi bien théoriques que pratiques.

Plusieurs approches sont disponibles pour modéliser les incertitudes dans le contexte des problèmes d'optimisation, principalement focalisées sur la programmation stochastique où les données incertaines sont modélisées par des variables aléatoires qui ont une distribution de probabilités connue (Wets (2002); Shapiro *et al.* (2008)). Néanmoins, une telle approche est limitée aux cas où les incertitudes ont une nature stochastique (ce qui n'est pas toujours le cas) et lorsqu'il est possible d'identifier la distribution de probabilité, qui peut être compliquée à obtenir, en particulier pour les problèmes à grande taille (Ben-Tal & Nemirovski (1998)). Un grand nombre d'études traite avec la programmation stochastique. Les modèles pionniers pour la programmation stochastique ont été proposés par Beale (1955) et Dantzig (1955). En particulier, pour le Problème de Tournées de Véhicules Stochastique (SVRP), les problèmes les plus étudiés ont été les VRP avec des demandes stochastiques et des clients stochastiques (Bertsimas (1992); Gendreau *et al.* (1995); Secomandi (2001)). En dépit de son importance dans les applications réelles, spécialement dans des grandes villes, peu de travaux ont été réalisés pour le VRP avec des temps de transport stochastiques (Laporte *et al.* (1992); Lambert *et al.* (1993)).

Une alternative à la programmation stochastique est l'optimisation robuste qui a été appliquée en général comme une façon d'éviter les impacts indésirables dus à des approximations vagues, incomplètes, imprécises, ou des données ambigües. Dans ce contexte, la littérature couvre un grand nombre d'applications telles que l'ordonnancement (Goren & Sabuncuoglu (2008); Hazir *et al.* (2010)), l'emplacement des installations (Minoux (2010); Baron *et al.* (2011); Alumur *et al.* (2012); Gülpinar *et al.* (2013)), la gestion des stocks (Bienstock & Özbay (2008); Ben-Tal *et al.* (2009a)), la finance (Fabozzi *et al.* (2007); Gülpinar & Rustem (2007); Ç. Pinar (2007); Bertsimas & Pachamanova (2008); Schoettle & Werner (2009)), les réseaux de files d'attente, les systèmes stochastiques et la théorie des jeux (Bertsimas & Doan (2010); Kırkızlar & Andradóttir (2010); Ordóñez & Stier-Moses (2010); Kardes *et al.* (2011)), l'apprentissage automatique et les statistiques (Xu *et al.* (2010); Ben-Tal *et al.* (2011); Xu *et al.* (2012)), et enfin les systèmes énergétiques (Ribas *et al.* (2010); Bertsimas *et al.* (2011)).

L'idée générale de cette approche est de fournir des solutions robustes en prenant en compte plusieurs difficultés techniques telles que l'évaluation de ces solutions, l'adaptation des méthodes heuristiques pour des problèmes de tournées à grande échelle et l'appréciation de l'efficience et des mesures de robustesse. Grosso modo, trois principales composantes doivent être considérées : (i) la manière dont les données incertaines sont modélisées (e.g., avec l'utilisation de scénarios), (ii) la sélection d'un critère d'optimisation robuste approprié tel que le min-max, l'écart min-max, l'écart relatif min-max, etc., appelés dans la suite Critères d'Optimisation Robuste (ROC), et (iii) le choix d'un modèle mathématique et de méthodes pour générer des solutions robustes (Voir Figure 1.1, adaptée de Kouvelis & Yu (1997)). Cette thèse de doctorat est dédiée à l'approche d'optimisation robuste abordant des incertitudes sur le VRP, dérivé du Problème de Tournées de Véhicules Robuste (RVRP). Premièrement, les incertitudes sont traitées sur les temps de transport, puis, une deuxième version du RVRP est considérée en prenant en compte les temps de transport ainsi que les demandes sur une version bi-objectif du RVRP (bi-RVRP). Pour résoudre le RVRP et le bi-RVRP, différents modèles et méthodes sont proposés pour déterminer des solutions robustes (moins affectées par les incertitudes) en minimisant le pire-cas. En particulier pour le RVRP, des heuristiques constructives, une approche évolutive, et des recherches locales itérées à démarrage multiple sont adaptées pour gérer des incertitudes. Concernant le bi-RVRP, différentes versions de méta-heuristiques évolutionnaires multi-objectif, couplées avec une procédure de recherche locale sont proposées.

La structure de ce document comprend quatre parties, la première est dédiée au compte rendu général de la littérature considérée dans ces travaux. La seconde partie présente les définitions générales et composantes pour l'Optimisation Robuste (RO). La troisième partie considère le RVRP avec coûts de transport incertains, et la dernière introduit le bi-RVRP avec incertitude sur les demandes ainsi que sur les temps de transport. Les principales contributions, classées par chapitres, sont présentées ci-dessous :

- Le chapitre 2 fournit les définitions générales pour la RO joint à une revue des ROC les plus communs, i.e., min-max, écart min-max, écart relatif min-max, critère lexicographique, α-robustesse, bw-robustesse et pw-robustesse, ... et leurs applications principales. Ensuite, deux nouveaux Critères de Classifications Robustes (RCC) sont proposés. Ils peuvent être appliquées pour mesurer la robustesse des scénarios et des solutions. Les RCC alternatifs proposés ici pourraient être utilisés indépendamment d'autres critères, comme un support additionnel du ROC classique, ou encore comme outil pour renforcer des stratégies de recherche locale. Il est important de souligner que les RCC proposés ne sont pas des critères d'optimisation, au contraire, ce sont des mesures de qualité basées sur le rang (classification) des solutions sur tous les scénarios considérés.

- Le chapitre 3 présente un état de l'art sur le VRP. En particulier, quelques articles-clés sont analysés sur le Problème de Tournées de Véhicules avec Capacités (CVRP) et les méthodes pour le résoudre tels que des heuristiques constructives et d'amélioration, des méthodes exactes et des méta-heuristiques. Une revue pour les VRPs avec incertitudes considérant le RVRP et SVRP est également fournie. Puisque le SVRP ne fait pas parti de nos travaux, seules quelques références sont citées pour ce problème.

- Dans le Chapitre 4, le RVRP avec coûts de transport incertains traités via l'optimisation robuste est étudié. Les coûts de transport incertains sont modélisés par des scénarios discrets dans un réseau complet et orienté (la littérature classique du VRP considère principalement des graphes non-orientés avec coûts symétriques), dans lequel chaque scénario définit un temps de transport pour chaque arc. L'objectif est de minimiser le pire coût (durée totale des tournées) sur tous les scénarios. Ces scénarios ne correspondent pas aux réalisations de variables aléatoires et ne sont pas associés à des probabilités d'occurrence. La situation reflète par exemple des problèmes de transit dans des réseaux urbains et a également une application dans l'urgence en cas de bioterrorisme à grande échelle. Pour résoudre le RVRP, une formulation mathématique est introduite. Ensuite, plusieurs heuristiques constructives sont proposées telles que celle de Clarke et Wright (CW), CW Aléatoire (RCW), l'Algorithme d'Insertion Parallèle (PBI), l'Algorithme d'Insertion Séquentielle (SBI), l'Algorithme d'Insertion Parallèle Pilote (Pilot PBI) et l'Algorithme d'Insertion Séquentielle Pilote (Pilot SBI). A cela s'ajoutent des stratégies plus sophistiquées, telles que l'approche évolutive i.e., l'Algorithme Génétique (GA), et les recherches locales itératives à démarrage multiple, qui sont adaptées pour résoudre le RVRP. Par exemple, une Recherche Locale Itérative (ILS), une ILS à Démarrage Multiple (MS-ILS) et une MS-ILS basée sur des Tours Géants (MS-ILS-GT) converties en tournées faisables via une procédure de découpage lexicographique ont été développées. Un test reposant sur 42 instances ayant jusqu'à 100 clients, 20 véhicules et 30 scénarios est généré. Des expérimentations numériques sont effectuées pour évaluer la formulation mathématique, les heuristiques constructives et les méta-heuristiques proposées.

- Finalement, le Chapitre 5 étend le RVRP à un problème d'optimisation multi-objectif, nommé bi-RVRP, avec incertitude en demandes et en temps de transport. Les paramètres incertains sont gérés dans la fonction objective en suivant le critère d'optimisation min-max. Les demandes incertaines par client sont modélisées par un ensemble de scénarios représentant les écarts à partir d'une demande attendue. Les temps de transport incertains sont indépendants des demandes des clients. Ensuite, les statistiques ou historiques de circulation sont considérés pour obtenir des scénarios discrets pour chaque arc du réseau de transport. Ce problème trouve des applications dans le transport urbain, par exemple pour livrer des petits commerces. Le bi-RVRP est résolu en développant différentes versions de l'Algorithme Génétique avec tri par non-domination (NSGAII) et l'Algorithme Evolutif Multi-objectif (MOEA). Deux ensembles d'instances sont testés en supposant une perturbation globale sur la demande totale des clients et une variation individuelle sur la demande définie dans un intervalle.

L'ensemble des instances contient 24 problèmes ayant jusqu'à 50 clients, 5 véhicules, 30 scénarios pour les temps de transport et 5 scénarios pour les demandes. Différentes métriques sont utilisées pour mesurer l'efficience de ces algorithmes, leur convergence, ainsi que la diversité des solutions obtenue.

Chapitre 2 - La Robustesse dans l'Optimisation

Dans ce chapitre, les concepts généraux sur la robustesse et une révision des différents Critères d'Optimisation Robuste (ROC) sont rappelés. Puis, deux Critères de Classifications Robustes (RCC) sont proposés et peuvent être appliqués pour mesurer la robustesse des scénarios et solutions. Ils constituent notre première contribution à la thèse. Les applications existantes sur l'optimisation robuste aux problèmes de tournées de véhicules sont présentées au Chapitre 3.

Définitions Générales pour l'Optimisation Robuste

Le mot "robuste" a différents sens dans l'optimisation. Dans la pratique, les données pour un problème d'optimisation peuvent être incertaines, inexactes ou susceptibles de changer dans le futur. Une solution optimale calculée en utilisant ce genre de paramètres peut être fortement affectée par des perturbations, devenant alors sous-optimale ou même infaisable. Ce que les décideurs appellent *solution robuste* est une solution résistant au maximum à de telles perturbations.

La robustesse peut être principalement adressée via l'optimisation stochastique lorsque l'incertitude a une description probabiliste. La programmation stochastique a été introduite par Dantzig & Ramser (1959) et deux méthodes ont été largement appliquées pour résoudre les problèmes stochastiques : la programmation par contraintes probabilistes et la programmation stochastique avec recours. Dans la programmation par contraintes probabilistes, la décision résultante garantit une probabilité donnée respectant les contraintes, i.e, le niveau de confiance de la faisabilité. Pour plus d'information, les lecteurs peuvent se référer à Charnes & Cooper (1959). Dans la programmation stochastique avec recours, quelques contraintes de faisabilité sont relaxées et incluses dans la fonction objective, supposant que des violations induites par des événements aléatoires après l'implémentation des décisions de la première étape peuvent être réparées par des actions récursoires. De bonnes références sur l'optimisation robuste sont Birge & Louveaux (1997) et Cordeau *et al.* (2007) pour des articles de synthèse sur le thème, Bianchi *et al.* (2009) pour les méta-heuristiques dans l'optimisation combinatoire stochastique. Les modèles stochastiques sont puissants mais ont deux inconvénients principaux : les distributions de probabilité sous-jacentes doivent être connues et les solutions peuvent devenir irréalisables pour certaines réalisations d'événements aléatoires.

L'optimisation robuste (RO) est une approche qui vise à traiter l'incertitude en évitant ces inconvénients. Cette approche n'est plus stochastique mais plutôt déterministe et basée sur un ensemble de données. Selon Bertsimas *et al.* (2011), "*au lieu de tenter de protéger la solution dans un certain sens probabiliste à l'incertitude stochastique, le décideur cherche une solution faisable pour toute réalisation de l'incertitude dans un ensemble donné*". En général, le but est d'optimiser la valeur du pire cas sur l'ensemble des données incertaines.

Selon Kouvelis & Yu (1997), des décisions importantes lors de l'application de la RO sont : (i) la manière de modéliser les données incertaines, (ii) la sélection de Critères d'Optimisation Robuste (ROC) appropriés tels que le min-max, l'écart min-max, l'écart min-max relatif, l'α-robustesse, la bw-robustesse, la pw-robustesse, et (iii) le choix d'un modèle mathématique et de méthodes pour générer des solutions robustes. À propos de la façon de structurer les données incertaines, la représentation la plus commune est donnée par un ensemble convexe, e.g., un polyèdre, un cône, ou un ellipsoïde (Ben-Tal *et al.* (2009b); Bertsimas *et al.* (2011)), ou par une affection de valeurs plausibles pour chaque paramètre du modèle, où deux manières communes de modéliser sont l'intervalle ou les scénarios discrets. Concernant un modèle et une solution robustes, des définitions intéressantes sont introduites par Mulvey *et al.* (1995) et Kouvelis & Yu (1997).

Une branche de la RO étudie des programmes mathématiques pour lesquels des approches traitables robustes peuvent être obtenues et développe des outils computationnels, voir par exemple Ben-Tal *et al.* (2009b) pour une revue. D'autres auteurs comme Kouvelis & Yu (1997) ou Aissi *et al.* (2005) considèrent les problèmes d'optimisation combinatoire et recherchent la complexité algorithmique de leurs versions robustes. Par exemple, résoudre un problème du plus court chemin dans un graphe est un problème polynomial, alors que la version robuste avec deux scénarios pour les coûts des arcs est NP-hard, de même dans des réseaux en couches (Yu & Yang (1998)). En dépit de leur difficulté, les problèmes NP-hard commencent à être résolus par la RO.

Les problèmes discrets NP-hard de la RO peuvent également être résolus en utilisant des approches exactes classiques comme le branch-and-cut ou la décomposition de Dantzig-Wolfe, voir les exemples dans Candia-Véjar *et al.* (2011) et Coco *et al.* (2014b). Un flux important de la recherche développe des heuristiques avec des garanties de performance (Aissi *et al.* (2005, 2009); Kasperski *et al.* (2012)). Étonnamment, peu de méta-heuristiques ont été développées. Par exemple, Nikulin (2008) a conçu un algorithme de recuit simulé pour un problème d'arbre de recouvrement. Comme les solutions robustes sont souvent considérées conservatives, Bertsimas & Sim (2003) proposent de limiter le nombre de paramètres incertains autorisés à dévier de leurs valeurs nominales à un nombre Γ, appelé *budget d'incertitude*. Appliquée aux coefficients de coûts et de contraintes d'un programme linéaire en nombres entiers ou mixtes, leur approche conduit à une version robuste avec une augmentation modérée en taille. En particulier, un programme linéaire 0–1 avec des incertitudes limitées aux n coefficients de coûts peut être traité pour résoudre au plus $n+1$ instances du problème d'origine.

Critères d'Optimisation Robuste

En ce qui concerne les critères de robustesse, les décisions peuvent être faites selon l'un des ROC suivantes : min-max, écart min-max, écart min-max relatif, min-max lexicographique, α-robustesse, bw-robustesse et pw-robustesse, etc., et leur utilisation dépend principalement de l'application et des objectifs d'optimisation. Ci-dessous une revue de ces critères et quelques cas pratiques sont fournis sur la base du travail que nous avons mené avec Coco *et al.* (2014a).

Les critères de la famille min-max, i.e., le min-max absolu, l'écart min-max et l'écart min-max relatif, sont généralement désignés comme critères conservatifs car ils visent à minimiser les pertes possibles chaque fois que le scénario du pire cas se produit (Kouvelis & Yu (1997)). Le critère min-max a été initialement développé pour la théorie des jeux par Von Neumann (1928) pour un jeu à somme nulle à deux joueurs. Le problème dual du min-max repose sur la maximisation du gain minimum, nommé max-min. Tant le min-max et le max-min sont utilisés chaque fois que le scénario du pire cas peut impliquer un impact majeur. En extension du travail de Von Neumann (1928), un modèle de prise de décision non-probabiliste basé sur le pire cas d'une décision est proposé par Wald (1939). Selon la définition de l'auteur, les décisions sont classées en fonction de leurs solutions du pire cas. Cette stratégie supprime le scénario du pire cas parmi les décisions possibles dans un processus d'optimisation. En réalité, le critère du min-max est l'un des ROC les plus étudiés et a été appliqué à différents problèmes (Kouvelis & Yu (1997); Ben-Tal & Nemirovski (2002); Kasperski (2008)), tels que les situations de compétition, des décisions risquées ponctuelles, le plus court chemin robuste (Yu & Yang (1998); Karasan *et al.* (2001); Montemanni *et al.* (2004)), et l'arbre de recouvrement minimal robuste (Yaman *et al.* (2001); Montemanni & Gambardella (2005); Kasperski & Zieliński (2011)), entre autres.

Un autre ROC est le min-max lexicographique introduit par Dresher (1961), qui prolonge les travaux de Von Neumann connu comme le nucléole d'un jeu matriciel dans la théorie des jeux. L'idée consiste à sélectionner un sous-ensemble de stratégies optimales, sur la base de la solution optimale du min-max, qui tire parti des erreurs des joueurs adversaires. Dans le min-max lexicographique, non seulement le pire cas est minimisé, comme dans le min-max classique, mais aussi le second pire cas, le troisième pire, etc. Par conséquent, le min-max lexicographique affine le concept de solution dans l'approche min-max, car il sélectionne un ensemble unique de résultats, mais pas nécessairement une solution unique (s'il y a plus d'une solution, toutes les solutions sélectionnées ont la même distribution).

En extension du travail de Dresher (1961), Orgryczak (1997) a étudié le min-max lexicographique pour améliorer le min-max lorsque les données incertaines sont modélisées avec des scénarios discrets. L'auteur a également montré que le min-max lexicographique est conforme à la fois avec le "Principe d'Optimalité du Pareto" et le " Principe des Transferts", alors que le min-max standard peut violer ces deux principes. Conformément au "Principe d'Optimalité de Pareto", la fonction objective est traitée de la même manière sans préférences et/ou des hypothèses spécifiques, et le "Principe des Transferts" est communément reconnu comme l'axiome essentiel pour la mesure de l'équité (Mandell (1991)). Enfin, Orgryczak (1997) a montré que les solutions obtenues par le min-max lexicographique sont meilleures que celles obtenues par le min-max sur les problèmes de localisation. Le min-max lexicographique a été utilisé sur les problèmes d'allocation (Luss (1999); Wang *et al.* (2004)), les problèmes de localisation (Ogryczak (1999); Narasimhan *et al.* (2005)), et le placement de noeud de capteur (Abidin & Din (2012)), ainsi qu'il a été modélisé et intégré à la programmation mathématique (Kostreva *et al.* (2004); Hooker & Williams (2012)).

Un ROC alternatif au min-max utilisant des scénarios discrets est présenté par Bertsimas & Sim (2003). Les auteurs ont proposé un modèle de programmation entier robuste qui permet de contrôler le degré de conservatisme d'une solution en utilisant les limites probabilistes et les violations de contraintes. En outre, Bertsimas & Sim (2003) ont proposé un algorithme pour flux de réseau robuste utilisant leur modèle. Le min-max avec scénarios discrets a été principalement testé sur les problèmes de localisation d'installation (Averbakh & Bereg (2005)), le problème de collecte de prix de l'arbre de Steiner robuste (Álvarez-Miranda *et al.* (2013)), le problème du sac-à-dos robuste (Monaci *et al.* (2013)) et le problème de chargement de réseau robuste avec routage dynamique (Mattia (2013)).

L'α-robustesse a été proposé par Kalaï *et al.* (2012) pour être moins conservateur. Il étend le min-max lexicographique et est appliqué à chaque fois que les données incertaines sont modélisées en utilisant des scénarios discrets. Sur ce critère, un paramètre α définit un seuil de tolérance pour limiter le degré de conservatisme d'une solution. Tant le min-max lexicographique et l'α-robustesse classent les solutions en considérant les pires scénarios. Concernant l'α-robustesse, un vecteur de solutions, appelé "solutions idéales" est défini. Une solution idéale est la meilleure solution obtenue pour un scénario, et elle est utilisé pour calculer les déviations. Donnant une solution ω, une solution idéale ω^*, et un ensemble de scénarios S, la différence entre le coût de ω et ω^* dans chaque scénario $k \in S$ est évaluée et la déviation maximale est maintenue. Par exemple, sur un problème avec deux scénarios, si les solutions idéales pour tous les scénarios sont données dans le vecteur $cost(\omega^*) = (10, 8)$ et le coût d'une solution ω dans le vecteur $cost(\omega) = (15, 10)$, alors la déviation maximale entre $cost(\omega)$ et $cost(\omega^*)$ est égale à 5. Récemment, ce critère a été utilisé pour traiter les évaluations d'attributs incertains dans les méthodes de surclassement (Durbach (2014)) qui construisent une liste de préférences suivant un ensemble d'alternatives prédéterminées.

Un autre ROC appelé bw-robustesse est proposé par Roy (2010). L'optimisation robuste est définie dans son travail comme la capacité à prévenir des impacts indésirables lorsque des données incertaines sont traitées. Ensuite, une valeur non-négative b établit un objectif, en termes de coût, qui doit être atteint ou amélioré dans le plus grand nombre de scénarios tandis que w définit la valeur du pire coût devant être garanti, quel que soit les scénarios ($w < b$). Le bw-robustesse est utilisé uniquement lorsque les scénarios correspondent à des valeurs discrètes et que leur nombre est grand. De même que pour les critères de la famille min-max, le bw-robustesse peut être adressé comme bw-robustesse absolu, bw-déviation absolu et bw-déviation relatif. Le bw-robustesse a été implémenté pour le problème du plus court chemin robuste (Gabrel *et al.* (2011)) et le problème de planification de mission militaire robuste (Tang *et al.* (2011)), qui consiste à allouer des ressources et à planifier des tâches au cours d'une mission militaire, afin de protéger les frontières de la ville.

Le critère du pw-robustesse proposé par Gabrel *et al.* (2013) est une extension du critère bw-robustesse. Une solution est dite robuste chaque fois qu'elle assure une valeur w pour tous les scénarios, et si elle atteint une valeur b dans un pourcentage p des scénarios.

Chapitre 3 - État de l'art

Comme une vaste littérature est consacrée aux problèmes de tournées de véhicules, cette section a été divisée en deux parties. La première cite quelques articles-clés sur le Problème de Tournées de Véhicules avec Capacité (CVRP) et ses méthodes de résolution, tandis que la seconde partie examine les Problèmes de Tournées de Véhicules (VRPs) avec incertitudes.

Problèmes de Tournées de Véhicules avec Incertitudes

Les deux principaux types de VRP avec incertitudes sont le problème de tournées de véhicules stochastique (SVRP) et le problème de tournées de véhicules robuste (de RVRP).

Comme le SVRP ne fait pas parti de ce travail, seulement quelques références sont présentées sur cette approche et une revue plus complète pour le RVRP est introduite.

Problèmes de Tournées de Véhicules Stochastiques

Nombreux sont les paramètres du problème du VRP qui dans la pratique ont un degré élevé d'incertitude. De bons articles sur les problèmes de tournées de véhicules stochastiques sont présentés par Gendreau *et al.* (1996); Bertsimas & Simchi-Levi (1996). Les cas les plus étudiés impliquent des clients stochastiques, où un client a besoin d'être desservi avec une probabilité donnée (Bertsimas (1988); Waters (1989)); des temps de trajet stochastiques, dans lesquels les temps de service ou de trajet sont modélisés par des variables aléatoires (voir, par exemple, Laporte *et al.* (1992); Kenyon & Morton (2003); Verweij *et al.* (2003)); et des demandes stochastiques, où les demandes des clients sont connues en tant que distributions de probabilité (Secomandi (2000); Laporte *et al.* (2002); Christiansen & Lysgaard (2007); Secomandi & Margot (2009); Sun (2014)). Une contribution importante à l'étude du Problème de Tournées de Véhicules avec Demandes Stochastiques (VRPSD) vient de Bertsimas (1992). Ce travail illustre la méthode avec différentes politiques de recours pour résoudre le VRPSD et dresse plusieurs bornes, des résultats asymptotiques et d'autres propriétés théoriques. Bertsimas & Simchi-Levi (1996) synthétisent le développement du VRPSD en mettant l'accent sur les connaissances apportées par les algorithmes proposés. Outre le cadre de la programmation stochastique conventionnelle, un processus de décision Markovien pour une seule étape et des modèles stochastiques multi-étapes sont introduits pour le VRPSD dans Dror (1993). Comme il a été déjà mentionné précédemment, les distributions de probabilité qui modélisent avec précision les incertitudes doivent être connues pour utiliser l'approche d'optimisation stochastique. Un article récent et intéressant de Solano-Charris *et al.* (2015) entrevoient des règles possibles pour les temps de trajet stochastiques.

Problèmes de Tournées de Véhicules Robustes

Dans la littérature, l'optimisation robuste est considérée sur les VRPs principalement pour gérer l'incertitude sur des fenêtres de temps, des temps de trajet, des coûts de trajet ou des demandes, donnant le RVRP. La Table 3.1 résume les travaux sur le RVRP avec des colonnes correspondant aux auteurs, aux méthodes, à la représentation de l'incertitude, et au modèle de données incertaines prenant en compte la localisation de l'incertitude dans le modèle mathématique.

En ce qui concerne les RVRPs, le cas des demandes incertaines est le plus étudié. Sungur *et al.* (2008) a été le premier à considérer ce problème. Dans leur travail, des vecteurs de demandes sont construits à partir d'une perturbation sur la valeur de demande attendue qui appartient à différents ensembles bornés. Leur formulation robuste gère les demandes incertaines dans les contraintes, suivant une approche robuste introduite par Ben-Tal & Nemirovski (1998). Pour résoudre le problème, les auteurs utilisent le VRP basé sur le branch-and-cut open source de la librairie SYMPHONY. Les expérimentations adressent trois différents ensembles d'instances, i.e., aléatoires, clustérisés et modifiés de la littérature, dans un rang de 15 à 100 clients, et démontrent que des solutions robustes peuvent protéger de la demande insatisfaite, mais augmentent un coût additionnel faible comparé à la version déterministe.

Moghaddam *et al.* (2012) ont considéré l'optimisation robuste pour traiter avec les distributions incertaines des demandes de clients. Un pourcentage de perturbation à partir des demandes nominales est défini sous les contraintes, dont l'objectif est de minimiser les distances des trajets. Une variante de l'Optimisation Par Essaims Particulaires est présentée et les résultats sont comparés avec ceux obtenus par Sungur *et al.* (2008).

Le VRP avec des flottes hétérogènes et des demandes incertaines est étudié par Noorizadegan *et al.* (2012). Les auteurs fournissent des formulations mathématiques pour la version robuste et pour une approche de programmation à contraintes probabilistes, tous les deux avec des incertitudes restrictives dans la partie droite des contraintes. Un algorithme B&C est considéré pour ajouter des plans de coupe qui réduisent la région polyédrique. Des instances avec 20 clients en utilisant CPLEX sont testées et les résultats sont analysés utilisant le coût extra requis pour atteindre un certain niveau de routes réalisables, de nombre de clients non satisfaits et de coût de recours.

Une autre étude du RVRP avec incertitudes sur les demandes est présentée dans Gounaris *et al.* (2013). Les Inégalités de Capacités Arrondies Robustes (RCI) sont développées. Les modèles gèrent les demandes de clients en tant que variables aléatoires sur le côté droit des contraintes et déterminent le plan de coût minimum de livraison qui est faisable pour toute réalisation anticipée de demandes. CPLEX est utilisé pour résoudre 90 instances de 15 à 135 clients : ses coupes génériques sont désactivées et remplacées par des coupes RCI. Les résultats montrent que la meilleure formulation robuste, soient les Deux Formulations de Flux de Véhicules indexés, est améliorée encore davantage si les coupes RCI sont utilisées : la plupart des instances ayant jusqu'à 50 noeuds sont résolues dans l'optimalité et l'écart moyen pour les autres instances est en dessous de 5%.

Le VRP Ouvert avec demandes incertaines est introduit par Cao *et al.* (2014). Un ensemble incertain limité de demandes de clients est décrit et des coûts de transportation et des demandes insatisfaites dans l'ensemble incertain borné spécifique sont minimisés. Un algorithme d'évolution différentielle est proposé et ses performances sont analysées pour différentes stratégies, en considérant les coûts supplémentaires et la demande insatisfaite.

Le travail le plus récent concernant les demandes incertaines est présenté dans Gounaris *et al.* (In press). La formulation robuste autorise des demandes incertaines de clients et l'objectif est de déterminer un ensemble de routes à moindre coût qui reste faisable pour toutes les réalisations des demandes dans un ensemble prédéfini d'incertitudes. Les auteurs ont implémenté une méta-heuristique de programmation à mémoire adaptive en utilisant deux classes d'ensembles incertains. Des expérimentations numériques sur des instances de référence ayant jusqu'à 483 clients et 38 véhicules sont récupérées et de nouvelles meilleures solutions pour un total de 123 instances de référence sont trouvées.

Tel qu'il est présenté dans Solano-Charris *et al.* (2014) et comme il peut être vu dans la Table 3.1, seuls quelques travaux ont traité les temps de trajet incertains ou les coûts de trajet. Une approche de scénario robuste pour le problème de tournées de véhicules avec temps de trajet incertains est étudiée par Han *et al.* (2013). Des classes de prévisions multiples dans un ensemble d'intervalles de temps sont présumées. L'incertitude est gérée en limitant le nombre de paramètres incertains sur les contraintes permises de déviations sur les valeurs nominales (Bertsimas & Sim (2003)). Ensuite, pour chaque réalisation (scénario) sur un intervalle de temps de trajet, une route robuste est identifiée, et la minimisation du pire cas parmi tous les scénarios est appliquée. Un programme stochastique avec recours à deux étapes résolu par un algorithme B&B est utilisé. Des tests sur des instances ayant jusqu'à 25 clients sont résolus en supposant des conditions normales et perturbées de trafic.

Les travaux de Toklu *et al.* (2013a,b) traitent le VRP avec coûts de trajet incertains. Le coût total de trajet est minimisé et l'incertitude est exprimée comme des intervalles. L'approche de Bertsimas & Sim (2003) (voir la Section 2.1) est implémentée pour contrôler le degré d'incertitude sur le modèle. Un système de colonie de fourmis et un système de colonie de fourmis multiple sont introduits dans Toklu *et al.* (2013a,b), respectivement, et sont testés sur des instances ayant jusqu'à 150 clients avec différentes configurations de degré de conservatisme.

Une approche pour le RVRP avec temps de trajet incertains a été étudiée récemment par Solano-Charris *et al.* (2015). L'ensemble des coûts des arcs est remplacé par un ensemble de scénarios discrets et l'objectif principal est de construire un ensemble de routes en utilisant le critère min-max lexicographique de Orgryczak (1997). Ainsi, le pire coût sur l'ensemble des scénarios est minimisé mais des liens sont supprimés en utilisant les autres scénarios, du pire au meilleur. Différentes méthodes sont adaptées, i.e., une Procédure Adaptée Randomisée de Recherche Gloutonne (GRASP), une Recherche Locale Itérative (ILS), une ILS à Démarrage

Multiple (MS-ILS), une MS-ILS basée sur des Tours Géants (MS-ILS-GT). Des tests sur des instances ayant jusqu'à 100 clients et 30 scénarios sont rapportés.

En termes de fenêtres de temps, Agra *et al.* (2013) ont adressé le VRP sans capacités élevé par la transportation maritime. Les coûts de trajet et les temps sont asymétriques et dépendent du navire utilisé. L'objectif est de trouvé un ensemble de routes à moindre coût respectant les fenêtres de temps et la faisabilité pour tous les temps de trajet dans un polytope incertain donné. Deux nouvelles formulations sont introduites et résolues par des algorithmes de décomposition ad-hoc, rendant le modèle plus efficace si l'approche du "budget d'incertitude" de Bertsimas & Sim (2003) est utilisé. Des instances avec 20 à 50 noeuds peuvent être résolues sur un PC de 2.5GHz.

Concernant des paramètres multiples sous incertitudes, le travail de Ordóñez (2010) a considéré des demandes incertaines, des temps de trajet, des coûts et des clients. L'auteur souligne différents modèles robustes dépendant de la source d'incertitude, la formulation du VRP, et la corrélation entre les coefficients incertains. L'incertitude sur les coûts de trajet est introduite dans la fonction objective, et les demandes incertaines et les coefficients de temps de trajet dans les contraintes. Un ensemble convexe et borné d'incertitudes est estimé et le problème est résolu via l'approche d'optimisation robuste de Ben-Tal & Nemirovski (1998). Des résultats pour des instances de petite taille sont fournis et comparés avec ceux obtenus par une méthode de programmation avec contraintes probabilistes et des modèles basés sur la programmation stochastique avec recours.

Finalement, des incertitudes sur les temps de trajet ainsi que sur les demandes sont trouvées dans Lee *et al.* (2012) pour un VRP avec échéances des clients. L'approche sur le budget d'incertitude de Bertsimas & Sim (2003) est définie en délimitant la somme des perturbations sur les temps de trajet, et la perturbation des demandes sur leurs valeurs nominales. La robustesse d'une solution est atteinte en cherchant une solution faisable pour tout temps de trajet et toute demande définis sur les ensembles d'incertitude, minimisant le temps de trajet. Une formulation d'ensemble de partitionnement est proposée et résolue par la génération de colonnes. Les incertitudes sont limitées au sous-problème de génération de colonnes, en utilisant une version robuste de l'algorithme du plus court chemin avec contraintes sur les ressources. Des instances ayant 20 à 40 clients sont résolues dans l'optimalité.

Chapitre 4 - Le Problème de Tournées de Véhicules Robuste avec Temps de Trajets

Incertains

Le RVRP considéré ici adresse une incertitude sur les coûts de trajet, modélisés par des scénarios discrets, dans lesquels chaque scénario définit un temps de trajet pour chaque arc. L'objectif est de minimiser le pire coût (durée totale de la route) sur l'ensemble des scénarios, en utilisant une méthode lexicographique pour supprimer des liens. Ces scénarios ne correspondent pas à des réalisations de variables aléatoires et ne sont pas associés à des probabilités d'occurrence. Dès lors, une question essentielle est la modélisation avec précision des temps de trajet. La plupart des auteurs utilisent des distributions de probabilité additives, e.g., normales tels que Kenyon & Morton (2003) ou les lois gamma comme Taş *et al.* (2013). Puisque les temps de trajet augmentent souvent rapidement d'un minimum à un maximum et ensuite diminue lentement avec une longue traîne, une distribution log-normale semble pertinente (Lecluyse *et al.* (2009)). Selon Gómez *et al.* (In press), une distribution précise ne peut pas exister et une spécifique doit être choisie pour chaque problème et même chaque arc. Ces auteurs proposent d'utiliser des distributions phase-type, qui peuvent approximer n'importe quelle distribution positive et continue, en calculant exactement leurs convolutions.

Concernant le réalisme d'utilisation de scénarios, ils peuvent être apportés par des centres de contrôle du trafic implémentés dans les grandes villes, avec deux avantages : ils correspondent à des données réelles et reflètent des corrélations possibles parmi les coûts des arcs. Des solutions intéressantes pourraient être obtenues si un nombre suffisant de registres de circulation est considéré. Le prix à payer est un temps d'exécution multiplié par $O(q \log q)$ pour q scénarios, comme il peut être vue dans la Section 4.4. La confirmation de l'exactitude de l'ensemble des modèles traitant des temps de trajet incertains requerrait des validations à grande échelle sur des réseaux de routes réels pour voir le coût réel atteint sur le terrain.

Le problème proposé peut être trouvé dans la ville de Troyes, où la municipalité envoie des techniciens pour remplacer des ampoules endommagées de lampadaires publics. Des routes sont préparées en utilisant des temps de trajet moyens mais, à cause des conditions de trafic, la durée actuelle des routes est souvent plus large (voir la Figure 4.1 par exemple). Le temps de travail quotidien maximum n'est pas excédé, mais les incertitudes retardent l'allocation des techniciens à d'autres tâches par la suite, comme des réparations dans des bâtiments publics.

Un point fort de l'approche du RVRP est qu'il peut être vu comme un problème multi-objectif, chaque objectif étant le coût total pour un scénario. Un optimum pour l'objectif min-max lexicographique est aussi le Pareto-optimal pour cette version multi-objectif (Orgryczak (1997)). Comme aucune autre solution ne la domine, c'est attirant pour un décideur.

Ce chapitre est structuré comme suit : les Section 4.2 et 4.3 présentent la définition formelle et le critère de robustesse ici implémenté pour le RVRP avec temps de trajet incertains. Les heuristiques gloutonnes proposées sont introduites dans la Section 4.4. La Section 4.5 est dédiée à une méta-heuristique à base de population, pendant que la Section 4.6 décrit les différents composants des stratégies itératives à démarrage multiple. Pour toutes les méthodes ici adaptées, les expérimentations numériques et les conclusions sont fournies.

Définition du Problème

Le RVRP adressé ici est motivé par des temps de trajet incertains induits par les conditions du trafic. Comme les deux directions d'une route peuvent être affectées distinctement lorsque le trafic est perturbé, un graphe complet et orienté $G = (N, A)$ est considéré, tandis que le CVRP est défini sur un réseau non-orienté. N dénote l'ensemble des noeuds, avec un dépôt (noeud 0) et n clients avec des demandes positives d_i, pendant que $A = \{(i,j) \mid i, j \in N, i \neq j\}$ se dresse pour l'ensemble des arcs. Une flotte de m véhicules identiques avec une capacité de Q est basée au dépôt.

Au lieu d'être des variables aléatoires, les coûts des arcs incertains sont modélisés par un ensemble de q scénarios discrets $S = \{1, 2, ...q\}$, où chaque scénario k définit un coût non-négatif c_{ij}^k pour chaque arc $(i,j) \in A$. Soit Ω définissant l'ensemble des solutions faisables. Comme dans le CVRP, une solution $\omega \in \Omega$ est un ensemble de routes : chaque route est couverte par un véhicule qui quitte le dépôt, dessert un sous-ensemble de clients dont la demande totale n'excède pas Q, et retourne au dépôt. En particulier, pour le RVRP le but principal est de construire un ensemble de routes considérant la minimisation du pire coût total sur tous les scénarios. Puisque le CVRP, connu pour être NP-hard, correspond au RVRP avec un scénario ($q = 1$), the RVRP est alors aussi NP-hard.

Le RVRP peut être spécifié par le MILP min-max suivant donné de l'Equation A.1 à A.9 dérivées d'un modèle du CVRP proposé par Kulkarni & Bhave (1985). Ce modèle compacte évite les indexes des véhicules : les routes sont définies par des variables binaires x_{ij}, égales à 1 si et seulement si un arc (i,j) est utilisé dans la solution. Dans (A.2), le coût total des arcs utilisé pour chaque scénario est majoré par la variable δ, définie dans la contrainte (A.9). Le but (A.1) est de minimiser δ. Les contraintes (A.3) et (A.4) signifient que chaque client est atteint par un arc d'entrée et un de sortie. La taille maximale d'une flotte est garantie via les contraintes (A.5). Les inégalités (A.6) et (A.7) généralisent le modèle de Miller *et al.* (1960) appelé contraintes MTZ pour l'élimination des sous-tours pour le Problème de Voyageurs de Commerce (TSP). Supposant que d_i est une quantité collectée, l_i est le chargement du véhicule quand on quitte i. Si l'arc (i, j) n'est pas utilisé, la contrainte (A.7) pour cet arc devient $l_i - l_j \leq Q - d_j$ et est trivialement satisfaite : comme $l_i \leq Q$ et $l_j \geq d_j$ via (A.6), le côté gauche ne peut pas excéder $Q - d_j$. Considérons maintenant une route $(r_1, r_2, \ldots, r_u)$, qui signifie que $x_{r_i,r_{i+1}} = 1$ pour $i = 1, 2, \ldots, u-1$: les contraintes (A.6) et (A.7) tiennent en définissant par exemple $l_{r_1} = 0$ et $l_{r_i} = l_{r_{i-1}} + d_{r_i}$ pour $i = 2, 3, \ldots, u$. La capacité du véhicule est aussi respectée depuis $l_{r_u} \leq Q$ de (A.6). Les variables binaires sont déclarées dans (A.8). Les autres équations mènent à une version min-max.

$$
\begin{aligned}
\min\ z = \delta & & \text{(A.1)}\\
\sum_{(i,j)\in A} c_{ij}^k x_{ij} \leq \delta & \quad \forall\, k \in S & \text{(A.2)}\\
\sum_{j\in N} x_{ji} = 1 & \quad \forall\, i \in N\backslash\{0\} & \text{(A.3)}\\
\sum_{j\in N} x_{ij} = 1 & \quad \forall\, i \in N\backslash\{0\} & \text{(A.4)}\\
\sum_{i\in N\backslash\{0\}} x_{0i} \leq m & & \text{(A.5)}\\
d_i \leq l_i \leq Q & \quad \forall\, i \in N\backslash\{0\} & \text{(A.6)}\\
l_i - l_j + Q\, x_{ij} \leq Q - d_j & \quad \forall\, (i,j) \in A, i \neq 0, j \neq 0 & \text{(A.7)}\\
x_{ij} \in \{0,1\} & \quad \forall\, (i,j) \in A & \text{(A.8)}\\
\delta \geq 0 & & \text{(A.9)}
\end{aligned}
$$

Il est notifié que les contraintes suivantes pour le CVRP peuvent être utilisées pour résoudre le MILP sensiblement plus rapide dans la pratique. Une borne inférieure simple pour le nombre de véhicules requis est définie par la contrainte (A.10), où d_{tot} indique la demande totale. Desrochers et Laporte ont proposé de remplacer les contraintes MTZ (A.7) par les inégalités levées (A.11), qui étaient écrites incorrectement dans leur papier. Ici la version corrigée par Kara *et al.* (2004) est considérée.

$$
\begin{aligned}
\sum_{i\in N\backslash\{0\}} x_{0i} \geq \lceil d_{tot}/Q \rceil & & \text{(A.10)}\\
l_i - l_j + Q\, x_{ij} + (Q - d_i - d_j)\, x_{ji} \leq Q - d_j & \quad \forall\, (i,j) \in A, i \neq 0, j \neq 0 & \text{(A.11)}
\end{aligned}
$$

La Figure 4.1 montre un exemple avec 16 noeuds et 2 scénarios, résolu en utilisant CPLEX. Les coûts des arcs sont égaux aux distances Euclidiennes. Le coût de la solution du CVRP sans incertitudes est 439 (gauche). Sur la droite, les deux scénarios (matin et soir) augmentent les distances de 20% sur les mêmes arcs. Notre modèle donne un coût de 464 sur les deux scénarios, seulement 5.7% plus élevé que la solution du CVRP. La mauvaise idée (pas encore considérée par beaucoup) est d'utiliser les déplacements de la solution déterministe et de recalculer leurs coûts en se servant des coûts des arcs des deux scénarios, mais dans ce cas, le pire coût sur les deux scénarios est 12% plus élevé. Cela montre l'intérêt d'utiliser une approche d'optimisation robuste.

Chapitre 5 - Le Problème de Tournées de Véhicules Robuste Bi-objectif avec Temps de Trajets et Demandes Incertains

Le Problème de Tournées de Véhicules Robuste bi-objectif avec incertitudes (bi-RVRP) est étudié dans ce chapitre. Les incertitudes sont traitées en demandes et en temps de trajet et sont supposées non corrélées. La demande pour chaque client est modélisée par un ensemble discret de scénarios représentant les déviations de la demande attendue. En ce qui concerne les temps de trajet, des registres de circulation sont considérés pour obtenir des scénarios discrets pour chaque arc du réseau de transportation, comme dans le RVRP du chapitre précédent. Le bi-RVRP vise à minimiser le pire coût total des arcs traversés et à minimiser la demande totale non satisfaite maximum sur l'ensemble des scénarios, au sens de Pareto. Comme il peut être vu dans le Chapitre 3, peu de travaux sont focalisés sur les incertitudes sur des paramètres multiples des VRPs (Lee *et al.* (2012); Ordóñez (2010)) et ont rarement traités des techniques d'optimisation multi-objectif (voir la Table 3.1). Ce problème peut trouver des applications pratiques dans la transportation urbaine, e.g., servant des petits commerces qui dans certains cas ne connaissent pas les quantités exactes nécessaires. Nous supposons que le conducteur découvre la demande réelle en arrivant chez chaque client. Les temps de trajet et les demandes ne suivent aucune distribution de probabilité, à la place, des scénarios représentatifs pour les deux paramètres incertains sont inclus. La robustesse sur les solutions faisables est affectée dépendant des réalisations des demandes et la capacité disponible pour servir le client.

Pour résoudre le problème, différentes versions de méta-heuristiques multi-objectif évolutives sont proposées : l'Algorithme Evolutif Multi-objectif (MOEA) et l'Algorithme Génétique avec Tri par Non-domination version 2 (NSGAII). Contrairement aux versions initiales de ces méta-heuristiques, nous les renforçons en utilisant une procédure de recherche locale, et nous évaluons les emplacements possibles de cette procédure dans la structure de chaque méta-heuristique. Différentes métriques sont utilisées pour mesurer la performance des algorithmes.

Ce chapitre est structuré comme suit : la Section 5.2 présente la définition formelle pour le bi-RVRP avec demandes et temps de trajet incertains. Des algorithmes multi-objectif proposés pour le bi-RVRP sont introduits dans la Section 5.3. La Section 5.4 est dédiée à une procédure de recherche locale et ses intégrations à des heuristiques multi-objectif et les Sections 5.5 et 5.6 décrivent les expérimentations numériques et les conclusions pour le bi-RVRP, respectivement.

Définition du Problème

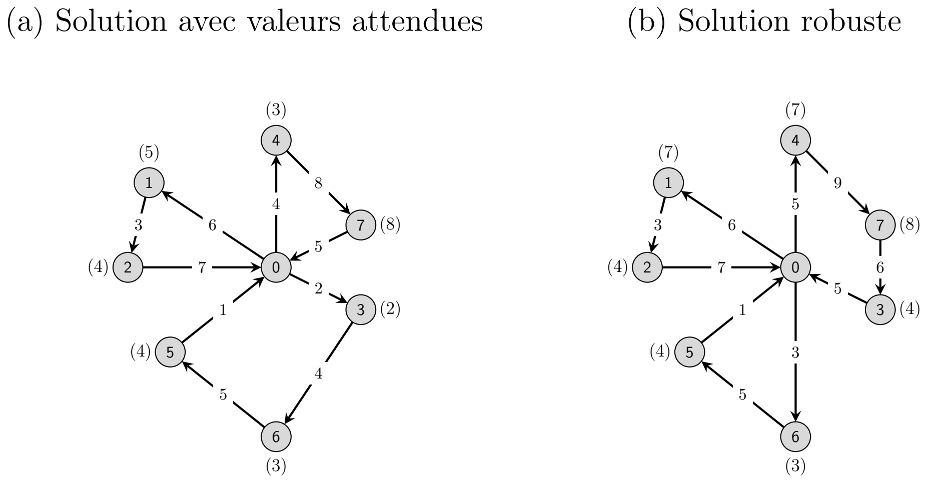
Le bi-RVRP peut être formellement défini par un digraphe complet $G = (N, A)$ avec un ensemble $N = \{0, 1, 2...n\}$ de $n+1$ noeuds, incluant le dépôt (0) et n clients, et un ensemble $A = \{(i, j)|i, j \in N, i \neq j\}$ d'arcs. Des données incertaines pour les temps de trajet sont modélisées comme un ensemble de q scénarios discrets $S = \{1, 2, ...q\}$, où chaque scénario $k \in S$ spécifie un coût $c_{ij}^k \in \mathbb{R}$ pour chaque arc $(i, j) \in A$. Une demande espérée d_i^* est associée à chaque client $i \in N$, et sa variation est définie dans un ensemble $\hbar$ de scénarios discrets $D = \{1, 2, ...\hbar\}$, où chaque scénario $\ell \in D$ spécifie une demande d_i^ℓ pour chaque client i, selon une perturbation sur la demande des clients. Des découpages des livraisons ne sont pas autorisés. Une flotte de m véhicules identiques est disponible au dépôt, chacun avec une capacité égale à Q. Une solution est un ensemble de routes de véhicules démarrant et finissant au dépôt, visitant chaque client une fois et respectant les capacités des véhicules. Le bi-RVRP vise à minimiser le pire coût total des arcs traversés et la demande non satisfaite totale maximum, à travers tous les scénarios. Pour une route planifiée, la politique de service suivante est appliquée lorsque l'on arrive chez un client i sous un scénario ℓ : si la quantité totale déjà servie au client précédent, plus d_i^ℓ, n'excède pas la capacité du véhicule, alors le client est desservi (i.e., la demande est non satisfaite), sinon il n'est pas desservi et la route procède avec le client suivant planifié.

Le bi-RVRP est déclaré dans un espace à 2 dimensions avec le pire coût total des arcs traversés z_1 et la pire demande non satisfaite totale sur tous les scénarios z_2, définis ainsi :

$$z_1 = \min \left(\max_{k \in S} \sum_{(i,j) \in A} c_{ij}^k x_{ij} \right), \quad z_2 = \min \left(\max_{\ell \in D} \sum_{i \in N} d_i^\ell y_i \right)$$

La variable binaire x_{ij} est égale à 1 si l'arc (i, j) est traversé par un véhicule, sinon $x_{ij} = 0$. y_i est une variable booléenne qui spécifie si un véhicule sert un client $y_i = 1$ ou pas $y_i = 0$, selon la capacité du véhicule Q. La relation dominante pour z_1 et z_2 dans le sens de Pareto est définie pour notre bi-RVRP comme suit : une solution η domine une solution ω si $z_1(\eta) \leq z_1(\omega)$ et $z_2(\eta) < z_2(\omega)$, ou $z_1(\eta) < z_1(\omega)$ et $z_2(\eta) \leq z_2(\omega)$. Une solution est non-dominée si aucune autre solution ne la domine. Le front optimal de Pareto est donné par l'ensemble des solutions robustes non-dominées.

Le bi-RVRP est illustré par un exemple simple dans la Figure A.1. Le problème considère un seul dépôt (0) et 7 clients à desservir par 3 véhicules avec une capacité égale à 10. La Figure A.1 montre les distances entre les clients (indiquées sur les flèches) dans des conditions normales de trafic dans la Figure A.1a et avec des temps de trajet perturbés dans la Figure A.1b, tous deux avec la demande par client (entre parenthèses).


(a) Solution avec valeurs attendues

(b) Solution robuste

Figure A.1: Exemples de solutions pour les VRP et bi-RVRP.

Route	Coût (z_1)	Demande Insatisfaite (z_2)
0-1-2-0	16	n.a.
0-4-7-0	17	n.a.
0-3-6-5-0	12	n.a.
Total	45	n.a.

Route	Coût (z_1)	Demande Insatisfaite (z_2)
0-1-2-0	16	4
0-4-7-3-0	25	12
0-6-5-0	9	0
Total	50	16

Table A.1: Résumé des résultats pour le VRP (à gauche) et le bi-RVRP (à droite).

Le récapitulatif des résultats se trouve dans la Table A.1. L'ensemble des routes pour une solution construite dans des conditions normales se sert de la valeur attendue pour les demandes et les temps de trajet, alors qu'en traitant avec les incertitudes, les scénarios perturbés pour des temps de trajet et des demandes sont gérés et le coût total z_1 et la demande insatisfaite z_2 sont calculés. Dans la Figure A.1b, un scénario perturbé est considéré. La solution est composée des trois routes avec les séquences, 0-1-2-0, 0-4-7-3-0, 0-6-5-0, respectivement. Puis, sous les demandes perturbées pour la route 0-1-2-0, le véhicule ne peut pas satisfaire la demande pour le client 2, traitant avec 4 unités de demandes insatisfaites et un coût total de 16. La route 0-4-7-3-0 peut satisfaire uniquement la demande du client 4 laissant 12 unités de demandes insatisfaites et un coût total de 25. Finalement, pour la séquence 0-6-5-0, tous les clients sont desservis et le coût total pour la route est 9. La même évaluation est faite pour le VRP (Figure A.1a), mais dans ce cas, un coût total de 45 est trouvé et tous les clients sont assistés. Pour chaque solution, nous totalisons le coût et la demande insatisfaite obtenant un coût total de 45 et 0 demandes non satisfaites dans le VRP, et un coût total de 50 avec 16 unités sur la demande insatisfaite dans le RVRP.

Chapitre 6 - Conclusions Générales et Orientations Futures de Recherche

Le Problème de Tournées de Véhicules (VRP) est un problème d'optimisation combinatoire bien connu et complexe avec un grand nombre d'applications réelles. Dans cette thèse, nous avons étudié des extensions du VRP avec données incertaines. Plusieurs méthodes sont disponibles pour modéliser des incertitudes, principalement focalisées sur la programmation stochastique où les données incertaines sont modélisées par des variables aléatoires qui ont une distribution de probabilités connue, et l'optimisation robuste qui n'est plus stochastique mais déterministe et basée sur un ensemble de données. Cette thèse est consacrée à des approches robustes d'optimisation du VRPs avec données incertaines, donnant ce que nous appelons le Problème de Tournées de Véhicules Robuste (RVRP). Pour le RVRP, deux principaux problèmes ont été étudiés, un RVRP avec temps de trajet/coûts incertains et une version bi-objectif du RVRP (bi-RVRP) avec temps de trajet et des demandes incertaines. De plus, deux Critères de Classifications Robustes (RCC) pour l'évaluation des solutions et des scénarios ont été introduits principalement comme un moyen de renforcer les procédures de recherche locale ou de combiner ces critères avec les critères d'optimisation robuste existants.

Le RVRP avec temps de trajet incertains est introduit dans le Chapitre 4. Les temps de trajet incertains sont modélisés par des scénarios discrets dans un réseau complet et orienté, dans lequel chaque scénario définit un temps de trajet pour chaque arc à une heure donnée. L'objectif est de minimiser la durée totale du tour à travers tous les scénarios. Pour la solution du RVRP, un modèle mathématique et plusieurs méthodes ont été développés. Les méthodes proposées incluent des versions déterministes et randomisées d'heuristiques constructives, i.e., le Clarke et Wright (CW), le CW Randomisé (RCW), l'Algorithme d'Insertion Parallèle (PBI), l'Algorithme d'Insertion Séquentielle (SBI), l'Algorithme d'Insertion Parallèle Pilote (Pilot PBI) et l'Algorithme d'Insertion Séquentielle Pilote (Pilot SBI). En outre, un Algorithme Génétique (GA) et quatre méta-heuristiques basées sur la recherche locale ont été proposés, i.e., une GRASP, une ILS, une ILS à Démarrage Multiple (MS-ILS) et une MS-ILS avec Tours Géants.

Des expérimentations numériques sur 18 instances de petites tailles et 24 instances de tailles moyennes aléatoirement générées ayant jusqu'à 100 clients et 30 scénarios sont fournies. Les résultats expérimentaux montrent la complexité du RVRP malgré sa déclaration assez simple. Parmi les heuristiques constructives, les méthodes les plus prometteuses en termes de temps d'exécution et d'écart moyen des solutions sont le CW et sa version randomisée. En ce qui concerne le GA, 7 sur 10 valeurs optima trouvées par le CPLEX sont récupérées sur les petites instances. Sur les méthodes de recherche locale, leur efficience a été démontrée en utilisant les instances de petites tailles et plusieurs instances du CVRP avec solution optimale connue. Ces méta-heuristiques ont trouvé des écarts moyens très proches en comparaison avec les résultats donnés par CPLEX. De plus, tous les optima prouvés sont récupérés et trois bornes supérieures sont améliorées. Toutefois, une alternance entre les solutions des tours géants et du RVRP dans le MS-ILS apporte une amélioration statistiquement signifiante en considérant un nombre fixe d'appels à la recherche locale et un budget de temps.

Dans le Chapitre 5, le RVRP est étendu à une version d'optimisation multi-objectif. Des incertitudes sont adressées dans la fonction objective, minimisant le coût total des arcs traversés et la demande totale non satisfaite sur tous les scénarios. Des ensembles d'incertitudes indépendants sont modélisés par des scénarios ou des intervalles représentant des perturbations possibles par rapport aux valeurs attendues. Pour résoudre le bi-RVRP, un Algorithme Génétique avec Tri par Non-domination version 2 (NSGAII) et un Algorithme Evolutif Multi-objectif (MOEA) couplés à une procédure de recherche locale dédiée (non inclue dans la version classique de ces méta-heuristiques) sont proposés.

Plusieurs localisations de la recherche locale et deux critères d'acceptation pour une perturbation ont été testés sur deux ensembles de 24 instances aléatoires assumant une perturbation globale à travers la demande totale des clients et une variation individuelle sur la demande définie dans un intervalle. Les ensembles des instances contiennent 24 instances qui ont jusqu'à 50 clients, 5 véhicules, 30 scénarios pour les temps de trajet et 5 scénarios pour les demandes.

Des analyses sur l'impact du nombre de scénarios et de la procédure de recherche locale ont été effectuées. Concernant l'impact du nombre de scénarios sur les solutions, l'évaluation du bi-RVRP est comparée au CVRP avec 10 et 20 clients et a révélé le prix à payer pour une solution robuste. Malgré cela, dans les cas où le CVRP donne des solutions sans demandes non satisfaites, nos méthodes proposées peuvent trouver des solutions qui supportent les perturbations qui peuvent surgir dans le futur avec un bon compromis entre le coût et la demande non satisfaite. Différentes métriques sont également utilisées pour mesurer l'efficience, la convergence mais aussi la diversité des solutions pour ces méta-heuristiques, i.e., le nombre de solutions classées dans le premier front, l'hypervolume et le spacing. Les résultats montrent que toutes les versions pour notre NSGAII et MOEA avec la procédure de recherche locale réduisent constamment le pire coût et la demande non satisfaite maximale sur le front de Pareto non-dominé, et ont une bonne performance sur les mesures multi-objectif.

En tant qu'orientations futures pour la thèse, une application des RCC pourrait être faite au travers des méta-heuristiques basées sur la recherche locale. Sur le RVRP et le bi-RVRP, il y a une opportunité d'inclure d'autres représentations des incertitudes et d'améliorer quelques-unes des méthodes introduites dans ce travail. L'impact positif du MS-ILS-GT serait intéressant à tester sur d'autres méta-heuristiques telles que les algorithmes génétiques hybrides, bien que cela puisse augmenter la difficulté sur le RVRP dû aux composants multiples qui sont impliqués.

Un inconvénient actuel de nos méta-heuristiques pour un VRP robuste avec q scénarios est un temps d'exécution multiplié par $q \log q$, comparé à la version déterministe. Cela peut conduire à des temps d'exécution excessifs si le décideur souhaite employer de nombreux scénarios (plus de 50 par exemple). En utilisant des structures de données ad-hoc ou des techniques de réduction du voisinage, une réduction substantielle du temps d'exécution (en moyenne) pourrait être espérée.

Le développement d'algorithmes exacts est prometteur, mais semble une tâche difficile. Actuellement, nous travaillons sur un modèle mathématique pour le bi-RVRP pour gérer les caractéristiques spécifiques de notre problème.

Il serait également intéressant d'étendre les méthodes ici proposées pour gérer d'autres critères d'optimisation tels que l'écart min-max et l'écart relatif min-max. Une adaptation à grande échelle du RVRP et du bi-RVRP est aussi envisagée. Finalement, des incertitudes sur d'autres paramètres du CVRP pourraient être adressées e.g., les fenêtres de temps, et la comparaison des résultats fournis par l'approche d'optimisation robuste par rapport à d'autres techniques pour gérer les incertitudes comme la programmation stochastique.

Appendix B

Statistical tests for the RVRP

No.	Instance	GRASP 1-5000			ILS 5000-2-4			MS-ILS 1-10-500-2-4			MS-ILS-GT 1-10-500-2-4		
	n-m-q-β	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	50-5-10-10	8,637.0	3.5	12.3	8,637.0	3.5	12.3	8,636.2	2.0	4.0	8,635.0	1.0	1.0
2	50-5-10-50	9,966.1	4.0	16.0	9,947.9	3.0	9.0	9,948.2	2.0	4.0	9,936.0	1.0	1.0
3	50-5-10-100	11,526.4	2.0	4.0	11,572.8	4.0	16.0	11,542.6	3.0	9.0	11,525.0	1.0	1.0
4	50-5-20-10	7,971.8	2.5	6.3	8,089.9	4.0	16.0	7,972.3	2.5	6.3	7,966.0	1.0	1.0
5	50-5-20-50	10,014.0	2.0	4.0	10,014.0	2.0	4.0	10,015.0	4.0	16.0	10,014.0	2.0	4.0
6	50-5-20-100	10,933.0	2.0	4.0	10,979.8	4.0	16.0	10,933.5	3.0	9.0	10,923.0	1.0	1.0
7	50-10-10-10	10,896.3	4.0	16.0	10,885.2	2.0	4.0	10,889.7	3.0	9.0	10,882.0	1.0	1.0
8	50-10-10-50	12,851.3	2.0	4.0	12,909.9	4.0	16.0	12,859.0	3.0	9.0	12,849.0	1.0	1.0
9	50-10-10-100	16,231.6	3.0	9.0	16,243.0	4.0	16.0	16,219.8	2.0	4.0	16,212.0	1.0	1.0
10	50-10-20-10	11,707.9	4.0	16.0	11,679.1	3.0	9.0	11,630.7	2.0	4.0	11,590.6	1.0	1.0
11	50-10-20-50	14,354.3	3.0	9.0	14,371.0	4.0	16.0	14,349.0	2.0	4.0	14,338.5	1.0	1.0
12	50-10-20-100	15,127.7	3.0	9.0	15,131.3	4.0	16.0	15,105.7	2.0	4.0	15,076.2	1.0	1.0
13	100-10-10-10	12,882.1	2.0	4.0	12,966.7	4.0	16.0	12,935.1	3.0	9.0	12,789.7	1.0	1.0
14	100-10-10-50	15,247.0	2.0	4.0	15,454.1	4.0	16.0	15,345.4	3.0	9.0	15,084.4	1.0	1.0
15	100-10-10-100	18,494.4	2.0	4.0	18,739.5	4.0	16.0	18,700.3	3.0	9.0	18,280.7	1.0	1.0
16	100-10-20-10	13,737.6	2.0	4.0	14,004.0	4.0	16.0	13,835.7	3.0	9.0	13,663.1	1.0	1.0
17	100-10-20-50	16,155.8	2.0	4.0	16,329.9	4.0	16.0	16,273.0	3.0	9.0	15,983.2	1.0	1.0
18	100-10-20-100	18,758.4	2.0	4.0	19,017.7	4.0	16.0	18,848.6	3.0	9.0	18,457.8	1.0	1.0
19	100-20-10-10	22,322.6	2.0	4.0	22,433.7	4.0	16.0	22,423.0	3.0	9.0	22,275.1	1.0	1.0
20	100-20-10-50	25,145.3	4.0	16.0	25,137.0	3.0	9.0	25,077.2	2.0	4.0	24,745.3	1.0	1.0
21	100-20-10-100	29,801.3	4.0	16.0	29,775.4	3.0	9.0	29,566.1	2.0	4.0	29,243.3	1.0	1.0
22	100-20-20-10	20,040.5	3.0	9.0	20,008.2	2.0	4.0	20,082.0	4.0	16.0	19,959.4	1.0	1.0
23	100-20-20-50	25,081.9	3.0	9.0	25,128.2	4.0	16.0	24,992.1	2.0	4.0	24,758.9	1.0	1.0
24	100-20-20-100	30,147.9	4.0	16.0	29,833.3	3.0	9.0	29,813.4	2.0	4.0	29,467.4	1.0	1.0
Average Rank			2.8			3.5			2.7			1.0	

Table B.1: Friedman test for the results with LS budget.

No.	Instance	GRASP 1-5000			ILS 5000-2-4			MS-ILS 1-10-500-2-4			MS-ILS-GT 1-10-500-2-4		
	n-*m*-*q*-β	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	50-5-10-10	8,637.0	3.5	12.3	8,637.0	3.5	12.3	8,633.4	1.0	1.0	8,635.0	2.0	4.0
2	50-5-10-50	9,986.8	4.0	16.0	9,940.8	3.0	9.0	9,937.9	2.0	4.0	9,936.0	1.0	1.0
3	50-5-10-100	11,576.3	4.0	16.0	11,557.2	3.0	9.0	11,532.8	2.0	4.0	11,525.0	1.0	1.0
4	50-5-20-10	8,019.4	3.0	9.0	8,089.6	4.0	16.0	7,966.0	1.5	2.0	7,966.0	1.5	2.3
5	50-5-20-50	10,040.2	4.0	16.0	10,014.0	2.0	4.0	10,014.0	2.0	4.0	10,014.0	2.0	4.0
6	50-5-20-100	10,962.7	4.0	16.0	10,958.0	3.0	9.0	10,931.3	2.0	4.0	10,923.0	1.0	1.0
7	50-10-10-10	10,905.6	4.0	16.0	10,884.0	2.0	4.0	10,885.4	3.0	9.0	10,882.0	1.0	1.0
8	50-10-10-50	12,860.0	3.0	9.0	13,032.3	4.0	16.0	12,849.0	1.5	2.3	12,849.0	1.5	2.3
9	50-10-10-100	16,261.0	4.0	16.0	16,229.2	3.0	9.0	16,212.0	1.5	2.3	16,212.0	1.5	2.3
10	50-10-20-10	11,748.1	4.0	16.0	11,733.7	3.0	9.0	11,628.5	2.0	4.0	11,590.6	1.0	1.0
11	50-10-20-50	14,387.0	4.0	16.0	14,359.2	3.0	9.0	14,340.1	2.0	4.0	14,338.5	1.0	1.0
12	50-10-20-100	15,183.4	4.0	16.0	15,135.7	3.0	9.0	15,080.6	2.0	4.0	15,076.2	1.0	1.0
13	100-10-10-d10	12,949.3	4.0	16.0	12,888.3	2.0	4.0	12,914.5	3.0	9.0	12,789.7	1.0	1.0
14	100-10-10-50	15,383.9	4.0	16.0	15,325.3	3.0	9.0	15,267.0	2.0	4.0	15,084.4	1.0	1.0
15	100-10-10-100	18,684.3	4.0	16.0	18,454.2	2.0	4.0	18,565.1	3.0	9.0	18,280.7	1.0	1.0
16	100-10-20-10	13,807.0	3.0	9.0	13,868.8	4.0	16.0	13,760.4	2.0	4.0	13,663.1	1.0	1.0
17	100-10-20-50	16,306.0	4.0	16.0	16,150.9	2.0	4.0	16,205.4	3.0	9.0	15,983.2	1.0	1.0
18	100-10-20-100	19,074.9	4.0	16.0	18,851.6	3.0	9.0	18,824.4	2.0	4.0	18,457.8	1.0	1.0
19	100-20-10-10	22,400.1	4.0	16.0	22,340.0	2.0	4.0	22,372.0	3.0	9.0	22,275.1	1.0	1.0
20	100-20-10-50	25,295.6	4.0	16.0	24,939.4	2.0	4.0	25,022.9	3.0	9.0	24,745.3	1.0	1.0
21	100-20-10-100	30,056.7	4.0	16.0	29,602.4	3.0	9.0	29,505.2	2.0	4.0	29,243.3	1.0	1.0
22	100-20-20-10	20,125.4	4.0	16.0	19,956.6	1.0	1.0	20,066.5	3.0	9.0	19,959.4	2.0	4.0
23	100-20-20-50	25,383.1	4.0	16.0	24,956.2	3.0	9.0	24,937.6	2.0	4.0	24,758.9	1.0	1.0
24	100-20-20-100	30,417.8	4.0	16.0	29,661.5	2.0	4.0	29,782.5	3.0	9.0	29,467.4	1.0	1.00
Average Rank			3.9			2.7			2.2			1.2	

Table B.2: Friedman test for the results with time budget.

Appendix C

Statistical tests for the bi-RVRP

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	2	4.5	20.3	2	4.5	20.3	1	7.5	56.3	1	7.5	56.3
2	30-3-20-50-5	7	1.0	1.0	4	6.0	36.0	2	8.0	64.0	5	3.0	9.0
3	30-3-20-100-3	4	1.5	2.3	2	6.0	36.0	4	1.5	2.3	2	6.0	36.0
4	30-3-20-100-5	1	8.0	64.0	4	3.5	12.3	4	3.5	12.3	2	7.0	49.0
5	30-3-30-50-3	2	2.0	4.0	1	6.0	36.0	2	2.0	4.0	1	6.0	36.0
6	30-3-30-50-5	4	1.0	1.0	3	2.5	6.3	1	6.0	36.0	1	6.0	36.0
7	30-3-30-100-3	4	5.0	25.0	7	1.0	1.0	2	7.0	49.0	5	3.0	9.0
8	30-3-30-100-5	2	7.5	56.3	5	1.5	2.3	3	5.5	30.3	3	5.5	30.3
9	40-4-20-50-3	2	7.5	56.3	11	1.0	1.0	2	7.5	56.3	5	5.0	25.0
10	40-4-20-50-5	14	1.0	1.0	9	2.0	4.0	6	4.5	20.3	8	3.0	9.0
11	40-4-20-100-3	8	1.0	1.0	6	4.5	20.3	6	4.5	20.3	7	2.5	6.3
12	40-4-20-100-5	5	8.0	64.0	7	3.5	12.3	7	3.5	12.3	6	6.0	36.0
13	40-4-30-50-3	9	4.0	16.0	4	7.5	56.3	6	5.5	30.3	4	7.5	56.3
14	40-4-30-50-5	7	6.5	42.3	7	6.5	42.3	8	4.5	20.3	6	8.0	64.0
15	40-4-30-100-3	2	7.0	49.0	3	6.0	36.0	5	2.0	4.0	1	8.0	64.0
16	40-4-30-100-5	9	1.5	2.3	4	7.0	49.0	8	3.5	12.3	8	3.5	12.3
17	50-5-20-50-3	7	1.5	2.3	6	3.5	12.3	4	6.0	36.0	5	5.0	25.0
18	50-5-20-50-5	7	3.0	9.0	5	5.0	25.0	8	2.0	4.0	5	5.0	25.0
19	50-5-20-100-3	11	1.0	1.0	7	2.5	6.3	4	5.5	30.3	7	2.5	6.3
20	50-5-20-100-5	5	5.0	25.0	4	7.0	49.0	4	7.0	49.0	6	3.0	9.0
21	50-5-30-50-3	7	1.5	2.3	6	3.5	12.3	7	1.5	2.3	5	6.5	42.3
22	50-5-30-50-5	4	6.5	42.3	6	4.0	16.0	4	6.5	42.3	3	8.0	64.0
23	50-5-30-100-3	4	5.0	25.0	5	2.5	6.3	3	7.5	56.3	6	1.0	1.0
24	50-5-30-100-5	8	1.0	1.0	4	5.0	25.0	7	2.0	4.0	6	3.0	9.0
Average Rank			3.8			4.3			4.8			5.1	

Table C.1: Friedman test for Θ with $\varrho = 5\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	4	1.0	1.0	3	2.0	4.0	2	4.5	20.3	2	4.5	20.3
2	30-3-20-50-5	4	6.0	36.0	4	6.0	36.0	5	3.0	9.0	5	3.0	9.0
3	30-3-20-100-3	3	3.0	9.0	2	6.0	36.0	2	6.0	36.0	2	6.0	36.0
4	30-3-20-100-5	3	6.0	36.0	5	1.0	1.0	4	3.5	12.3	4	3.5	12.3
5	30-3-30-50-3	1	6.0	36.0	2	2.0	4.0	1	6.0	36.0	1	6.0	36.0
6	30-3-30-50-5	1	6.0	36.0	3	2.5	6.3	1	6.0	36.0	1	6.0	36.0
7	30-3-30-100-3	2	7.0	49.0	2	7.0	49.0	5	3.0	9.0	5	3.0	9.0
8	30-3-30-100-5	2	7.5	56.3	5	1.5	2.3	4	3.5	12.3	4	3.5	12.3
9	40-4-20-50-3	8	2.0	4.0	6	3.0	9.0	5	5.0	25.0	5	5.0	25.0
10	40-4-20-50-5	5	6.0	36.0	6	4.5	20.3	4	7.5	56.3	4	7.5	56.3
11	40-4-20-100-3	7	2.5	6.3	5	6.0	36.0	3	7.5	56.3	3	7.5	56.3
12	40-4-20-100-5	6	6.0	36.0	6	6.0	36.0	11	1.5	2.3	11	1.5	2.3
13	40-4-30-50-3	10	3.0	9.0	6	5.5	30.3	12	1.5	2.3	12	1.5	2.3
14	40-4-30-50-5	10	3.0	9.0	8	4.5	20.3	13	1.5	2.3	13	1.5	2.3
15	40-4-30-100-3	4	4.0	16.0	6	1.0	1.0	4	4.0	16.0	4	4.0	16.0
16	40-4-30-100-5	6	5.0	25.0	9	1.5	2.3	4	7.0	49.0	4	7.0	49.0
17	50-5-20-50-3	6	3.5	12.3	7	1.5	2.3	3	7.5	56.3	3	7.5	56.3
18	50-5-20-50-5	9	1.0	1.0	5	5.0	25.0	4	7.5	56.3	4	7.5	56.3
19	50-5-20-100-3	3	8.0	64.0	4	5.5	30.3	4	5.5	30.3	4	5.5	30.3
20	50-5-20-100-5	9	1.0	1.0	4	7.0	49.0	6	3.0	9.0	6	3.0	9.0
21	50-5-30-50-3	6	3.5	12.3	5	6.5	42.3	5	6.5	42.3	5	6.5	42.3
22	50-5-30-50-5	6	4.0	16.0	6	4.0	16.0	9	1.5	2.3	9	1.5	2.3
23	50-5-30-100-3	5	2.5	6.3	3	7.5	56.3	4	5.0	25.0	4	5.0	25.0
24	50-5-30-100-5	5	4.0	16.0	1	7.0	49.0	1	7.0	49.0	1	7.0	49.0
Average Rank			4.2			4.3			4.8			4.8	

Table C.2: Friedman test for Θ with $\varrho = 5\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	0.00	3.5	12.3	0.00	3.5	12.3	0.00	3.5	12.3	0.00	3.5	12.3
2	30-3-20-50-5	0.16	3.0	9.0	0.27	4.5	20.3	0.00	1.5	2.3	0.27	4.5	20.3
3	30-3-20-100-3	0.04	6.0	36.0	0.00	3.0	9.0	0.21	7.0	49.0	0.00	3.0	9.0
4	30-3-20-100-5	0.00	2.0	4.0	0.20	4.0	16.0	0.22	5.0	25.0	0.00	2.0	4.0
5	30-3-30-50-3	0.00	4.5	20.3	0.00	4.5	20.3	0.00	4.5	20.3	0.00	4.5	20.3
6	30-3-30-50-5	0.11	7.0	49.0	0.01	5.0	25.0	0.00	2.5	6.3	0.00	2.5	6.3
7	30-3-30-100-3	0.48	6.0	36.0	0.17	5.0	25.0	0.00	2.0	4.0	0.55	8.0	64.0
8	30-3-30-100-5	0.00	2.0	4.0	0.14	4.0	16.0	0.18	5.0	25.0	0.30	7.0	49.0
9	40-4-20-50-3	0.00	1.5	2.3	0.08	3.5	12.3	0.00	1.5	2.3	0.18	8.0	64.0
10	40-4-20-50-5	0.08	1.0	1.0	0.19	3.0	9.0	0.30	7.0	49.0	0.21	4.0	16.0
11	40-4-20-100-3	0.10	3.0	9.0	0.04	1.0	1.0	0.14	5.0	25.0	0.13	4.0	16.0
12	40-4-20-100-5	0.22	5.5	30.3	0.09	1.5	2.3	0.22	5.5	30.3	0.20	4.0	16.0
13	40-4-30-50-3	0.10	4.0	16.0	0.02	1.5	2.3	0.19	7.0	49.0	0.02	1.5	2.3
14	40-4-30-50-5	0.12	7.0	49.0	0.04	1.0	1.0	0.30	8.0	64.0	0.11	6.0	36.0
15	40-4-30-100-3	0.00	1.5	2.3	0.38	8.0	64.0	0.37	7.0	49.0	0.00	1.5	2.3
16	40-4-30-100-5	0.10	1.5	2.3	0.41	8.0	64.0	0.18	5.5	30.3	0.10	1.5	2.3
17	50-5-20-50-3	0.29	7.0	49.0	0.02	2.0	4.0	0.43	8.0	64.0	0.27	6.0	36.0
18	50-5-20-50-5	0.26	7.0	49.0	0.23	6.0	36.0	0.09	2.0	4.0	0.21	5.0	25.0
19	50-5-20-100-3	0.19	4.0	16.0	0.24	5.0	25.0	0.36	6.0	36.0	0.10	2.0	4.0
20	50-5-20-100-5	0.32	4.5	20.3	0.47	8.0	64.0	0.35	6.0	36.0	0.18	3.0	9.0
21	50-5-30-50-3	0.12	3.0	9.0	0.21	5.0	25.0	0.24	6.0	36.0	0.30	8.0	64.0
22	50-5-30-50-5	0.45	8.0	64.0	0.30	6.0	36.0	0.38	7.0	49.0	0.20	3.0	9.0
23	50-5-30-100-3	0.36	4.0	16.0	0.25	2.0	4.0	0.47	6.0	36.0	0.06	1.0	1.0
24	50-5-30-100-5	0.12	7.0	49.0	0.07	2.0	4.0	0.11	5.5	30.3	0.11	5.5	30.3
Average Rank			4.3			4.0			5.2			4.1	

Table C.3: Friedman test for Y_1 with $\varrho = 5\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.26	8.0	64.0	0.17	7.0	49.0	0.00	3.5	12.3	0.00	3.5	12.3
2	30-3-20-50-5	0.35	6.0	36.0	0.00	1.5	2.3	0.38	7.0	49.0	0.47	8.0	64.0
3	30-3-20-100-3	0.48	8.0	64.0	0.00	3.0	9.0	0.00	3.0	9.0	0.00	3.0	9.0
4	30-3-20-100-5	0.32	7.0	49.0	0.25	6.0	36.0	0.40	8.0	64.0	0.00	2.0	4.0
5	30-3-30-50-3	0.00	4.5	20.3	0.00	4.5	20.3	0.00	4.5	20.3	0.00	4.5	20.3
6	30-3-30-50-5	0.00	2.5	6.3	0.03	6.0	36.0	0.00	2.5	6.3	0.38	8.0	64.0
7	30-3-30-100-3	0.00	2.0	4.0	0.00	2.0	4.0	0.53	7.0	49.0	0.01	4.0	16.0
8	30-3-30-100-5	0.00	2.0	4.0	0.38	8.0	64.0	0.19	6.0	36.0	0.00	2.0	4.0
9	40-4-20-50-3	0.12	6.0	36.0	0.10	5.0	25.0	0.08	3.5	12.3	0.16	7.0	49.0
10	40-4-20-50-5	0.29	6.0	36.0	0.23	5.0	25.0	0.52	8.0	64.0	0.14	2.0	4.0
11	40-4-20-100-3	0.25	8.0	64.0	0.21	7.0	49.0	0.20	6.0	36.0	0.08	2.0	4.0
12	40-4-20-100-5	0.40	8.0	64.0	0.13	3.0	9.0	0.09	1.5	2.3	0.29	7.0	49.0
13	40-4-30-50-3	0.09	3.0	9.0	0.24	8.0	64.0	0.11	5.0	25.0	0.16	6.0	36.0
14	40-4-30-50-5	0.09	4.0	16.0	0.09	4.0	16.0	0.07	2.0	4.0	0.09	4.0	16.0
15	40-4-30-100-3	0.24	5.0	25.0	0.07	3.0	9.0	0.31	6.0	36.0	0.20	4.0	16.0
16	40-4-30-100-5	0.34	7.0	49.0	0.11	3.0	9.0	0.18	5.5	30.3	0.16	4.0	16.0
17	50-5-20-50-3	0.17	4.0	16.0	0.20	5.0	25.0	0.08	3.0	9.0	0.00	1.0	1.0
18	50-5-20-50-5	0.09	2.0	4.0	0.09	2.0	4.0	0.35	8.0	64.0	0.14	4.0	16.0
19	50-5-20-100-3	0.08	1.0	1.0	0.37	7.0	49.0	0.44	8.0	64.0	0.12	3.0	9.0
20	50-5-20-100-5	0.06	1.0	1.0	0.12	2.0	4.0	0.44	7.0	49.0	0.32	4.5	20.3
21	50-5-30-50-3	0.08	2.0	4.0	0.25	7.0	49.0	0.05	1.0	1.0	0.16	4.0	16.0
22	50-5-30-50-5	0.22	4.0	16.0	0.23	5.0	25.0	0.15	1.0	1.0	0.16	2.0	4.0
23	50-5-30-100-3	0.26	3.0	9.0	0.75	8.0	64.0	0.55	7.0	49.0	0.43	5.0	25.0
24	50-5-30-100-5	0.10	3.5	12.3	0.00	1.0	1.0	0.10	3.5	12.3	0.33	8.0	64.0
Average Rank			4.5			4.7			4.9			4.3	

Table C.4: Friedman test for Y_1 with $\varrho = 5\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	1.00	3.5	12.3	0.99	7.5	56.3	1.00	3.5	12.3	1.00	3.5	12.3
2	30-3-20-50-5	0.99	6.0	36.0	1.00	2.5	6.3	1.00	2.5	6.3	0.98	8.0	64.0
3	30-3-20-100-3	0.99	6.5	42.3	0.99	6.5	42.3	0.99	6.5	42.3	1.00	2.5	6.3
4	30-3-20-100-5	1.00	2.5	6.3	0.99	6.5	42.3	1.00	2.5	6.3	1.00	2.5	6.3
5	30-3-30-50-3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3
6	30-3-30-50-5	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3
7	30-3-30-100-3	0.98	8.0	64.0	0.99	6.0	36.0	1.00	2.5	6.3	1.00	2.5	6.3
8	30-3-30-100-5	0.99	6.5	42.3	1.00	2.5	6.3	0.99	6.5	42.3	0.99	6.5	42.3
9	40-4-20-50-3	0.99	3.0	9.0	0.98	7.0	49.0	0.99	3.0	9.0	0.98	7.0	49.0
10	40-4-20-50-5	0.97	7.0	49.0	0.99	3.0	9.0	0.99	3.0	9.0	0.97	7.0	49.0
11	40-4-20-100-3	0.99	7.0	49.0	1.00	3.0	9.0	1.00	3.0	9.0	0.99	7.0	49.0
12	40-4-20-100-5	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3
13	40-4-30-50-3	1.00	2.0	4.0	0.99	6.0	36.0	0.99	6.0	36.0	1.00	2.0	4.0
14	40-4-30-50-5	1.00	1.5	2.3	0.98	5.5	30.3	0.98	5.5	30.3	0.97	8.0	64.0
15	40-4-30-100-3	0.97	8.0	64.0	1.00	3.0	9.0	1.00	3.0	9.0	1.00	3.0	9.0
16	40-4-30-100-5	0.97	3.5	12.3	0.96	5.5	30.3	0.97	3.5	12.3	0.94	8.0	64.0
17	50-5-20-50-3	0.99	6.0	36.0	0.99	6.0	36.0	1.00	2.0	4.0	0.99	6.0	36.0
18	50-5-20-50-5	1.00	2.0	4.0	0.98	7.5	56.3	0.99	5.0	25.0	1.00	2.0	4.0
19	50-5-20-100-3	0.99	8.0	64.0	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0
20	50-5-20-100-5	0.97	6.0	36.0	0.97	6.0	36.0	1.00	1.5	2.3	0.99	3.5	12.3
21	50-5-30-50-3	0.98	7.0	49.0	0.99	3.5	12.3	0.98	7.0	49.0	0.98	7.0	49.0
22	50-5-30-50-5	0.99	6.5	42.3	1.00	3.0	9.0	1.00	3.0	9.0	1.00	3.0	9.0
23	50-5-30-100-3	1.00	2.5	6.3	0.99	6.5	42.3	1.00	2.5	6.3	0.99	6.5	42.3
24	50-5-30-100-5	0.98	7.0	49.0	1.00	2.0	4.0	0.98	7.0	49.0	0.99	4.5	20.3
Average Rank			5.1			4.9			4.0			4.9	

Table C.5: Friedman test for Y_2 with $\varrho = 5\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.99	7.5	56.3	1.00	3.5	12.3	1.00	3.5	12.3	1.00	3.5	12.3
2	30-3-20-50-5	1.00	2.5	6.3	0.99	6.0	36.0	0.99	6.0	36.0	1.00	2.5	6.3
3	30-3-20-100-3	0.99	6.5	42.3	1.00	2.5	6.3	1.00	2.5	6.3	1.00	2.5	6.3
4	30-3-20-100-5	0.99	6.5	42.3	0.99	6.5	42.3	0.99	6.5	42.3	1.00	2.5	6.3
5	30-3-30-50-3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3
6	30-3-30-50-5	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	0.99	4.5	20.3
7	30-3-30-100-3	0.99	6.0	36.0	0.99	6.0	36.0	1.00	2.5	6.3	1.00	2.5	6.3
8	30-3-30-100-5	1.00	2.5	6.3	1.00	2.5	6.3	1.00	2.5	6.3	0.99	6.5	42.3
9	40-4-20-50-3	0.98	7.0	49.0	0.99	3.0	9.0	0.99	3.0	9.0	0.99	3.0	9.0
10	40-4-20-50-5	0.99	3.0	9.0	0.97	7.0	49.0	0.99	3.0	9.0	0.99	3.0	9.0
11	40-4-20-100-3	1.00	3.0	9.0	0.99	7.0	49.0	1.00	3.0	9.0	1.00	3.0	9.0
12	40-4-20-100-5	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3
13	40-4-30-50-3	0.99	6.0	36.0	1.00	2.0	4.0	0.99	6.0	36.0	0.99	6.0	36.0
14	40-4-30-50-5	0.98	5.5	30.3	1.00	1.5	2.3	0.98	5.5	30.3	0.99	3.0	9.0
15	40-4-30-100-3	1.00	3.0	9.0	0.98	7.0	49.0	0.99	6.0	36.0	1.00	3.0	9.0
16	40-4-30-100-5	0.96	5.5	30.3	0.95	7.0	49.0	1.00	1.0	1.0	0.98	2.0	4.0
17	50-5-20-50-3	0.99	6.0	36.0	0.99	6.0	36.0	1.00	2.0	4.0	1.00	2.0	4.0
18	50-5-20-50-5	0.98	7.5	56.3	0.99	5.0	25.0	1.00	2.0	4.0	0.99	5.0	25.0
19	50-5-20-100-3	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0
20	50-5-20-100-5	0.97	6.0	36.0	0.99	3.5	12.3	0.96	8.0	64.0	1.00	1.5	2.3
21	50-5-30-50-3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3	1.00	1.0	1.0
22	50-5-30-50-5	1.00	3.0	9.0	1.00	3.0	9.0	0.98	8.0	64.0	0.99	6.5	42.3
23	50-5-30-100-3	0.99	6.5	42.3	0.99	6.5	42.3	1.00	2.5	6.3	1.00	2.5	6.3
24	50-5-30-100-5	1.00	2.0	4.0	1.00	2.0	4.0	0.99	4.5	20.3	0.98	7.0	49.0
Average Rank			4.9			4.5			4.1			3.6	

Table C.6: Friedman test for Y_2 with $\varrho = 5\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	2	7.5	56.3	5	3.5	12.3	5	3.5	12.3	5	3.5	12.3
2	30-3-20-50-5	10	1.0	1.0	2	5.5	30.3	5	2.5	6.3	1	7.5	56.3
3	30-3-20-100-3	7	3.0	9.0	8	2.0	4.0	4	6.5	42.3	6	4.0	16.0
4	30-3-20-100-5	2	8.0	64.0	7	2.0	4.0	6	4.0	16.0	5	6.0	36.0
5	30-3-30-50-3	10	1.0	1.0	4	7.5	56.3	7	2.0	4.0	5	5.5	30.3
6	30-3-30-50-5	8	2.5	6.3	10	1.0	1.0	3	6.5	42.3	8	2.5	6.3
7	30-3-30-100-3	8	3.0	9.0	5	6.5	42.3	10	2.0	4.0	6	5.0	25.0
8	30-3-30-100-5	7	3.0	9.0	12	1.0	1.0	8	2.0	4.0	5	5.0	25.0
9	40-4-20-50-3	10	1.5	2.3	10	1.5	2.3	4	7.5	56.3	4	7.5	56.3
10	40-4-20-50-5	7	3.5	12.3	6	6.0	36.0	8	2.0	4.0	6	6.0	36.0
11	40-4-20-100-3	8	4.0	16.0	7	6.0	36.0	9	2.5	6.3	7	6.0	36.0
12	40-4-20-100-5	7	1.5	2.3	4	5.5	30.3	5	3.0	9.0	3	8.0	64.0
13	40-4-30-50-3	9	1.0	1.0	6	6.0	36.0	5	8.0	64.0	8	3.0	9.0
14	40-4-30-50-5	1	6.0	36.0	3	1.0	1.0	2	2.5	6.3	1	6.0	36.0
15	40-4-30-100-3	12	1.0	1.0	6	7.0	49.0	9	2.5	6.3	7	6.0	36.0
16	40-4-30-100-5	12	1.5	2.3	12	1.5	2.3	10	5.0	25.0	8	7.0	49.0
17	50-5-20-50-3	11	4.5	20.3	13	2.5	6.3	15	1.0	1.0	7	7.0	49.0
18	50-5-20-50-5	13	2.0	4.0	6	8.0	64.0	10	5.0	25.0	10	5.0	25.0
19	50-5-20-100-3	9	4.5	20.3	11	2.5	6.3	4	8.0	64.0	8	6.0	36.0
20	50-5-20-100-5	13	2.0	4.0	17	1.0	1.0	4	8.0	64.0	8	7.0	49.0
21	50-5-30-50-3	9	5.5	30.3	11	2.5	6.3	7	7.0	49.0	11	2.5	6.3
22	50-5-30-50-5	10	1.5	2.3	7	5.0	25.0	8	4.0	16.0	6	6.5	42.3
23	50-5-30-100-3	7	6.0	36.0	13	1.0	1.0	9	4.0	16.0	8	5.0	25.0
24	50-5-30-100-5	8	6.5	42.3	10	4.0	16.0	8	6.5	42.3	13	1.0	1.0
Average Rank			3.4			3.8			4.4			5.4	

Table C.7: Friedman test for Θ with $\varrho = 25\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	2	7.5	56.3	5	3.5	12.3	3	6.0	36.0	6	1.0	1.0
2	30-3-20-50-5	1	7.5	56.3	2	5.5	30.3	5	2.5	6.3	3	4.0	16.0
3	30-3-20-100-3	13	1.0	1.0	5	5.0	25.0	4	6.5	42.3	2	8.0	64.0
4	30-3-20-100-5	4	7.0	49.0	8	1.0	1.0	6	4.0	16.0	6	4.0	16.0
5	30-3-30-50-3	5	5.5	30.3	6	3.5	12.3	6	3.5	12.3	4	7.5	56.3
6	30-3-30-50-5	7	4.0	16.0	5	5.0	25.0	3	6.5	42.3	2	8.0	64.0
7	30-3-30-100-3	11	1.0	1.0	7	4.0	16.0	4	8.0	64.0	5	6.5	42.3
8	30-3-30-100-5	3	8.0	64.0	4	7.0	49.0	5	5.0	25.0	5	5.0	25.0
9	40-4-20-50-3	7	5.5	30.3	7	5.5	30.3	9	3.0	9.0	8	4.0	16.0
10	40-4-20-50-5	6	6.0	36.0	2	8.0	64.0	9	1.0	1.0	7	3.5	12.3
11	40-4-20-100-3	7	6.0	36.0	11	1.0	1.0	5	8.0	64.0	9	2.5	6.3
12	40-4-20-100-5	7	1.5	2.3	4	5.5	30.3	4	5.5	30.3	4	5.5	30.3
13	40-4-30-50-3	6	6.0	36.0	8	3.0	9.0	8	3.0	9.0	6	6.0	36.0
14	40-4-30-50-5	1	6.0	36.0	2	2.5	6.3	1	6.0	36.0	1	6.0	36.0
15	40-4-30-100-3	8	4.5	20.3	8	4.5	20.3	9	2.5	6.3	5	8.0	64.0
16	40-4-30-100-5	7	8.0	64.0	11	3.0	9.0	10	5.0	25.0	10	5.0	25.0
17	50-5-20-50-3	11	4.5	20.3	8	6.0	36.0	13	2.5	6.3	6	8.0	64.0
18	50-5-20-50-5	12	3.0	9.0	14	1.0	1.0	9	7.0	49.0	10	5.0	25.0
19	50-5-20-100-3	12	1.0	1.0	11	2.5	6.3	9	4.5	20.3	5	7.0	49.0
20	50-5-20-100-5	12	3.0	9.0	10	4.5	20.3	9	6.0	36.0	10	4.5	20.3
21	50-5-30-50-3	15	1.0	1.0	9	5.5	30.3	10	4.0	16.0	5	8.0	64.0
22	50-5-30-50-5	6	6.5	42.3	5	8.0	64.0	9	3.0	9.0	10	1.5	2.3
23	50-5-30-100-3	10	3.0	9.0	11	2.0	4.0	6	7.5	56.3	6	7.5	56.3
24	50-5-30-100-5	12	2.0	4.0	9	5.0	25.0	11	3.0	9.0	5	8.0	64.0
Average Rank			4.5			4.3			4.7			5.6	

Table C.8: Friedman test for Θ with $\varrho = 25\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	0.00	1.5	2.3	0.19	6.0	36.0	0.18	5.0	25.0	0.34	7.0	49.0
2	30-3-20-50-5	0.07	5.0	25.0	0.00	2.5	6.3	0.15	8.0	64.0	0.00	2.5	6.3
3	30-3-20-100-3	0.07	2.0	4.0	0.12	3.0	9.0	0.38	8.0	64.0	0.18	5.0	25.0
4	30-3-20-100-5	0.00	1.0	1.0	0.11	4.0	16.0	0.26	7.0	49.0	0.07	2.5	6.3
5	30-3-30-50-3	0.12	3.0	9.0	0.20	5.0	25.0	0.10	1.0	1.0	0.29	6.0	36.0
6	30-3-30-50-5	0.32	8.0	64.0	0.14	4.0	16.0	0.26	7.0	49.0	0.19	6.0	36.0
7	30-3-30-100-3	0.07	2.0	4.0	0.29	8.0	64.0	0.08	3.0	9.0	0.18	7.0	49.0
8	30-3-30-100-5	0.29	7.0	49.0	0.04	1.0	1.0	0.12	3.0	9.0	0.20	5.0	25.0
9	40-4-20-50-3	0.10	2.0	4.0	0.08	1.0	1.0	0.43	8.0	64.0	0.25	7.0	49.0
10	40-4-20-50-5	0.15	5.0	25.0	0.24	7.0	49.0	0.09	3.0	9.0	0.35	8.0	64.0
11	40-4-20-100-3	0.05	1.0	1.0	0.24	7.0	49.0	0.12	3.5	12.3	0.13	5.0	25.0
12	40-4-20-100-5	0.12	2.0	4.0	0.40	8.0	64.0	0.24	3.0	9.0	0.32	5.0	25.0
13	40-4-30-50-3	0.05	1.0	1.0	0.40	8.0	64.0	0.22	6.0	36.0	0.08	2.0	4.0
14	40-4-30-50-5	0.00	4.0	16.0	0.40	8.0	64.0	0.00	4.0	16.0	0.00	4.0	16.0
15	40-4-30-100-3	0.07	2.0	4.0	0.20	7.0	49.0	0.19	5.5	30.3	0.19	5.5	30.3
16	40-4-30-100-5	0.06	1.0	1.0	0.07	2.0	4.0	0.10	5.5	30.3	0.20	7.0	49.0
17	50-5-20-50-3	0.07	1.5	2.3	0.09	3.0	9.0	0.07	1.5	2.3	0.10	4.0	16.0
18	50-5-20-50-5	0.04	2.5	6.3	0.28	8.0	64.0	0.12	6.0	36.0	0.04	2.5	6.3
19	50-5-20-100-3	0.10	3.0	9.0	0.13	5.5	30.3	0.12	4.0	16.0	0.06	2.0	4.0
20	50-5-20-100-5	0.11	4.0	16.0	0.07	2.0	4.0	0.57	8.0	64.0	0.16	7.0	49.0
21	50-5-30-50-3	0.08	2.0	4.0	0.09	3.5	12.3	0.07	1.0	1.0	0.09	3.5	12.3
22	50-5-30-50-5	0.04	2.0	4.0	0.20	7.0	49.0	0.11	4.0	16.0	0.28	8.0	64.0
23	50-5-30-100-3	0.16	6.5	42.3	0.15	5.0	25.0	0.13	3.5	12.3	0.13	3.5	12.3
24	50-5-30-100-5	0.15	6.5	42.3	0.15	6.5	42.3	0.22	8.0	64.0	0.10	3.5	12.3
Average Rank			3.1			5.1			4.9			4.9	

Table C.9: Friedman test for Y_1 with $\varrho = 25\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.00	1.5	2.3	0.17	4.0	16.0	0.46	8.0	64.0	0.15	3.0	9.0
2	30-3-20-50-5	0.00	2.5	6.3	0.00	2.5	6.3	0.08	6.0	36.0	0.14	7.0	49.0
3	30-3-20-100-3	0.23	6.0	36.0	0.17	4.0	16.0	0.33	7.0	49.0	0.00	1.0	1.0
4	30-3-20-100-5	0.39	8.0	64.0	0.16	5.0	25.0	0.22	6.0	36.0	0.07	2.5	6.3
5	30-3-30-50-3	0.31	7.0	49.0	0.32	8.0	64.0	0.11	2.0	4.0	0.13	4.0	16.0
6	30-3-30-50-5	0.18	5.0	25.0	0.09	3.0	9.0	0.05	2.0	4.0	0.00	1.0	1.0
7	30-3-30-100-3	0.12	4.0	16.0	0.17	6.0	36.0	0.01	1.0	1.0	0.17	5.0	25.0
8	30-3-30-100-5	0.30	8.0	64.0	0.05	2.0	4.0	0.13	4.0	16.0	0.23	6.0	36.0
9	40-4-20-50-3	0.24	6.0	36.0	0.17	4.0	16.0	0.15	3.0	9.0	0.20	5.0	25.0
10	40-4-20-50-5	0.05	2.0	4.0	0.00	1.0	1.0	0.14	4.0	16.0	0.16	6.0	36.0
11	40-4-20-100-3	0.19	6.0	36.0	0.07	2.0	4.0	0.34	8.0	64.0	0.12	3.5	12.3
12	40-4-20-100-5	0.10	1.0	1.0	0.36	6.0	36.0	0.28	4.0	16.0	0.38	7.0	49.0
13	40-4-30-50-3	0.15	4.0	16.0	0.16	5.0	25.0	0.13	3.0	9.0	0.23	7.0	49.0
14	40-4-30-50-5	0.00	4.0	16.0	0.00	4.0	16.0	0.00	4.0	16.0	0.00	4.0	16.0
15	40-4-30-100-3	0.13	4.0	16.0	0.12	3.0	9.0	0.04	1.0	1.0	0.24	8.0	64.0
16	40-4-30-100-5	0.23	8.0	64.0	0.09	4.0	16.0	0.10	5.5	30.3	0.08	3.0	9.0
17	50-5-20-50-3	0.20	7.0	49.0	0.12	5.0	25.0	0.14	6.0	36.0	0.29	8.0	64.0
18	50-5-20-50-5	0.20	7.0	49.0	0.05	4.0	16.0	0.10	5.0	25.0	0.03	1.0	1.0
19	50-5-20-100-3	0.15	7.0	49.0	0.04	1.0	1.0	0.13	5.5	30.3	0.25	8.0	64.0
20	50-5-20-100-5	0.09	3.0	9.0	0.05	1.0	1.0	0.14	6.0	36.0	0.12	5.0	25.0
21	50-5-30-50-3	0.14	6.0	36.0	0.10	5.0	25.0	0.20	7.0	49.0	0.21	8.0	64.0
22	50-5-30-50-5	0.02	1.0	1.0	0.17	5.0	25.0	0.09	3.0	9.0	0.19	6.0	36.0
23	50-5-30-100-3	0.09	2.0	4.0	0.05	1.0	1.0	0.20	8.0	64.0	0.16	6.5	42.3
24	50-5-30-100-5	0.09	1.5	2.3	0.10	3.5	12.3	0.14	5.0	25.0	0.09	1.5	2.3
Average Rank			4.6			3.7			4.8			4.9	

Table C.10: Friedman test for Y_1 with $\varrho = 25\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0
2	30-3-20-50-5	0.97	8.0	64.0	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0
3	30-3-20-100-3	0.99	4.0	16.0	0.98	7.0	49.0	0.98	7.0	49.0	1.00	1.5	2.3
4	30-3-20-100-5	1.00	1.5	2.3	0.98	8.0	64.0	0.99	5.0	25.0	1.00	1.5	2.3
5	30-3-30-50-3	0.98	8.0	64.0	1.00	2.0	4.0	0.99	5.5	30.3	1.00	2.0	4.0
6	30-3-30-50-5	0.95	8.0	64.0	0.99	4.5	20.3	1.00	2.0	4.0	0.98	6.0	36.0
7	30-3-30-100-3	0.98	6.5	42.3	0.98	6.5	42.3	0.98	6.5	42.3	0.99	2.5	6.3
8	30-3-30-100-5	0.98	7.0	49.0	0.97	8.0	64.0	1.00	2.0	4.0	1.00	2.0	4.0
9	40-4-20-50-3	0.99	5.0	25.0	0.99	5.0	25.0	1.00	1.5	2.3	0.99	5.0	25.0
10	40-4-20-50-5	0.99	7.5	56.3	1.00	3.5	12.3	1.00	3.5	12.3	1.00	3.5	12.3
11	40-4-20-100-3	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3	0.99	4.5	20.3
12	40-4-20-100-5	0.98	8.0	64.0	1.00	3.5	12.3	1.00	3.5	12.3	1.00	3.5	12.3
13	40-4-30-50-3	0.98	7.0	49.0	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3
14	40-4-30-50-5	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3
15	40-4-30-100-3	0.97	8.0	64.0	0.99	3.5	12.3	0.98	6.0	36.0	0.98	6.0	36.0
16	40-4-30-100-5	0.95	6.5	42.3	0.95	6.5	42.3	0.97	3.0	9.0	0.97	3.0	9.0
17	50-5-20-50-3	0.98	5.5	30.3	0.98	5.5	30.3	0.97	7.5	56.3	0.99	2.5	6.3
18	50-5-20-50-5	0.95	8.0	64.0	1.00	1.5	2.3	0.97	7.0	49.0	0.99	3.5	12.3
19	50-5-20-100-3	0.98	7.5	56.3	0.98	7.5	56.3	1.00	2.5	6.3	0.99	5.5	30.3
20	50-5-20-100-5	0.99	4.0	16.0	0.98	7.5	56.3	1.00	1.0	1.0	0.99	4.0	16.0
21	50-5-30-50-3	0.98	6.5	42.3	0.99	3.5	12.3	1.00	1.5	2.3	0.98	6.5	42.3
22	50-5-30-50-5	0.97	8.0	64.0	0.99	5.5	30.3	1.00	2.5	6.3	1.00	2.5	6.3
23	50-5-30-100-3	0.99	4.5	20.3	0.99	4.5	20.3	1.00	1.5	2.3	0.99	4.5	20.3
24	50-5-30-100-5	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3	0.97	7.5	56.3
Average Rank			6.1			4.9			3.9			3.9	

Table C.11: Friedman test for Y_2 with $\varrho = 25\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	1.00	4.0	16.0	0.99	8.0	64.0	1.00	4.0	16.0	1.00	4.0	16.0
2	30-3-20-50-5	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0	1.00	4.0	16.0
3	30-3-20-100-3	0.98	7.0	49.0	0.99	4.0	16.0	0.99	4.0	16.0	1.00	1.5	2.3
4	30-3-20-100-5	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0
5	30-3-30-50-3	0.99	5.5	30.3	0.99	5.5	30.3	0.99	5.5	30.3	1.00	2.0	4.0
6	30-3-30-50-5	0.96	7.0	49.0	0.99	4.5	20.3	1.00	2.0	4.0	1.00	2.0	4.0
7	30-3-30-100-3	0.99	2.5	6.3	0.98	6.5	42.3	0.99	2.5	6.3	0.99	2.5	6.3
8	30-3-30-100-5	1.00	2.0	4.0	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0
9	40-4-20-50-3	0.99	5.0	25.0	1.00	1.5	2.3	0.98	8.0	64.0	0.99	5.0	25.0
10	40-4-20-50-5	0.99	7.5	56.3	1.00	3.5	12.3	1.00	3.5	12.3	1.00	3.5	12.3
11	40-4-20-100-3	0.98	7.0	49.0	0.97	8.0	64.0	1.00	1.5	2.3	1.00	1.5	2.3
12	40-4-20-100-5	0.99	7.0	49.0	1.00	3.5	12.3	1.00	3.5	12.3	1.00	3.5	12.3
13	40-4-30-50-3	0.99	3.5	12.3	0.97	8.0	64.0	0.99	3.5	12.3	0.99	3.5	12.3
14	40-4-30-50-5	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3	1.00	4.5	20.3
15	40-4-30-100-3	1.00	1.5	2.3	0.99	3.5	12.3	0.98	6.0	36.0	1.00	1.5	2.3
16	40-4-30-100-5	0.98	1.0	1.0	0.96	5.0	25.0	0.94	8.0	64.0	0.97	3.0	9.0
17	50-5-20-50-3	0.99	2.5	6.3	0.97	7.5	56.3	0.99	2.5	6.3	0.99	2.5	6.3
18	50-5-20-50-5	0.98	5.5	30.3	0.98	5.5	30.3	1.00	1.5	2.3	0.99	3.5	12.3
19	50-5-20-100-3	1.00	2.5	6.3	1.00	2.5	6.3	0.99	5.5	30.3	1.00	2.5	6.3
20	50-5-20-100-5	0.98	7.5	56.3	0.99	4.0	16.0	0.99	4.0	16.0	0.99	4.0	16.0
21	50-5-30-50-3	0.98	6.5	42.3	0.98	6.5	42.3	0.99	3.5	12.3	1.00	1.5	2.3
22	50-5-30-50-5	0.99	5.5	30.3	1.00	2.5	6.3	0.98	7.0	49.0	1.00	2.5	6.3
23	50-5-30-100-3	0.96	8.0	64.0	0.98	7.0	49.0	0.99	4.5	20.3	1.00	1.5	2.3
24	50-5-30-100-5	0.99	3.5	12.3	0.99	3.5	12.3	0.97	7.5	56.3	0.99	3.5	12.3
Average Rank			4.8			5.0			4.4			3.1	

Table C.12: Friedman test for Y_2 with $\varrho = 25\%$ Part 2.

No.	Instance	NSGAII(1)			NSGAII(2)			NSGAII(3)			NSGAII(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_1}	$\Gamma(\Psi_{\varsigma_1})$	$\Gamma(\Psi_{\varsigma_1})^2$	Ψ_{ς_2}	$\Gamma(\Psi_{\varsigma_2})$	$\Gamma(\Psi_{\varsigma_2})^2$	Ψ_{ς_3}	$\Gamma(\Psi_{\varsigma_3})$	$\Gamma(\Psi_{\varsigma_3})^2$	Ψ_{ς_4}	$\Gamma(\Psi_{\varsigma_4})$	$\Gamma(\Psi_{\varsigma_4})^2$
1	30-3-20-50-3	11.00	1.0	1.0	2.00	8.0	64.0	5.00	3.0	9.0	4.00	5.0	25.0
2	30-3-20-50-5	9.00	2.0	4.0	10.00	1.0	1.0	4.00	7.5	56.3	6.00	3.5	12.3
3	30-3-20-100-3	7.00	1.5	2.3	4.00	5.5	30.3	3.00	7.5	56.3	4.00	5.5	30.3
4	30-3-20-100-5	11.00	1.0	1.0	5.00	6.0	36.0	4.00	8.0	64.0	5.00	6.0	36.0
5	30-3-30-50-3	8.00	1.5	2.3	8.00	1.5	2.3	4.00	6.5	42.3	5.00	4.5	20.3
6	30-3-30-50-5	7.00	3.5	12.3	10.00	2.0	4.0	5.00	5.5	30.3	4.00	7.5	56.3
7	30-3-30-100-3	8.00	1.5	2.3	7.00	4.5	20.3	5.00	7.0	49.0	7.00	4.5	20.3
8	30-3-30-100-5	6.00	3.0	9.0	5.00	6.0	36.0	5.00	6.0	36.0	6.00	3.0	9.0
9	40-4-20-50-3	8.00	3.5	12.3	5.00	7.0	49.0	6.00	5.5	30.3	9.00	2.0	4.0
10	40-4-20-50-5	14.00	2.0	4.0	12.00	4.5	20.3	4.00	8.0	64.0	12.00	4.5	20.3
11	40-4-20-100-3	10.00	2.5	6.3	3.00	8.0	64.0	8.00	4.5	20.3	8.00	4.5	20.3
12	40-4-20-100-5	8.00	3.5	12.3	4.00	7.5	56.3	5.00	6.0	36.0	11.00	2.0	4.0
13	40-4-30-50-3	14.00	3.0	9.0	5.00	8.0	64.0	9.00	7.0	49.0	12.00	4.0	16.0
14	40-4-30-50-5	12.00	3.0	9.0	5.00	7.5	56.3	5.00	7.5	56.3	8.00	4.5	20.3
15	40-4-30-100-3	6.00	3.0	9.0	5.00	4.5	20.3	4.00	6.0	36.0	9.00	2.0	4.0
16	40-4-30-100-5	9.00	2.0	4.0	4.00	8.0	64.0	6.00	6.5	42.3	9.00	2.0	4.0
17	50-5-20-50-3	8.00	5.5	30.3	4.00	8.0	64.0	8.00	5.5	30.3	13.00	2.5	6.3
18	50-5-20-50-5	6.00	7.0	49.0	4.00	8.0	64.0	12.00	2.0	4.0	10.00	5.0	25.0
19	50-5-20-100-3	15.00	1.0	1.0	10.00	2.5	6.3	5.00	7.0	49.0	4.00	8.0	64.0
20	50-5-20-100-5	12.00	2.5	6.3	11.00	6.0	36.0	11.00	6.0	36.0	12.00	2.5	6.3
21	50-5-30-50-3	9.00	5.5	30.3	10.00	4.0	16.0	9.00	5.5	30.3	7.00	8.0	64.0
22	50-5-30-50-5	16.00	2.0	4.0	9.00	5.0	25.0	11.00	3.5	12.3	6.00	8.0	64.0
23	50-5-30-100-3	10.00	1.0	1.0	7.00	3.0	9.0	6.00	5.0	25.0	1.00	8.0	64.0
24	50-5-30-100-5	9.00	6.0	36.0	9.00	6.0	36.0	15.00	1.5	2.3	14.00	3.0	9.0
Average Rank			2.8			5.5			5.8			4.6	

Table C.13: Friedman test for Θ with $\epsilon = 50\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	4.00	5.0	25.0	8.00	2.0	4.0	4.00	5	25.0	3.00	7.0	49.0
2	30-3-20-50-5	5.00	5.5	30.3	6.00	3.5	12.3	4.00	7.5	56.3	5.00	5.5	30.3
3	30-3-20-100-3	5.00	4.0	16.0	6.00	3.0	9.0	3.00	7.5	56.3	7.00	1.5	2.3
4	30-3-20-100-5	7.00	3.0	9.0	9.00	2.0	4.0	6.00	4	16.0	5.00	6.0	36.0
5	30-3-30-50-3	6.00	3.0	9.0	5.00	4.5	20.3	3.00	8	64.0	4.00	6.5	42.3
6	30-3-30-50-5	11.00	1.0	1.0	7.00	3.5	12.3	4.00	7.5	56.3	5.00	5.5	30.3
7	30-3-30-100-3	8.00	1.5	2.3	7.00	4.5	20.3	7.00	4.5	20.3	4.00	8.0	64.0
8	30-3-30-100-5	5.00	6.0	36.0	3.00	8.0	64.0	6.00	3	9.0	14.00	1.0	1.0
9	40-4-20-50-3	12.00	1.0	1.0	6.00	5.5	30.3	8.00	3.5	12.3	4.00	8.0	64.0
10	40-4-20-50-5	13.00	3.0	9.0	19.00	1.0	1.0	7.00	7	49.0	10.00	6.0	36.0
11	40-4-20-100-3	12.00	1.0	1.0	10.00	2.5	6.3	5.00	7	49.0	6.00	6.0	36.0
12	40-4-20-100-5	14.00	1.0	1.0	4.00	7.5	56.3	6.00	5	25.0	8.00	3.5	12.3
13	40-4-30-50-3	16.00	1.0	1.0	11.00	5.5	30.3	11.00	5.5	30.3	15.00	2.0	4.0
14	40-4-30-50-5	17.00	1.0	1.0	8.00	4.5	20.3	6.00	6	36.0	13.00	2.0	4.0
15	40-4-30-100-3	10.00	1.0	1.0	5.00	4.5	20.3	3.00	7	49.0	2.00	8.0	64.0
16	40-4-30-100-5	7.00	5.0	25.0	9.00	2.0	4.0	8.00	4	16.0	6.00	6.5	42.3
17	50-5-20-50-3	9.00	4.0	16.0	13.00	2.5	6.3	7.00	7	49.0	16.00	1.0	1.0
18	50-5-20-50-5	11.00	3.5	12.3	14.00	1.0	1.0	9.00	6	36.0	11.00	3.5	12.3
19	50-5-20-100-3	8.00	4.5	20.3	8.00	4.5	20.3	10.00	2.5	6.3	6.00	6.0	36.0
20	50-5-20-100-5	12.00	2.5	6.3	11.00	6.0	36.0	12.00	2.5	6.3	10.00	8.0	64.0
21	50-5-30-50-3	19.00	1.0	1.0	17.00	2.0	4.0	12.00	3	9.0	8.00	7.0	49.0
22	50-5-30-50-5	7.00	7.0	49.0	8.00	6.0	36.0	21.00	1	1.0	11.00	3.5	12.3
23	50-5-30-100-3	4.00	7.0	49.0	8.00	2.0	4.0	6.00	5	25.0	6.00	5.0	25.0
24	50-5-30-100-5	11.00	4.0	16.0	9.00	6.0	36.0	7.00	8	64.0	15.00	1.5	2.3
Average Rank			3.2			3.9			5.3			4.9	

Table C.14: Friedman test for Θ with $\epsilon = 50\%$ Part 2.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.10	3.5	12.3	0.00	1.0	1.0	0.08	2.0	4.0	0.18	6.0	36.0
2	30-3-20-50-5	0.17	5.0	25.0	0.06	1.0	1.0	0.19	7.0	49.0	0.15	3.0	9.0
3	30-3-20-100-3	0.09	3.0	9.0	0.38	7.0	49.0	0.04	1.0	1.0	0.08	2.0	4.0
4	30-3-20-100-5	0.17	3.0	9.0	0.15	2.0	4.0	0.36	8.0	64.0	0.33	7.0	49.0
5	30-3-30-50-3	0.14	4.0	16.0	0.09	3.0	9.0	0.37	8.0	64.0	0.23	6.0	36.0
6	30-3-30-50-5	0.15	5.5	30.3	0.15	5.5	30.3	0.29	8.0	64.0	0.26	7.0	49.0
7	30-3-30-100-3	0.07	2.0	4.0	0.15	4.5	20.3	0.35	7.0	49.0	0.15	4.5	20.3
8	30-3-30-100-5	0.21	4.0	16.0	0.27	6.0	36.0	0.21	4.0	16.0	0.17	2.0	4.0
9	40-4-20-50-3	0.12	3.0	9.0	0.19	7.0	49.0	0.18	5.5	30.3	0.15	4.0	16.0
10	40-4-20-50-5	0.08	4.0	16.0	0.10	5.0	25.0	0.24	8.0	64.0	0.07	3.0	9.0
11	40-4-20-100-3	0.10	4.0	16.0	0.31	8.0	64.0	0.13	5.0	25.0	0.08	2.0	4.0
12	40-4-20-100-5	0.13	5.0	25.0	0.29	8.0	64.0	0.12	3.5	12.3	0.07	2.0	4.0
13	40-4-30-50-3	0.05	3.0	9.0	0.12	5.0	25.0	0.25	8.0	64.0	0.03	1.0	1.0
14	40-4-30-50-5	0.06	2.0	4.0	0.12	5.0	25.0	0.19	6.5	42.3	0.10	4.0	16.0
15	40-4-30-100-3	0.25	7.0	49.0	0.19	5.0	25.0	0.23	6.0	36.0	0.10	3.0	9.0
16	40-4-30-100-5	0.13	3.0	9.0	0.26	8.0	64.0	0.25	7.0	49.0	0.08	2.0	4.0
17	50-5-20-50-3	0.12	5.5	30.3	0.04	1.0	1.0	0.17	7.0	49.0	0.05	2.5	6.3
18	50-5-20-50-5	0.18	8.0	64.0	0.07	2.0	4.0	0.08	4.0	16.0	0.08	4.0	16.0
19	50-5-20-100-3	0.04	1.0	1.0	0.06	3.0	9.0	0.24	7.0	49.0	0.29	8.0	64.0
20	50-5-20-100-5	0.17	7.0	49.0	0.03	1.0	1.0	0.06	2.5	6.3	0.11	4.0	16.0
21	50-5-30-50-3	0.19	7.0	49.0	0.10	5.0	25.0	0.22	8.0	64.0	0.11	6.0	36.0
22	50-5-30-50-5	0.11	3.5	12.3	0.11	3.5	12.3	0.04	1.5	2.3	0.28	8.0	64.0
23	50-5-30-100-3	0.11	3.0	9.0	0.20	7.5	56.3	0.12	4.0	16.0	0.00	1.0	1.0
24	50-5-30-100-5	0.08	5.0	25.0	0.16	7.0	49.0	0.06	3.0	9.0	0.03	1.0	1.0
Average Rank			4.2			4.6			5.5			3.9	

Table C.15: Friedman test for Y_1 with $\epsilon = 50\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.10	3.5	12.3	0.18	6.0	36.0	0.18	6.0	36.0	0.49	8.0	64.0
2	30-3-20-50-5	0.18	6.0	36.0	0.16	4.0	16.0	0.35	8	64.0	0.12	2.0	4.0
3	30-3-20-100-3	0.29	6.0	36.0	0.10	4.0	16.0	0.68	8	64.0	0.11	5.0	25.0
4	30-3-20-100-5	0.23	6.0	36.0	0.18	4.0	16.0	0.08	1	1.0	0.19	5.0	25.0
5	30-3-30-50-3	0.17	5.0	25.0	0.08	2.0	4.0	0.04	1	1.0	0.28	7.0	49.0
6	30-3-30-50-5	0.11	2.0	4.0	0.13	3.0	9.0	0.10	1	1.0	0.14	4.0	16.0
7	30-3-30-100-3	0.14	3.0	9.0	0.22	6.0	36.0	0.06	1	1.0	0.66	8.0	64.0
8	30-3-30-100-5	0.21	4.0	16.0	0.70	8.0	64.0	0.28	7	49.0	0.10	1.0	1.0
9	40-4-20-50-3	0.07	1.0	1.0	0.18	5.5	30.3	0.11	2	4.0	0.38	8.0	64.0
10	40-4-20-50-5	0.06	1.5	2.3	0.06	1.5	2.3	0.21	7	49.0	0.13	6.0	36.0
11	40-4-20-100-3	0.07	1.0	1.0	0.17	6.0	36.0	0.29	7	49.0	0.09	3.0	9.0
12	40-4-20-100-5	0.16	6.0	36.0	0.21	7.0	49.0	0.12	3.5	12.3	0.05	1.0	1.0
13	40-4-30-50-3	0.12	5.0	25.0	0.15	7.0	49.0	0.12	5.0	25.0	0.04	2.0	4.0
14	40-4-30-50-5	0.05	1.0	1.0	0.19	6.5	42.3	0.24	8	64.0	0.08	3.0	9.0
15	40-4-30-100-3	0.07	2.0	4.0	0.18	4.0	16.0	0.60	8	64.0	0.00	1.0	1.0
16	40-4-30-100-5	0.19	5.0	25.0	0.07	1.0	1.0	0.16	4	16.0	0.20	6.0	36.0
17	50-5-20-50-3	0.05	2.5	6.3	0.12	5.5	30.3	0.08	4	16.0	0.21	8.0	64.0
18	50-5-20-50-5	0.11	6.0	36.0	0.04	1.0	1.0	0.16	7	49.0	0.08	4.0	16.0
19	50-5-20-100-3	0.12	5.0	25.0	0.19	6.0	36.0	0.05	2	4.0	0.09	4.0	16.0
20	50-5-20-100-5	0.06	2.5	6.3	0.14	6.0	36.0	0.22	8	64.0	0.12	5.0	25.0
21	50-5-30-50-3	0.07	4.0	16.0	0.04	1.0	1.0	0.06	3	9.0	0.05	2.0	4.0
22	50-5-30-50-5	0.12	5.5	30.3	0.27	7.0	49.0	0.04	1.5	2.3	0.12	5.5	30.3
23	50-5-30-100-3	0.04	2.0	4.0	0.20	7.5	56.3	0.18	6	36.0	0.13	5.0	25.0
24	50-5-30-100-5	0.12	6.0	36.0	0.07	4.0	16.0	0.24	8	64.0	0.04	2.0	4.0
Average Rank			3.8			4.7			4.9			4.4	

Table C.16: Friedman test for Y_1 with $\epsilon = 50\%$ Part 2.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.97	8.0	64.0	0.99	3.5	12.3	0.98	6.5	42.3	0.99	3.5	12.3
2	30-3-20-50-5	0.98	5.0	25.0	0.97	7.5	56.3	1.00	1.0	1.0	0.97	7.5	56.3
3	30-3-20-100-3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3
4	30-3-20-100-5	0.97	7.5	56.3	0.98	5.0	25.0	0.99	2.0	4.0	0.99	2.0	4.0
5	30-3-30-50-3	0.97	8.0	64.0	0.98	7.0	49.0	0.99	3.5	12.3	0.99	3.5	12.3
6	30-3-30-50-5	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0	1.00	2.0	4.0
7	30-3-30-100-3	0.99	5.0	25.0	1.00	1.5	2.3	0.99	5.0	25.0	0.99	5.0	25.0
8	30-3-30-100-5	0.97	7.0	49.0	0.99	3.5	12.3	1.00	1.5	2.3	0.98	5.5	30.3
9	40-4-20-50-3	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0	0.99	5.0	25.0
10	40-4-20-50-5	0.98	5.0	25.0	0.98	5.0	25.0	1.00	1.5	2.3	0.97	7.5	56.3
11	40-4-20-100-3	0.99	5.5	30.3	1.00	1.5	2.3	0.99	5.5	30.3	0.99	5.5	30.3
12	40-4-20-100-5	0.99	6.5	42.3	0.99	6.5	42.3	1.00	2.5	6.3	1.00	2.5	6.3
13	40-4-30-50-3	0.99	4.0	16.0	0.99	4.0	16.0	1.00	1.0	1.0	0.99	4.0	16.0
14	40-4-30-50-5	0.97	6.5	42.3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3
15	40-4-30-100-3	0.98	7.5	56.3	1.00	2.5	6.3	1.00	2.5	6.3	0.98	7.5	56.3
16	40-4-30-100-5	0.99	2.0	4.0	0.99	2.0	4.0	0.98	5.0	25.0	0.97	7.0	49.0
17	50-5-20-50-3	0.98	3.5	12.3	0.96	8.0	64.0	0.99	1.5	2.3	0.97	6.0	36.0
18	50-5-20-50-5	1.00	1.0	1.0	0.98	7.5	56.3	0.98	7.5	56.3	0.99	4.0	16.0
19	50-5-20-100-3	0.99	3.5	12.3	0.98	5.5	30.3	1.00	1.5	2.3	1.00	1.5	2.3
20	50-5-20-100-5	0.99	3.5	12.3	0.98	6.5	42.3	0.98	6.5	42.3	0.99	3.5	12.3
21	50-5-30-50-3	0.99	4.0	16.0	0.98	7.0	49.0	0.99	4.0	16.0	1.00	1.5	2.3
22	50-5-30-50-5	0.97	7.5	56.3	1.00	1.5	2.3	0.98	6.0	36.0	1.00	1.5	2.3
23	50-5-30-100-3	0.97	8.0	64.0	0.99	4.5	20.3	0.99	4.5	20.3	1.00	1.5	2.3
24	50-5-30-100-5	0.98	6.0	36.0	0.99	2.5	6.3	0.98	6.0	36.0	0.99	2.5	6.3
Average Rank			5.3			4.6			3.8			4.0	

Table C.17: Friedman test for Y_2 with $\epsilon = 50\%$ Part 1.

No.	Instance	MOEA(1)			MOEA(2)			MOEA(3)			MOEA(4)		
	n–m–p–β–$\hbar$	Ψ_{ς_5}	$\Gamma(\Psi_{\varsigma_5})$	$\Gamma(\Psi_{\varsigma_5})^2$	Ψ_{ς_6}	$\Gamma(\Psi_{\varsigma_6})$	$\Gamma(\Psi_{\varsigma_6})^2$	Ψ_{ς_7}	$\Gamma(\Psi_{\varsigma_7})$	$\Gamma(\Psi_{\varsigma_7})^2$	Ψ_{ς_8}	$\Gamma(\Psi_{\varsigma_8})$	$\Gamma(\Psi_{\varsigma_8})^2$
1	30-3-20-50-3	0.99	3.5	12.3	0.99	3.5	12.3	0.98	6.5	42.3	1.00	1.0	1.0
2	30-3-20-50-5	1.00	2.0	4.0	0.98	5.0	25.0	0.98	5.0	25.0	0.99	3.0	9.0
3	30-3-20-100-3	0.99	3.5	12.3	0.97	8.0	64.0	0.99	3.5	12.3	0.98	7.0	49.0
4	30-3-20-100-5	0.98	5.0	25.0	0.97	7.5	56.3	0.99	2.0	4.0	0.98	5.0	25.0
5	30-3-30-50-3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3	0.99	3.5	12.3
6	30-3-30-50-5	0.98	7.5	56.3	0.98	7.5	56.3	1.00	2.0	4.0	1.00	2.0	4.0
7	30-3-30-100-3	0.98	8.0	64.0	0.99	5.0	25.0	1.00	1.5	2.3	0.99	5.0	25.0
8	30-3-30-100-5	1.00	1.5	2.3	0.96	8.0	64.0	0.99	3.5	12.3	0.98	5.5	30.3
9	40-4-20-50-3	0.99	5.0	25.0	1.00	1.0	1.0	0.99	5.0	25.0	0.99	5.0	25.0
10	40-4-20-50-5	0.99	3.0	9.0	0.97	7.5	56.3	1.00	1.5	2.3	0.98	5.0	25.0
11	40-4-20-100-3	0.99	5.5	30.3	0.99	5.5	30.3	0.99	5.5	30.3	1.00	1.5	2.3
12	40-4-20-100-5	0.99	6.5	42.3	0.99	6.5	42.3	1.00	2.5	6.3	1.00	2.5	6.3
13	40-4-30-50-3	0.97	8.0	64.0	0.98	7.0	49.0	0.99	4.0	16.0	0.99	4.0	16.0
14	40-4-30-50-5	0.96	8.0	64.0	0.97	6.5	42.3	1.00	1.0	1.0	0.99	3.5	12.3
15	40-4-30-100-3	0.99	5.5	30.3	0.99	5.5	30.3	1.00	2.5	6.3	1.00	2.5	6.3
16	40-4-30-100-5	0.98	5.0	25.0	0.96	8.0	64.0	0.99	2.0	4.0	0.98	5.0	25.0
17	50-5-20-50-3	0.98	3.5	12.3	0.97	6.0	36.0	0.99	1.5	2.3	0.97	6.0	36.0
18	50-5-20-50-5	0.99	4.0	16.0	0.99	4.0	16.0	0.99	4.0	16.0	0.99	4.0	16.0
19	50-5-20-100-3	0.97	7.5	56.3	0.98	5.5	30.3	0.97	7.5	56.3	0.99	3.5	12.3
20	50-5-20-100-5	0.96	8.0	64.0	0.99	3.5	12.3	0.99	3.5	12.3	1.00	1.0	1.0
21	50-5-30-50-3	0.98	7.0	49.0	0.98	7.0	49.0	1.00	1.5	2.3	0.99	4.0	16.0
22	50-5-30-50-5	0.99	4.0	16.0	0.99	4.0	16.0	0.97	7.5	56.3	0.99	4.0	16.0
23	50-5-30-100-3	0.99	4.5	20.3	0.98	7.0	49.0	0.99	4.5	20.3	1.00	1.5	2.3
24	50-5-30-100-5	0.98	6.0	36.0	0.98	6.0	36.0	1.00	1.0	1.0	0.98	6.0	36.0
Average Rank			5.2			5.8			3.4			3.8	

Table C.18: Friedman test for Y_2 with $\epsilon = 50\%$ Part 2.

Bibliography

Abidin, H. Z., & Din, N. M. 2012. Sensor node placement based on lexicographic minimax. *Pages 82–87 of: 2012 International Symposium on Telecommunication Technologies, ISTT 2012.*

Achuthan, N. R., Caccetta, L., & Hill, S. P. 2003. An Improved Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem. *Transportation Science*, **37**(2), 153–169.

Afsar, H. M., Prins, C., & Santos, A. C. 2014. Exact and heuristic algorithms for solving the generalized vehicle routing problem with fixed fleet size. *International Transactions in Operational Research*, **21**(1), 153–175.

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., & Requejo, C. 2013. The robust vehicle routing problem with time windows. *Computers & Operations Research*, **40**(3), 856–866.

Aissi, H., Bazgan, C., & Vanderpooten, D. 2005. Complexity of the min-max and min-max regret assignment problems. *Operations Research Letters*, **33**(6), 634–640.

Aissi, H., Bazgan, C., & Vanderpooten, D. 2009. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, **197**(2), 427–438.

Alumur, S., Nickel, S., & Saldanha-da-Gama, F. 2012. Hub location under uncertainty. *Transportation Research Part B: Methodological*, **46**(4), 529–543.

Álvarez-Miranda, E., Ljubić, I., & Toth, P. 2013. Exact approaches for solving robust prize-collecting Steiner tree problems. *European Journal of Operational Research*, **229**(3), 599–612.

Araque, J. R., Kudva, G., Morin, T. L., & Pekny, J. F. 1994. A branch-and-cut algorithm for vehicle routing problems. *Annals of Operations Research*, **50**(1), 37–59.

Averbakh, I., & Bereg, S. 2005. Facility location problems with uncertainty on the plane. *Discrete Optimization*, **2**(1), 3–34.

Baker, B. M., & Ayechew, M. A. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, **30**(5), 787 – 800.

Baldacci, R., Bartolini, E., Mingozzi, A., & Roberti, R. 2010. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, **7**(3), 229–268.

Barbarosoglu, G., & Ozgur, D. 1999. A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, **26**(3), 255–270.

Baron, O., Milner, J., & Naseraldin, H. 2011. Facility location: A robust optimization approach. *Production and Operations Management*, **20**(5), 772–785.

Beale, E. M. L. 1955. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)*, **17**(2), 173–184.

Beasley, J. E. 1983. Route first-cluster second methods for vehicle routing. *Omega*, **11**(4), 403–408.

Ben-Tal, A., & Nemirovski, A. 1998. Robust convex optimization. *Mathematics of Operations Research*, **23**(4), 769–805.

Ben-Tal, A., & Nemirovski, A. 2002. Robust optimization – methodology and applications. *Mathematical Programming*, **92**(3), 453–480.

Ben-Tal, A., Boaz, G., & Shimrit. 2009a. Robust multi-echelon multi-period inventory control. *European Journal of Operational Research*, **199**(3), 922–935.

Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. 2009b. *Robust optimization.* Princeton, NJ: Princeton University Press.

Ben-Tal, A., Bhadra, S., Bhattacharyya, C., & Nath, J. S. 2011. Chance constrained uncertain classification via robust optimization. *Mathematical Programming Series B*, **127**(1), 145–173.

Berger, J., & Barkaoui, M. 2003. A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem. *Pages 646–656 of: Genetic and Evolutionary Computation GECCO 2003.* Lecture Notes in Computer Science, vol. 2723. Springer, Berlin Heidelberg.

Bertsimas, D. 1988. *Probabilistic combinatorial optimization problems.* Ph.D. thesis, Massachusetts Institute of Technology, Dept. of Mathematics.

Bertsimas, D. 1992. A vehicle routing problem with stochastic demand. *Operations Research*, **40**(3), 574–585.

Bertsimas, D., & Doan, X. 2010. Robust and data-driven approaches to call centers. *European Journal of Operational Research*, **207**(2), 1072–1085.

Bertsimas, D., & Pachamanova, D. 2008. Robust multiperiod portfolio management in the presence of transaction costs. *Computers and Operations Research*, **35**(1), 3–17.

Bertsimas, D., & Sim, M. 2003. Robust discrete optimization and network flows. *Mathematical Programming Series B*, **98**(1-3), 49–71.

Bertsimas, D., & Simchi-Levi, D. 1996. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, **44**(2), 286–304.

Bertsimas, D., Gamarnik, D., & Rikun, A. A. 2011. Performance analysis of queueing networks via robust optimization. *Operations Research*, **59**(2), 455–466.

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, **8**(2), 239–287.

Bienstock, D., & Özbay, N. 2008. Computing robust basestock levels. *Discrete Optimization*, **5**(2), 389–414.

Birge, J., & Louveaux, F. 1997. *Introduction to stochastic programming.* Springer-Verlag, New York.

Breedam, A. Van. 1994. *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.* Ph.D. thesis, University of Antwerp, Belgium.

Candia-Véjar, A., Álvarez-Miranda, E., & Maculan, N. 2011. Minmax regret combinatorial optimization problems: an algorithmic perspective. *RAIRO-Operations Research*, **45**(2), 101–129.

Cao, E., Lai, M., & Yang, H. 2014. Open vehicle routing problem with demand uncertainty and its robust strategies. *Expert Systems with Applications*, **41**(7), 3569–3575.

Ç. Pinar, M. 2007. Robust scenario optimization based on downside-risk measure for multi-period portfolio selection. *OR Spectrum*, **29**(2), 295–309.

Charnes, A., & Cooper, W. W. 1959. Chance-Constrained Programming. *Management Science*, **6**(1), 73–79.

Christiansen, C., & Lysgaard, J. 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operation Research Letters*, **35**(6), 773–781.

Christofides, N., Mingozzi, A., & Toth, P. 1979. The vehicle routing problem. *Pages 315–338 of: Combinatorial Optimization.* Wiley, Chichester.

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

Coco, A. A., Solano-Charris, E. L., Santos, A. C., Prins, C., & Noronha, T. 2014a. *Robust optimization criteria: state-of-the-art and new issues.* Technical Report UTT-LOSI-14001. ISSN:2266-5064. Université de Technologie de Troyes.

Coco, Amadeu Almeida, Júnior, João Carlos Abreu, Noronha, Thiago Ferreira, & Santos, Andréa Cynthia. 2014b. An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. *Journal of Global Optimization*, **60**(2), 265–287.

Conover, W. J. 1999. *Practical nonparametric statistics.* New-York: 3rd edition, Wiley.

Contardo, C., & Martinelli, R. 2014. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, **12**, 129–146.

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

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. 2002. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, **53**(5), 512–522.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. 2005. New heuristics for the vehicle routing problem. *Pages 279–297 of: Logistics Systems: Design and Optimization.* Springer, US.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., & Vigo, D. 2007. Vehicle routing. *Pages 367–428 of: Handbooks in Operations Research and Management Science*. Elsevier.

Dantzig, G. B. 1955. Linear Programming under uncertainty. *Management Science*, **1**(3-4), 197–206.

Dantzig, G. B., & Ramser, J. H. 1959. The truck dispatching problem. *Management Science*, **6**(1), 80–91.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**(2), 182–197.

Dresher, M. 1961. *Games of Strategy*. Prentice-Hall.

Dror, M. 1993. Modeling vehicle routing with uncertain demands as a stochastic program: properties of the corresponding solution. *European Journal of Operational Research*, **64**(3), 432–441.

Duis, C. W., & Voß, S. 1999. The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks*, **34**(3), 181–191.

Durbach, I. N. 2014. Outranking under uncertainty using scenarios. *European Journal of Operational Research*, **232**(1), 98–108.

Eksioglu, B., Vural, A. V., & Reisman, A. 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, **57**(4), 1452–1483.

Ergun, Ö., Orlin, J. B., & Steele-Feldman, A. 2006. *Journal of Heuristics*, **12**(1-2), 115–140.

Fabozzi, F. J., Kolm, P. N., Pachamanova, D., & Focardi, S. M. 2007. *Robust portfolio optimization and management*. 5 edn. Wiley.

Feo, T. A., & Resende, M. G. C. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**(2), 109–133.

Fisher, M. 1995. Vehicle routing. *Pages 1–33 of: Handbooks in Operations Research and Management Science*. Elsevier Science.

Fisher, M., & Jaikumar, R. 1981. A generalized assignment heuristic for vehicle routing. *Networks*, **11**(2), 109–124.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P., Reis, M., Uchoa, E., & Werneck, R. F. 2006. Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming Series A*, **106**(3), 491–511.

Gabrel, V., Murat, C., & Wu, L. 2011. New models for the robust shortest path problem: complexity, resolution and generalization. *Annals of Operations Research*, **207**(1), 97–120.

Gabrel, V., Murat, C., & Thièle, A. 2013. La pw-robustesse: pourquoi un nouveau critère de robustesse et comment l'appliquer? *In: 14ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF).*

Gendreau, M., Hertz, A., & Laporte, G. 1994. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, **40**(10), 1276–1290.

Gendreau, M., Laporte, G., & Seguin, R. 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, **29**(2), 143–155.

Gendreau, M., Laporte, G., & Seguin, R. 1996. Stochastic vehicle routing. *European Journal of Operational Research*, **88**(1), 3–12.

Gendreau, M., Potvin, J.-Y., Bräysy, O., Hasle, G., & Løkketangen, A. 2008. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. *Pages 143–169 of: The vehicle routing problem: latest advances and new challenges*. Operations Research/Computer Science Interfaces, vol. 43. Springer, New York.

Ghaziri, H. 1991. Solving routing problems by a self-organizing map. *Pages 829–834 of: Artificial Neural Networks.* North-Holland, Amsterdam.

Ghaziri, H. 1996. Supervision in the self-organizing feature map: Application to the vehicle routing problem. *Pages 651–660 of: Meta-Heuristics: Theory and Applications*. Kluwer, Boston.

Gillett, B. E., & Miller, L. R. 1974. A heuristic algorithm for the Vehicle-Dispatch problem. *Operations Research*, **22**(2), 340–349.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Golden, B., Raghavan, S., & Wasil, E. 2008. *The vehicle routing problem: latest advances and new challenges.* New York: Springer.

Gómez, A., Mariño, R., Akhavan-Tabatabaei, R., Medaglia, A., & Mendoza, J. E. In press. On modeling stochastic travel and service times in vehicle routing. *Transportation Science*.

Goren, S., & Sabuncuoglu, I. 2008. Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions*, **40**(1), 66–83.

Gounaris, C. E., Wiesemann, W., & Floudas, C. A. 2013. The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty. *Operations Research*, **61**(3), 677–693.

Gounaris, C. E., Repoussis, P. P., Tarantilis, C. D., Wiesemann, W., & Floudas, C. A. In press. An Adaptive Memory Programming Framework for the Robust Capacitated Vehicle Routing Problem. *Transportation Science*.

Gülpinar, N., & Rustem, B. 2007. Worst-case robust decisions for multi-period mean-variance portfolio optimization. *European Journal of Operational Research*, **183**(3), 981–1000.

Gülpinar, N., Pachamanova, D., & Çanakoğlu, E. 2013. Robust strategies for facility location under uncertainty. *European Journal of Operational Research*, **225**(1), 21–35.

Han, J., Lee, C., & Park, S. 2013. A Robust Scenario Approach for the Vehicle Routing Problem with Uncertain Travel Times. *Transportation Science*, **48**(3), 373–390.

Hazir, Ö., Haouari, M., & Erel, E. 2010. Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, **207**(2), 633–643.

Holland, J.H. 1975. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor.

Hooker, J. N., & Williams, H. P. 2012. Combining equity and utilitarianism in a mathematical programming model. *Management Science*, **58**(9), 1682–1693.

Jozefowiez, N., Semet, F., & Talbi, E.-G. 2009. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, **195**(3), 761–769.

Kalaï, R., Lamboray, C., & Vanderpooten, D. 2012. Lexicographic α-robustness: An alternative to min-max criteria. *European Journal of Operational Research*, **220**(3), 722–728.

Kara, I., Laporte, G., & Bektas, T. 2004. A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research*, **158**(3), 793–795.

Karasan, O. E., Ç. Pinar, M., & Yaman, H. 2001. *The robust shortest path problem with interval data.* Tech. rept. Bilkent University, Department of Industrial Engineering.

Kardes, E., Ordóñez, F., & Hall, R. W. 2011. Discounted robust stochastic games and an application to queueing control. *Operations Research*, **59**(2), 365–382.

Kasperski, A. 2008. *Discrete Optimization with Interval Data - Minmax Regret and Fuzzy Approach.* Studies in Fuzziness and Soft Computing, vol. 228. Springer.

Kasperski, A., & Zieliński, P. 2011. On the approximability of robust spanning tree problems. *Theoretical Computer Science*, **412**(4-5), 365–374.

Kasperski, A., Kurpisz, A., & Zieliński, P. 2012. Approximating a two-machine flow shop scheduling under discrete scenario uncertainty. *European Journal of Operational Research*, **217**(1), 36–43.

Kenyon, A. S., & Morton, D. 2003. Stochastic vehicle routing with random travel times. *Transportation Science*, **37**(1), 69–82.

Kindervater, G. A. P., & Savelsbergh, M. W. P. 1997. Vehicle routing: handling edge exchanges. *Pages 337–360 of: Local search in combinatorial optimization.* Wiley, Chichester.

Kırkızlar, E., & Andradóttir, S. H. A. 2010. Robustness of efficient server assignment policies to service time distributions in finite-buffered lines. *Naval Research Logistic*, **57**(6), 563–582.

Kostreva, M. M., Ogryczak, W., & Wierzbicki, A. 2004. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, **158**(2), 362–377.

Kouvelis, P., & Yu, G. 1997. *Robust discrete optimization and its applications.* Kluwer Academic Publishers, Norwell, MA.

Kulkarni, R. V., & Bhave, P. R. 1985. Integer programming formulations for vehicle routing problems. *European Journal of Operational Research*, **20**(1), 58–67.

Lacomme, P., Prins, C., & Ramdane-Chérif, W. 2004. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, **131**(1-4), 159–185.

Lacomme, P., Prins, C., & Sevaux, M. 2006. A genetic algorithm for a bi-objective capacitated arc routing problem. *Computers & Operations Research*, **33**(12), 3473 – 3493.

Lambert, V., Laporte, G., & Louveaux, F. 1993. Designing collection routes through bank branches. *Computers & Operations Research*, **20**(7), 783–791.

Laporte, G. 2009. Fifty Years of Vehicle Routing. *Transportation Science*, **43**(4), 408–416.

Laporte, G., & Semet, F. 2002. Classical heuristics for the capacitated VRP. *Pages 109–128 of: The Vehicle routing problem.* SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia.

Laporte, G., Louveaux, F., & Mercure, H. 1992. The vehicle routing problem with stochastic travel times. *Transportation Science*, **26**(3), 161–170.

Laporte, G., Gendreau, M., Potvin, J.-Y., & Semet, F. 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, **7**(4-5), 285–300.

Laporte, G., V., Louveaux F., & Hamme, L. V. 2002. An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. *Operations Research*, **50**(3), 415–423.

Lecluyse, C., Woensel, T. Van, & Peremans, H. 2009. Vehicle routing with stochastic time-dependent travel times. *4OR: A Quarterly Journal of Operations Research*, **7**(4), 363–377.

Lee, C., Lee, K., & Park, S. 2012. Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the OR Society*, **63**, 1294–1306.

Li, F., Golden, B., & Wasil, E. 2005. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, **32**(5), 1165 – 1179.

Lin, S. 1965. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, **44**(10), 2245–2269.

Lourenço, H. R., Martin, O. C., & Stützle, T. 2010. Iterated Local Search: Framework and Applications. *Pages 363–397 of: Handbook of Metaheuristics.* International Series in Operations Research & Management Science. Springer, US.

Luss, H. 1999. On equitable resource allocation problems: A lexicographic minimax approach. *Operations Research*, **47**(3), 361–378.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, **100**(2), 423–445.

Mandell, M. B. 1991. Modelling effectiveness-equity trade-offs in public service delivery systems. *Management Science*, **37**(4), 467–482.

Matsuyama, Y. 1991. Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. *Pages 385–390 of: Proceedings of the International Joint Conference on Neural Networks*. Seatle, WA.

Mattia, S. 2013. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, **54**(3), 619–643.

Mester, D., & Bräysy, O. 2007. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, **34**(10), 2964–2975.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. 1960. Integer programming formulations and traveling salesman problems. *Journal of the ACM*, **7**(4), 326–329.

Minoux, M. 2010. Robust network optimization under polyhedral demand uncertainty is NP-hard. *Discrete Applied Mathematics*, **158**(5), 597–603.

Moghaddam, B. F., Ruiz, R., & Sadjadi, S. J. 2012. Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. *Computers & Industrial Engineering*, **62**(1), 306–317.

Mole, R. H., & Jameson, S. R. 1976. A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly*, **27**(2), 503–511.

Monaci, M., Pferschy, U., & Serafini, P. 2013. Exact solution of the robust knapsack problem. *Computers & Operations Research*, **40**(11), 2625–2631.

Montemanni, R., & Gambardella, L. M. 2005. A branch and bound algorithm for the robust spanning tree problem with interval data. *European Journal of Operational Research*, **161**(3), 771–779.

Montemanni, R., Gambardella, L. M., & Donati, A. V. 2004. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, **32**(3), 225–232.

Mulvey, J. M., Vanderbei, R. J., & Zenios, S. A. 1995. Robust Optimization of Large-Scale Systems. *Operations Research*, **43**(2), 264–281.

Murata, T., Nozawa, H., Ishibuchi, H., & Gen, M. 2003. Modification of local search directions for non-dominated solutions in cellular multiobjective genetic algorithms for pattern classification problems. *Pages 593–607 of: Evolutionary Multi-Criterion Optimization.* Lecture Notes in Computer Science, vol. 2632. Springer Berlin Heidelberg.

Narasimhan, R., Talluri, S., Sarkis, J., & Ross, A. 2005. Efficient service location design in government services: A decision support system framework. *Journal of Operations Management*, **23**(2), 163–178.

Neumann, J. V. 1928. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, **100**(1), 295–320.

Nikulin, Y. 2008. Simulated annealing algorithm for the robust spanning tree problem. *Journal of Heuristics*, **14**(4), 391–402.

Noorizadegan, M., Galli, L., & Chen, B. 2012. On the heterogeneous vehicle routing problem under demand uncertainty. *Pages 1–25 of: 21st International Symposium on Mathematical Programming.*

Ogryczak, W. 1999. On the distribution approach to location problems. *Computers & Industrial Engineering*, **37**(3), 595–612.

Or, I. 1976. *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking.* Ph.D. thesis, Northwestern University, Evanston, USA.

Ordóñez, F. 2010. *Robust Vehicle Routing.* Chap. 8, pages 153–178.

Ordóñez, F., & Stier-Moses, N. 2010. Wardrop equilibria with risk-averse users. *Transportation Science*, **44**, 63–86.

Orgryczak, W. 1997. On the lexicographic minimax approach to location problems. *European Journal of Operational Research*, **100**(3), 566–585.

Osman, I. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, **41**(4), 421–451.

Pecin, D., Pessoa, A., Poggi, M., & Uchoa, E. 2014. Improved Branch-Cut-and-Price for Capacitated Vehicle Routing. *Pages 393–403 of: Integer Programming and Combinatorial Optimization.* Lecture Notes in Computer Science, vol. 8494. Springer International Publishing.

Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, **31**(12), 1985–2002.

Prins, C. 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, **22**(6), 916–928.

Prins, C., Labadi, N., & Reghioui, M. 2009. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, **47**(2), 507–535.

Prins, C., Lacomme, P., & Prodhon, C. 2014. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C*, **40**, 179–200.

Ribas, G. P., Hamacher, S., & Street, A. 2010. Optimization under uncertainty of the integrated oil supply chain using stochastic and robust programming. *International Transactions in Operational Research*, **17**(6), 777–796.

Rochat, Y., & Taillard, É. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, **1**(1), 147–167.

Roy, B. 2010. Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research*, **200**(3), 629–638.

Santos, A. C., Lima, D. R, & Aloise, D. J. 2013. Modeling and solving the bi-objective minimum diameter-cost spanning tree problem. *Journal of Global Optimization*, **60**(2), 1–22.

Schoettle, K., & Werner, R. 2009. Robustness properties of mean-variance portfolios. *Optimization:A journal of Mathematical Programming and Operations Research*, **58**(6), 641–663.

Schott, J. R. 1995. *Fault tolerant design using single and multicriteria genetic algorithm optimization.* M.Phil. thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics.

Schumann, M., & Retzko, R. 1995. Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. *Pages 401–406 of: Proceedings of the International Conference on Artificial Neural Networks.* Paris.

Secomandi, N. 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, **27**(11-12), 1201–1225.

Secomandi, N. 2001. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, **49**(5), 796–802.

Secomandi, N., & Margot, F. 2009. Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands. *Operations Research*, **57**(1), 214–230.

Shapiro, A., Dentcheva, S., & Ruszczyński, A. 2008. *Lectures on stochastic programming: Modeling and Theory.* Vol. 9.

Solano-Charris, E. L., Prins, C., & Santos, A. C. 2014. Heuristic Approaches for the Robust Vehicle Routing Problem. *Pages 384–395 of: Combinatorial Optimization.* Lecture Notes in Computer Science. Springer International Publishing.

Solano-Charris, E. L., Prins, C., & Santos, A. C. 2015. Local Search Based Metaheuristics for the Robust Vehicle Routing Problem with Discrete Scenarios. *Applied Soft Computing*, **32**, 518–531.

Sun, L. 2014. A New Robust Optimization Model for the Vehicle Routing Problem with Stochastic Demands. *Journal of Interdisciplinary Mathematics*, **17**(3), 287–309.

Sungur, I., Ordóñez, F., & Dessouky, M. 2008. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, **40**(5), 509–523.

Taş, D., Dellaert, N., van Woensel, T., & de Kok, T. 2013. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, **40(1)**, 214–224.

Taillard, É. 1993. Parallel iterative search methods for vehicle routing problems. *Networks*, **23**(8), 661–673.

Tang, L., Zhu, C., Zhang, W., & Liu, Z. 2011. Robust mission planning based on nested genetic algorithm. *Pages 45–49 of: Proceedings of 4th International Workshop on Advanced Computational Intelligence, IWACI 2011.*

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. 2002. A list based threshold accepting algorithm for the capacitated vehicle routing problem. *International Journal on Computer Mathematics*, **79**(5), 537–553.

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, **152**(1), 148–158.

Thompson, P. M., & Psaraftis, H. N. 1993. Cyclic Transfer Algorithm for Multivehicle Routing and Scheduling Problems. *Operations Research*, **41**(5), 935–946.

Toklu, N. E., Montemanni, R., & Gambardella, L. M. 2013a. An ant colony system for the capacitated vehicle routing problem with uncertain travel costs. *Pages 32–39 of: IEEE Symposium on Swarm Intelligence (SIS).*

Toklu, N. E., Montemanni, R., & Gambardella, L. M. 2013b. A Robust Multiple Ant Colony System for the Capacitated Vehicle Routing Problem. *Pages 1871–1876 of: IEEE International Conference on Systems, Man, and Cybernetics (SMC).*

Toth, P., & Vigo, D. 1998. Exact solution of the Vehicle Routing problem. *Pages 1–31 of: Fleet Management and Logistics*. Springer, US.

Toth, P., & Vigo, D. 2003. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, **15**(4), 333–346.

Toth, P., & Vigo, D. 2014. *Vehicle routing: Problems, Methods and Applications, Second Edition*. Philadelphia: SIAM.

Verweij, B., Shabbir, A., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. 2003. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, **24**(2-3), 289–333.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, **60**(3), 611–624.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. 2013. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, **231**(1), 1–21.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, **234**(3), 658–673.

Wald, A. 1939. Contributions to the theory of statistical estimation and testing hypotheses. *Annals of Mathematical Statistics*, **10**(4), 299–326.

Wang, L., Fang, L., & Hipel, K. W. 2004. Lexicographic minimax approach to fair water allocation problem. *Pages 1038–1043 of: 2004 IEEE International Conference on Systems, Man and Cybernetics, SMC 2004*, vol. 1.

Waters, C. 1989. Vehicle-scheduling problems with uncertainty and omitted customer. *Journal of the Operational Research Society*, **4**(12), 1099–1108.

Wets, R. J. B. 2002. Stochastic Programming Models: Wait-and-See Versus Here-and-Now. *Pages 1–15 of: Decision Making Under Uncertainty.* The IMA Volumes in Mathematics and its Applications, vol. 128. Springer, New York.

Xu, H., Caramanis, C., & Mannor, S. 2010. Robust regression and Lasso. *IEEE Transactions on Information Theory*, **56**(7), 3561–3574.

Xu, H., Caramanis, C., & Sanghavi, S. 2012. Robust PCA via outlier pursuit. *IEEE Transactions on Information Theory*, **58**(5), 3047–3064.

Xu, J., & James, K. 1996. A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. *Transportation Science*, **30**(4), 379–393.

Yaman, H., Karasan, O. E., & Ç. Pinar, M. 2001. The robust spanning tree problem with interval data. *Operations Research Letters*, **29**(1), 31–40.

Yu, G., & Yang, J. 1998. On the robust shortest path problem. *Computers & Operations Research*, **25**(6), 457–468.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & da Fonseca, V. G. 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, **7**(2), 117–132.

Elyn Lizeth SOLANO CHARRIS

Doctorat : Optimisation et Sûreté des Systèmes

Année 2015

Méthodes d'optimisation pour le problème de tournées de véhicules robuste

Cette thèse aborde le problème de tournées de véhicules (VRP) adressant des incertitudes via l'optimisation robuste, en donnant le VRP Robuste (RVRP). D'abord, les incertitudes sont intégrées sur les temps de trajet. Ensuite, une version bi-objectif du RVRP (bi-RVRP) est considérée en prenant en compte les incertitudes sur les temps de trajet et les demandes. Pour résoudre le RVRP et le bi-RVRP, différentes méthodes sont proposées pour déterminer des solutions robustes en minimisant le pire cas. Un Programme Linéaire à Variables Mixtes Entières (MILP), six heuristiques constructives, un algorithme génétique (GA), une procédure de recherche locale et quatre stratégies itératives à démarrage multiple sont proposées : une procédure de recherche constructive adaptive randomisée (GRASP), une recherche locale itérée (ILS), une ILS à démarrage multiple (MS-ILS), et une MS-ILS basée sur des tours géants (MS-ILS-GT) convertis en tournées réalisables grâce à un découpage lexicographique. Concernant le bi-RVRP, le coût total des arcs traversés et la demande totale non satisfaite sont minimisés sur tous les scénarios. Pour résoudre le problème, différentes versions de métaheuristiques évolutives multi-objectif sont proposées et couplées à une recherche locale : l'algorithme évolutionnaire multi-objectif (MOEA) et l'algorithme génétique avec tri par non-domination version 2 (NSGAII). Différentes métriques sont utilisées pour mesurer l'efficience, la convergence, ainsi que la diversité des solutions pour tous ces algorithmes.

Mots clés : recherche opérationnelle - optimisation combinatoire - logistique (organisation) - transport - incertitude - algorithmes - métaheuristiques.

Optimization Methods for the Robust Vehicle Routing Problem

This work extends the Vehicle Routing Problem (VRP) for addressing uncertainties via robust optimization, giving the Robust VRP (RVRP). First, uncertainties are handled on travel times/costs. Then, a bi-objective version (bi-RVRP) is introduced to handle uncertainty in both, travel times and demands. For solving the RVRP and the bi-RVRP different models and methods are proposed to determine robust solutions minimizing the worst case. A Mixed Integer Linear Program (MILP), several greedy heuristics, a Genetic Algorithm (GA), a local search procedure and four local search based algorithms are proposed: a Greedy Randomized Adaptive Search Procedure (GRASP), an Iterated Local Search (ILS), a Multi-Start ILS (MS-ILS), and a MS-ILS based on Giant Tours (MS-ILS-GT) converted into feasible routes via a lexicographic splitting procedure. Concerning the bi-RVRP, the total cost of traversed arcs and the total unmet demand are minimized over all scenarios. To solve the problem, different variations of multiobjective evolutionary metaheuristics are proposed and coupled with a local search procedure: the Multiobjective Evolutionary Algorithm (MOEA) and the Non-dominated Sorting Genetic Algorithm version 2 (NSGAII). Different metrics are used to measure the efficiency, the convergence as well as the diversity of solutions for all these algorithms.

Keywords: operations research - combinatorial optimization - business logistics - transportation - uncertainty - algorithms - metaheuristics.

Thèse réalisée en partenariat entre :

RÉGION CHAMPAGNE ARDENNE

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"