par Sohaib LAFIFI

Problèmes de Tournées de Véhicules avec Synchronisation de Ressources

Thèse présentée pour l'obtention du grade
de Docteur de l'UTC

Soutenue le : 25 septembre 2014
Spécialité : Technologie de l'Information et des Systèmes

A thesis presented for the degree of Doctor
UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

Technologies de l'Information et des Systèmes
HeuDiaSyC, UMR CNRS 7253

Vehicle Routing Problems with Resources Synchronization

By Sohaib LAFIFI

Thesis defense: 25 september 2014

Jury

Jacques Carlier	Université de Technologie de Compiègne	Examiner
Duc-Cuong Dang	University of Nottingham	Invited
Laure Devendeville	Université Picardie Jules Verne	Examiner
Nacima Labadie	Université de Technologie de Troyes	Reporter
Corinne Lucet	Université Picardie Jules Verne	Examiner
Aziz Moukrim	Université de Technologie de Compiègne	Supervisor
Marc Sevaux	Université de Bretagne-Sud	Reporter

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

Abstract

Technologies de l'Information et des Systèmes
HeuDiaSyC, UMR CNRS 7253

Doctor of UTC

by Sohaib LAFIFI

Title: Vehicle Routing Problems with Resources Synchronization

This dissertation focuses on vehicle routing problems, one of the major academic problems in logistics. We address NP-Hard problems that model some real world situations particularly those with different temporal constraints including time windows, visit synchronization and service balance.

The aim of this research is to develop new algorithms for the considered problems, investigate their performance and compare them with the literature approaches. Two cases are carried out. The first case studies the Vehicle Routing Problem with Time Windows (VRPTW). We propose new lower bound methods for the number of vehicles. Then we present a Particle Swarm Optimization algorithm dealing with the Solomon objective. The second case studies the Vehicle Routing Problem with Time Windows and Synchronized Visits (VRPTWsyn). Both exact methods and heuristics are proposed and compared to the literature approaches.

keywords: Vehicle Routing Problems, Synchronization, Metaheuristics, Exact Methods.

Thesis supervisor: Aziz Moukrim

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

Résumé

Technologies de l'Information et des Systèmes
HeuDiaSyC, UMR CNRS 7253

Docteur de l'UTC

par Sohaib LAFIFI

Titre: Problèmes de Tournées de Véhicules avec Synchronisation des Ressources

Cette thèse porte sur la résolution de problèmes de transport qui intègrent des contraintes temporelles considérant les fenêtres de temps, la synchronisation des visites et l'équilibrage des services. Ces problèmes trouvent plusieurs applications dans le monde réel.

L'objectif de nos recherches est l'élaboration de nouvelles méthodes de résolution pour les problèmes considérés en examinant leur performance avec une étude comparative par rapport aux différentes approches de la littérature. Deux variantes sont traitées. Le premier cas étudie le Problème de Tournées de Véhicules avec Fenêtres de Temps (VRPTW). Nous proposons de nouveaux prétraitements et bornes inférieures pour déterminer le nombre de véhicules nécessaires en s'inspirant de travaux menés en ordonnancement (raisonnement énergétique) et d'autres problèmes combinatoires comme la clique maximum et les problèmes de bin-packing. Nous présentons également un algorithme d'optimisation par essaim particulaire qui traite de la minimisation du nombre de véhicules puis de celle du temps de trajet total. Le deuxième cas étudie le Problème de Tournées de Véhicules avec des Fenêtres de Temps et des Visites Synchronisées (VRPTWSyn). Nous proposons plusieurs méthodes basées sur des approches heuristiques et des formulations linéaires avec l'incorporation d'inégalités valides pour tenir compte de la contrainte de synchronisation.

Mots-clés : Problèmes de tournées de véhicules, Synchronisation, Metaheuristiques, Méthodes exactes.

Directeur de thèse : Aziz Moukrim

To my parents

sisters and brothers . . .

Acknowledgements

Thank God for the wisdom and perseverance that he has been bestowed upon me during this research project.

I would like to express my special appreciation and thanks to my advisor Professor Aziz Moukrim, he has been a tremendous mentor for me. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His advices on research as well as on my career have been priceless.

Besides my advisor, I would like to thank the rest of my thesis committee: Jacques Carlier, Laure Devendeville, Nacima Labadie, Corinne Lucet and Marc Sevaux for their encouragement, insightful comments and questions.

This work was supported by the Regional Council of Picardy and the European Regional Development Fund (ERDF), under PRIMA project with Aziz Moukrim and in collaboration with Corinne Lucet and Laure Devendeville from Laboratoire MIS at Université Picardie Jules Verne.

To my family, I am particularly thankful. Words cannot express how grateful I am to my mother and father for all the sacrifices that they've made on my behalf. Their prayer for me was what sustained me thus far.

Last but not least, I would also like to thank all of my friends and labmates who supported me and assisted me to strive towards my goal.

Contents

Page

Acknowledgements vii

Contents ix

List of Figures xiii

List of Tables xv

List of Algorithms xvii

Preface xix

Publications xxi

Symbols and Notation xxiii

Introduction 1

1 Field of Study 5

1.1 Combinatorial optimization 6

1.1.1 Algorithm complexity 7

1.1.2 Problem complexity 8

1.1.3 Solution methods 9

1.2 Vehicle routing problems 10

1.2.1 Some variants of VRP 10

1.2.2 Solution methods 12

1.3 Conclusion 15

2 Bounding Methods On The Number Of Vehicles For VRPTW 17

2.1 Trivial lower bounds 19

2.2 Lower bounds inspired from Energetic Reasoning 19

2.3 Bin-packing lower bounds and Energetic Reasoning 20

2.4 Lower bounds based on problem decomposition techniques 21

2.5 Preprocessing 22

2.6 Numerical results 26

2.7 Conclusion 29

3 Particle Swarm Optimization for VRPTW 31

3.1 PSO based algorithm 32

3.1.1 Solution representation and evaluation 32

3.1.2 Crossover and position update 33

3.2 Initialization algorithm 37

3.3 Best insertion heuristic . . . 38
3.4 Local search . . . 39
3.5 Parameter configuration and experimentation . . . 41
3.6 Conclusion . . . 46

4 Exact methods for VRPTWSyn 47
4.1 Problem definition . . . 48
4.2 Literature . . . 49
4.3 Problem formulation . . . 51
4.4 New reduced formulation . . . 53
4.5 Constraint programming model . . . 55
4.6 Preprocessing . . . 57
4.7 Additional cuts . . . 57
4.7.1 Incompatibilities and clique cuts . . . 57
4.7.2 Subtour eliminations . . . 59
4.7.3 MIP overall algorithm . . . 59
4.8 Experimentation . . . 60
4.8.1 Travel time . . . 62
4.8.2 Preferences . . . 63
4.8.3 Workload balance . . . 63
4.9 The efficiency of the cuts . . . 66
4.10 Conclusion . . . 67

5 Heuristic Solutions for VRPTWSyn 71

5.1 Simulated annealing based iterative local search algorithm 73

5.1.1 Constructive heuristic 74

5.1.2 Diversification process 78

5.1.3 Local search procedure 79

5.2 Experimentation 81

5.2.1 Parameter settings 82

5.2.2 Efficiency of the neighborhood structure 84

5.2.3 Comparative results 85

5.3 Conclusion 90

Conclusions and future works 93

Appendices 99

A Paper: New Lower Bounds on the Number of Vehicles for VRPTW 101

B Detailed results for Chapter 2 119

C Detailed results for Chapter 3 129

Bibliography 133

List of Figures

1.1 Euler diagram for $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-*complete*, and $\mathcal{NP}$-*hard* set of problems 8

1.2 Example of a solution to a vehicle routing problem 11

1.3 Some variants of VRP 12

3.1 Example of a sequence evaluation. 34

3.2 Example of the 2-opt* move exchange of links (i, j), (u, v) for links (i, v), (u, j). 40

3.3 Example of the Or-opt move moving sequence (i, i+1) after customer j. 40

3.4 Example of the swap move, between customers c and f. 41

3.5 Example of the shift move, on the customer f. 41

4.1 Example of VRPTWSyn solution. 49

4.2 MIP solver overall algorithm for VRPTWSyn. 60

4.3 Comparison of the Computational times for the travel time. 67

4.4 Comparison of the Computational times for the sum of negative preferences. 67

4.5 Comparison of the Computational times for the workload balance. . 68

5.1 Cross synchronization. 77

5.2 2-opt*: exchange of links (1, 2), (5, 6) for links (1, 6), (5, 2) 79

5.3 Or-opt: moving visit 3 between the depot and visit 1 80

5.4 Tradeoff between performance and computational time for different parameter settings when minimizing the total travel time. 83

5.5 Tradeoff between performance and computational time for different parameter settings when minimizing the sum of negative preferences. 84

5.6 Success rates of the neighborhoods in minimizing the total travel time. 85

5.7 Success rates of the neighborhoods in optimizing the preference. . . 86

5.8 Computational times to reach the travel time equivalent to the one in the literature. 90

5.9 Computational times to reach the sum of preferences equivalent to the one in the literature. 91

List of Tables

2.1 Average lower bound results and CPU times for Solomon [81] instances 27

2.2 Average lower bound results and CPU times for Gehring and Homberger [37] instances 28

3.1 Results on Solomon and Desrosiers [82] instances. 43

3.2 Results on Homberger and Gehring [46] instances with 200 customers. 43

3.3 Results on Homberger and Gehring [46] instances with 400 customers. 44

3.4 Results on Homberger and Gehring [46] instances with 600 customers. 44

3.5 Results on Homberger and Gehring [46] instances with 800 customers. 44

3.6 Results on Homberger and Gehring [46] instances with 1000 customers. 45

4.1 Characteristics of the benchmark instances. 61

4.2 Comparison of the solutions and computational times for the total travel time. 64

4.3 Comparison of the solutions and computational times for the sum of negative preferences. 65

4.4 Comparison of the solutions and computational times for the workload balance. 66

4.5 Comparative of the efficiency of the cuts using the reduced model on the preference objective. 68

5.1 Comparison of the solutions and computational times for the total travel time. 87

5.2 Comparison of the solutions and computational times for the sum of negative preferences. 88

5.3 Comparison of the solutions and computational times for the fairness objective. 89

B.1 Results on Solomon and Desrosiers [82] instances with 25 customers. 120

B.2 Results on Solomon and Desrosiers [82] instances with 50 customers. 121

B.3 Results on Solomon and Desrosiers [82] instances with 100 customers. 122

B.4 Results on Homberger and Gehring [46] instances with 200 customers. 123

B.5 Results on Homberger and Gehring [46] instances with 400 customers. 124

B.6 Results on Homberger and Gehring [46] instances with 600 customers. 125

B.7 Results on Homberger and Gehring [46] instances with 800 customers. 126

B.8 Results on Homberger and Gehring [46] instances with 1000 customers. 127

C.1 Results on Solomon and Desrosiers [82] instances. 129

C.2 Results on Homberger and Gehring [46] instances with 200 customers. 130

C.3 Results on Homberger and Gehring [46] instances with 400 customers. 130

C.4 Results on Homberger and Gehring [46] instances with 600 customers. 130

C.5 Results on Homberger and Gehring [46] instances with 800 customers. 131

C.6 Results on Homberger and Gehring [46] instances with 1000 customers. 131

List of Algorithms

2.1 Solving disjunctions 24

2.2 Reducing Time windows 25

3.1 PSO basic algorithm 33

3.2 Split algorithm for VRPTW. 35

3.3 Cross algorithm for VRPTW. 36

3.4 Local search for VRPTW. 39

4.1 CP model for VRPTWSyn. 56

5.1 SA-ILS algorithm for VRPTWSyn. 73

5.2 BestInsertion algorithm for VRPTWSyn. 77

5.3 Algorithm to diversify a solution. 78

5.4 Local search for VRPTWSyn. 81

Preface

This work was supported by the Regional Council of Picardy and the European Regional Development Fund (ERDF), under PRIMA project in collaboration with Corinne Lucet and Laure Devendeville from Laboratoire MIS at Université Picardie Jules Verne.

The following sections of this thesis are substantially based upon work published or submitted for publication elsewhere, or include collaborative work:

1. Parts of Chapter 2 are based on the publication:

   Sohaib Afifi et al. "New Lower Bounds on the Number of Vehicles for the Vehicle Routing Problem with Time Windows". In: CPAIOR 2014, Cork, Ireland. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014, pp. 422–437

2. Chapter 3 includes collaborative work with Rym Nesrine Guibadj

3. Chapter 4 includes collaborative work with Ali-Bey Amar during his internship

4. Parts of Chapter 5 are based on the publications:

   Sohaib Afifi et al. "A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints". In: LION 7, Catania, Italy. Vol. 7997. Lecture Notes in Computer Science. Springer, 2013, pp. 259–265

   Sohaib Afifi et al. "Heuristic Solutions for the Vehicle Routing Problem with Time Windows and Synchronized Visits". Submitted to Optimization Letters, 2014

Publications

Journal article

- Sohaib Afifi et al. “Heuristic Solutions for the Vehicle Routing Problem with Time Windows and Synchronized Visits”. Submitted to Optimization Letters, 2014

In proceedings

- Sohaib Afifi et al. “A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints”. In: LION 7, Catania, Italy. Vol. 7997. Lecture Notes in Computer Science. Springer, 2013, pp. 259–265
- Sohaib Afifi et al. “New Lower Bounds on the Number of Vehicles for the Vehicle Routing Problem with Time Windows”. In: CPAIOR 2014, Cork, Ireland. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014, pp. 422–437

Seminar and communication

- Sohaib Afifi et al. “Un algorithme de recuit simulé pour le problème de tournées de véhicules avec fenêtres de temps et contraintes de synchronisation”. (In french) ROADEF 14ième, Troyes, France. 2013

Symbols and Notation

V	Set of customers
E	Set of arcs
P^{Sync}	Set of synchronization pairs (see Chapter 4)
Q	Vehicle capacity
$tmax$	Maximum route length
δ_{ij}	Distance between nodes i and j

For a customer i:

e_i	The earliest start time
l_i	The latest start time
s_i	Service duration
q_i	Customer demand

Introduction

In today's business world in which many activities and exchanges are carried, either industrial or private, operations costs and resources consumption become a major concern. Optimizing those resources takes a very important place in decision-making companies, including those operating in the field of transport and logistics. Considerable efforts are being made to reduce the costs and consumptions, both materials and humans.

In this context, we wish to focus on Vehicle Routing Problem (VRP), one of the major academic field in logistics. VRP is a widely studied combinatorial optimization problem in which the aim is the determination of optimal tours for a group of vehicles serving a set of customers respecting some side constraints. After decades since its first definition by Dantzig and Ramser [24], it is still one of the most attractive problems because of its various practical applications and all the challenges it exposes. Dozens of variants have been treated in order to take into consideration the real-world constraints. In this thesis, we are interested in two variants which deal with extra temporal constraints. The capacitated Vehicle Routing Problem with Time Windows (VRPTW) and the Vehicle Routing Problem with Time Windows and Synchronized constraints (VRPTWSyn).

An optimization problem is easy to solve in the case where we can enumerate and evaluate all the possible solutions naively. However, in practice, enumerating

all the solutions needs an exponential calculation time. The challenge is to propose better, faster and smarter approaches. Chapter 1 gives an overview on the optimization field and some essential definitions. We also discuss some known and major methods used to deal with the vehicle routing problems in general.

After presenting the field of Study, the thesis is divided into two parts. The first part covers the capacitated Vehicle Routing Problem with Time Windows (VRPTW). It concerns a very well-known variant of VRP where each customer has a time window constraint and a demand to be delivered by capacitated vehicles. Defined first by Fisher et al. [36], it became among the most studied variants of routing problems due to its wide range of applications. Common examples are newspaper delivery, beverage and food delivery and commercial and industrial waste collection [41]. This part contains two chapters, Chapters 2 and 3.

After presenting the problem, reviewing the literature and giving a mathematical formulation, Chapter 2 presents a study of lower bounds on the number of vehicles required to serve all the customers. Our method is an adaptation of Energetic Reasoning. This algorithm, originally developed for scheduling problems, allows the detection of infeasibilities and the adjustment of the time windows during which the task execution is permitted. We proposed some procedures to define the minimal necessary time succeeding or preceding the service of each customer. The objective is to find a strong relaxation of the problem to a Parallel Machine Scheduling Problem. Then, an extension of the energetic reasoning is proposed using the bin-packing problem with conflicts. In comparison with the results of the literature, we were able to prove the optimality of the number of vehicles for the majority of the instances thanks to the preprocessing proposed.

Then, in Chapter 3, we present a solution for VRPTW which deals with the Solomon objective (Minimizing the number of vehicles used then the total travel cost [82]). We present a Particle Swarm Optimization algorithm for VRPTW. PSO is a swarm intelligence technique which takes inspiration from the collective behavior of wild animals in the nature. The proposed PSO works with permutation

encoding and uses an adapted split. A combination with a set partitioning problem and a neighborhood-based local search framework is used.

The second part deals with a relatively new variant of VRP which extends from the last one and studies the combination of some extra temporal constraints. A synchronization constraint requires more than one vehicle to serve a customer at the same time. The problem is therefore called the VRP with time windows and synchronized visits (VRPTWSyn). Such a requirement appears frequently in vehicle routing applications in various domains, ranging from individual health care services to dispatching technicians for on-site reparation/maintenance. Nevertheless, only few studies have been dedicated to the synchronization aspect of vehicle routing.

Chapter 4 introduces the problem and explores the idea of using efficiently some exact methods on the vehicle routing problem with time windows and synchronized visits (VRPTWSyn). A new linear model is proposed and compared to the classical one after strengthening them with some cuts and preprocessing techniques. Then, we propose a new approach based on constraint programming where we consider the problem as a scheduling one. A detailed discussion and comparisons are presented at the end of the chapter.

In Chapter 5, we develop a simulated annealing based iterative local search algorithm (SA-ILS) that incorporates several techniques to deal with VRPTWSyn. Experiments conducted on the instances from the literature show that our SA-ILS is fast and outperforms the existing approaches. To the best of our knowledge, this is the first time that dedicated local search methods have been proposed and evaluated on this variant of VRP.

At the end, we finish with a general conclusion including a synthesis of the contributions presented in this dissertation and some perspectives and future works.

1 | Field of Study

If you can't explain it simply, you don't understand it well enough.

Albert Einstein

1.1 Combinatorial optimization 6
1.1.1 Algorithm complexity 7
1.1.2 Problem complexity 8
1.1.3 Solution methods 9
1.2 Vehicle routing problems 10
1.2.1 Some variants of VRP 10
1.2.2 Solution methods 12
1.3 Conclusion 15

This chapter is intended to recall the basics of optimization in general and especially the vehicle routing problems. It provides an overview on all the issues addressed in the thesis. We begin by presenting some essential definitions of a combinatorial optimization problem, the vehicle routing one and some related

terminologies. Then, we present general methods for solving these problems. At the end of this chapter, we give a comprehensive idea on all issues involved and treated in this thesis with some definitions and motivations. The chapter ends with a discussion on the exact and heuristic methods used in the literature.

1.1 Combinatorial optimization

Combinatorial optimization is a subset of mathematical optimization that consists of finding optimal solutions for complex problems. In many such problems, exhaustive search is not feasible. While some problems are relatively well understood and admit solution to optimality in polynomial time, many other are NP-Hard and need some sophisticated techniques. Combinatorial optimization is very related to operations research, algorithm theory, and computational complexity theory. It has important applications in several fields, including artificial intelligence, machine learning, mathematics and software engineering.

Definition 1.1. An optimization combinatorial problem (COP) $\Phi = (\Omega, f)$ can be defined using:

- A set of variables $X = \{x_1, \ldots, x_n\}$.
- Every variable x_i is associated to a domain D_i.
- A set of constraints between the variables.
- An objective function to minimize (or to maximize) $f : D_1 \times \cdots \times D_n \rightarrow \mathbb{R}$.

A feasible solution to the problem Φ is an element $s \in \Omega$ such that:
$s = \{v_1, \ldots, v_n | v_i \in D_i$ and all the constraints are verified$\}$.
The resolution of a COP with a minimization objective function (resp. maximization) consists in finding s^* such that $\forall s \in \Omega, f(s^*) \leq f(s)$ (resp. $f(s^*) \geq f(s)$). The set Ω is called the search space.

1.1.1 Algorithm complexity

The formal definition of an algorithm is introduced by Turing [90]. This definition is based on the notion of a formal language using an abstract machine called the Turing machine. Simply, an algorithm A is a finite list of well-defined instructions for calculating a function in order to solve a problem Φ. The time and space are the most well-known complexity measures. The first is often presented by the number of instructions an algorithm needs. We consider that every instruction is done by a simple elementary operation (e.g. a test, an addition, a multiplication …).

Definition 1.2. A complexity C_A of an algorithm A on a problem Φ of size n is defined using the number of instructions required to solve any instance of the problem Φ.

C_A is often expressed asymptotically to n using the $\mathcal{O}$ notation. A is said to be of complexity $\mathcal{O}(g(n))$ if $\exists M > 0, \exists n_0$ such that $\forall n \geq n_0, C_A \leq Mg(n)$. According to g the most used algorithmic complexities are:

- $\mathcal{O}(1)$: constant and independent of the size of Ω.
- $\mathcal{O}(log\, n)$: logarithmic to the size of Ω.
- $\mathcal{O}(n)$: linear to the size of Ω.
- $\mathcal{O}(n^c)$ ($c \geq 2$ a constant): polynomial to the size of Ω.
- $\mathcal{O}(a^n)$ ($a > 1$ a constant): exponential to the size of Ω.

A practical case where the complexity is polynomial to the size of the problem and in the same time to the input values, e.g. $g(n) = n^p v^q$ where v is a value of the input and q is a constant. The algorithm is said Pseudo-polynomial.

1.1.2 Problem complexity

There are two types of problems:

- The optimization problem: $\mathbf{min}\{f(s)|s \in \Omega\}$.
- And its corresponding decision problem: Is there a solution $s \in \Omega$ with the value $f(s) \leq k$?

Decision problems are one of the central objects of study in computational complexity theory. A decision problem is a special type of computational problem whose answer is either yes or no. There are different classes of decision problems.

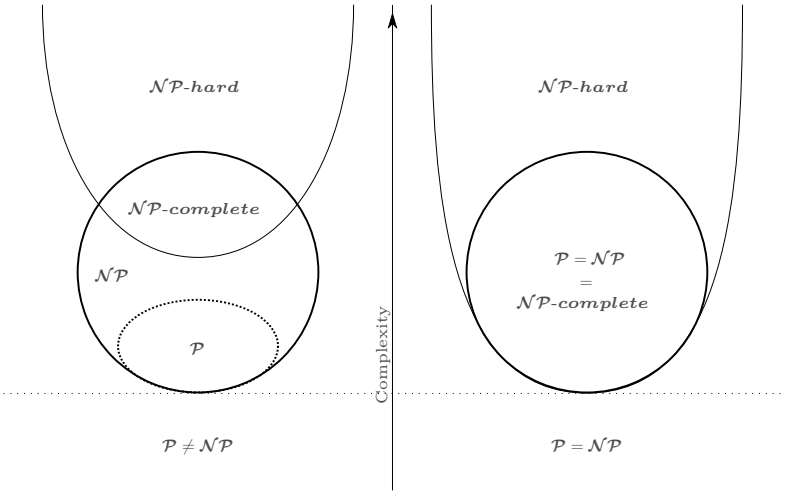


Figure 1.1 – Euler diagram for $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{NP}}$, $\boldsymbol{\mathcal{NP}}$**-complete**, and $\boldsymbol{\mathcal{NP}}$**-hard** set of problems

$\boldsymbol{\mathcal{NP}}$ is the class of decision problems, where it can be proven in polynomial time, that the answer is "yes", if this is the case. The optimization problem with a corresponding decision problem in the $\mathcal{NP}$ class can be solved by answering the decision problem a polynomial number of times.

$\boldsymbol{\mathcal{P}}$ is the class of decision problems in $\mathcal{NP}$ that can be solved in polynomial time.

$\mathcal{NP}$-complete is a subset of $\mathcal{NP}$. An instance of a problem in $\mathcal{NP}$ can be converted in polynomial time to an instance of a problem in ***$\mathcal{NP}$-complete***. (The problems in $\mathcal{NP}$ are polynomial reducible to the problems in ***$\mathcal{NP}$-complete***). This means that the problems in $\mathcal{NP}$ are not more difficult than the problems in ***$\mathcal{NP}$-complete***. The optimization problem with a corresponding ***$\mathcal{NP}$-complete*** decision problem is ***$\mathcal{NP}$-hard***.

It has not yet proven that $\mathcal{P} = \mathcal{NP}$ nor $\mathcal{P} \neq \mathcal{NP}$. For all the problems in ***$\mathcal{NP}$-complete*** there have not yet been found polynomial solution methods, and hence it is so far conjectured that $\mathcal{P} \neq \mathcal{NP}$.

1.1.3 Solution methods

In this section we will try to present an overview of the methods used to solve some COP from the ***$\mathcal{NP}$-hard*** class. These methods are usually categorized into two subsets: exact methods (e.g. linear programing, tree methods …) and Approximate methods (e.g. heuristics, meta-heuristics …).

1.1.3.a Exact methods

Exact algorithms are methods which guarantee to find an optimal solution and to prove its optimality for every instance of a COP. Among the exact methods are branch-and-bound (B&B), dynamic programming, Lagrangian relaxation based methods, and linear and integer programming based methods such as branch-and-cut, branch-and-price and branch-price-and-cut.

However, their problem is that the run-time often increases dramatically with the instance size and often only small or moderately sized instances can be practically solved to provable optimality. In this case, the only possibility for larger instances is to trade optimality for run-time, yielding heuristic algorithms.

1.1.3.b Approximate methods

With the approximate method, the guarantee of finding optimal solutions is sacrificed for the sake of getting good solutions in a limited time.

Metaheuristics include, among others, simulated annealing, tabu search, iterated local search, variable neighborhood search, and various population-based methods such as evolutionary algorithms, scatter search, memetic algorithms, and various estimation of distribution algorithms.

Recently there have been very different attempts to combine ideas and methods from these two streams. Some approaches using MP combined with metaheuristics have begun to appear regularly in the metaheuristics literature. This combination can go two-ways, both in MP used to improve or design metaheuristics and in metaheuristics used for improving known MP techniques. Even though the first of these two directions is by far more studied and refereed in some papers as Matheuristics.

1.2 Vehicle routing problems

In this section, we consider only the problems concerning the distribution of goods between depots and final users (customers). These problems are generally known as Vehicle Routing Problems or Vehicle Scheduling Problems.

1.2.1 Some variants of VRP

The vehicle Routing Problem (VRP) is a widely studied combinatorial optimization problem in which the aim is the determination of the optimal set of routes to be performed by a fleet of vehicles in order to serve a given set of customers.

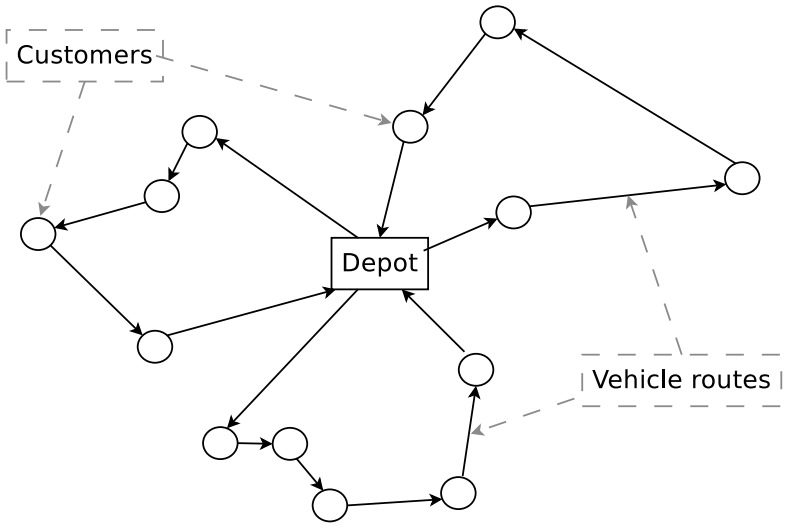


Figure 1.2 – Example of a solution to a vehicle routing problem

Typical applications of this type are, for instance, waste collection, street cleaning, school bus routing, transportation of handicapped persons, salespeople transportation …

Several variants and specializations of the vehicle routing problem exist. We first present a basic variation called Capacitated Vehicle Routing Problem (CVRP). In the CVRP, all the customers have deterministic demands known in advance and may not be split. The vehicles are identical with a capacity restriction. They start from and return to a single central depot. The objective is to minimize the total cost which can be a weighted function of the number of routes and their length or travel time in order to serve all the customers. In other words, a solution of CVRP is a set of routes which all begin and end in the depot, and which satisfies the constraint that all the customers are served only once. The transportation cost can be improved by reducing the total traveled distance and by reducing the number of required vehicles.

The majority of the real world problems are often much more complex than the classical VRP. It may usually be augmented by constraints, for example, time intervals in which each customer has to be served. In the last sixty years many

real-world problems have required extended formulation that resulted in the multiple depot VRP, periodic VRP, split delivery VRP, stochastic VRP, VRP with backhauls, VRP with pickup and delivering and many others.

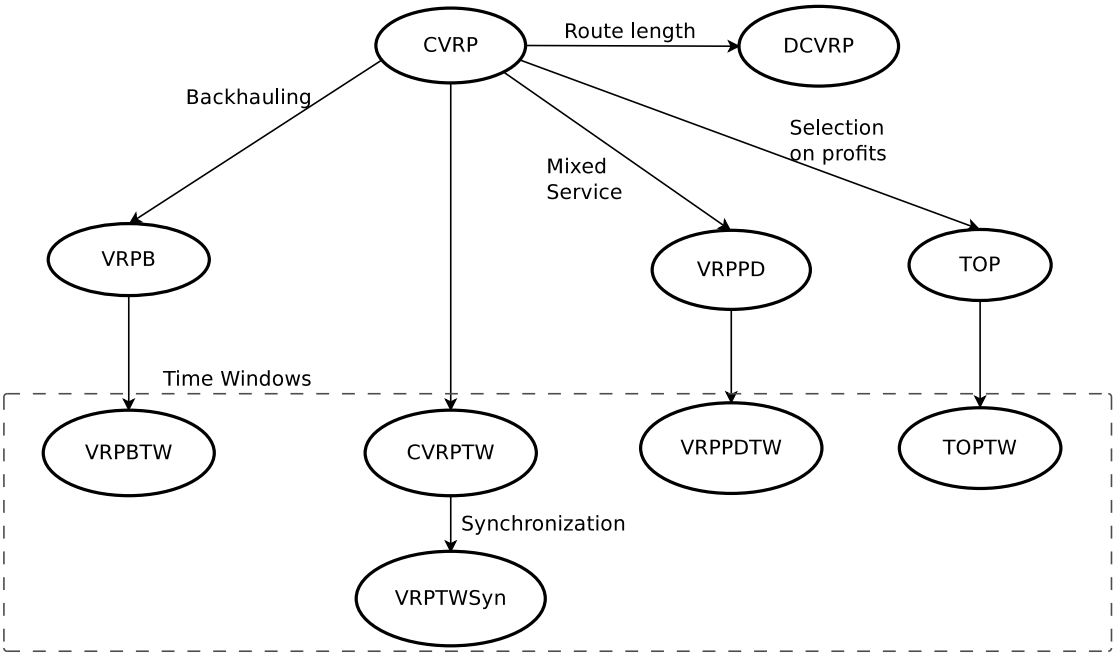


Figure 1.3 – Some variants of VRP

1.2.2 Solution methods

1.2.2.a Exact methods

Many exact approaches for the VRP and its variants were inherited from the extensive and successful work done for the TSP. Here are some of these methods:

a.1) Branch-and-bound: The branch and bound method has been extensively used to solve many VRP variants. This method sill represents the state-of-the-art with respect to exact solution methods. Laporte and Nobert [60] gave a complete and detailed analysis of the branch-and-bound algorithms proposed until 1990.

a.2) Branch-and-cut: These methods have been extremely successful in finding optimal solutions for large instances.

Branch and cut involves running a branch and bound algorithm and using cutting planes to tighten the linear programming relaxations. For an extensive and comprehensive description of these methods and successful applications on VRP variants, see [11, 67].

a.3) Dynamic programming: Dynamic programming (DP) has been applied to several types of VRPs. The idea behind dynamic programming is quite simple. In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution. Often when using a more naive method, many of the subproblems are generated and solved many times. The dynamic programming approach seeks to solve each subproblem only once, thus reducing the number of computations. DP has been used successfully to solve some VRPs to optimality or to obtain very sharp bounds on the value of their optimal solutions. Some examples can be found in [28, 71, 80].

a.4) Set covering based ILP formulations Over the years, several integer linear programming formulations have been suggested for VRP. Among these, set partitioning formulations cover a wide range of problems. Unfortunately, due to the large number of variables they contain, it will be rarely practicable to use them to derive optimal solutions. The idea is to enumerate all the feasible routes for example and then select a minimum cost set of routes such that each customer is included in some route. The formulation of set covering is equivalent to the original set partitioning one since we have assumed that the distance matrix satisfies the triangle inequality. Hence each customer will be visited exactly once in the optimal solution. Many exact methods based on these formulations were

developed by [26, 27, 45, 79]. The survey of Desrosiers et al. [29] is an excellent source for column generation based approaches.

1.2.2.b Approximate methods

In many cases, exact methods are not capable to solve VRP instances with more than $50 - 100$ nodes in reasonable time, which is generally needed in real-life applications [44]. Many works in the literature decided to sacrifice the guarantee of optimality in order to achieve good solutions in reasonable calculation time. Actually this is the tendency. A simple query at scopus with the keywords "vehicle routing" and "heuristic" shows more than 2000 papers. The methods used can be categorized into two sets: simple heuristics and metaheuristics.

b.1) Constructive heuristics and local searches: Many constructive heuristics have been developed for VRP and its variants in order to efficiently generate very good solutions keeping in mind some interesting characteristics like their short calculation time, simplicity and flexibility. The common idea is to start from an empty solution and then repeatedly extend it until a complete solution is constructed. Heuristics rules are used in the extension part. Some example of powerful heuristics can be found in the review of Cordeau et al. [21]. On the other hand, local searches have been exhaustively used in combination to the heuristics and been proved to be effective for finding good solutions. It is mainly used to move from a solution to its neighborhood through some defined operations like moving clients or exchanging paths. There are dozens of successful local search methods for the VRP and its variants. The reader is refereed to the reviews [14, 15, 40] for some examples.

b.2) Metaheuristics: Several metaheuristics have been proposed for the variants of VRP. The objective is to use many components and explore a large solution

space even by allowing deteriorating or infeasible solutions. Despite they use more calculation time in general, they hit better solutions and identify local optima. For the VRP and its variants, we count for example the following metaheuristics: Simulated annealing (SA), Tabu search (TS), Genetic algorithms (GA), Ant systems (AS). While some algorithms start from an initial solution and try to move iteratively using neighborhoods, others use a population of solutions and generate at every iteration a new solution based on some recombinations so that at the end the best part are kept during generations. A good survey on the most popular metaheuristics for VRP can be found in the chapter of Gendreau et al. [38].

1.3 Conclusion

After 60 years of its first definition, the vehicle routing problem (VRP) was and still is one of the most challenging fields of combinatorial optimization because of its various practical applications and considerable hardness. Dozens of methods were proposed including exact and heuristic techniques. Many variants have been created for the problem in order to take into consideration the real-world constraints. It is remarkable that one of the most important variants are those which deal with extra temporal constraints. In the following chapters, we study and propose some techniques for the well known vehicle routing problem with time windows and for another variant which takes into account some synchronization constraints.

2 Bounding Methods On The Number Of Vehicles For VRPTW

> An approximate answer to the right question is worth far more than a precise answer to the wrong one.
>
> John Tuley

2.1 Trivial lower bounds 19
2.2 Lower bounds inspired from Energetic Reasoning 19
2.3 Bin-packing lower bounds and Energetic Reasoning 20
2.4 Lower bounds based on problem decomposition techniques . . . 21
2.5 Preprocessing 22
2.5.0.c Preprocessing based on Energetic Reasoning . 22
2.5.0.d Preprocessing based on shaving techniques . . 23
2.6 Numerical results 26
2.7 Conclusion 29

This chapter covers a very well-known variant of VRP where each customer has a time window constraint when he can accept visits and a demand to be delivered by a capacity limited vehicle. The problem is known as the capacitated Vehicle Routing Problem with Time Windows (VRPTW). In this chapter we present a study of lower bounds on the number of vehicles required to serve all the customers. This work includes collaborative work with Rym Nesrine Guibadj. A preliminary version has been published first in the thesis of Guibadj [42] and then in the paper:

Sohaib Afifi et al. "New Lower Bounds on the Number of Vehicles for the Vehicle Routing Problem with Time Windows". In: CPAIOR 2014, Cork, Ireland. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014, pp. 422–437

The sections of this chapter include an overview of the paper and additional contributions to the work. We present at the end a comparison between all the methods. The preprint version of the paper is attached in Appendix A

The main objective of this work is to define the number of vehicles needed to visit all the customers within a VRPTW problem. This objective is very important to evaluate the fixed costs for operating the fleet. We provide an analysis of several lower bounds based on incompatibility between customers and on vehicle capacity constraints. We also develop an adaptation of Energetic Reasoning algorithm for VRPTW with a limited fleet. The main idea is to focus on some time-intervals and exploits time constraints, incompatibility graph, bin packing models and decomposition techniques in order to obtain new valid lower bounds for the fleet size. Time windows can be later adjusted based on those approaches. Experiments conducted on the standard benchmarks show that our algorithms outperform the classical lower bound techniques and give the minimum number of vehicles for 377 out of 468 instances.

Despite the extensive literature on the problem, there were only few attempts to propose lower bounds for VRPTW when the objective is to minimize the number

of vehicles. To the best of our knowledge, the most competitive results are offered by Kontoravdis and Bard [56]. Most of the works use some trivial lower bounds.

2.1 Trivial lower bounds

An obvious lower bound for VRPTW can be obtained considering the vehicle as a bin with size Q and each customer demand as an item with size q_i. A lower bound can be computed using Equation (2.1) or any other stronger lower bound for the associated bin packing problem.

$$LB_{Capacity} = \left\lceil \sum_{i=1}^{n} q_i / Q \right\rceil \tag{2.1}$$

Another lower bound can be deduced from the incompatibility constraints. Customers i and j are incompatible and denoted $i||j$, if they cannot be in the same route due to their time window constraints or if the sum of their demands is greater than the vehicle capacity.

We define the graph of incompatibilities between customers as $G_{inc} = (V, E_V)$ where $E_V = \{(i, j) \in V \times V : i||j\}$. Therefore, the minimum number of routes to be used is equal to the size of the maximum clique extracted from G_{inc}.

$$LB_{Clique} = MaximumClique(G_{inc}) \tag{2.2}$$

2.2 Lower bounds inspired from Energetic Reasoning

Energetic Reasoning is one of the known propagation techniques applied generally on some scheduling problems. We briefly present the main idea behind the energetic reasoning.

Given a time interval $[t_1, t_2]$, the approach is first based on the computation of the part of the jobs that must be processed between t_1 and t_2. This part of a job i is called work and denoted $W(i, t_1, t_2)$. The total work over $[t_1, t_2]$ is then calculated by considering it as the sum of all the works and denoted $W(t_1, t_2)$. The instance is said to be infeasible if the total work between t_1 and t_2 is greater than the available energy $m \times (t_2 - t_1)$.

Starting from a limited number of vehicles m defined by the trivial methods, the instance of mVRPTW is transformed into a PMSP instance. Then we apply the same satisfiability test on the relaxed instance. If the instance is infeasible then $m + 1$ is a valid lower bound. The process is iterated until no infeasibility is detected. The lower bound found is denoted LB_{ER}.

Baptiste et al. [10] have proved that the only relevant time intervals $[t_1, t_2]$ that need to be considered are those where $t_1 \in T_1$ and $t_2 \in T_2$ such as $t_1 < t_2$, $T_1 = \{e_i, i \in V\} \cup \{l_i, i \in V\} \cup \{e_i + s_i, i \in V\}$ and $T_2 = \{l_i + s_i, i \in V\} \cup \{e_i + s_i, i \in V\} \cup \{l_i, i \in V\}$. Therefore, the satisfiability test algorithm runs in $O(n^3)$.

2.3 Bin-packing lower bounds and Energetic Reasoning

The Energetic Reasoning approach can be extended and enforced using bin packing modeling. For each interval $[t_1, t_2]$, one can consider the bin packing instance defined by bins of size $t_2 - t_1$ and items of weights $W(i, t_1, t_2)$. The instance is feasible if all the sub-instance of bin-packing are feasible using m bins. This can be improved furthermore embedding the graph of incompatibilities into this instance. The two lower bounds are denoted respectively LB_{ERBPP} and LB_{ERBPPC}.

2.4 Lower bounds based on problem decomposition techniques

In this section, we try to integrate all the constraints and solve the problem for some subset of clients using an exact method. Lets $S \subset V$ be a set of clients from the graph G where all the travel costs satisfy the triangle inequality since $\delta_{i,k} + s_k + \delta_{k,j} \geq \delta_{i,j}\ \forall i,j,k \in V$ and $q_i \geq 0\ \forall i \in V$.

Property 1. If k is the minimum number of vehicles needed to serve the customers of the subset S such as $S \subset V$ then k is a valid lower bound of the number of vehicle needed to serve all the customers in the set V.

Using the model presented in the paper, we try to solve the problem defined by the graph $G_S = (S^+, E_{S^+})$ for every subset $S \subset V$. We denote the set of clients subsets by Ω. Since the number of subset is exponential, Ω should be reduced to include just the most interesting ones. Thus, we use the same intervals defined in Section 2.2 to trap the clients as following: for every interval $[t_1, t_2]$, if $W(i, t_1, t_2) > 0$ then the client i is inserted into the subset which represents that interval. We consider only the subsets that have a cardinality bigger than the best lower bound found so far.

$$\Omega = \{\ S_{t_1,t_2} \mid S_{t_1,t_2} = \{i \mid W(i,t_1,t_2) > 0\} \text{ and } |S_{t_1,t_2}| > LB_{best}\ \} \tag{2.3}$$

Every execution should be stopped once it hits an upper bound smaller than or equal to the best overall lower bound found so far.

2.5 Preprocessing

We describe the preprocessing steps applied on VRPTW. The objective is to improve the efficiency of the bounding algorithms. First, travel times $\delta_{i,j}$ between any customers i and j are updated to eliminate the waiting time at customer j:

$$\delta_{i,j} \leftarrow \max(\delta_{i,j}, e_j - (l_i + s_i)) \quad \forall i \in V \; \forall \; j \in V^+ \tag{2.4}$$

Then, time windows' width is reduced using the following basic conditions:

$$e_i = \max\{e_i, e_0 + \delta_{0,i}\} \quad \forall i \in V \tag{2.5}$$

$$l_i = \min\{l_i, l_0 - \delta_{i,0}\} \quad \forall i \in V \tag{2.6}$$

$$e_i = \max\{e_i, \min_{(j,i)\in V}\{e_j + \delta_{j,i}\}\} \quad \forall i \in V \tag{2.7}$$

$$e_i = \max\{e_i, \min\{l_i, \min_{(i,j)\in V}\{e_j - \delta_{i,j}\}\}\} \quad \forall i \in V \tag{2.8}$$

$$l_i = \min\{l_i, \max\{l_i, \max_{(j,i)\in V}\{l_j + \delta_{j,i}\}\}\} \quad \forall i \in V \tag{2.9}$$

$$l_i = \min\{l_i, \max_{(i,j)\in V}\{l_j - \delta_{i,j}\}\} \quad \forall i \in V \tag{2.10}$$

The travel time δ_{ij} is set to infinity if j cannot be served after i due to its time window. This aims to reduce the number of potential successors of customer i.

$$if(e_i + s_i + \delta_{i,j} > l_j) \quad then \qquad \delta_{i,j} \leftarrow \infty \qquad \forall i \in V^+ \; \forall \; j \in V^+ \setminus \{i\} \tag{2.11}$$

2.5.0.c Preprocessing based on Energetic Reasoning

We define a slack of an activity i on a time interval $[t_1, t_2]$ as the available energy that can be used to process i. We denote by Slack(i, t_1, t_2) the slack of $V\setminus\{i\}$. It

is calculated according to the following equation:

$$\text{Slack}(i, t_1, t_2) \quad = \quad m \cdot (t_2 - t_1) - W(t_1, t_2) + W(i, t_1, t_2) \tag{2.12}$$

According to these definitions, the slack allows adjusting the time-bounds of activities. If the right work of the activity $i \in V$ is strictly greater than $\text{Slack}(i, t_1, t_2)$, the activity cannot be right-shifted, i.e., it cannot start at its latest start time. Hence, only a part of activity i, smaller than or equal to the slack $\text{Slack}(i, t_1, t_2)$ can be processed on $[t_1, t_2]$. More precisely, the adjustment scheme can be defined as follows:

Proposition 1. Release date adjustments

$\forall i \in V$, if $W_{left}(i, t_1, t_2) > \text{Slack}(i, t_1, t_2)$ then $e_i \leftarrow \max(e_i, t_2 - \text{Slack}(i, t_1, t_2))$

Proposition 2. Latest start time adjustments

$\forall i \in V$, if $W_{right}(i, t_1, t_2) > \text{Slack}(i, t_1, t_2)$ then

$$l_i \leftarrow \max(l_i, t_1 - s_i - \text{Slack}(i, t_1, t_2))$$

2.5.0.d Preprocessing based on shaving techniques

Better adjustments can be obtained by using "shaving techniques" [18]. The principle of this process is to assign restricted values to a variable. If an inconsistency arises, the assigned values are suppressed from the variable domain. We apply two types of shaving techniques on VRPTW: the first one aims to solve disjunctions by eliminating arcs and reducing the number of decision variables (see Algorithm 2.1) and the second tries to reduce the time windows of customers using a dichotomy strategy (see Algorithm 2.2).

Algorithm 2.1: Solving disjunctions

```
Data: I: VRPTW instance;
UB an upper bound for I;
1  begin
2      m ← UB − 1;
3      foreach i ∈ V do
4          foreach (i, j) ∈ E    where δ_{i,j} is big do
5              fix (i, j) to 1 on the conflict matrix;
6              if not feasible then
7                  fix (i, j) to 0;
8                  δ_{i,j} ← ∞;
9              else
10                 unfix (i, j)
11         foreach (i, j) ∈ E    where T_{ij} is small do
12             fix (i, j) to 0 on the conflict matrix;
13             if not feasible then
14                 fix (i, j) to 1;
15             else
16                 unfix (i, j)
```

Algorithm 2.2: Reducing Time windows

Data: I: VRPTW instance;
UB an upper bound for I;

```
1  begin
2      m ← UB − 1;
3      foreach i ∈ V do
           // Dichotomy
4          repeat
5              EST ← e_i,   LST ← l_i;
6              if EST = LST then next i ;
               // Try to adjust to the right
7              e_i = EST + (LST−EST)/2,   LST = l_i;
8              if not feasible then
9                  e_i = EST,   l_i = LST − (LST−EST)/2;
10                 improved ← true;
11             else
12                 e_i = EST,   l_i = LST;
13             if improved then continue;
               // Try to adjust to the left
14             e_i = EST,   l_i = LST − (LST−EST)/2;
15             if not feasible then
16                 e_i = EST + (LST−EST)/2,   l_i = LST;
17                 improved ← true;
18             else
19                 e_i = EST,   l_i = LST;
20         until (no improvement);
```

2.6 Numerical results

We tested our algorithms on the well known instances of Solomon [81] and Gehring and Homberger [37]. The benchmark comprises 6 sets ($R1$, $C1$, $RC1$, $R2$, $C2$, $RC2$). Each data set contains $25, 50, 100, 200, 400, 600, 800$ and 1000 customers who have specific euclidean coordinates. Customers' locations are determined using a random uniform distribution for the problem sets $R1$ and $R2$, but are restricted to be within clusters for the sets $C1$ and $C2$. Sets $RC1$ and $RC2$ have a combination of clustered and randomly placed customers. Sets $R1$, $C1$ and $RC1$ have a short scheduling horizon with tight time windows, while $R2$, $C2$ and $RC2$ are based on wide time windows. Our algorithms are coded in C++ using the Standard Template Library (STL) for data structures and IBM Ilog Cplex 12.6 for the linear programming. All the experiments were conducted on an Intel Xeon 2.67GHz.

Table 2.1 and Table 2.2 compare the performance of our Energetic Reasoning bounds: LB_{ER}, LB_{ERBPP}, LB_{ERBPPC} and $LB_{SubSets}$ to the elementary bounds present in the literature: LB_{Clique}, $LB_{Capacity}$ and LB_{BP}. The column $BestUB$ represents the overall best-published upper bounds. The maximum of the lower bounds is reported in column $BestLB$. In $AvgGAP$, we present the average gap between $BestUB$ and $BestLB$. Detailed results are presented in Appendix B.

In general, the proposed techniques give the minimum number of vehicles of 377 instances among the 468 instances tested and give near optimal solution for the rest. Beside the results of the methods presented and discussed in Appendix A, the $SubSets$ method improves the lower bounds of 11 instances of 25 customers, 17 for 50 customers, 10 for 100 customers and 1 for 200 or 400 customers. The method needs huge cpu time one considering the bigger instances.

Data Set	n	*Classical*						*New*								*BestLB*	*BestUB*	*AvgGAP*
		Clique		*Capacity*		*BP*		*ER*		*ERBPP*		*ERBPPC*		*SubSets*				
		LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU			
C1	25	1.89	0	3	0.02	2	0	3	0	3	0	3	0	3	0	3	3	0
C2	25	1	0	1	0.02	1	0	1.13	0	1.13	0	1.13	0	1.13	0	1.13	1.13	0
R1	25	3.58	0	2	0.02	3.25	0	3.92	0	3.92	0.01	4	0.34	4.5	200.83	4.5	4.67	0.17
R2	25	1	0	1	0.01	1	0	1.09	0	1.09	0	1.09	0.04	1.27	0.01	1.27	1.27	0
RC1	25	2.75	0	3	0.01	2.25	0	3.13	0	3.13	0	3.13	0.04	3.25	0	3.25	3.25	0
RC2	25	1	0	1	0.01	1	0	1	0	1	0	1	0.04	1.25	0	1.25	1.25	0
C1	50	3	0	5	0.03	4	0	5	0	5	0	5	0	5	0	5	5	0
C2	50	1	0	2	0.02	2	0	2	0	2	0	2	0	2	0	2	2	0
R1	50	5.25	0	4	0.02	5.17	0	6.33	0.02	6.33	0.09	6.42	64.41	6.67	6635.42	6.67	7.42	0.75
R2	50	1	0	1	0.02	1.18	0	1.27	0.01	1.27	0.06	1.27	2.39	1.55	7094.08	1.55	2	0.45
RC1	50	4.25	0	5	0.03	4.13	0	5.25	0.01	5.25	0.06	5.25	15.3	6.25	5631.28	6.25	6.5	0.25
RC2	50	1.13	0	1	0.03	1	0	1.13	0.01	1.13	0.06	1.13	2.48	1.88	11685.6	1.88	2	0.13
C1	100	4.89	0	10	0.03	8	0	10	0	10	0	10	0	10	0	10	10	0
C2	100	1.38	0	3	0.03	3	0	3	0	3	0	3	0	3	0	3	3	0
R1	100	7.92	0	8	0.03	8.33	0	10.42	0.11	10.42	0.48	10.42	367.74	10.75	12521.39	10.75	11.92	1.17
R2	100	1.18	0	2	0.03	2	0	2.09	0.07	2.09	0.32	2.09	29.18	2.27	8226.61	2.27	2.73	0.45
RC1	100	6.38	0	9	0.03	7.63	0	9.63	0.1	9.63	0.47	9.63	103.78	10.38	12425.09	10.38	11.5	1.13
RC2	100	1.13	0	2	0.03	2	0	2.13	0.11	2.13	0.45	2.13	35.48	2.5	12957.95	2.5	3.25	0.75

Table 2.1 – Average lower bound results and CPU times for Solomon [81] instances

Data Set	n	*Classical*						*New*								*BestLB*	*BestUB*	*AvgGAP*
		Clique		*Capacity*		*BP*		*ER*		*ERBPP*		*ERBPPC*		*SubSets*				
		LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU			
C1	200	8.8	0	18	0.09	15	0	18.3	0.39	18.3	1.86	18.3	456.89	18.4	5438.07	18.4	18.9	0.5
C2	200	2.1	0	6	0.1	6	0	6	0	6	0	6	0	6	0	6	6	0
R1	200	10.4	0	18	0.09	7.3	0	18.2	0	18.2	0	18.2	0	18.2	0	18.2	18.2	0
R2	200	1.6	0	4	0.09	2	0	4	0	4	0	4	0	4	0	4	4	0
RC1	200	6.8	0	18	2.5	6.5	0	18	0	18	0	18	0	18	0	18	18	0
RC2	200	1.9	0	4	0.09	2	0	4	0.14	4	0.76	4	151.09	4	3604.49	4	4.3	0.3
C1	400	16.9	0.01	36	0.4	26.5	0.01	36.5	2.64	36.5	12.56	36.5	1440.06	36.7	7207.52	36.7	37.6	0.9
C2	400	2.5	0.01	11	0.4	11	0.01	11.2	2.43	11.2	12.38	11.2	1440.04	11.2	7204.93	11.2	11.6	0.4
R1	400	17.9	0.01	36	0.4	11.7	0.01	36.4	0	36.4	0	36.4	0	36.4	0	36.4	36.4	0
R2	400	2.4	0.01	8	0.38	2.7	0.01	8	0	8	0	8	0	8	0	8	8	0
RC1	400	12.8	0.01	36	0.38	10.8	0.01	36	0	36	0	36	0	36	0	36	36	0
RC2	400	3.1	0.01	8	0.39	2.8	0.01	8	1.07	8	5.6	8	720.02	8	3608.16	8	8.4	0.4
C1	600	24.2	0.03	56	1.02	40.4	0.02	56.4	4.81	56.4	30.83	56.4	1080.14	56.4	5406.35	56.4	57.2	0.8
C2	600	4.3	0.02	17	1.1	15.9	0.02	17.1	5.54	17.1	28.76	17.1	1080.1	17.1	5403.12	17.1	17.4	0.3
R1	600	28.1	0.03	54	1.16	13.9	0.02	54.5	0	54.5	0	54.5	0	54.5	0	54.5	54.5	0
R2	600	4.3	0.02	11	1.15	3.4	0.02	11	0	11	0	11	0	11	0	11	11	0
RC1	600	19.7	0.05	55	1.21	12.2	0.02	55	0	55	0	55	0	55	0	55	55	0
RC2	600	4.7	0.02	11	1.21	2.9	0.02	11	3.49	11	18.47	11	720.08	11	3606.24	11	11.4	0.4
C1	800	34.4	0.07	72	2.38	49.3	0.05	72.8	16.31	72.8	93.69	72.8	1441.34	72.8	7200.43	72.8	75	2.2
C2	800	5.7	0.05	22	2.43	21	0.05	22.2	44.61	22.2	221.16	22.2	3242.16	22.2	16202.36	22.2	23.3	1.1
R1	800	35.3	0.07	72	2.81	15.8	0.06	72.8	0	72.8	0	72.8	0	72.8	0	72.8	72.8	0
R2	800	5.3	0.06	15	2.67	3.5	0.05	15	0	15	0	15	0	15	0	15	15	0
RC1	800	27.5	0.41	72	100.27	14.9	0.06	72	0	72	0	72	0	72	0	72	72	0
RC2	800	6.6	0.06	15	2.61	3.5	0.05	15	9.03	15	44.38	15	720.56	15	3600.12	15	15.4	0.4
C1	1000	44.6	0.32	90	4.77	58	0.1	91	32.25	91	177.37	91	1443.45	91	7200.39	91	94.1	3.1
C2	1000	7.9	0.09	28	4.4	25.1	0.08	28.1	63.56	28.1	275.5	28.1	2163.64	28.1	10807.33	28.1	28.8	0.7
R1	1000	44.5	0.13	91	4.96	19.5	0.1	91.9	0	91.9	0	91.9	0	91.9	0	91.9	91.9	0
R2	1000	6.5	0.1	19	5.03	4	0.09	19	0	19	0	19	0	19	0	19	19	0
RC1	1000	31.5	0.79	90	5.72	17.7	0.12	90	0	90	0	90	0	90	0	90	90	0
RC2	1000	7.8	0.11	18	5.68	4.1	0.09	18	8.58	18	40.57	18	360.82	18	1801.3	18	18.2	0.2

Table 2.2 – Average lower bound results and CPU times for Gehring and Homberger [37] instances

2.7 Conclusion

In this chapter, we introduced an overview of several combinatorial optimization methods which can be used to get lower bounds for the Vehicle Routing Problem with Time Windows (VRPTW). Some of them were presented in details in the paper attached in Appendix A. Investigating the concept of Energetic Reasoning, we were able to propose new lower bounding techniques based on the transformation of m-VRPTW instance to PMSP. The numerical results confirm the contribution brought by the new proposed techniques. With a very fast computing time, we were able to provide the exact number or a reasonable approximation of the minimum number of vehicles required to visit all the customers. This suggests that our lower bounds techniques can quickly produce a good estimation of the fleet size. A challenging area for future research is to develop an exact method using the proposed lower bound procedures. A new method based on the problem decomposition was also tested and proved its efficiency on the instances up to 400 customers. As a future work, we plan to test these several approaches on more complex variants of VRPTW, those with non homogeneous vehicles for example or on the instances of the Pickup and Delivery with Time Windows. Since this problem has a considerable number of incompatibilities between the customers due to the precedence constraints, this may be promising when solving our bin-packing problems.

3 | Particle Swarm Optimization for VRPTW

All art is but imitation of nature.

Lucius Annaeus Seneca

3.1 PSO based algorithm 32
3.1.1 Solution representation and evaluation 32
3.1.2 Crossover and position update 33
3.2 Initialization algorithm 37
3.3 Best insertion heuristic 38
3.4 Local search 39
3.5 Parameter configuration and experimentation 41
3.6 Conclusion 46

3.1 PSO based algorithm

In this chapter, we present a Particle Swarm Optimization algorithm for VRPTW. PSO is a swarm intelligence technique which takes inspiration from the collective behavior of wild animals in the nature. The proposed PSO works with permutation encoding and uses an adapted split to transform it into a valid solution. A particle position update and a combination with a neighborhood-based local search framework are used.

PSO was first proposed by Kennedy and Eberhart [53] for optimization problem in continuous space. It uses a set of particles called swarm S. Each particle represents one candidate solution and moves through the search space by the adjustment of its trajectory according to its previous best performance and the historical best performance in its neighborhood. This allows high operating efficiency and fast convergence speed. Each particle i memorizes its best known position as x_i^{lbest}. The best known position for the swarm is denoted as x^{gbest}. Our PSO algorithm is presented in Algorithm 3.1. Details about solution representation, position update, initialization algorithm and local searches are described in the next sections.

3.1.1 Solution representation and evaluation

A position in PSO is a permutation $\pi = (s_1, s_2, \dots s_n)$ of all accessible customers in V. Algorithm 3.2 presents a splitting procedure that transforms the permutation into a solution of VRPTW in a similar manner to the split of Prins [76] and Labadi et al. [57]. This algorithm splits the so called giant tour into a minimum number of routes so that the total distance is also minimized. The main idea is to calculate the shortest path which minimizes first the number of arcs then the total distance where an arc represents a feasible path (a potential route). First, we construct an auxiliary graph Π containing the nodes $\{s_0, s_1, s_2, \dots s_n\}$ where $s_0 = 0$. Each

Algorithm 3.1: PSO basic algorithm

Input: S a swarm of N particles;
Output: x^{gbest} best position found;

```
initialize and evaluate each particle in S (see Section 3.2);
iter ← 1;
while iter ≤ itermax do
    foreach x_i in S do
        update x_i (see Section 3.1.2);
        if rand(0,1) < p_ls then
            apply local search on x_i (see Section 3.4 );
        evaluate x_i (see Section 3.1.1);
        determine x_j^lbest ∈ S to be updated ;
        if (update is applied) then
            iter ← 1;
        else
            iter ← iter + 1;
```

arc in the graph Π represents a feasible route. i.e. if there exists an arc (s_i, s_j) between s_i and s_j where $i < j$, it represents the feasible route visiting the customers $s_{i+1}, \ldots s_j$ in this order. The cost of this arc is the travel time of such a route to which we add a big $\mathcal{M}$ in order to prioritize the minimization of the number of routes. The fitness of the permutation is determined by calculating the shortest path from s_0 to s_n using Ballman's algorithm [12]. An example is illustrated in Figure 3.1.

3.1.2 Crossover and position update

The original PSO algorithm was designed to deal with continuous function. Thus, it is not suitable for combinatorial problems with a discrete solution space. We have experimented different position updating strategies. The particle position update in our PSO is a recombination of three positions x_i^{lbest}, x^{gbest} and x_i according to inertia, cognitive and social parameters. This takes inspiration from the works of [23] and [65]. The first tested approach was to use a crossover in a

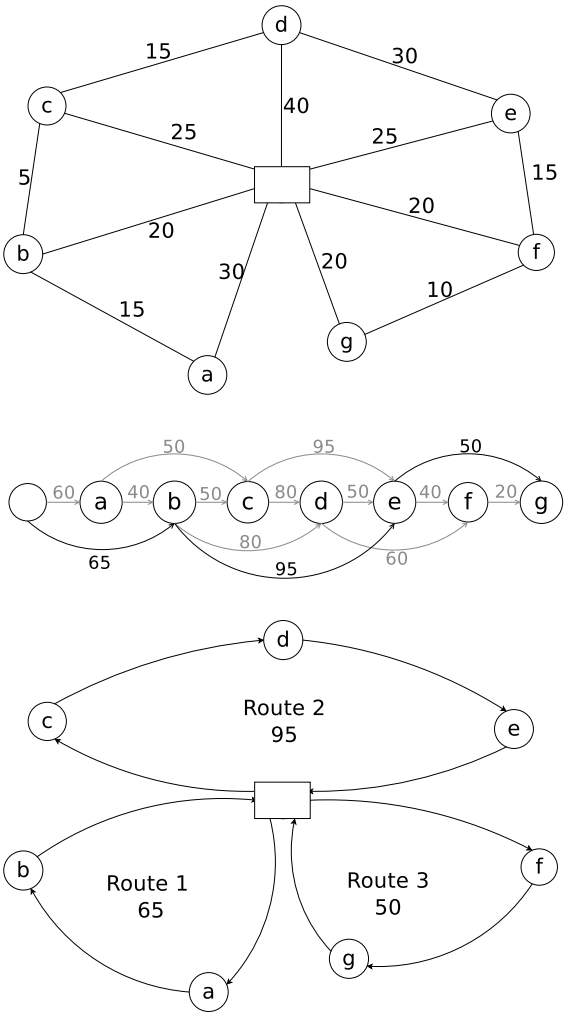


Figure 3.1 – Example of a sequence evaluation.

similar way to the genetic one, a subsequence of size l may be extracted from each position and combined into the new position. A set $\mathcal{X}$ of extracted customers would be used to avoid duplications. With respect of three parameters $inertia$, c_1 and c_2, the size l is equal to:

- $l_i \leftarrow inertia \cdot n$ when considering x_i.
- $l_l \leftarrow n \cdot (1 - inertia) \cdot \frac{r1 \cdot c_1}{r1 \cdot c_1 + r2 \cdot c_2}$ when considering x_i^{lbest} where $r1$, $r2$ are two real numbers generated randomly between $[0, 1[$ with a uniform distribution.
- $l_b \leftarrow n - l_i - l_l$ when considering x^{gbest}.

Algorithm 3.2: Split algorithm for VRPTW.

Input: π, the permutation;
Output: $cost$, the solution cost;

$V_0 \leftarrow 0$;
for $i \leftarrow 1$ to n do $V_i \leftarrow +\infty$;
for $i \leftarrow 1$ to n do
  $time \leftarrow 0$; $load \leftarrow 0$; $distance \leftarrow 0$;
  $j \leftarrow i$; $stop \leftarrow false$;
  repeat
    $load \leftarrow load + q_{s_j}$;
    if $i = j$ then
      $time \leftarrow max(\delta_{0,s_j}, e_{s_j}) + s_{s_j} + \delta_{s_j,0}$;
      $distance \leftarrow \delta_{0,s_j} + \delta_{s_j,0}$;
    else
      $time \leftarrow max(cost - \delta_{s_{j-1},0} + \delta_{s_{j-1},s_j}, e_{s_j})$;
      $distance \leftarrow distance - T_{s_{j-1},0} + \delta_{s_{j-1},s_j}$;
    if $time > l_{s_j}$ or $load > C$ then
      $stop \leftarrow true$;
    else
      $time \leftarrow time + s_{s_j} + \delta_{s_j,0}$;
      $distance \leftarrow distance + \delta_{s_j,0}$;
      if $V_{i-1} + \mathcal{M} + distance < V_j$ then
        $V_j \leftarrow V_{i-1} + \mathcal{M} + distance$;
      $j \leftarrow j + 1$;
  until $stop$ or $(j > n)$;
  $cost \leftarrow V_n$;

We noticed that this type of cross is inefficient when optimizing the fleet size. Indeed, when splitting the resulting particle we often obtain solution with a greater number of routes. Moreover, we spent more time in the local search procedure to improve the quality of this solution. We develop a new update strategy which takes into consideration the number of vehicles and tries to inherit at most the same number from the parent. The algorithm, described in Algorithm 3.3, builds the offspring from the three parents by inheriting the best routes. First, a random parent is selected according to the weights described before (l_i for x_i, l_l for x_i^{lbest} and l_b for x^{gbest}). Then, from the solution presented by its parents, the best route

Algorithm 3.3: Cross algorithm for VRPTW.

Input: x_i, x_i^{lbest}, x^{gbest} the selected parents.
Output: s the offspring;

$forbid_i, forbid_l, forbid_b \leftarrow false$;
repeat
  $r \leftarrow \mathcal{U}(0, n)$; /* generate a random number */
  switch $r \in$ do
    case $[0, l_i[$ and $forbid_i = false$
      $rbest \leftarrow$ the best route in $Split(x_i)$;
      $forbid_i \leftarrow true$;
      $forbid_l, forbid_b \leftarrow false$;
      break;
    case $[l_i, l_i + l_l[$ and $forbid_l = false$
      $rbest \leftarrow$ the best route in $Split(x_i^{lbest})$;
      $forbid_l \leftarrow true$;
      $forbid_i, forbid_b \leftarrow false$;
      break;
    case $[l_i + l_l, n[$ and $forbid_b = false$
      $rbest \leftarrow$ the best route in $Split(x^{gbest})$;
      $forbid_b \leftarrow true$;
      $forbid_i, forbid_l \leftarrow false$;
      break;
  add $rbest$ to s ;
  remove c from x_i, x_i^{lbest}, x_i^{lbest} $\forall c \in rbest$;
until stop condition;
Insert remaining customers using Best insertion;

r with the minimal travel time per client (according to the function: $\text{TD}_r/\text{Size}_r$) is considered and added to the offspring where Size_r denoted the number of clients on the route r. This should privilege condensed routes. All the customers on this route should be removed from the two other parents as well. For the purpose of varying the offspring, the selected parent is not considered for the next step. This strategy avoids transmitting routes always from the same parent and introduces some diversity [50]. The algorithm stops by serving all the customers or by hitting the maximum number of routes of the parents. In the latter case, the remaining customers are inserted using the best insertion heuristic described in Section 3.3.

After moving to the new position x_i, the algorithm will search for an appropriate particle j in the swarm and update x_j^{lbest}. The update rule is similar to [23]:

1. The fitness of the new position is better than the worst local best x_{worst}^{lbest}.
2. If there exists a particle j in S such that x_j^{lbest} is similar to x_i, then replace x_j^{lbest} with x_i, otherwise replace x_{worst}^{lbest}.

3.2 Initialization algorithm

The initial population of PSO is generated randomly. A small part is generated using an iterative Destruction/Repair algorithm as follows. Starting from a random initial solution, a random number of customers are removed. Then, a best insertion algorithm is applied to repair the solution. These two steps are repeated until n^2 iterations are performed without improvement. This routine is described in Section 3.3. After n iterations without improvement the route with the minimum number of customers is the destructed part. During the algorithm, all the routes are saved in a pool $\mathcal{T}$ with their fitness TD_k $\forall k \in \mathcal{T}$. The $0-1$ linear program described by (3.1), (3.2) and (3.3) presents a set covering problem. The objective is to find a VRPTW solution at minimum cost using the collected routes in the pool. θ_k indicates whether route k is selected ($\theta_k = 1$) in the solution or not. Constraints (3.2) guarantee that each customer is served at least once, where a_{ik} indicates that route k contains customer i.

$$\min \sum_{k \in \mathcal{T}} (\mathcal{M}\theta_k + \theta_k \mathrm{TD}_k) \tag{3.1}$$

subject to:

$$\sum_{k \in \mathcal{T}} \theta_k a_{ik} \geq 1 \qquad \forall i \in V \tag{3.2}$$

$$\theta_k \in \{0,1\} \qquad \forall k \in \mathcal{T} \tag{3.3}$$

The solution may contain duplicated customers. Despite it can be considered as an upper bound, it should be repaired and cleaned up. The $0-1$-LP presented by (3.4 – 3.8) is used for this purpose. $C_{p,k}$ represents the customer at position p in the route k. Variable $z^k_{p,q}$ indicates whether the arc between the position p and q is to be considered. Constraints (3.5) ensure that every customer is served only once ($\mathcal{A}_i$ defines the set of positions of customer i). Constraints (3.6) ensure that $z^k_{p,q} = 1$ if $\xi^k_p = \xi^k_q = 1$ and $\xi^k_r = 0 \, \forall r | p < r < q$.

$$\min \quad \sum_{p,q,k} z^k_{p,q} T_{C_{p,k} C_{q,k}} \tag{3.4}$$

subject to:

$$\sum_{(p,k) \in \mathcal{A}_i} \xi^k_p = 1 \qquad \forall i \in V \tag{3.5}$$

$$2 + z^k_{p,q} + \sum_{r=p+1}^{q-1} \xi^k_r \geq \xi^k_p + \xi^k_q \qquad \forall k \in \mathcal{T} \; \forall p,q < \text{Size}_k \text{ and } p < q \tag{3.6}$$

$$\xi^k_p, z^k_{p,q} \in \{0,1\} \qquad \forall k \in \mathcal{T} \; \forall p,q < \text{Size}_k \tag{3.7}$$

$$\mathcal{A}_i = \{(p,k) \text{ such that } C_{p,k} = 1\} \qquad \forall i \in V \tag{3.8}$$

3.3 Best insertion heuristic

The best insertion algorithm is a constructive heuristic which repairs a partial solution by inserting the customers one by one into the best evaluated positions. In order to evaluate an insertion in a constant complexity, we record for each served

customer i in route t its waiting time Wait_i and the maximum delay allowed for the start of its service noted by MaxShift_i. The insertion of customer c after the position p in the route k is evaluated using the function: $\text{InsertionCost}_c^{p,k} = \delta_{p,c} + \delta_{c,p+1} - \delta_{p,p+1}$. The feasible insertion that minimizes the cost is performed.

3.4 Local search

Whenever a solution (new position) is found, it has a p_{ls} probability of being improved using a local search. Our local search contains five neighborhoods and is executed in a parallel way as shown in Algorithm 3.4. Each neighborhood operator is selected with a given probability $\mathcal{P}_w$. The algorithm ends by exploring at least each neighborhood once. The probabilities of choosing the neighborhoods are adjusted according to their success during the PSO algorithm. This helps to identify operators suited for each structure of the instances. We noticed, for example, that the Or-opt rarely improve the solution for instances with small time windows. In this case, its probability should be reduced compared to other operators.

Algorithm 3.4: Local search for VRPTW.

```
Input: X : the solution to improve
Output: X the new solution if improved
W_0 ← {2-opt*, Or-opt, Swap, Shift, Destruction/Repair};
W ← W_0;
repeat
    Select a random neighborhood w from W according to its probability P_w;
    if w(X) = true then
        W ← W_0;
        Adjust probabilities ;
    else
        W ← W\{w};
until W = ∅;
```

2-opt* operator:

We explore the possibility of exchanging two links with two others from different routes to find a local improvement. Figure 3.2 illustrates an example of such operator. The 2-opt* move is tested on a random set of customers and the best one is selected and applied.

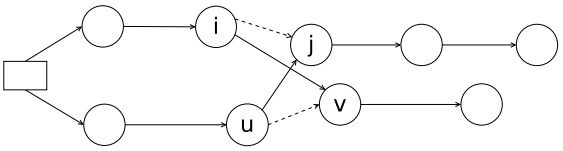


Figure 3.2 – Example of the 2-opt* move
exchange of links (i, j), (u, v) for links (i, v), (u, j).

Or-opt operator:

In this operator, we look for the possibilities of moving a sequence of (1, 2 or 3) customers to another position in the same route. In a similar way to 2-opt*, we test it on a random selection at the beginning then the best move is applied. An example is illustrated in Figure 3.3.

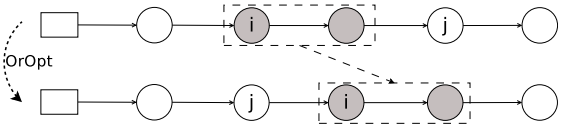


Figure 3.3 – Example of the Or-opt move
moving sequence (i, i+1) after customer j.

Swap operator:

This operator is applied on the permutation of customers. It evaluates the possibility of exchanging the position of two customers in this set. The first improvement of the particle is considered.

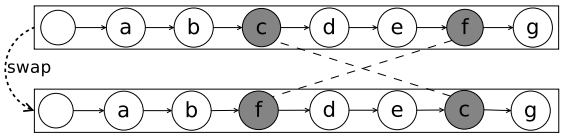


Figure 3.4 – Example of the swap move, between customers c and f.

Shift operator:

Similarly to the Swap operator, it is also applied on the permutations. It evaluates the possibility of moving a customer into another position on the particle. The first improvement is also considered in this local search.

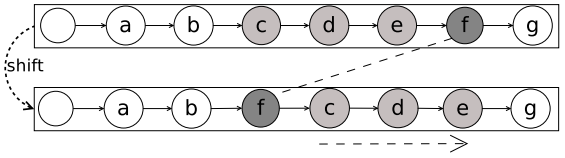


Figure 3.5 – Example of the shift move, on the customer f.

Destruction/repair operator:

This move is similar to the initialization algorithm and uses the same Best Insertion heuristic described in Section 3.3. It evaluates the possibility of removing a random number of customers from a solution and then rebuilding it with the best insertion algorithm. The destruction/repair heuristic runs iteratively and stops after n iterations without improvements.

3.5 Parameter configuration and experimentation

We first tested our algorithms on the well known instances of Solomon and Desrosiers [82]. The benchmark comprises 6 sets ($R1$, $C1$, $RC1$, $R2$, $C2$, $RC2$). Each data set contains $25, 50$ and 100 customers which have specific euclidean coordinates. Customers' locations are determined using a random uniform distribution for the

problem sets $R1$ and $R2$, but are restricted to be within clusters for the sets $C1$ and $C2$. Sets $RC1$ and $RC2$ have a combination of clustered and randomly placed customers. Sets $R1$, $C1$ and $RC1$ have a short scheduling horizon with tight time windows, while $R2$, $C2$ and $RC2$ are based on wide time windows. Later, an experimentation has been done on bigger instances. Homberger and Gehring [46] generated many other instances using the same characteristics and considering a larger number of customers. Their benchmark contains 10 instances for each type and includes sets with $200, 400, 600, 800$ and 1000 customers. Our algorithms are coded in C++ using the Standard Template Library (STL) for data structures and IBM Ilog Cplex 12.6 as MIP solver. All experiments were conducted on an Intel Xeon 2.67GHz.

We present preliminary results of our method compared to the literature on the instances of Solomon and Desrosiers [82] in Table 3.1 and for the large scale instances of Homberger and Gehring [46] in Tables 3.2 to 3.6. Detailed results are presented in Appendix C. The first column displays the instance set. The next columns compare the average of the best results including the number of vehicles (NV) and the total distance (TD) as well as the Cpu time. We refer by CNV and CTD the sum of the results over all the instances. The state-of-the-art methods shown are coded as:

- ALNS for the ALNS of Pisinger and Ropke [73],
- EP-GEC for the Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains of Lim and Zhang [64],
- BP for the branch-and-price of Prescott-Gagnon et al. [75],
- ArcEA for the Arc-guided evolutionary algorithm of Repoussis et al. [78],
- HGSADC for the hybrid genetic algorithm of Vidal et al. [93],
- EAMA for the penalty-based edge assembly memetic algorithm of Nagata et al. [68].

We decided to reuse most of the parameter values that were chosen in the similar work of Dang et al. [23]. The parameters $c1$ and $c2$ are equal hence the probability of moving to local or global best is similar.

- N, the population size, is set to 40,
- K, the part of the population initialized using the heuristic, is set to 5,
- $itermax$, the maximum number of iterations without improvements, is set to n,
- p_{ls}, the local search probability, is set to $1 - \frac{iter}{itermax}$,
- $c1$ and $c2$ are set to 0.5,
- *inertia* parameter is set to 0.1.

Table 3.1 – Results on Solomon and Desrosiers [82] instances.

Data	ALNS		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	10	828.38	10	828.38	10	828.38	10	828.38	10	828.38	10	828.38
C2	3	589.86	3	589.86	3	589.86	3	589.86	3	589.86	3	589.86
R1	11.92	1 212.39	11.92	1 210.34	11.92	1 210.34	11.92	1 211.49	11.92	1 210.34	11.92	1 212.79
R2	2.73	957.72	2.73	955.74	2.73	952.08	2.73	952.05	2.73	951.03	2.73	952.97
RC1	11.5	1 385.78	11.5	1 384.16	11.5	1 384.72	11.5	1 384.81	11.5	1 384.16	11.5	1 384.17
RC2	3.25	1 123.49	3.25	1 119.44	3.25	1 119.45	3.25	1 119.4	3.25	1 119.24	3.25	1 119.37
CNV	405		405		405		405		405		405	
CTD	57 332		57 240		57 216		57 196		57 187		57 238	
Cpu	2.5		30		17.9		2.68		5		20.32	

Table 3.2 – Results on Homberger and Gehring [46] instances with 200 customers.

Data	EP-GEC		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	18.9	2 726.11	18.9	2 718.77	18.9	2 721.9	18.9	2 718.41	18.9	2 718.41	18.9	2 727.29
C2	6	1 834.24	6	1 831.59	6	1 833.36	6	1 831.59	6	1 831.64	6	1 833.41
R1	18.2	3 639.6	18.2	3 615.69	18.2	3 640.11	18.2	3 613.16	18.2	3 612.36	18.2	3 664.39
R2	4	2 950.09	4	2 937.67	4	2 941.99	4	2 929.41	4	2 929.41	4	2 937.46
RC1	18	3 205.51	18	3 192.56	18	3 224.63	18	3 180.48	18	3 178.68	18	3 294.72
RC2	4.3	2 574.1	4.3	2 559.32	4.3	2 554.33	4.3	2 536.2	4.3	2 536.22	4.3	2 546.34
CNV	694		694		694		694		694		694	
CTD	169 296		168 556		169 163		168 092		168 067		170 036	
Cpu	93.2		53		90		8.4		4.1		60.06	

Table 3.3 – Results on Homberger and Gehring [46] instances with 400 customers.

Data	EP-GEC		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	37.6	7 229.04	37.6	7 182.75	37.6	7 179.71	37.6	7 170.47	37.6	7 175.72	38	7 438.68
C2	11.7	3 942.93	11.9	3 874.58	11.7	3 898.02	11.6	3 952.95	11.7	3 899	12	3 859.65
R1	36.4	8 489.53	36.4	8 420.52	36.4	8 413.23	36.4	8 402.57	36.4	8 403.24	36.4	9 154.6
R2	8	6 271.57	8	6 213.48	8	6 149.49	8	6 152.92	8	6 148.57	8	6 221.34
RC1	36	8 005.25	36	7 940.65	36	7 931.66	36	7 907.14	36	7 922.23	36.3	8 517.48
RC2	8.5	5 431.15	8.6	5 269.09	8.4	5 293.74	8.5	5 215.21	8.4	5 297.86	8.9	5 290.76
CNV	1 382		1 385		1 381		1 381		1 381		1 396	
CTD	393 695		389 011		395 936		388 697		388 466		404 825	
Cpu	295.9		89		180		34.1		16.2		243.01	

Table 3.4 – Results on Homberger and Gehring [46] instances with 600 customers.

Data	EP-GEC		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	57.4	14 103.61	57.4	14 106.03	57.3	14 236.86	57.4	14 058.46	57.4	14 067.34	58.1	14 468.13
C2	17.4	7 725.86	17.5	7 632.37	17.4	7 729.8	17.4	7 594.41	17.4	7 605.07	17.8	8 108.57
R1	54.5	18 381.28	54.5	18 252.13	54.5	18 781.79	54.5	18 023.18	54.5	18 186.24	54.8	21 086.39
R2	11	12 847.31	11	12 808.59	11	12 804.6	11	12 352.38	11	12 330.49	11	13 052.17
RC1	55	16 274.17	55	16 266.14	55	16 767.72	55	16 097.05	55	16 183.95	55.1	18 744.25
RC2	11.5	10 935.91	11.7	10 990.85	11.4	11 311.81	11.5	10 511.86	11.4	10 586.14	12.1	11 171.88
CNV	2 068		2 071		2 066		2 068		2 067		2 089	
CTD	802 681		800 797		816 326		786 373		789 592		866 313	
Cpu	646.9		105		270		99.4		25.3		280.55	

Table 3.5 – Results on Homberger and Gehring [46] instances with 800 customers.

Data	EP-GEC		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	75.4	25 026.42	75.4	25 093.38	75.2	25 911.44	75.4	24 876.38	75.2	25 151.83	77	26 310.85
C2	23.4	11 598.81	23.5	11 569.39	23.4	11 835.72	23.3	11 475.05	23.4	11 447.27	24.5	12 529.77
R1	72.8	31 755.57	72.8	31 797.42	72.8	32 734.57	72.8	31 311.38	72.8	31 492.81	72.8	38 018.25
R2	15	20 601.22	15	20 651.81	15	20 618.21	15	19 933.39	15	19 914.97	15	21 180.79
RC1	72	31 267.84	72	33 170.01	72	33 795.61	72	29 404.32	72	31 278.28	73	33 518.75
RC2	15.6	16 992.79	15.8	16 852.38	15.5	17 536.54	15.4	16 495.82	15.4	16 484.31	16.2	17 228.95
CNV	2 742		2 745		2 739		2 739		2 738		2 785	
CTD	1 372 427		1 391 344		1 424 321		1 334 963		1 357 695		1 487 873	
Cpu	1 269.4		129		360		215		27.6		301.29	

To discuss the effectiveness of our algorithm we refer to `http://www.sintef.no/` which contains an updated list of all the best solutions found in the literature. Concerning the number of vehicles, our algorithm found 262 from the 356 best known solutions. This includes all the instances with up to 200 customers. It found difficulties with the big instances. The algorithm found 83 solutions from the best total travel results. We believe that further research and calibration on

Table 3.6 – Results on Homberger and Gehring [46] instances with 1000 customers.

Data	EP-GEC		BP		ArcEA		HGSADC		EAMA		PSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C1	94.4	41 699.32	94.3	41 783.27	94.2	43 111.6	94.1	41 572.86	94.1	41 748.6	96.8	44 350.18
C2	29.3	16 589.74	29.5	16 657.06	29.3	16 810.22	28.8	16 796.45	29.1	16 534.36	31.1	18 041.28
R1	91.9	48 827.23	91.9	49 702.32	91.9	51 414.26	91.9	47 759.66	91.9	48 369.71	92.6	55 772.58
R2	19	30 164.6	19	30 495.26	19	30 804.79	19	29 076.45	19	29 003.42	19	31 378.11
RC1	90	44 818.54	90	45 574.11	90	46 753.61	90	44 333.4	90	44 860.6	90.1	54 116.96
RC2	18.3	25 064.88	18.5	25 470.33	18.4	25 588.52	18.2	24 131.13	18.3	24 055.31	19	27 079.26
CNV	2 429		2 432		3 428		3 420		3 424		3 486	
CTD	2 071 643		2 096 823		2 144 830		2 036 700		2 045 720		2 307 383	
Cpu	1 864.4		162		450		349		35.3		294.63	

our PSO algorithm may improve the solution qualities presented in this work.

3.6 Conclusion

The Vehicle Routing Problem with Time Windows (VRPTW) consists in determining the routing plan of vehicles with identical capacity in order to supply the demands of a set of customers with predefined time windows. This complex multi-constrained problem has been widely studied due to its industrial, economic and environmental implications.

Motivated by the potential applications, this variant is considered to be one of the main challenging problems in vehicle routing. This chapter presented a new PSO algorithm for VRPTW. It incorporates many new components which deal with VRPTW specific objective. This includes the sequence evaluation, an adapted position update and dedicated local search operators. The numerical results show the competitiveness of these components and their combination. Nevertheless we consider these results, which are preliminary, quite encouraging for further improvements. Another further research issue relates to the application of our PSO algorithm on other objective functions especially the one which takes into consideration only the minimization of the total travel time as the main goal.

4 | Exact methods for VRPTWSyn

> We can not solve our problems with the same level of thinking that created them.
>
> Albert Einstein

4.1 Problem definition . . . 48
4.2 Literature . . . 49
4.3 Problem formulation . . . 51
4.4 New reduced formulation . . . 53
4.5 Constraint programming model . . . 55
4.6 Preprocessing . . . 57
4.7 Additional cuts . . . 57
4.7.1 Incompatibilities and clique cuts . . . 57
4.7.2 Subtour eliminations . . . 59
4.7.3 MIP overall algorithm . . . 59
4.8 Experimentation . . . 60
4.8.1 Travel time . . . 62
4.8.2 Preferences . . . 63
4.8.3 Workload balance . . . 63
4.9 The efficiency of the cuts . . . 66

4.10 Conclusion 67

The vehicle routing problem (VRP) [89] is a widely studied combinatorial optimization problem in which the aim is to design optimal tours for a set of vehicles serving a set of clients or customers geographically distributed and respecting some side constraints. We are interested in a particular variant of VRP, the VRP with time windows and synchronized visits (VRPTWSyn).

In this chapter, we present new and improved exact methods for VRPTWSyn. Our aim is to compare several approaches on the problem. The classical model is first used. Then, we propose a new reduced formulation which uses fewer variables and constraints. We apply some cutting planes and preprocessing techniques on both models. A constraint programming model is also given and tested on VRPTWSyn. A comparison discussion with the literature methods is presented at the end of this chapter.

4.1 Problem definition

In this problem, similarly to VRPTW presented in Chapter 3, each customer is associated with a time window, e.g. a time interval representing the availability of the customer to start receiving the vehicle service. This means that if the vehicle arrives too soon it should wait until the opening of the time window to serve the customer, while late arrivals are not allowed. Additionally, for some customers, more than one visit are allowed, e.g., two visits from two different vehicles, are required to complete the service. Visits associated to a particular customer also need to be synchronized, e.g. having the same start time. Figure 4.1 presents an example of a solution to the problem with 7 customers, 2 points of synchronization and 9 visits to be done with 3 vehicles.

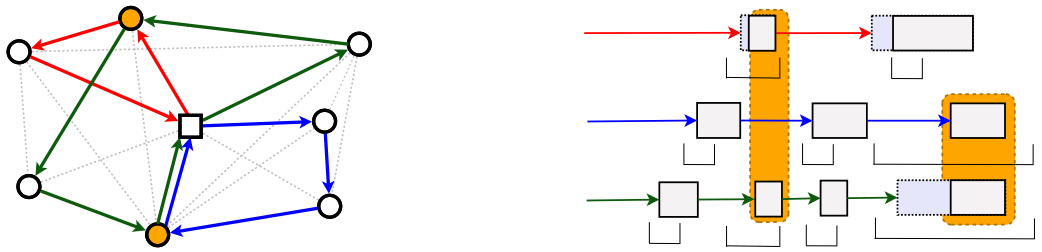


Figure 4.1 – Example of VRPTWSyn solution.

4.2 Literature

VRPTWSyn was first studied in [17] with an application in home-care services for elders. In such services, some operations may require more than one staff to be accomplished, for example the ones demanding heavy lifts or requiring different sets of skills. Timing and coordination are crucial for the success of the operations and the associated temporal constraints must be taken into account during the construction of the schedule.

As an extension of VRP, VRPTWSyn is clearly NP-Hard [62]. To the best of our knowledge, there were only few attempts in the literature to solve this variant of the problem [16, 17] and its generalizations [30, 77]. In those methods, solutions are obtained by approximately or optimally solving integer linear programs. Thus, good solutions often require extensive computational times.

In detail, Bredström and Rönnqvist [17] studied the role of the synchronization constraints found in real world applications and proposed a mathematical formulation of VRPTWSyn. Three objectives of optimization were considered: (i) minimizing the total travel time; (ii) maximizing the sum of preferences, this is due to the fact that each customer may like or dislike being served by a specific vehicle; (iii) minimizing the difference between the longest and the shortest service times among the vehicles in order to optimize the workload balance. For convenience, we shall call the three objectives respectively travel cost, service reward and attribution fairness. In practical applications, they can be addressed simultaneously,

e.g. by aggregating them into a single objective using a set of weights. To avoid negative weights in the aggregation, the minimization of the sum of the negative preferences, e.g. dislike measures, is used instead of the maximization of service reward. The authors proposed a variant of the local-branching approach [34] to solve the problem. A set of benchmark instances was created to evaluate the methods. For analytical purposes, e.g. identify the strengths and weaknesses of the approaches, the three objectives were studied independently on those instances.

As a continuity of [17], the same authors proposed in [16] a branch-and-price algorithm focusing on the first two objectives, the cost and reward. The approach is influenced by the fleet assignment techniques of [48]. In the root node, the synchronization constraints are relaxed and the linear model is basically a set partitioning formulation. Then, during the solving steps, the constraints are strengthened with a branch-and-bound. This was done by repeatedly adjusting the arrival times of the vehicles at the customers' and by branching on time windows. The branch-and-price algorithm was able to solve 44 out of 60 proposed instances to optimality. Later, Labadie et al. [58], in a preliminary work, proposed an iterative local search algorithm for the problem and presented some results on the small and medium sized instances.

In [77], a similar application in home-care services was studied with a particular focus on the reward objective. The authors proposed a clustering scheme based on the customer's preferences and a branch-and-price algorithm to solve the problem. The algorithm was able to find good approximate solutions for instances that were not solved to the optimality. The synchronization constraints were modeled as generalized precedence constraints and then reinforced through branching. This approach was tested on the standard instances of [17], as well as on real-world instances collected from two Danish municipalities.

Later, Dohn et al. [30] presented a generalization of VRPTWSyn, the Vehicle Routing Problem with Time Windows and Temporal Dependencies (VRPTWTD).

In addition to the standard synchronization, more general requirements are studied, such as the maximum/minimum overlap and/or gap between the starting time or ending time of visits. On the objective, only travel cost was considered. A branch-and-cut-and-price algorithm was proposed to solve the generalized problem and was tested on a set of instances derived from the 56 well-known instances of Solomon's benchmark for VRPTW [81]. Note that the results reported in both papers [30] and [77] are general statistics, such as the number of instances being solved to the optimality by the proposed method. Readers interested in other variants or applications are referred to [32, 63, 95]. General perspectives of temporal constraints for vehicle routing can be also found on the survey [31] and on the paper [58].

4.3 Problem formulation

The problem is modeled using an oriented graph $G = (V^+, E)$, where $V^+ = \{0, \ldots, n+1\}$ is the vertex set and E is the arc set. Vertices 0 and $n+1$ are the departure and arrival points respectively. The other vertices $V = \{1, \ldots, n\}$ are the visit points where each one is associated to a customer. A customer can have multiple visit points depending on the number of vehicles needed to deliver the service. For example, if two visit points i and j are associated to the same customer, then the points are superposed and required to be visited by two distinct vehicles. These two visits must be synchronized and we use $[i, j] \in P^{Sync}$ to denote the set of synchronizations. Also, for each i we denote by $P_i^{sync} = \{j \in V^+ \text{ such that } [i, j] \in P^{Sync}\}$ the set of visits to be synchronized with visit i.

A travel time δ_{ij} is associated to each arc $(i, j) \in E$. For convenience, we associate an infinite travel time $\delta_{i,i} = +\infty$ ($\delta_{i,j} = +\infty$) to non-existent arcs. Each visit point i is associated with a service time s_i and a time window $[e_i, l_i]$ where e_i and l_i specify the earliest and the latest possible starting time of the service ($l_i \geq e_i \geq 0$). For a given customer, these data are identical for all the associated

visit points. The departure and arrival points are also associated with a time window $[E, L]$ ($[e_0, l_0] = [e_{n+1}, l_{n+1}] = [0, tmax]$ and $s_0 = s_{n+1} = 0$).

The fleet of vehicles is denoted by the set $K = \{1, \ldots, m\}$. Related to reward objective, $Pref_{ik}$ defines the negative preference of the assignment of vehicle k to the service of the customer at point i. Let $x_{ijk} \in \{0,1\}\ \forall k \in K\ \forall (i,j) \in E$ be the binary routing variables. $x_{ijk} = 1$ if a vehicle k travels along arc (i,j), 0 otherwise. Let t_{ik} be the scheduling variables which represent the time when vehicle k starts the service i, this variable is equal to 0 if visit i is not performed by k. We have the following linear formulation [17].

$$\min \omega_1 \sum_{k \in K} \sum_{(i,j) \in E} \delta_{ij} x_{ijk} + \omega_2 \sum_{k \in K} \sum_{(i,j) \in E} Pref_{ik} x_{ijk} + \omega_3 \mathcal{W} \tag{4.1}$$

$$\sum_{k \in K} \sum_{j:(i,j) \in E} x_{ijk} = 1 \quad \forall i \in V \tag{4.2}$$

$$\sum_{j:(0,j) \in E} x_{0jk} = \sum_{j:(j,n+1) \in E} x_{in+1k} = 1 \quad \forall k \in K \tag{4.3}$$

$$\sum_{j:(i,j) \in E} x_{ijk} - \sum_{j:(j,i) \in E} x_{jik} = 0 \quad \forall k \in K \quad \forall i \in V \tag{4.4}$$

$$t_{ik} + (\delta_{ij} + s_i) x_{ijk} \leq t_{jk} + l_i (1 - x_{ijk}) \quad \forall k \in K \quad \forall (i,j) \in E \tag{4.5}$$

$$e_i \sum_{j:(i,j) \in E} x_{ijk} \leq t_{ik} \leq l_i \sum_{j:(i,j) \in E} x_{ijk} \quad \forall k \in K \quad \forall i \in V \tag{4.6}$$

$$e_i \leq t_{ik} \leq l_i \quad \forall k \in K \quad \forall i \in \{0, n+1\} \tag{4.7}$$

$$\sum_{k \in K} t_{ik} = \sum_{k \in K} t_{jk} \quad \forall [i,j] \in P^{Sync} \tag{4.8}$$

$$\sum_{(i,j) \in E} s_i x_{ijk} - \sum_{(i,j) \in E} s_i x_{ijl} \leq \mathcal{W} \quad \forall k \in K \quad \forall l \in K \setminus \{k\} \tag{4.9}$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in K \quad \forall (i,j) \in E \tag{4.10}$$

$$t_{ik} \geq 0 \quad \forall k \in K \quad \forall i \in V^+$$

$$\mathcal{W} \geq 0$$

The objective (4.1) is to minimize either the total travel time, the sum of

assigned negative preferences or the difference in the attribution of the workload. While constraints (4.2) ensure that each visit point is served by exactly one vehicle, constraints (4.3) ensure that every vehicle starts from the departure and returns to the arrival. Constraints (4.4) guarantee that the same vehicle enters and leaves a given customer. The connectivity of each tour is set by in constraints (4.5) and the time windows are respected with constraints (4.5), (4.6) and (4.7). Constraints (4.8) ensure that synchronized visits start simultaneously. Inequalities (4.9) record the gap between the longest and shortest service times of the fleet according to the minimization objective. Finally, (4.10) are the variables definition constraints.

4.4 New reduced formulation

The formulation presented in Section 4.3 uses $\mathcal{O}(m \cdot n^2)$ binary variables and $\mathcal{O}(m \cdot n^2)$ constraints, where n is the number of customers and m the number of vehicles. In this section, we propose a new linear formulation for VRPTWSyn. It uses only $\mathcal{O}(n^2)$ binary variables and $\mathcal{O}(n^2)$ constraints. Let $z_{ij}, (i,j) \in E$ be the flow variable. It is equal to one if a vehicle travels along arc (i,j) and zero otherwise and let $y_{ik} (k \in K, i \in V)$ equal to one if costumer i is served by the vehicle k and zero otherwise. ξ_i records the start of the service at the visit i. We have the following mixed-integer linear programming model:

Objective function

$$\min \quad \omega_1 \sum_{(i,j)\in E} \delta_{ij} z_{ij} + \omega_2 \sum_{k\in K} \sum_{i\in V} Pref_{ik} y_{ik} + \omega_3 \mathcal{W} \tag{4.11}$$

Basic constraints

$$1 \leq \sum_{j \in V} z_{0j} \leq m \tag{4.12}$$

$$\sum_{j \in V^+ \setminus \{o\}} z_{ij} - \sum_{j \in V^+ \setminus \{d\}} z_{ji} = 0, \quad \forall i \in V \tag{4.13}$$

$$\sum_{j \in V} z_{0j} - \sum_{i \in V} z_{in+1} = 0 \tag{4.14}$$

$$\sum_{k \in K} y_{ik} = 1, \quad \forall i \in V \tag{4.15}$$

$$y_{ik} - y_{jk} \leq 1 - z_{ij}, \quad \forall (i,j) \in E, \quad \forall k \in K \tag{4.16}$$

$$y_{jk} - y_{ik} \leq 1 - z_{ij}, \quad \forall (i,j) \in E, \quad \forall k \in K \tag{4.17}$$

$$y_{jk} \leq 3 - (z_{0i} + z_{0j} + y_{ik}), \quad \forall i,j \in V, \quad \forall k \in K \tag{4.18}$$

Time constraints

$$e_i \leq \xi_i \leq l_i, \quad \forall i \in V^+ \tag{4.19}$$

$$\xi_i + (s_i + \delta_{ij}) z_{ij} \leq \xi_j + l_i (1 - z_{ij}), \quad \forall (i,j) \in E, \ \forall k \in K \tag{4.20}$$

$$\sum_{i \in V} s_i y_{ip} - \sum_{i \in V} s_i y_{iq} \leq \mathcal{W} \quad \forall p \in K, \forall q \in K \setminus \{p\} \tag{4.21}$$

$$\xi_i - \xi_j = 0 \quad \forall [i,j] \in P^{Sync} \tag{4.22}$$

$$z_{ij} \in \{0,1\} \forall (i,j) \in E \tag{4.23}$$

$$\xi_i \in \mathbb{R}_+, \ \forall i \in V^+$$

$$\mathcal{W} \in \mathbb{R}_+$$

The objective function (4.11) is to minimize the local travel time, the sum of negative preferences or the difference on workload. Constraints (4.12) ensure that there are at least one vehicle and at most m vehicles stating from the depot. Constraints (4.13) and (4.14) are the flow conservation equations enforcing route

continuity so that every vehicle starts from the departure and returns to the arrival. Constraints (4.15) ensure that each customer is served by exactly one vehicle. (4.16) and (4.17) ensure that if the arc (i, j) is used then the visits i and j are done by the same vehicle. Constraints (4.18) guarantee that if there are two clients reached directly from the depot then they must be served by two different vehicles. (4.19) and (4.20) are the time window constraints. They also eliminate the sub-tours. Constraints (4.21) calculate the maximum difference of the vehicle workloads. Constraints (4.22) ensure that synchronized visits start simultaneously. Finally, (4.4) are the variables definition constraints.

4.5 Constraint programming model

In this section, we present a constraint programing (CP) model for VRPTWSyn. Unfortunately, there is no standard language or presentation for this type of modeling which support the scheduling layer [70]. We use a syntax close to the one used for the implementation. Equivalent versions of the constraints exist in many other CP solvers. Our model uses the interval variables as decision variables [59]. Each variable has some proprieties, which include the time window $[a, b]$, the minimum length of the interval d and a boolean o used to indicate whether the interval must be obligatory or optional to be scheduled. We assign for each visit i a set of k optional interval variables Visit_{ik}. A visit is done by the vehicle k if the interval variable Visit^*_{ik} is present and scheduled.

For each vehicle k, the n interval variables Visit^*_{ik} $\forall i \in V$ are grouped by a sequence variable Route_k (Line 4). A NoOverlapSequence constraint is applied to the sequence to ensure that the intervals respect a time lag between them which corresponds to the travel time between each pair of visits δ_{ij} (Line 8). An alternative constraint is used to ensure that exactly one interval is executed for each visit which should be then represented by the variable Visit_i (Line 5). Finally

a start-at-start constraint is used to make sure that each pair of synchronized tasks start simultaneously (Lines 6 and 7).

We add k artificial but obligatory interval variables each represents the start of every route in order to take into consideration the arc outgoing from the depots (Line 9). The algorithm is configured to branch first on the sequence variables then on the intervals and to use the multi-point search strategy in order to diversify the solutions.

Algorithm 4.1: CP model for VRPTWSyn.

Variables:

1 $\text{Visit}^*_{ik} = \text{Interval}(e_i, l_i, s_i, \textit{optional}) \; \forall i \in V, k \in K$;
2 $\text{Departure}_k = \text{Interval}(0, 0, 0, \textit{obligatory}) \; \forall k \in K$;
3 $\text{Visit}_i = \text{Interval}(e_i, l_i, s_i, \textit{obligatory}) \; \forall i \in V$;
4 $\text{Route}_k = \text{Sequence}(\bigcup_{i \in V} \text{Visit}^*_{ik} \cup \text{Departure}_k) \; \forall k \in K$;

Constraints:

5 $\text{Alternative}(\text{Visit}_i, \bigcup_{k \in K} \text{Visit}^*_{ik})$;

6 $\text{StartAtStart}(\text{Visit}_i, \text{Visit}_j) \; \forall [i, j] \in P^{Sync}$;
7 $\text{PresenceOf}(\text{Visit}^*_{ik}) + \text{PresenceOf}(\text{Visit}^*_{jk}) \leq 1 \; \forall [i, j] \in P^{Sync} \quad \forall k \in K$;

8 $\text{NoOverlapSequence}(\text{Route}_k, \delta) \; \forall k \in K$;
9 $\text{First}(\text{Departure}_k, \text{Route}_k) \; \forall k \in K$;

The CP constraints used in the model are described as follows:

Alternative(i, J) models an exclusive alternative between the intervals in J. If interval i is present then exactly one interval in J is present and i starts and ends together with this chosen one.

StartAtStart(i, j) states that whenever both interval variables i and j are present, they should start at the same time.

PresenceOf(i) states whether the interval variable i is present or not.

NoOverlapSequence(s, M) states that all the present intervals in the sequence s are pairwise non-overlapping and a minimal distance $M_{i,j}$ is to be maintained between the end of i and the start of j.

First(i, s) states that whenever an interval variable i is present, it must be ordered first in the sequence variable s.

4.6 Preprocessing

Based on the characteristics of the time windows and the distances, we deduce a set of precedence relationships between the visits. For example, if two visits i and j have $l_i < e_j + s_j + \delta_{ji}$, then i has to precede j in any route. As a consequence, arc (j,i) is removed from E and its associated variables $x_{jik} \forall k \in K$ and z_{ji} are set to 0 while δ_{ji} is set to ∞.

Since the problem considers a limited number of vehicles, the time windows are adjusted using the energetic reasoning algorithm presented earlier in Section 2.5.

4.7 Additional cuts

In this section, we present some cuts and valid inequalities added to the three models in order to accelerate their solution.

4.7.1 Incompatibilities and clique cuts

By definition, two visits i and j are said to be incompatible if they cannot be done by the same vehicle. In other word, the incompatibility can be summarized in the following conditions where R represents a route and Cost(R) the travel cost of the

route R:

$$[i,j] \in P^{Sync} \tag{4.24}$$

$$(e_i + s_i + \delta_{i,j} > l_j) \wedge (e_j + s_j + \delta_{j,i} > l_i) \tag{4.25}$$

$$\begin{gathered}(\text{Cost}(R_1) > l_{n+1}) \wedge (\text{Cost}(R_2) > l_{n+1}) \\ \text{where } R_1 = (0,i,j,n+1) \wedge R_2 = (0,j,i,n+1)\end{gathered} \tag{4.26}$$

We denote by $i||j$ the incompatibility between the visits i and j. Using these conditions, we build the graph of incompatibilities between visits defined as: $G^V_{inc} = (V, E_V)$ where $E_V = \{(i,j) \in V \times V : i||j\}$. Based on this graph we extract all the maximal cliques and for each clique $\mathcal{C}$ we add the cut presented by Equation (4.27) for the classical model, Equation (4.28) for the reduced model and Equation (4.29) for the CP model.

$$\sum_{i \in \mathcal{C}} \left(\sum_{j:(i,j) \in E} x_{ijk} \right) \leq 1 \quad \forall k \in K \tag{4.27}$$

$$\sum_{i \in \mathcal{C}} y_{ik} \leq 1 \quad \forall k \in K \tag{4.28}$$

$$\sum_{i \in \mathcal{C}} \text{PresenceOf}(\text{Visit}_{ik}) \leq 1 \quad \forall k \in K \tag{4.29}$$

4.7.2 Subtour eliminations

The following cuts are applied only on the MIP models. We first relax the constraints (4.5) or (4.20) depending on the model. Then, we try to replace them with stronger ones known as the generalized subtour elimination constraints (GSECs) [35]. Since a feasible solution of VRPTWSyn should be a set of open directed paths, after relaxing the constraints (4.5) or (4.20), it may contain some cycles that should be eliminated. Hence, we use the inequality presented by (4.30) for the classical model and (4.31) for the reduced model where S represents a cycle.

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \quad \forall k \in K \tag{4.30}$$

$$\sum_{i,j \in S} z_{ij} \leq |S| - 1 \tag{4.31}$$

4.7.3 MIP overall algorithm

The overall algorithm is presented in Figure 4.2. The incomplete model (i.e. while relaxing the constraints (4.5) and (4.20)) is first solved to optimality and the solution is then checked for the existence of subtours. This can be done by determining all the strongly connected components in the resulting subgraph associated to each tour. We note that in a directed graph, a pair of vertices is said to be strongly connected to each other if there is a path in both directions that links them together. Consequently, a directed graph is called strongly connected if there is a path in each direction between each pair of vertices in the graph. Since in our case, the graph is directed and the depots are separated, thus the strongly connected components determined for a specific tour represent the vertices of the subtours. We note that it is possible to test the strong connectivity of a graph, or to find

its strongly connected components, in a polynomial time using Tarjan's algorithm [84]. This algorithm is applied on each tour of the obtained solution to determine the subtours. The corresponding subtour elimination constraints are then added to the model and the resolution is recalled. The process is iterated until no subtour is detected. In this case, we added all the missing constraints previously relaxed and we solve the new enforced model for the last time.

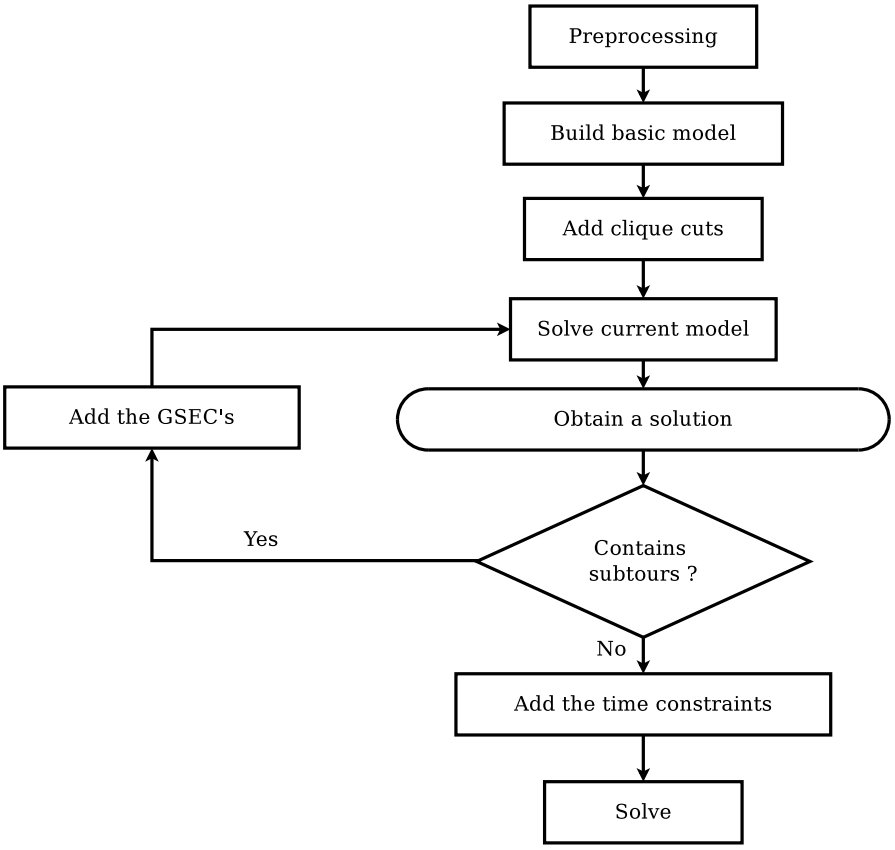


Figure 4.2 – MIP solver overall algorithm for VRPTWSyn.

4.8 Experimentation

We tested our algorithm on the standard instances of [17]. The benchmark, which was generated to simulate the scheduling problem in homecare services, comprises

Table 4.1 – Characteristics of the benchmark instances.

Instance	n	m	syn	$\sum s_i/m$ (h)	S(h)	M(h)	L(h)
1	20	4	2	4.9	1.5	2.1	2.9
2	20	4	2	4.2	1.7	2.2	3.0
3	20	4	2	5.3	1.5	2.4	3.0
4	20	4	2	5.9	1.8	2.9	3.9
5	20	4	2	5.0	1.3	2.1	3.2
6	50	10	5	4.7	1.4	2.3	3.1
7	50	10	5	5.0	1.6	2.5	3.4
8	50	10	5	6.2	1.5	2.4	3.2
9	80	16	8	6.1	1.5	2.3	2.9
10	80	16	8	5.1	1.6	2.6	3.6

10 data sets. These sets are grouped in 3 categories based on the number of clients. Each set has 5 varieties of instances that are named after the width of the time windows. In each instance, about 10% of the visits need to be pairewisly synchronized. An overview of the characteristics of the instances is found in Table 4.1. In this table, columns n, m and syn show respectively the number of visits, the number of vehicles and the number of synchronizations. The other column headers are $\sum s_i/m$ for the average service duration per vehicle and S, M, L for the average widths of the time windows associated with instances of the three varieties small, medium and large time windows respectively. Note that the two other varieties are instances with fixed appointments and the ones with no time windows at all. Thus those two are out of the scope of VRPTWSyn research [16]. The time unit of the table is in hours.

Our algorithm is coded in C++ using the Standard Template Library (STL) for data structures, IBM Ilog Cplex 12.6 for the linear programming and IBM Ilog CP optimizer 12.6 for the constraint programming. The program is compiled with GNU GCC in a Linux environment and all experiments were conducted on an Intel Xeon 2.67GHz. Our configuration is similar to the computational environment used by Bredström and Rönnqvist [16, 17]. According to the protocol proposed in [16], all the methods were tested with the three varieties S, M and L

as mentioned earlier. We consider the three objectives separately: minimizing the total travel time, minimizing the sum of negative preferences and minimizing the maximal difference in service times of the vehicles.

Tables 4.2 to 4.4 report our results and compare them with the existing methods in the literature. Column *Best* shows the best known solution collected from all methods (including ours) for each instance. A star mark (*) is used in *Best* to indicate that a solution has been found and proved to be optimal. The other column are: BP for the results of the branch-and-price algorithms presented in [16]; MIP for the results of the reduced linear model solver reported in Section 4.4; VMIP for the classical formulation and finally CP for the constraints programming based approach. Columns Sol and CPU report respectively the best solution found by each method and the associated computational time. LB is used with the MIP based methods to report the lower bound found. Bold numbers in Sol indicate that the solution quality reaches the best found. The time unit in those tables for the objective values like travel time or fairness is in hours, and for computational time is in seconds.

4.8.1 Travel time

We first present a comparison between the methods when considering the optimization of the travel time (see Table 4.2). Among 30 instances BP found 23 of the best solutions while MIP was able to find all the solutions (30). On the other hand the VMIP method which uses the classical formulation found difficulties to reach feasible solutions for the large instances. We also found that CP is not efficient on this objective compared to the other methods. In term of cpu time, MIP looks to be the fastest to find the best solutions. We also noticed that CP finds some difficulties to solve the instances of type S, which remains true for the later results.

4.8.2 Preferences

When considering the preferences (see Table 4.3), CP found all the best solutions of the 30 instances within a reasonable cpu time. MIP misses most of them even one from the small instances while VMIP acts better. From the literature BP method found 24 of the best solutions and missed the big instances with 80 visits.

4.8.3 Workload balance

Concerning the third objective, it is clear that the CP model is the most adapted for the fairness. It finds all the best solutions of the 30 instances. Using the reduced model, its relaxation seems to be weak compared to the one of the classical formulation. All the lower bounds are equal to 0 and it could find only 2 solutions from the best while MIP found 14 solutions.

Table 4.2 – Comparison of the solutions and computational times for the total travel time.

		BP1			BP2			VMIP			MIP			CP	
Instance	Best	Sol	LB	CPU	Sol	LB	CPU	Sol	LB	CPU	Sol	LB	CPU	Sol	CPU
1S	3.55*	3.55	3.55	1.96	3.55	3.55	1.12	3.55	3.55	2	3.55	3.55	0.02	3.55	2.29
1M	3.55*	3.55	3.55	221.93	3.55	3.55	3.69	3.55	3.55	11.29	3.55	3.55	0.06	3.55	2.9
1L	3.39*	3.39	3.39	107.41	3.39	3.39	11.91	3.39	3.39	82.24	3.39	3.39	0.09	3.39	2.12
2S	4.27*	4.27	4.27	3.28	4.27	4.27	0.56	4.27	4.27	0.11	4.27	4.27	0.02	4.27	1.37
2M	3.58*	3.58	3.58	8.12	3.58	3.58	3.2	3.58	3.58	1.96	3.58	3.58	0.09	3.58	6.36
2L	3.42*	3.42	3.42	2.72	3.42	3.42	7.41	3.42	3.42	368.25	3.42	3.42	0.16	3.42	13.63
3S	3.63*	3.63	3.63	14.17	3.63	3.63	3.84	3.63	3.63	2.79	3.63	3.63	0.03	3.63	4.67
3M	3.33*	3.33	3.33	17.57	3.33	3.33	4.31	3.33	3.33	5.95	3.33	3.33	0.04	3.33	1.64
3L	3.29*	3.29	3.29	42.78	3.29	3.29	1.44	3.29	3.29	141.24	3.29	3.29	0.14	3.29	2.19
4S	6.14*	6.14	6.14	14.02	6.14	6.14	1.54	6.14	6.14	28.52	6.14	6.14	0.06	6.14	3.02
4M	5.67*	5.67	5.67	27.53	5.67	5.67	2.55	5.67	5.67	1006.2	5.67	5.67	0.28	5.67	3.22
4L	5.13*	5.13	5.13	9.74	5.13	5.13	7.69	5.23	4.17	3600	5.13	5.13	5.51	5.13	1.25
5S	3.93*	3.93	3.93	2.84	3.93	3.93	2.9	3.93	3.93	18.39	3.93	3.93	0.04	3.93	0.94
5M	3.53*	3.53	3.53	57.04	3.53	3.53	9.1	3.53	3.53	4.6	3.53	3.53	0.03	3.53	139.15
5L	3.34*	3.34	3.34	9.11	3.34	3.34	5.15	3.34	3.34	533.43	3.34	3.34	0.16	3.34	27.15
6S	8.14*	8.14	8.13	3600	8.14	8.14	197	8.75	7.75	3600	8.14	8.14	0.43	8.2	618.25
6M	7.7*	7.71	7.67	3600	7.7	7.7	3600	-	6.37	3600	7.7	7.7	32.14	7.85	156.89
6L	7.14*	7.14	7.14	3279	7.14	7.13	3600	-	5.23	3600	7.14	6.86	3600	7.54	161.15
7S	8.39*	8.39	8.39	14.72	8.39	8.39	169	12.9	7.78	3600	8.39	8.39	6.91	8.56	2151.73
7M	7.48*	7.67	7.36	3600	7.56	7.41	3600	-	6.31	3600	7.48	7.48	896.02	7.63	1343.49
7L	6.88	6.88	6.87	3600	6.88	6.86	3600	-	5.05	3600	6.88	6.36	3600	6.95	18.21
8S	9.54*	9.54	9.54	931	9.54	9.54	850	-	8.78	3600	9.54	9.54	24.43	10.1	256.53
8M	8.54*	8.54	8.53	3600	8.54	8.54	3490	-	7.09	3600	8.54	8.54	756.77	8.94	230.02
8L	8.11	8.62	7.91	3600	8.11	7.94	3600	-	6.07	3600	8.11	7.39	3600	8.23	270.13
9S	11.95	-	11.7	3600	12.21	11.68	3600	-	10.05	3600	11.95	11.73	3600	12.53	1120.43
9M	10.93	11.74	10.79	3600	11.04	10.79	3600	-	7.88	3600	10.93	9.7	3600	11.68	2678.73
9L	10.64	11.11	10.37	3600	10.89	10.37	3600	-	6.61	3600	10.64	9.24	3600	11.22	2776.35
10S	8.54*	-	8.4	3600	9.13	8.43	3600	-	7.41	3600	8.54	8.54	1789.57	9.62	589.86
10M	7.67	8.54	7.5	3600	8.1	7.52	3600	-	5.34	3600	7.67	6.98	3600	8.15	1223.57
10L	7.84	-	7.05	3600	-	7.05	3600	-	4.16	3600	7.84	5.76	3600	8.4	1793.67

Table 4.3 – Comparison of the solutions and computational times for the sum of negative preferences.

		BP|			VMIP			MIP			CP	
Instance	Best	Sol	LB	CPU	Sol	LB	CPU	Sol	LB	CPU	Sol	CPU
1S	-114.03*	-114.03	-114.03	1.27	-114.03	-114.03	1.31	-114.03	-114.03	10.74	-114.03	0.64
1M	-117.8*	-117.8	-117.8	1.68	-117.8	-117.8	1.54	-117.8	-117.8	23.85	-117.8	0.33
1L	-118.51*	-118.51	-118.51	2.55	-118.51	-118.51	24.43	-118.51	-118.51	44.87	-118.51	0.42
2S	-92.09*	-92.09	-92.09	0.6	-92.09	-92.09	0.43	-92.09	-92.09	19.48	-92.09	0.23
2M	-104.81*	-104.81	-104.81	2.3	-104.81	-104.81	38.53	-104.81	-104.81	98.66	-104.81	0.32
2L	-107.64*	-107.64	-107.64	6.44	-107.64	-107.64	795.11	-107.64	-107.64	1624.45	-107.64	1.14
3S	-99.49*	-99.49	-99.49	1.66	-99.49	-99.49	0.69	-99.49	-99.49	4.41	-99.49	0.25
3M	-106.59*	-106.59	-106.59	2.01	-106.59	-106.59	2.31	-106.59	-106.59	66.52	-106.59	0.8
3L	-107.87*	-107.87	-107.87	2.63	-107.87	-107.87	7.13	-107.87	-107.87	267.63	-107.87	1.01
4S	-100*	-100	-100	1.72	-100	-100	1.12	-100	-100	1.87	-100	0.82
4M	-106.72*	-106.72	-106.72	2.36	-106.72	-106.72	56.04	-106.72	-106.72	136.55	-106.72	0.33
4L	-109.27*	-109.27	-109.27	5.04	-109.27	-109.27	864.37	-107.1	-114.25	3600	-109.27	1.1
5S	-76.29*	-76.29	-76.29	0.64	-76.29	-76.29	0.13	-76.29	-76.29	2.1	-76.29	0.63
5M	-76.29*	-76.29	-76.29	1.28	-76.29	-76.29	2.42	-76.29	-76.29	16.87	-76.29	0.45
5L	-84.21*	-84.21	-84.21	2.21	-84.21	-84.21	43.96	-84.21	-84.21	67.25	-84.21	0.8
6S	-370.06*	-370.06	-370.06	150.63	-370.06	-370.1	1058.22	-288.08	-394.85	3600	-370.06	28.9
6M	-379.88*	-379.88	-379.88	247.88	-	-394.62	3600	-352.2	-402.05	3600	-379.88	47.96
6L	-387.2*	-387.2	-387.2	474.15	-	-403.82	3600	-313.63	-404.94	3600	-387.2	8.38
7S	-401.11*	-401.11	-401.11	291.29	-401.11	-407.43	3600	-308.75	-413.83	3600	-401.11	86.89
7M	-406.17*	-406.17	-406.17	86.7	-	-413.22	3600	-245.58	-416.16	3600	-406.17	7.68
7L	-407.48*	-407.48	-407.48	710.62	-	-415.97	3600	-237.99	-416.75	3600	-407.48	10.3
8S	-380.76*	-380.76	-380.76	135.39	-288.63	-393.91	3600	-178.57	-411.86	3600	-376.42	131.18
8M	-403.57*	-403.57	-403.57	290.77	-	-414.41	3600	-235.6	-422.81	3600	-403.57	248
8L	-407.48*	-407.48	-407.48	362.18	-	-423.01	3600	-208.98	-426.03	3600	-407.48	163.11
9S	-583.03	-552.65	-633.91	3600	-	-661.55	3600	-335.57	-695.66	3600	-583.03	1458.29
9M	-654.43	-463.82	-661.77	3600	-	-684.05	3600	-320.42	-701.67	3600	-654.43	1563
9L	-669.97	-663.47	-637.09	3600	-	-693.48	3600	-306.29	-703.75	3600	-669.97	1344.73
10S	-675.90	-675.81	-676.14	3600	-	-688.03	3600	-393.19	-708.94	3600	-675.90	2403.72
10M	-686.77	-685.31	-687.9	3600	-	-705.28	3600	-343.19	-710.08	3600	-686.77	630.79
10L	-691.86	-691.34	-693.98	3600	-	-709.75	3600	-367.37	-710.49	3600	-691.86	346.85

Table 4.4 – Comparison of the solutions and computational times for the workload balance.

		VMIP			MIP			CP	
Instance	Best	Sol	LB	CPU	Sol	LB	CPU	UB	CPU
1S	0*	0	0	51.01	0.04	0	3600	0	13.95
1M	0*	0	0	435.57	0.04	0	3600	0	11.36
1L	0*	0	0	3025.43	0.06	0	3600	0	4.06
2S	0.01*	0.01	0.01	132.06	0.05	0	3600	0.01	15.84
2M	0.01	0.01	0	3600	0.02	0	3600	0.01	2.54
2L	0.01	0.01	0	3600	0.05	0	3600	0.01	8.87
3S	0.01*	0.01	0.01	1868.16	0.04	0	3600	0.01	3.13
3M	0.01*	0.01	0.01	2690.94	0.01	0	3600	0.01	1.46
3L	0.01	0.01	0	3600	0.06	0	3600	0.01	1.19
4S	0.06*	0.06	0.06	1399.77	0.06	0	3600	0.06	2.77
4M	0.02	0.02	0	3600	0.03	0	3600	0.02	14.17
4L	0.02	0.03	0	3600	0.02	0	3600	0.02	4.52
5S	0.01	0.01	0.01	1168	0.08	0	3600	0.01	0.67
5M	0.01	0.01	0	3600	0.09	0	3600	0.01	4.5
5L	0.01	0.03	0	3600	0.04	0	3600	0.01	2.31
6S	0.01	0.55	0	3600	2.01	0	3600	0.01	1595.76
6M	0.01	0.55	0	3600	2	0	3600	0.01	575.75
6L	0.01	-	0	3600	2.02	0	3600	0.01	819.01
7S	0.03	0.3	0	3600	2.39	0	3600	0.03	596.91
7M	0.01	1.27	0	3600	3.3	0	3600	0.01	245.18
7L	0.01	-	0	3600	-	-	3600	0.01	519.16
8S	0.05	-	0	3600	2.03	0	3600	0.05	436.32
8M	0.04	-	0	3600	-	-	3600	0.04	216.83
8L	0.03	-	0	3600	-	-	3600	0.03	697.88
9S	0.08	-	0	3600	-	-	3600	0.08	1050.03
9M	0.06	-	0	3600	-	-	3600	0.06	1118.03
9L	0.06	-	0	3600	-	-	3600	0.06	915.6
10S	0.03	-	0	3600	-	-	3600	0.03	1164.32
10M	0.03	-	0	3600	-	-	3600	0.03	1075.03
10L	0.01	-	0	3600	-	-	3600	0.01	993.47

4.9 The efficiency of the cuts

In this section, we present and discuss the efficiency of the used cuts. The model solvers were first run without any cuts, then compared with all the cuts activated. As an example, Table 4.5 presents a comparison of these two runs when dealing with the preference objective using the reduced model.

Column GAP shows the gap corresponding to the results of the version with cuts compared to the basic one. It is computed as $100 \times (WithCuts - Withoutcuts)/|WithCuts|)$. Using the cuts in this case improves both the lower bounds and the upper bounds by 1.89% and 7.56% in average respectively. The improvements are particularly

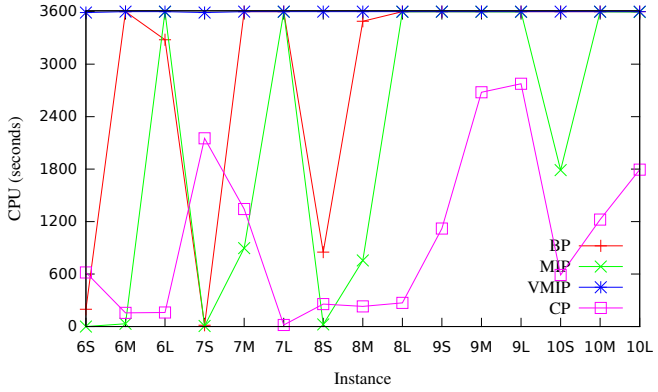


Figure 4.3 – Comparison of the Computational times for the travel time.

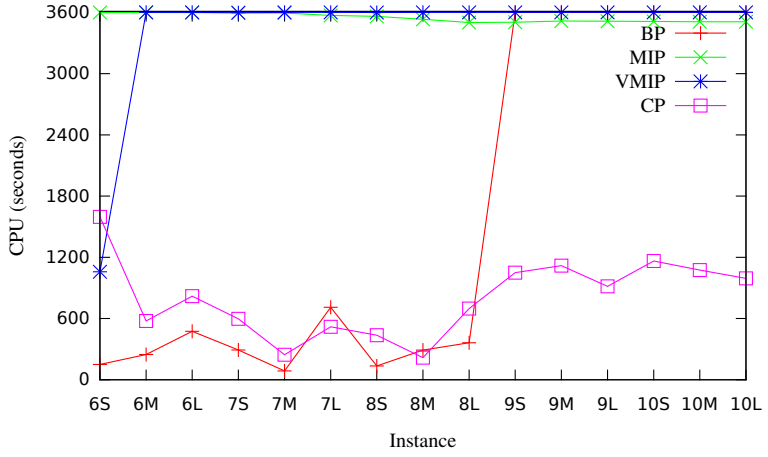


Figure 4.4 – Comparison of the Computational times for the sum of negative preferences.

high for the instances with small time windows. This is due to the high density of the incompatibility graph which yields to more clique constraints. Similar results were found for the remaining cases.

4.10 Conclusion

In this chapter, we explored the idea of using efficiently some exact methods on the vehicle routing problem with time windows and synchronized visits (VRPTWSyn). A new linear model has been proposed which uses fewer variables and constraints.

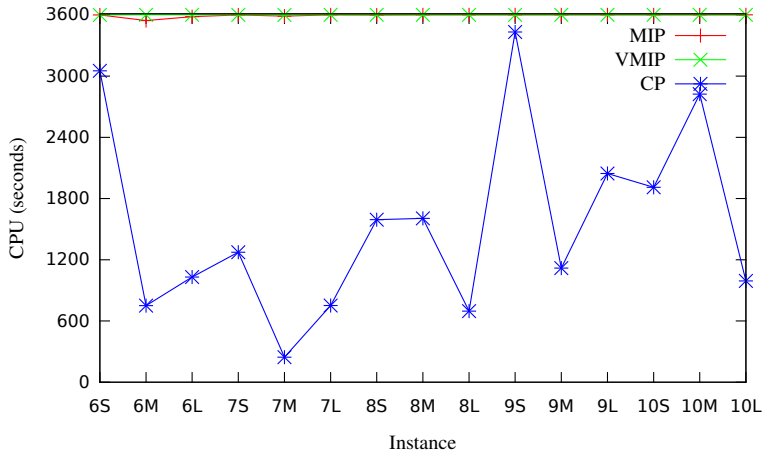


Figure 4.5 – Comparison of the Computational times for the workload balance.

	Without Cuts		With Cuts		GAP %	
	UB	LB	UB	LB	UB	LB
S	-160.03	-267.52	-177.32	-262.74	-9.75	1.82
M	-167.32	-273.41	-178.19	-269.00	-6.10	1.64
L	-141.51	-277.67	-151.92	-271.63	-6.86	2.22
All	-157.02	-272.86	-169.87	-267.80	-7.56	1.89

Table 4.5 – Comparative of the efficiency of the cuts using the reduced model on the preference objective.

The new model has been tested and compared to the classical one on the three objectives which minimize the local travel time, the sum of negative preferences or the difference on workloads. The results confirm the benefits of using fewer variables and constraints especially when considering the total travel time. Unlike the classical formulation, the model is capable of solving large instances. Some additional cuts and preprocessing have been also applied to both models.

A new approach has been also tested on VRPTWSyn based on constraints programming. The problem has been modeled as a scheduling one and enforced using some preprocessing and incompatibility constraints. This new approach produced a significant improvement especially when dealing with the second and the third objectives. Further research will focus on developing a hybrid method

using the main keys found on both studies and improve the performance of the MIP algorithms by using more sophisticated cuts. A study of a multi-objective approach using the CP solver is also planned as well as the support of visitors with different qualifications. This can be provisionally managed by assigning very small preference values to the unqualified visitors.

5 Heuristic Solutions for VRPTWSyn

Originality is nothing but judicious imitation.

Voltaire

5.1 Simulated annealing based iterative local search algorithm . . . 73
5.1.1 Constructive heuristic . . . 74
5.1.2 Diversification process . . . 78
5.1.3 Local search procedure . . . 79
5.1.3.a 2-opt* . . . 79
5.1.3.b Or-opt . . . 80
5.1.3.c Replacement . . . 80
5.1.3.d Single-move . . . 80
5.2 Experimentation . . . 81
5.2.1 Parameter settings . . . 82
5.2.2 Efficiency of the neighborhood structure . . . 84
5.2.3 Comparative results . . . 85
5.3 Conclusion . . . 90

In this chapter, we present a simulated annealing based iterative local search algorithm (SA-ILS) for the Vehicle Routing Problem with Time Windows and Synchronized Visits (VRPTWSyn). This problem described in Chapter 4, is a variant of the vehicle routing problem (VRP), in which a time window is associated with each client service and some services require simultaneous visits from different vehicles to be accomplished. The algorithm features a set of local improvement methods to deal with various objectives of the problem. Experiments conducted on the benchmark instances from the literature clearly show that our method is fast and outperforms the existing approaches especially when considering the total travel time or the sum of preferences as objective. It produces all known optimal solutions of the benchmark in very short computational times, and improves the best results on some remaining instances of the benchmark.

Motivated by the potential applications and by the challenge of computational time, we propose, in this work, a simulated annealing based iterative local search algorithm (SA-ILS) for solving VRPTWSyn. Our SA-ILS incorporates several local search methods dedicated to the problem. It produces high quality solutions in a very short computational time compared to the other methods of the literature. New best solutions are discovered. A statistical report on the performance of each local search operator on each objective is also given to provide the insights. The remainder of the chapter is organized as follows. In Section 5.1, the detailed description of the proposed SA-ILS algorithm is given. The results of the experimental studies are reported in Section 5.2. Finally, some concluding remarks are drawn in the Section 5.3.

Algorithm 5.1: SA-ILS algorithm for VRPTWSyn.

Output: X_{best}, the best solution found so far by the algorithm;

```
1  X ← BestInsertion(∅);
2  X ← LocalSearch(X);
3  X_best ← X;
4  reheat ← 0;
5  repeat
6      T ← T_0;
7      iter ← 0;
8      X_lbest ← X;
9      repeat
10         X' ← Diversification(X, 1, d);
11         X' ← LocalSearch(X');
12         Δ ← Fitness(X') − Fitness(X);
13         iter ← iter + 1;
14         T ← α × T;
15         r ∼ Unif(0,1);
16         if (r < e^(−Δ/T)) then
17             X ← X';
18             if (Fitness(X) < Fitness(X_lbest)) then
19                 iter ← 0;
20                 X_lbest ← X;
21                 if (Fitness(X) < Fitness(X_best)) then
22                     X_best ← X;
23                     reheat ← 0;
24     until (iter = itermax);
25     X ← Diversification(X, n/2, n);
26     reheat ← reheat + 1;
27 until (reheat = rhmax);
```

5.1 Simulated annealing based iterative local search algorithm

Our motivation in this work is to propose a fast dedicated heuristic solution for VRPTWSyn. The global scheme of our approach is a Simulated Annealing algorithm (SA) [54]. SA is a stochastic local search which is often used to address

discrete optimization problems. The main idea of a Simulated Annealing algorithm is to occasionally accept degraded solutions in the hope of escaping the current local optimum. The probability of accepting a newly created solution is computed as $e^{-\frac{\Delta}{T}}$, where Δ is the difference of fitness between the new solution and the current one and T is a parameter called the current temperature. This parameter is evolved during the search by imitating the cooling process in metallurgy. Successful applications of SA in VRP and its variants can be found in [9, 19, 22, 92].

Our SA-ILS is summarized in Algorithm 5.1. The algorithm is implemented with a reheating mechanism, due to Lines 4, 26 and 27. The simulated annealing routine is from Line 9 to Line 24. In the algorithm, we use $Fitness()$ to denote the process of computing the objective value according to Equation (4.1). The other functions: $BestInsertion(X)$, $Diversification(X, d_{min}, d_{max})$ and $LocalSearch(X')$ are described as follows.

5.1.1 Constructive heuristic

The procedure $BestInsertion(X)$ described in Algorithm 5.2 is a constructive heuristic to build a solution from scratch ($X = \emptyset$) or from a partial solution. A solution is called partial if some visits are not routed. In each iteration of $BestInsertion(X)$, a visit is heuristically selected to be inserted in a route so that the increasing cost is minimized. The algorithm is stopped when no more insertion are possible. The obtained solution can be either complete, i.e. a feasible solution with all the visits being routed, or still partial, i.e. an infeasible solution. This can happen for the instances of VRPTWSyn [17], particularly for the ones with small or strict time windows. In that case, unrouted visits are put in a pool for later attempts and a penalty cost proportional to the number of visits in the pool is added to the objective value.

In order to evaluate each insertion cost in constant time, a calculation of possible positions to insert visits is first performed. Then, information for each visit is archived and updated during the process as follows. For each visit i, we use Wait_i to memorize the waiting time in case the arrival takes place before the beginning of the time window, MaxShift_i to compute the maximal delay of the visit and LMaxShift_i for the maximal delay of the visit considering only the route where it belongs and ignoring the synchronization constraints. Supposing that Arrival_i and Start_i are the arrival time and the starting time of the service respectively, it holds that

$$\text{Wait}_i = \text{Start}_i - \text{Arrival}_i \tag{5.1}$$

Because of the synchronization constraints, Start_i for some visits may be delayed so that the client is served simultaneously by the assigned vehicles. For a given route r, we also use function $r(p)$ to denote the visit at position p in the route. We now notice that $\text{LMaxShift}_{r(p)}$ is equal to the sum of the $\text{Wait}_{r(p+1)}$ and $\text{MaxShift}_{r(p+1)}$, unless there is a time window bound.

$$\text{LMaxShift}_{r(p)} \leftarrow \mathbf{min}(b_{r(p)} - \text{Start}_{r(p)}, \text{Wait}_{r(p+1)} + \text{MaxShift}_{r(p+1)}) \tag{5.2}$$

If two visits need to be synchronized, the minimal value of LMaxShift is taken for both of them.

$$\text{if } [i,j] \in P^{Sync} \text{MaxShift}_i \leftarrow \mathbf{min}(\text{LMaxShift}_i, \text{LMaxShift}_j) \tag{5.3}$$

Therefore, an insertion of a visit k in a route r between p and $p+1$ will be considered to be valid if the generated shift: $\text{Shift}_k^{r,p}$ is smaller than the sum of

$\text{Wait}_{r(p+1)} + \text{MaxShift}_{r(p+1)}$.

$$\text{Shift}_k^{r,p} \leftarrow \delta_{r(p)k} + \text{Wait}_k + s_k + \delta_{kr(p+1)} - \delta_{r(p)r(p+1)} \tag{5.4}$$

As mentioned earlier, the insertion with the least cost is applied in each iteration. The insertion cost is considered to be $\delta_{r(p)k} + \delta_{kr(p+1)} - \delta_{r(p)r(p+1)}$ for the case of minimizing the travel cost, $Pref_{kr}$ when minimizing the preference, and the new $\mathcal{W}$ (denoted $\mathcal{W}_k^{r,p}$) if optimizing the workload balance. For general objective, the insertion cost of a visit k at a position p in the route r, denoted by $\text{Cost}_k^{r,p}$, is calculated as follows.

$$\text{Cost}_k^{r,p} \leftarrow \omega_1(\delta_{r(p)k} + \delta_{kr(p+1)} - \delta_{r(p)r(p+1)}) + \omega_2 Pref_{kr} + \omega_3 \mathcal{W}_k^{r,p} \tag{5.5}$$

When an insertion is applied, the update is propagated through different routes because of the synchronization constraints. The propagation may loop infinitely if the cross synchronizations are not prohibited, e.g. visiting u then v by the first vehicle, visiting i then j by the second vehicle, and finally realizing that u and j are the same client as well as v and i, e.g. $[r(u), r(j)]$, $[r(v), r(i)] \in P^{Sync}$ (see Figure 5.1). To avoid such issues, transitive closures [5] are computed to filter out cross synchronizations from the set of possible positions for insertion (see Line 6 in the Algorithm 5.2). The reduced solution $\mathcal{R}_{Sync}(X)$ refers to a structure equivalent to the solution X with only the synchronization visits. This computation takes $O(s^3)$ where s is the number of synchronizations. Therefore, the complexity of constructing a solution completely from scratch is $O(n \cdot \max\{s^3, n^2\})$.

In addition, the computation of possible positions for insertions and the evaluation of insertion costs is accelerated using the preprocessed data of the input instance. Based on characteristics of the time windows, we deduce a set of precedence relationships between the visits. For example, if two visits i and j have

Algorithm 5.2: BestInsertion algorithm for VRPTWSyn.

```
1  Procedure BestInsertion(X: a solution)
2      Ω ← the set of unrouted visits ;
3      while Ω ≠ ∅ do
4          foreach k ∈ Ω do
5              if ∃j ∈ V/[k, j] ∈ P^Sync then
6                  calculate positions from the reduced solution R_Sync(X) ;
7              else
8                  consider all the positions in X ;
9              foreach (r, p) ∈ positions do
                   // r is the route and p the position within this route
10                 if Insertion_k^{r,p} is feasible then
11                     Insertions ← Insertions ∪ (k, (r, p), Cost_k^{r,p}) ;
12             if Insertions ≠ ∅ then
13                 Best ← best(Insertions);
14                 Insert Best.k in the position Best.(r, p) ;
15                 Propagate the updates;
16                 Ω ← Ω \ Best.k ;
17             else
18                 Update the penalties ;
19                 break ;
```

$b_i < a_j + s_j + \delta_{ji}$, then i has to precede j in any route. As a consequence, arc (j, i) is removed from A, or δ_{ji} is set to ∞.

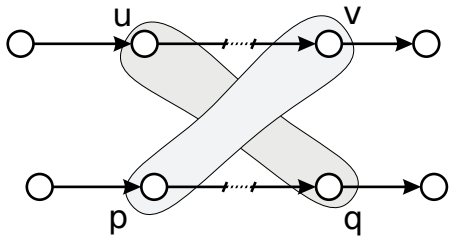


Figure 5.1 – Cross synchronization.

5.1.2 Diversification process

The function $Diversification(X, d_{min}, d_{max})$ (see Algorithm 5.3) first removes a number (randomly generated between d_{min} and d_{max}) of visits from the current solution (between 0 and d in our case) and runs a local search procedure (described in Section 5.1.3) to optimize the partial solution. A reconstruction phase is then processed using the above constructive heuristic. This iterative approach is similar to the destruction/repair operator used in [13]. The aim is to obtain a new solution from the current one without losing much of the quality, thanks to the constructive heuristic.

Algorithm 5.3: Algorithm to diversify a solution.

Input:
X, a solution;
d_{min}, d_{max}, parameters of the diversification;
Output: X', a new solution derived from X;

1 $X' \leftarrow X$;
2 $d \sim \mathcal{U}(d_{min}, d_{max})$;
3 remove randomly d clients from routes of X';
4 $X' \leftarrow$ `LocalSearch(X');`
5 $X' \leftarrow$ `BestInsertion(X');`

In addition, a dynamic priority management is also administered to identify critical visits. Each visit is associated with a priority number initialized to 0. This number is increased by 1 unit whenever the insertion of the visit cannot be done. Visits having the highest priority, i.e. frequently caused infeasible solutions, are in fact critical. Therefore, they need to be inserted during the early stages of the constructive heuristic. With this dynamic management, the search is guided back to the feasible space whenever it hits the infeasible one. In general, we remarked that the portion of explored infeasible solutions over feasible ones varies from one instance to another. This solely depends on the size of the time windows, e.g. the algorithm hits infeasible solutions more frequently with instances having small time windows.

5.1.3 Local search procedure

The following neighborhoods were adapted to the synchronization constraints and used in our local search procedure:

5.1.3.a 2-opt* (Exchanges the tails of two routes [74])

In a 2-opt operator, the possibilities of exchanging two links with two others in the same route are explored to find a local improvement. For the case of multiple vehicles, we use 2-opt* to denote the same principle of exchange but related to two distinct routes. This operator consequently implies the exchanges of paths between two routes. It is particularly suitable for our case since it is hardly possible for the classical 2-opt to find an improvement due to the preserved order of visits from the time windows. Our 2-opt* is implemented as follows.

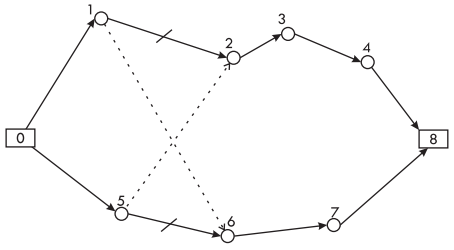


Figure 5.2 – 2-opt*: exchange of links (1, 2), (5, 6) for links (1, 6), (5, 2)

A subset of d visits is randomly selected and for each couple of visits $\{r(i), r'(j)\}$, we consider the arcs $(r(i), r(i+1))$ and $(r'(j), r'(j+1))$. If the exchange of these two arcs for $(r(i), r'(j+1))$ and $(r'(j), r(i+1))$ ensures the feasibility then the associated cost is recorded. The feasibility check is handled by the same process as the one used in the constructive heuristic to avoid cross synchronizations. Therefore, the exchange cost is evaluated in a constant time for each couple $\{r(i), r'(j)\}$. After testing all the possible couples, the best one is then memorized and the improving exchange is applied.

5.1.3.b Or-opt (Relocation of visits in the same route [82])

In this operator, we look for the possibilities of relocating a sequence of (1, 2 or 3) visits from its original place to another one in the same route. The implementation of this operator is similar to 2-opt* operator: a random selection at the beginning with a feasibility check. The best move is applied. Although this operator does not directly improve the objective when minimizing the sum of preferences, it compacts the routes and makes room for further insertions.

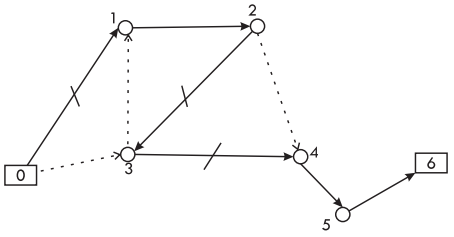


Figure 5.3 – Or-opt: moving visit 3 between the depot and visit 1

5.1.3.c Replacement (Exchanges between the routed and unrouted visits)

In this operator, we try to insert unrouted visits by mean of exchanging them with routed visits. The operator is implemented with a full enumeration. That is to say for each routed visit we try to exchange its position with all the unrouted visits. Among the feasible exchanges that improve the objective, the best one is applied.

5.1.3.d Single-move (Change the position of the routed visits)

This operator tries to move every routed visit from its current position to another position so that the objective value is improved. Similar to replacement, a full enumeration is considered and the best improving move is applied. Unlike Or-opt, the operator looks for potential positions in all routes and only one routed visit is considered at a time.

Our *LocalSearch*(X) function is described in Algorithm 5.4. At each iteration, a random neighborhood w is chosen from the set W of unexplored neighborhoods, initialized to {2-opt*, Or-opt, Replacement, Single-move} denoted W_0. Neighborhood w is then removed from W and applied on the current solution. If at least one improvement is detected by the current neighborhood w, the set of unexplored neighborhoods will be set back to W_0. The procedure is terminated when W is empty.

Algorithm 5.4: Local search for VRPTWSyn.

```
Input: X : the solution to improve
Output: X the new solution if improved
1 W_0 ← {2-opt*, Or-opt, Replacement, Single − move};
2 W ← W_0;
3 repeat
4 |   Select a random neighborhood w from W;
5 |   if w(X) = true then                    /* perform the local search */
6 |   |   W ← W_0;
7 |   else
8 |   |   W ← W\{w};
9 until W = ∅;
```

5.2 Experimentation

We tested our algorithm on the same instances introduced by [17] and presented in Section 5.2. The benchmark comprises 10 sets grouped in 3 categories based on the number of customers. Each set has 3 varieties of instances, those are named after the width of the time windows: S (small), M (medium) and L (large) time windows. Our algorithm is coded in C++ and all experiments were conducted on an Intel Xeon 2.67GHz, the same configuration used in Chapter 4 and the work of Bredström and Rönnqvist [17].

5.2.1 Parameter settings

Our algorithm has five following parameters:

- T_0, the initial temperature of the cooling schedule
- α, a parameter that controls the speed of the cooling schedule
- d, an integer that influences the degree of the diversification process
- $itermax$, the number of iterations without improvements to initiate a reheating phase
- $rhmax$, the maximal number of reheating phases

We first identify and fix the parameters that do not much influence the outcome of our algorithm and the runtime. After some small experiments, those parameters are chosen as follows. By recording the maximum number of iterations and phases needed to achieve the best solutions, values of $itermax$ and $rhmax$ are fixed to $8 \times n$ and 3 respectively. Parameter d is fixed to n/m as the average number of visits per vehicle, this is the common setting mentioned and used in [13].

The two remaining parameters required to be tuned are: the initial temperature T_0 and the control parameter α of the cooling schedule. Note that selecting the rightful set of those parameters is a common issue for simulated annealing. The manual exploration of all the combinations of the settings is tedious and generally hard to interpret. For this reason, we limited our exploration to subset of possible settings and on a subset of training instances as follows.

The initial temperature T_0 were tested with values 0.1, 0.5, 1, 10, 100. For parameter α, we used values 0.9, 0.95, 0.99, 0.995, 0.999. This results 25 different combinations of the pair $\{T_0, \alpha\}$ for the test. The training set is picked from the instances of the benchmark with more than 50 clients. For each combination of

the settings, the algorithm was executed 10 times per instance. Two following quantitative measures are used to compare the combinations: the relative gap to the best solutions found, denoted by rpe and the average computation time, denoted by cpu.

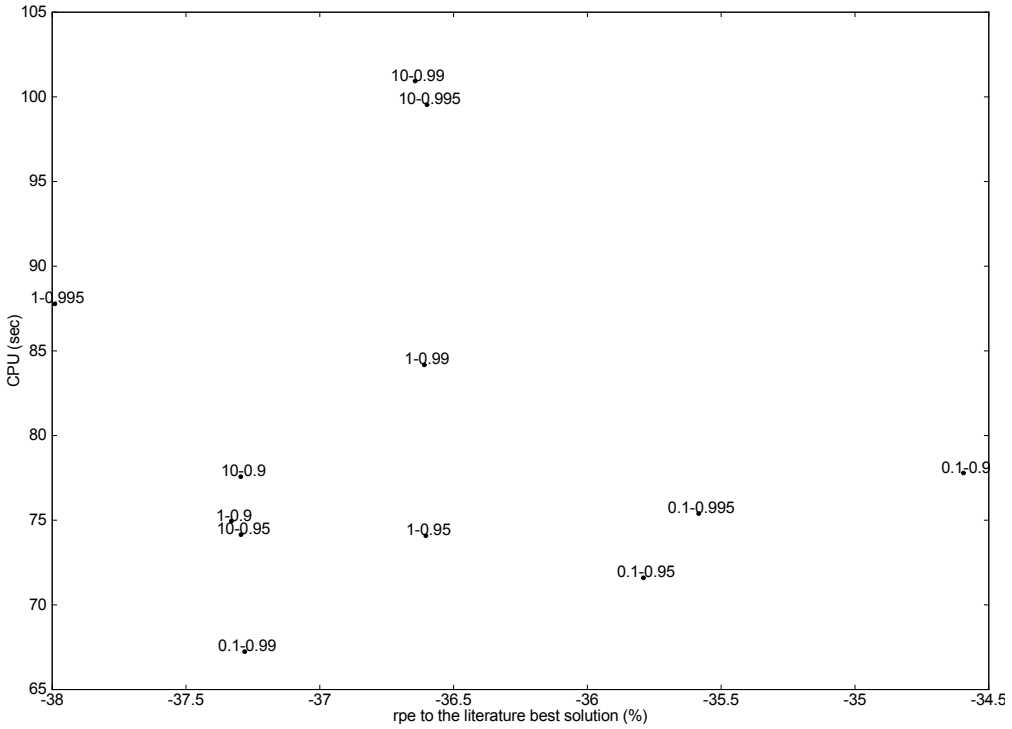


Figure 5.4 – Tradeoff between performance and computational time for different parameter settings when minimizing the total travel time.

Examples of the outcome results for the case of minimizing the total travel time and the negative preferences are illustrated in Figures 5.4 and 5.5. In order to find the best configuration, we first calculate the ideal point which give both the best values for cpu and rpe. Then the best combinations is selected among the configuration points so that the euclidean distance to the ideal point is minimized. Note that this step requires rpe and cpu to be normalized into the $[0,1]$ interval. Using this technique, we adopt the following parameter settings: $\{T_0 = 0.1, \alpha = 0.99\}$ when minimizing the travel cost, $\{T_0 = 1, \alpha = 0.95\}$ when working the preference and $\{T_0 = 1, \alpha = 0.99\}$ if optimizing the workload balance. The

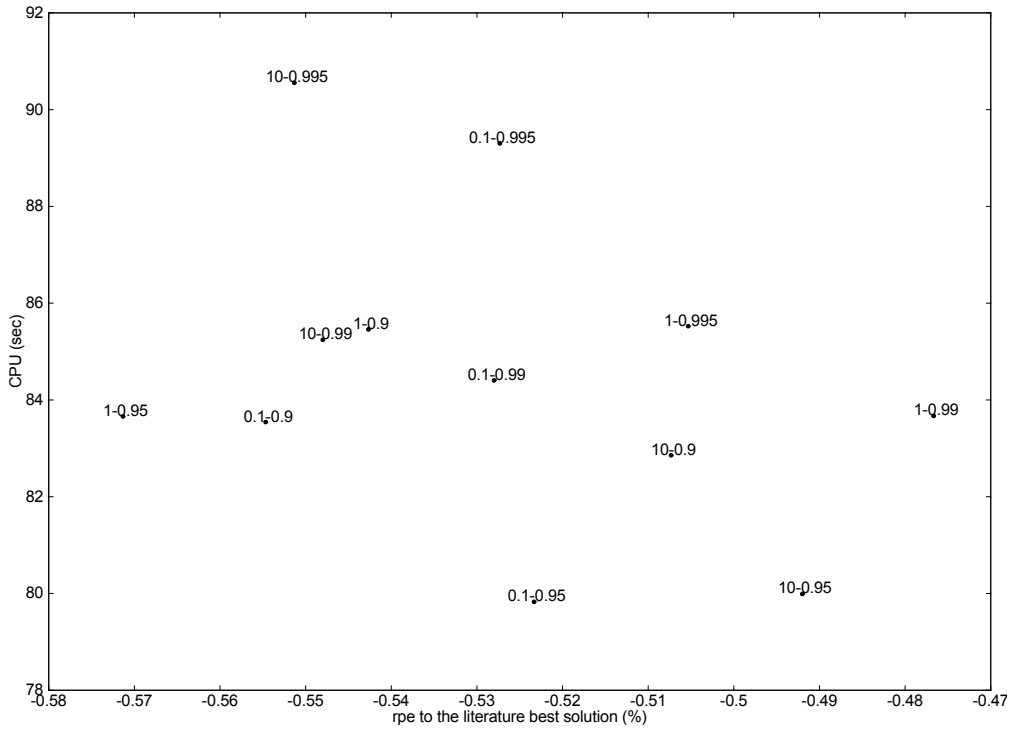


Figure 5.5 – Tradeoff between performance and computational time for different parameter settings when minimizing the sum of negative preferences.

same approach can be used to find the appropriate parameters for the general (aggregated) objective function.

5.2.2 Efficiency of the neighborhood structure

Local search is the essential ingredient in modern design of metaheuristics. In our SA-ILS, it is also the most expensive component. After our observation, more than 90% of the runtime is spent in the local search. Therefore, it is important to understand the contribution of each neighborhood to the success of the search. For this purpose, we record for each neighborhood the success rate, i.e. the number of attempts with improved outcome over the total number of attempts. Figures 5.6 and 5.7 present the average success rate (in percent) for each neighborhood and the overall local search, denoted by LS, on each category of instances related to

the time windows while optimizing the two objectives, the travel time and the preference.

We noticed that some neighbors had difficulties with specific instances. For example, the success rate of Or-opt is often below 2.5% for instances with small time windows. For this reason, we adapt the following strategy to identify and permanently drop useless neighborhood during the execution of the algorithm: after 100 attempts, if a neighborhood has a poor success rate, i.e. below a threshold, it is permanently removed, i.e. it is no longer considered initializing W in Algorithm 5.4. The threshold is actually fixed to 10%.

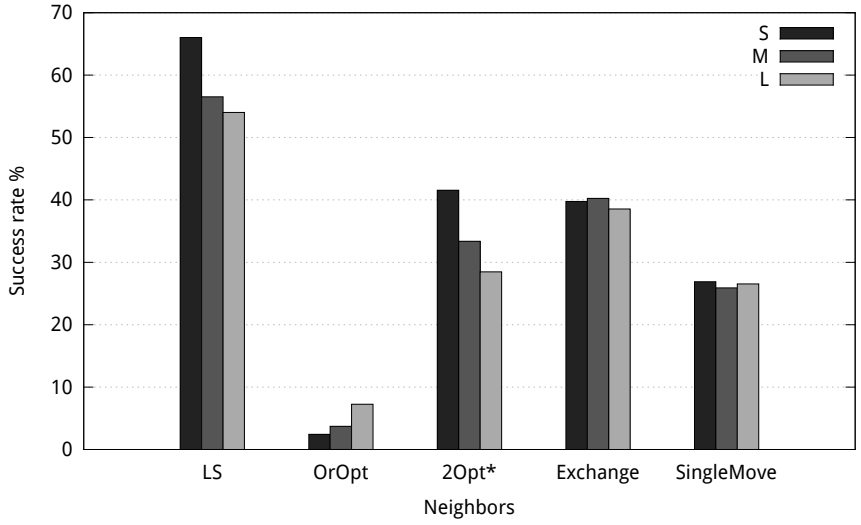


Figure 5.6 – Success rates of the neighborhoods in minimizing the total travel time.

5.2.3 Comparative results

With the parameters found in the previous sections, our algorithm is then tested on the whole benchmark. In addition, since our approach is heuristic and it serves the purpose of being fast, the runtime for our algorithm is also limited to 200 seconds, compared to the 1 hour limit commonly used by the exact approaches. Tables 5.1 to 5.3 report our results and compare them with the existing methods in

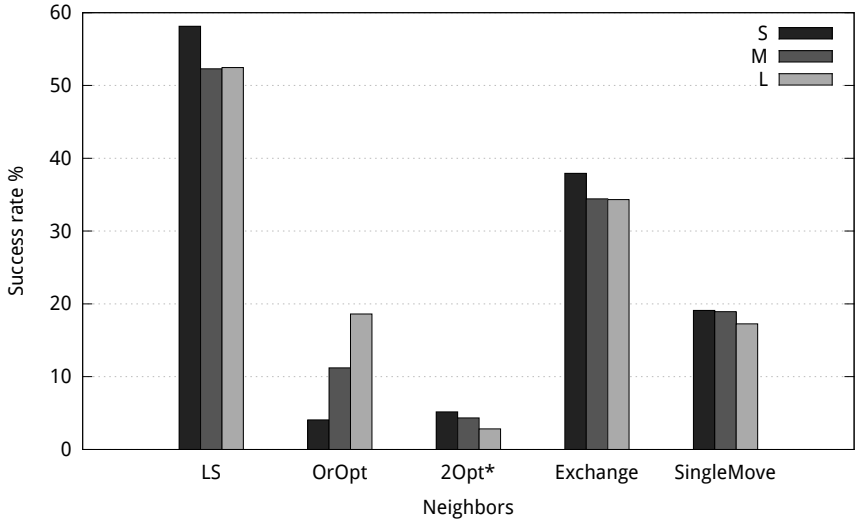


Figure 5.7 – Success rates of the neighborhoods in optimizing the preference.

the literature and the exact methods presented earlier in Chapter 4. Column *Best* shows the best known solution collected from all methods (including ours) for each instance. A star mark (*) is used in *Best* to indicate that the solution has been proved to be optimal by an exact method. The other column are: VMIP, MIP and CP for the results of the methods presented in Chapter 4; BP for the results of the branch-and-price algorithms presented in [16] and finally SA-ILS for our simulated annealing based iterative local search algorithm. Columns Sol and CPU report the best solution found by each method and the associated computational time. Bold numbers in Sol indicate that the solution quality reaches *Best*. The time unit in those tables for the objective values like travel time or fairness is in hours, and for the computational time is in seconds.

Table 5.1 – Comparison of the solutions and computational times for the total travel time.

Data	*Best*	BP		VMIP		MIP		CP		SA-ILS	
		Sol	CPU	Sol	CPU	Sol	CPU	Sol	CPU	Sol	CPU
1S	3.55*	3.55	1.12	3.55	2	3.55	0.02	3.55	2.29	3.55	0
1M	3.55*	3.55	3.69	3.55	11.29	3.55	0.06	3.55	2.9	3.55	0
1L	3.39*	3.39	11.91	3.39	82.24	3.39	0.09	3.39	2.12	3.39	0.01
2S	4.27*	4.27	0.56	4.27	0.11	4.27	0.02	4.27	1.37	4.27	0
2M	3.58*	3.58	3.2	3.58	1.96	3.58	0.09	3.58	6.36	3.58	0.01
2L	3.42*	3.42	7.41	3.42	368.25	3.42	0.16	3.42	13.63	3.42	0.01
3S	3.63*	3.63	3.84	3.63	2.79	3.63	0.03	3.63	4.67	3.63	0.02
3M	3.33*	3.33	4.31	3.33	5.95	3.33	0.04	3.33	1.64	3.33	0.03
3L	3.29*	3.29	1.44	3.29	141.24	3.29	0.14	3.29	2.19	3.29	0.02
4S	6.14*	6.14	1.54	6.14	28.52	6.14	0.06	6.14	3.02	6.14	0.02
4M	5.67*	5.67	2.55	5.67	1006.2	5.67	0.28	5.67	3.22	5.67	0.05
4L	5.13*	5.13	7.69	5.23	3600	5.13	5.51	5.13	1.25	5.13	0.59
5S	3.93*	3.93	2.9	3.93	18.39	3.93	0.04	3.93	0.94	3.93	0.03
5M	3.53*	3.53	9.1	3.53	4.6	3.53	0.03	3.53	139.15	3.53	0.03
5L	3.34*	3.34	5.15	3.34	533.43	3.34	0.16	3.34	27.15	3.34	0.02
6S	8.14*	8.14	197	8.75	3600	8.14	0.43	8.2	618.25	8.14	3.99
6M	7.7*	7.7	3600	-	3600	7.7	32.14	7.85	156.89	7.7	7.96
6L	7.14*	7.14	3600	-	3600	7.14	3600	7.54	161.15	7.14	5.01
7S	8.39*	8.39	169	12.9	3600	8.39	6.91	8.56	2151.73	8.39	4.62
7M	7.48*	7.56	3600	-	3600	7.48	896.02	7.63	1343.49	7.48	8.5
7L	6.88	6.88	3600	-	3600	6.88	3600	6.95	18.21	6.88	6.76
8S	9.54*	9.54	850	-	3600	9.54	24.43	10.1	256.53	9.54	6.6
8M	8.54*	8.54	3490	-	3600	8.54	756.77	8.94	230.02	8.54	7.08
8L	8.02	8.11	3600	-	3600	8.11	3600	8.23	270.13	8.02	8.49
9S	11.91	12.21	3600	-	3600	11.95	3600	12.53	1120.43	11.91	87.61
9M	10.93	11.04	3600	-	3600	10.93	3600	11.68	2678.73	10.93	61.81
9L	10.43	10.89	3600	-	3600	10.64	3600	11.22	2776.35	10.43	67.66
10S	8.54*	9.13	3600	-	3600	8.54	1789.57	9.62	589.86	8.57	78.25
10M	7.63	8.1	3600	-	3600	7.67	3600	8.15	1223.57	7.63	93.4
10L	7.38	-	3600	-	3600	7.84	3600	8.4	1793.67	7.38	63.1

Table 5.2 – Comparison of the solutions and computational times for the sum of negative preferences.

		BP|		VMIP		MIP		CP		SA-ILS	
Instance	Best	Sol	CPU	Sol	CPU	Sol	CPU	Sol	CPU	Sol	CPU
1S	-114.03*	-114.03	1.27	-114.03	1.31	-114.03	10.74	-114.03	0.64	-114.03	0.04
1M	-117.8*	-117.8	1.68	-117.8	1.54	-117.8	23.85	-117.8	0.33	-117.8	0.01
1L	-118.51*	-118.51	2.55	-118.51	24.43	-118.51	44.87	-118.51	0.42	-118.51	0.07
2S	-92.09*	-92.09	0.6	-92.09	0.43	-92.09	19.48	-92.09	0.23	-92.09	0.25
2M	-104.81*	-104.81	2.3	-104.81	38.53	-104.81	98.66	-104.81	0.32	-104.81	0.04
2L	-107.64*	-107.64	6.44	-107.64	795.11	-107.64	1624.45	-107.64	1.14	-107.64	0.24
3S	-99.49*	-99.49	1.66	-99.49	0.69	-99.49	4.41	-99.49	0.25	-99.49	0.19
3M	-106.59*	-106.59	2.01	-106.59	2.31	-106.59	66.52	-106.59	0.8	-106.59	0.01
3L	-107.87*	-107.87	2.63	-107.87	7.13	-107.87	267.63	-107.87	1.01	-107.87	0.37
4S	-100*	-100	1.72	-100	1.12	-100	1.87	-100	0.82	-100	0.11
4M	-106.72*	-106.72	2.36	-106.72	56.04	-106.72	136.55	-106.72	0.33	-106.72	0.07
4L	-109.27*	-109.27	5.04	-109.27	864.37	-107.1	3600	-109.27	1.1	-109.27	1.9
5S	-76.29*	-76.29	0.64	-76.29	0.13	-76.29	2.1	-76.29	0.63	-76.29	0.14
5M	-76.29*	-76.29	1.28	-76.29	2.42	-76.29	16.87	-76.29	0.45	-76.29	0.01
5L	-84.21*	-84.21	2.21	-84.21	43.96	-84.21	67.25	-84.21	0.8	-84.21	0.09
6S	-370.06*	-370.06	150.63	-370.06	1058.22	-288.08	3600	-370.06	28.9	-370.06	3.66
6M	-379.88*	-379.88	247.88	-	3600	-352.2	3600	-379.88	47.96	-379.88	5.9
6L	-387.2*	-387.2	474.15	-	3600	-313.63	3600	-387.2	8.38	-387.2	9.58
7S	-401.11*	-401.11	291.29	-401.11	3600	-308.75	3600	-401.11	86.89	-401.11	0.79
7M	-406.17*	-406.17	86.7	-	3600	-245.58	3600	-406.17	7.68	-406.17	4.23
7L	-407.48*	-407.48	710.62	-	3600	-237.99	3600	-407.48	10.3	-407.34	1.57
8S	-380.76*	-380.76	135.39	-288.63	3600	-178.57	3600	-376.42	131.18	-380.76	11.14
8M	-403.57*	-403.57	290.77	-	3600	-235.6	3600	-403.57	248	-403.36	11.37
8L	-407.48*	-407.48	362.18	-	3600	-208.98	3600	-407.48	163.11	-407.48	10.72
9S	-605.46	-552.65	3600	-	3600	-335.57	3600	-583.03	1458.29	-605.46	200.04
9M	-658.68	-463.82	3600	-	3600	-320.42	3600	-654.43	1563	-658.68	71.46
9L	-670.39	-663.47	3600	-	3600	-306.29	3600	-669.97	1344.73	-670.39	200.02
10S	-675.81	-675.81	3600	-	3600	-393.19	3600	-675.66	2403.72	-675.74	64.73
10M	-686.77	-685.31	3600	-	3600	-343.19	3600	-686.77	630.79	-686.77	155.32
10L	-691.86	-691.34	3600	-	3600	-367.37	3600	-691.86	346.85	-691.86	162.04

Table 5.3 – Comparison of the solutions and computational times for the fairness objective.

Data	*Best*	VMIP		MIP		CP		SA-ILS	
		Sol	CPU	Sol	CPU	Sol	CPU	Sol	CPU
1S	0*	0	51.01	0.04	3600	0	13.95	0	0.8
1M	0*	0	435.57	0.04	3600	0	11.36	0	0.4
1L	0*	0	3025.43	0.06	3600	0	4.06	0	0.61
2S	0.01*	0.01	132.06	0.05	3600	0.01	15.84	0.01	0.08
2M	0.01	0.01	3600	0.02	3600	0.01	2.54	0.01	0.7
2L	0.01	0.01	3600	0.05	3600	0.01	8.87	0.01	0.07
3S	0.01*	0.01	1868.16	0.04	3600	0.01	3.13	0.01	0.02
3M	0.01*	0.01	2690.94	0.01	3600	0.01	1.46	0.01	0.9
3L	0.01	0.01	3600	0.06	3600	0.01	1.19	0.01	0.8
4S	0.06*	0.06	1399.77	0.06	3600	0.06	2.77	0.06	0.98
4M	0.02	0.02	3600	0.03	3600	0.02	14.17	0.02	1.17
4L	0.02	0.03	3600	0.02	3600	0.02	4.52	0.02	1.8
5S	0.01	0.01	1168	0.08	3600	0.01	0.67	0.01	0.9
5M	0.01	0.01	3600	0.09	3600	0.01	4.5	0.01	0.39
5L	0.01	0.03	3600	0.04	3600	0.01	2.31	0.01	3.4
6S	0.01	0.55	3600	2.01	3600	0.01	1595.76	0.08	27.74
6M	0.01	0.55	3600	2	3600	0.01	575.75	0.05	36.99
6L	0.01	-	3600	2.02	3600	0.01	819.01	0.05	57.56
7S	0.03	0.3	3600	2.39	3600	0.03	596.91	0.08	29.54
7M	0.01	1.27	3600	3.3	3600	0.01	245.18	0.08	29.97
7L	0.01	-	3600	-	3600	0.01	519.16	0.08	44.68
8S	0.05	-	3600	2.03	3600	0.05	436.32	0.09	28.91
8M	0.04	-	3600	-	3600	0.04	216.83	0.08	24.22
8L	0.03	-	3600	-	3600	0.03	697.88	0.09	38.86
9S	0.08	-	3600	-	3600	0.08	1050.03	0.16	187.73
9M	0.06	-	3600	-	3600	0.06	1118.03	0.14	200.06
9L	0.06	-	3600	-	3600	0.06	915.6	0.11	200.03
10S	0.03	-	3600	-	3600	0.03	1164.32	0.08	143.37
10M	0.03	-	3600	-	3600	0.03	1075.03	0.1	200.21
10L	0.01	-	3600	-	3600	0.01	993.47	0.08	200.12

From these results, we notice that SA-ILS finds 23 known optimal solutions from 24 when minimizing the total travel time and all the optimal solutions (24) when minimizing the sum of preference in very short computational times compared to the other methods. Some solution qualities for the remaining instances are also better than the ones found by the other methods. In addition, Figures 5.8 and 5.9 illustrate the comparison between the computational times required by the methods to reach their best solutions and the ones required by our method to achieve the same solution quality.

For the total travel time, the algorithm strictly improved the best known solutions for 4 instances of the data sets. Those instances are $8L$, $9M$, $9L$ and $10L$. For the sum of negative preferences, we could improve 1 instance ($9M$). For

those two objectives, the improved solutions have been discovered by our methods within 3 minutes.

To summarize, our SA-ILS is clearly fast and efficient in optimizing these two objectives.

For the fairness objective, SA-ILS finds all the best known solutions for the instances with 20 visits. It finds some difficulties for the larger instances where CP still dominates all the other methods.

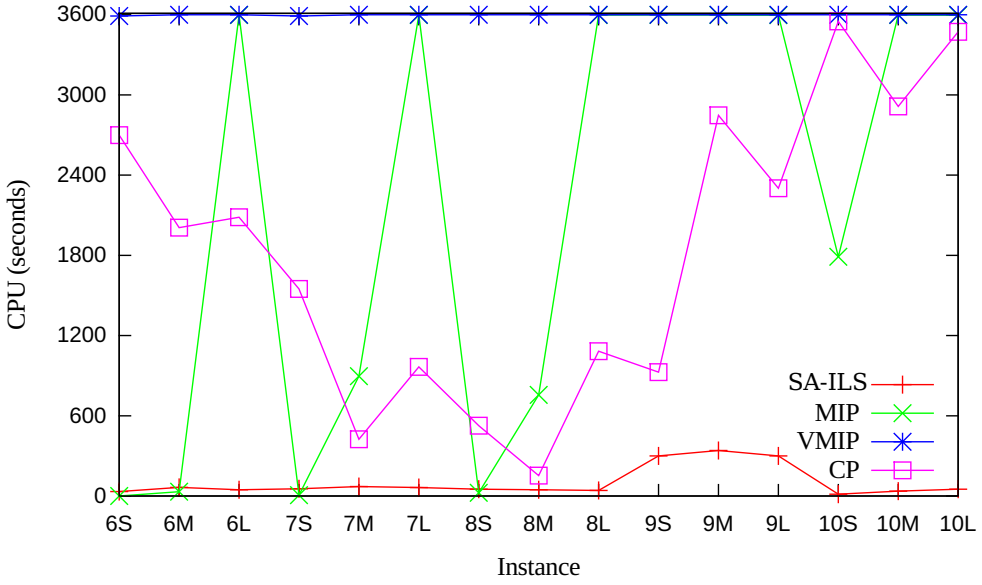


Figure 5.8 – Computational times to reach the travel time equivalent to the one in the literature.

5.3 Conclusion

In this chapter, we have proposed a new algorithm to address VRPTWSyn. The approach is based on a Simulated Annealing mechanism with and ILS based diversification and an adaptive multiple neighborhoods. To the best of our knowledge, this is the first time that dedicated local search heuristics have been proposed and evaluated on this variant of VRP. The experiments conducted on the standard

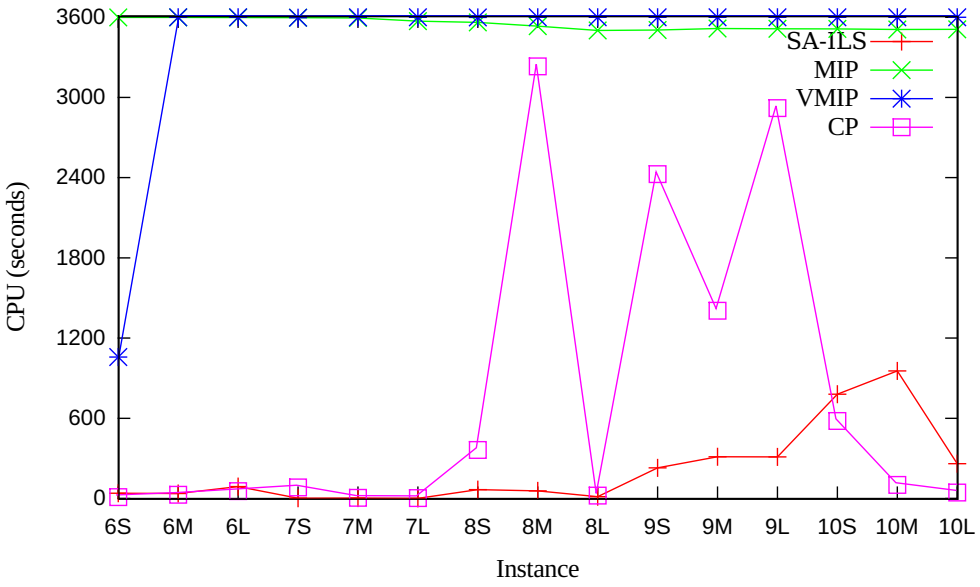


Figure 5.9 – Computational times to reach the sum of preferences equivalent to the one in the literature.

benchmark [17] for VRPTWSyn clearly demonstrate the efficiency and the competitiveness of our approach compared to the existing methods in the literature especially when considering the total travel time or the sum of preferences as objective. The results also confirm that destruction/repair operator and local search heuristics can be efficiently adapted to support the synchronization constraints.

As a future work, we plan to develop efficient hybrid methods to solve VRPTWSyn since there are still unsolved instances in the benchmark. In particular, having an efficient algorithm being able to produce a high quality feasible solutions in short computational times, such as our SA-ILS, is a significant advantage. This can, for example, be used as a warm start for the exact methods. Another obvious challenge will be to tackle larger problems, derived from those of VRPTW for example or to adapt the algorithm for more generalized cases especially those described in [30].

étxc

Conclusions and future works

In this dissertation, we explored the idea of developing efficient methods to solve two main problems that model many real world issues in the field of the vehicle routing problems. The solution proposed approaches include both exact methods and heuristics and propose either lower bounds or upper bounds. To evaluate these methods, numerical experimentations and comparison with the best results in the literature were provided.

After giving some essential definitions and notations in the first chapter, the dissertation included two parts where each focused on one of the variants of the vehicle routing problem.

The first part investigated a well known problem referred as the capacitated Vehicle Routing Problem with Time Windows (VRPTW) and included two chapters. In Chapter 2 we presented an improved version of our published paper which proposes new lower bounds techniques on the number of vehicles needed to serve all the VRPTW customers. We provided an analysis of several lower bounds based on the incompatibility between customers and on vehicle capacity constraints. We also developed an adaptation of Energetic Reasoning algorithm and used decomposition techniques to reduce the size of the problem. The numerical results confirmed the contribution brought by our proposed approaches within a very fast computing time. Chapter 3 proposed a Particle Swarm Optimization method that deals with the Solomon objective which consists on minimizing the

number of vehicles and then the total travel time. We tested our method on the instances of the literature which include up to 1000 customers. We showed how the combination of the initial phase using a destruction/repair heuristic with a Set Partitioning problem improved the results. It reached competitive results especially for the instances of Solomon and Desrosiers [82]. As a future work we intend to study in depth the parameters settings and the performance of the algorithm notably when considering big instances.

In the second part of the dissertation, we focused on a new variant of VRPTW and studied a new type of temporal constraints. We considered the synchronization constraints which require more than one vehicle to serve a customer at the same time. The problem is called the Vehicle Routing Problem with Time Windows and Synchronized Visits (VRPTWSyn). Chapter 4 covered three exact methods and compared them with the existing ones from the literature. After using the classical model, a new reduced formulation was proposed. Both methods were executed using a branch-and-cut like scheme in which the subtours elimination constraints were used. Later, a constraint programming model, which explores the scheduling characteristics of the problem, was also proposed. A comparison was presented at the end between all the methods while dealing the three different problem objectives. At the end, a new approach based on a Simulated Annealing mechanism with and ILS based diversification and an adaptive multiple neighborhoods was proposed for VRPTWSyn in Chapter 5. The experiments conducted on the standard benchmark clearly demonstrated the efficiency and the competitiveness of the algorithm compared to the existing methods especially when considering the total travel time or the sum of preferences as an objective. The results also confirmed that destruction/repair operator and local search heuristics can be efficiently adapted to support the synchronization constraints.

Future works

The primary goal of this thesis was the study of new solution approaches for the vehicle routing problem with or without synchronization. The observations made in these works present some interesting open questions. In the following, we outline some directions for future research.

We have seen various fast algorithms that propose tight lower bounds for VRPTW. Some of them might even be fast enough to be embedded in a search framework. This constitutes an encouraging first step towards the development of an effective branch-and-bound or a constraint programming procedure for solving VRPTW. The development of such an exact solution approach is a part of our ongoing research. Future research also needs to be focused on investigating similar techniques to more complex vehicle routing problems such as those requiring non heterogeneous fleet with different capacities or/and different availabilities, or those containing customers with multiple time windows. This can be modeled using as many nodes as time windows for each customer. Only one node will be visited among the nodes associated with a given customer. An adaptation to the Pickup and Delivery with Time Windows would also be a promising path. Aiming to satisfy transportation requests, each requiring both pickup and delivery under capacity, time window and precedence constraints may considerably enforce the incompatibility graph and consequently the lower bounds that use the Bin-packing relaxation.

A future work will be conducted to further improve the proposed PSO algorithm for VRPTW. More sensitivity analyses on the parameters are planned. The use of a more sophisticated set partitioning based approach which can be extended to a branch-and-price scheme is also investigated. This may yield better solutions when considering the number of vehicles. A more sophisticated and dedicated local search routine to the algorithm is likely to improve the result. In addition to that, several neighborhood searches, which benefit from the relaxation of the

problem to explore more solution areas [94], can be applied to the problem. Many relaxation schemes should be tested and compared.

Regarding the algorithms dedicated to the synchronization case, we were restricted to problems with up to 80 customers. An obvious challenge will be to tackle larger problems, derived from those of VRPTW for example. It will be interesting to study how the use of more sophisticated cuts can have an impact on the efficiency of the algorithms, by simply starting with the cuts tested on similar variants. This may include for example the Comb Inequalities, Box Inequalities, Path-Box Inequalities and path inequalities [11].

We also plan to develop efficient hybrid methods combining the fast heuristic with the exact algorithms proposed in both chapters (Chapters 4 and 5). Using solutions from the SA-ILS as warm start for the exact methods should be a promising technique. We are also investigating the use of the exact methods for the repair phase within the SA-ILS algorithm.

The proposition of a multi-objective algorithm which deals with all the presented criteria would be an interesting area, dealing with the travel cost, the sum of preferences and the workload balance in the same time. The support of visitors with different qualifications is contemplated. It is promising to study how this can be done in practice and increase the application of this problem to more real-life cases.

It is known how these types of problems have experienced tremendous growth of applications in recent years. However, variability in data affects considerably the quality of the solutions. Indeed, for example, some real life logistics systems demands might arrive randomly and continuously in time or when deliveries need to be made to external visitors. Therefore, there is a real need to develop routing tools that manage uncertain cases. Robust extensions of both problems are being discussed and studied by considering some cases where the travel times or the demands may belong to an uncertainty set. Hence, our plans fall into the

development of a framework of robust programming, where a solution is said to be feasible only if it is feasible for all realizations of the uncertain data.

Appendices

A | Paper: New Lower Bounds on the Number of Vehicles for VRPTW

New Lower Bounds on the Number of Vehicles for the Vehicle Routing Problem with Time Windows

Sohaib AFIFI, Rym Nesrine GUIBADJ and Aziz MOUKRIM

Université de Technologie Compiègne
Laboratoire Heudiasyc, UMR 7253 CNRS, 60205 Compiègne, France
{sohaib.afifi,rym-nesrine.guibadj,aziz.moukrim}@hds.utc.fr

Abstract. The Vehicle Routing Problem with Time Windows (VRPTW) consists in determining the routing plan of vehicles with identical capacity in order to supply the demands of a set of customers with predefined time windows. This complex multi-constrained problem has been widely studied due to its industrial, economic and environmental implications. In this work, we are interested in defining the number of vehicles needed to visit all the customers. This objective is very important to evaluate the fixed costs for operating the fleet. In this paper, we provide an analysis of several lower bounds based on incompatibility between customers and on vehicle capacity constraints. We also develop an adaptation of Energetic Reasoning algorithm for VRPTW with a limited fleet. The proposed approach focuses on some time-intervals and exploits time constraints, incompatibility graph and bin packing models in order to obtain new valid lower bounds for the fleet size. Experiments conducted on the standard benchmarks show that our algorithms outperform the classical lower bound techniques and give the minimum number of vehicles for 339 out of 468 instances.

Keywords: vehicle routing, time windows, lower bounds, energetic reasoning.

1 Introduction

In today's business world, transportation costs become a major share of the total logistic expenses of companies. That is why many companies try to improve their transportation by using rational manners and effective tools. The objective of these problems is to make a vehicle scheduling strategy in order to minimize the number of routes and the corresponding total travel distance or cost. In the literature such problems are referred to as *routing problems*.

The vehicle routing problem with time windows (VRPTW) [10] is among the most studied variants of routing problems due its wide range of applications. Common examples are newspaper delivery, beverage and food delivery, commercial and industrial waste collection [13]. In VRPTW, a set of customers must

be served by a fleet of vehicles located in a single depot. A quantity of goods should be delivered to each customer whose service takes an amount of time. Each customer is associated with a time window that represents the interval of time when the customer is available to receive the service. This means that if the vehicle arrives too soon, it should wait until the opening of the time window to serve the customer while too late arrival is not allowed. Since deliveries cannot be split, a customer must be served by a single vehicle. All vehicles are identical and have a maximum capacity Q. The aim is to plan the minimal number of routes starting and ending in a unique depot in order to serve all the customers while respecting all the time windows and capacity constraints.

VRPTW was first introduced by Solomon [25]. Both exact and heuristic algorithms have been proposed to solve VRPTW. Most of the exact methods focus on the variant of the problem where the number of available vehicles is not fixed. A review on the exact methods up to 2002 is reported in [7]. Kallehauge in [17] gave a detailed analysis of existing formulations. More recently, Baldacci et al. [3] reviewed mathematical formulations, relaxations and recent exact methods. They reported the computational comparison between the methods proposed in [15], [8] and [2] that are considered as the most effective exact methods in the literature. These approaches have significantly improved the quality of the lower bounds for instances with up to 100 customers. The key factor of their success is the effective combination between the set partitioning formulation and the column generation based algorithms.

Since, VRPTW is an NP-Hard problem [21], the computational times for exact methods can be very high, even for instances with a moderate size. This has been the motivation for some researches to focus on approximate methods. It is worth pointing out that the literature concerning VRPTW is split according to the objective considered. While exact methods usually minimize the total traveled distance, most heuristics consider a hierarchical objective which first minimizes the number of vehicles used and then the total distance. Thus, a solution that employs fewer vehicles is always better than a one using more, even if its total traveled distance is worse. A good survey of heuristic methods is reported in the papers of Bräysy and Gendreau [5] [6]. Among the best performing heuristics are the hybrid genetic algorithm of [16], the column generation heuristic of [1] and the memetic algorithm of [20]. A new optimization framework was later developed by Ursani et al. [27] for the distance minimization objective only. This framework is an iterative procedure between optimization and deterioration phases and uses a genetic algorithm as an optimization methodology. In the recent paper of Vidal et al. [28], a hybrid genetic solver is developed to deal with a large class of time-constrained vehicle routing problem. A third stream of research focuses on solving VRPTW as a multi-objective problem in which both vehicles and cost are considered depending on the needs of the user [26] [24].

The goal of this paper is to use scheduling methods via Energetic Reasoning in order to develop new lower bounding procedures for VRPTW. This is mainly based on constraint propagation concept. The objective is to reduce the computational effort by removing some values from the variables of the problem

because a given subset of the constraints cannot be satisfied. The remained of the paper is organized as follows. Section 2 briefly describes the problem. In Sections 3 and 4, the detailed description of the proposed lower bound methods is given and in Section 5 the results of a computational study are reported. Finally, Section 6 provides some concluding remarks.

2 Problem formulation

In the following, we present a mixed integer formulation for VRPTW. The problem is modeled using an oriented graph $G = (V^+, E)$, where $V^+ = \{0, 1, 2, ..., n\}$ is the vertex set representing the set of customers $V = \{1, 2, ..., n\}$ and the depot 0. $E = \{(i, j) : i \neq j, i, j \in V^+\}$ is the edge set. The capacities of all vehicles are equal and are denoted by Q. A demand q_i, a service time s_i and a time window $[e_i, l_i]$ are associated to each vertex $i \in V$. Vehicle v cannot arrive later than l_i and if it arrives earlier than e_i, it must wait before the service can start. Each edge $(i, j) \in E$ is associated with a travel cost $\delta_{i,j}$ which satisfies the triangle inequality. Each vehicle must start and finish its tour at the depot. Each customer must be served within a predefined time window and assigned to exactly one vehicle. The total size of deliveries for customers assigned to the same vehicle must not exceed the vehicle capacity Q and the travel cost/time $C(R)$ of each tour R must not exceed l_0 which is the latest possible arrival time to the depot.

The model involves three types of variables: the binary routing variables $x_{ij} \in \{0, 1\}\,(i, j \in V^+)$, the scheduling variables $w_i \geq 0\,(i \in V)$ and the vehicle load variables $y_i\,(i \in V)$. The routing variables x_{ij} is one if a vehicle traverses the arc $(i, j) \in E$. The scheduling variable w_i denotes the time the vehicle arrives at customer $i \in V$. y_i denotes the vehicle load at departure from customer i. The formulation is as follows:

$$\min \sum_{i \in V} x_{0i} \tag{1}$$

subject to:

$$\sum_{j \in V^+} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V^+} x_{ij} - \sum_{j \in V^+} x_{ji} = 0 \qquad \forall i \in V^+ \tag{3}$$

$$w_j \geq w_i + x_{ij}(\max(\delta_{i,j} + s_i, e_j - l_i)) - (1 - x_{ij})(l_i - e_j) \qquad \forall i, j \in V \tag{4}$$

$$e_i \leq w_i \leq l_i \qquad \forall i \in V \tag{5}$$

$$y_j \geq y_i + q_j - (1 - x_{ij})Q \qquad \forall i, j \in V \tag{6}$$

$$q_i \leq y_i \leq Q \qquad \forall i \in V \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V^+ \tag{8}$$

The objective function (1) is to minimize the total number of vehicles used to serve all the customers. Constraints (2) and (3) define the routing network and the constraints (4) and (5) guarantee the connectivity of each tour and ensure that the time windows are respected. We assume that the time windows are adjusted such that $e_i = \max(e_i, \delta_{0,i})$ and $l_i = \min(l_i, l_0 - (\delta_{i,0} + s_i))\ \forall i \in V$. Constraints (6) and (7) ensure that the vehicle's capacity is not exceeded. Also constraints (6) eliminate subtours in a manner similar to (4). Finally, (8) are integral constraints.

3 Classical lower bounding techniques

There were only few attempts to propose lower bounds for VRPTW when the objective is to minimize the number of vehicles. To the best of our knowledge, the most competitive results are currently offered by Kontoravdis and Bard [19]. In this section, we briefly review the main features of their lower bounding heuristics.

3.1 A lower bound based on incompatibilities between customers

The first lower bound is deduced from the incompatibility constraints. Let i and j be two customers. If there is no feasible route containing i and j then they define an incompatible pair denoted by $i||j$. Such a situation occurs if one of the following conditions is verified:

1. Customers i and j cannot be in the same route due to their time window constraints:
   $(e_i + s_i + \delta_{i,j} > l_j) \wedge (e_j + s_j + \delta_{j,i} > l_i) \Rightarrow i||j$.
2. The travel cost of any tour with i and j exceeds the cost limit l_0:
   $(C(R_1) > l_0) \wedge (C(R_2) > l_0)$ where $R_1 = (0, i, j, 0) \wedge R_2 = (0, j, i, 0) \Rightarrow i||j$.
3. The sum of the demands is greater than the vehicle capacity:
   $q_i + q_j > Q \Rightarrow i||j$.

Using these conditions, we build the graph of incompatibilities between customers defined as: $G^V_{inc} = (V, E_V)$ where $E_V = \{(i, j) \in V \times V : i||j\}$. Based on this graph the minimum number of routes to be used, denoted LB_{Clique}, is equal to the size of the maximum clique extracted from G^V_{inc}.

3.2 A lower bound based on vehicle capacity constraints

The second bound is based on a relaxation of time window constraints. When considering only the capacity constraints, VRPTW can be reduced to a Bin Packing Problem (BPP). Each vehicle is considered as a bin with fixed size Q and each customer demand as an item with size q_i that should be put in a bin. Any lower bound $LB_{Capacity}$ on the number of bins required to pack all the items is considered as a valid lower bound for VRPTW.

3.3 A lower bound based on the amount of needed travel time

This lower bound consists of calculating the minimum number of bins LB_{BP} of capacity l_0 to pack $n+m$ items. The size θ_i of an item $i, 1 \leq i \leq n$, represents the necessary amount of time that a vehicle needs to serve customer i and to travel to its closest neighbor. This time is defined by:

$$\theta_i \leftarrow \min_{j \in V^+} \{\max(\delta_{i,j} + s_i, e_j - l_i)\} \tag{9}$$

The sizes of the other m items correspond to the m least travel times from the depot to the first served customers where $m = \max(LB_{Clique}, LB_{Capacity})$.

4 New lower bounds inspired from Energetic Reasoning

In this section, we first present a brief overview of Energetic Reasoning, then we discuss its adaptation to VRPTW.

4.1 Energetic Reasoning

Energetic Reasoning (ER) is one of the most powerful propagation algorithms. It has been originally developed by Erschler et al. [9] for Cumulative Scheduling Problems (CuSP). The idea is to propose a smart way to simultaneously consider time and resource constraints in a unique reasoning. In this context, the energy is generally defined by multiplying the time duration by the resource quantity of a given time interval. Considering the quantities of energy supplied by the resources and consumed by the tasks within given intervals, the energetic approach aims to develop satisfiability tests to ensure that a given schedule is feasible. Since its inception, Energetic Reasoning has gained popularity and has been used for solving more complex scheduling problems [23].

In order to keep the same notation used for vehicle routing problem, we describe the CuSP as follows. We consider a set V of n activities to be scheduled on a resource of quantity m. Each activity i has a release time e_i, a latest start time l_i and a processing time s_i. Moreover, the activity i requires a constant amount b_i of resource throughout its processing. We will deal here only with the case where $b_i = 1, \forall i \in V$. This is equivalent to the problem of scheduling n activities on m identical parallel machines. For ease of presentation, we denote this problems as PMSP.

Given a time interval $[t_1, t_2]$, with $t_1 < t_2$, the part of an activity i that must be processed between t_1 and t_2 is called *work* of i in the time interval $[t_1, t_2]$. To compute this *work*, the activities are either *left-shifted* or *right-shifted* on their time window, which means that, they can start either at their release date e_i, or at their latest start time l_i. Thus, the work of an activity i over $[t_1, t_2]$ is equal to the minimum between its *left work* and its *right work*. For convenience, the left work, the right work and the work of an activity i over $[t_1, t_2]$ are denoted

respectively $W_{left}(i,t_1,t_2)$, $W_{right}(i,t_1,t_2)$ and $W(i,t_1,t_2)$. They are formally defined as follows:

$$W_{left}(i,t_1,t_2) = \min\{\ t_2 - t_1, s_i, \max(0, e_i + s_i - t_1)\} \quad (10)$$

$$W_{right}(i,t_1,t_2) = \min\{\ t_2 - t_1, s_i, \max(0, t_2 - l_i)\} \quad (11)$$

$$W(i,t_1,t_2) = \min(\ W_{left}(i,t_1,t_2), W_{right}(i,t_1,t_2)) \quad (12)$$

Finally, we define the total work over $[t1,t2]$ as the sum of the works of all the activities $W(t_1,t_2) = \sum_{i=1}^{i=n} W(i,t_1,t_2)$ and the available energy in the considered interval as $E(t_1,t_2) = m * (t_2 - t_1)$. If the total work is greater than the available energy then no feasible solution exists.

Proposition 1 ***satisfiability test***
if $\exists[t_1,t_2]$, $W(t_1,t_2) > E(t_1,t_2)$ *then the instance is infeasible.*

Note that one crucial point to apply efficiently Energetic Reasoning is to determine the relevant time-intervals on which it may be useful to check feasibility conditions. Baptiste et al. [4] have proved that the only relevant time intervals $[t_1,t_2]$ that need to be considered are those where $t_1 \in T_1$ and $t_2 \in T_2$ such as $t_1 < t_2$, $T_1 = \{e_i, i \in V\} \cup \{l_i, i \in V\} \cup \{e_i + s_i, i \in V\}$ and $T_2 = \{l_i + s_i, i \in V\} \cup \{e_i + s_i, i \in V\} \cup \{l_i, i \in V\}$. Therefore, the satisfiability test algorithm runs in $O(n^3)$. The detailed steps are summarized in Algorithm 1.

Algorithm 1: satisfiability test of Energetic Reasoning

```
   Data: I : PMSP instance;
 1 begin
 2     initialization;
 3     T1 = {e_i, i ∈ V} ∪ {l_i, i ∈ V} ∪ {e_i + s_i, i ∈ V};
 4     T2 = {l_i + s_i, i ∈ V} ∪ {e_i + s_i, i ∈ V} ∪ {l_i, i ∈ V};
 5     foreach t1 ∈ T1 do
 6         foreach t2 ∈ T2 such as t1 < t2 do
 7             W ← 0;
 8             foreach i ∈ V do
 9                 W ← W + W(i, t1, t2);
10             if W > m * (t2 − t1) then
11                 Infeasible instance ;
```

4.2 From VRPTW to PMSP

Our approach is to relax a VRPTW instance, where a limited number of vehicles is given, in order to obtain a PMSP instance. Once the transformation is performed, we apply the same satisfiability test on the relaxed m-VRPTW instance,

using Algorithm 1. Starting from a trivial value $m = \max(LB_{Clique}, LB_{Capacity})$, feasibility tests are carried out to detect an infeasibility (if in at least one of the time intervals, the minimum number of vehicles found exceeds m). If an infeasibility is detected, then $m + 1$ is a valid lower bound. The process is iterated until no infeasibility is detected.

A trivial relaxation of an m-VRPTW instance can be done by ignoring travel times, customer demands and vehicle capacities. We obtain a PMSP where the vehicles are considered as m identical parallel machines, the number of activities is equal to the number of customers n and each activity i has to be processed for s_i units of time by only one machine. The processing of activity i cannot be started before its release date e_i or after its lastest start time l_i.

In vehicle routing problems, travel times are not negligible compared to the service times. Ignoring the travel time would undervalue the energy consumed. Therefore, few adjustments could be performed and Energetic Reasoning becomes inefficient. Better results are obtained by considering the time that a vehicle needs to travel in order to visit each customer.

First, the travel time $\delta_{i,j}$ between the customers i and j is updated to eliminate the waiting time at the customer j.

$$\delta_{i,j} \leftarrow \max(\delta_{i,j}, e_j - (l_i + s_i)) \quad \forall i \in V \ \forall \ j \in V^+ \tag{13}$$

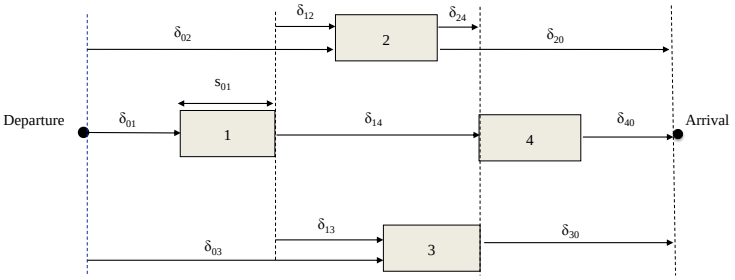
Then, the number of potential successors of customer i is reduced. This is performed by eliminating the transition $\delta_{i,j}$ if j cannot be served after i due to its time window:

$$if(e_i + s_i + \delta_{i,j} > l_j) \quad then \quad \delta_{i,j} \leftarrow \infty \qquad \forall i \in V^+ \ \forall \ j \in V^+ \setminus \{i\} \tag{14}$$

Before giving the detail of our travel evaluation procedure, we note by I' the instance derived from the m-VRPTW instance I. We associate I' with a graph $G' = (V', E')$ after performing the following transformations:

1. We introduce m artificial departure vertices V_d and m artificial arrival vertices V_a. Then, we define the set $V' = V \cup V_d \cup V_a$ with $V = \{1, ..., n\}$, $V_d = \{n + 1, ..., n + m\}$ and $V_a = \{n + m + 1, ..., n + 2 \times m\}$.
2. The set of arcs is defined by $E' = E \cup \{(i, j) : i \neq j, i \in V \cup V_d, j \in V \cup V_a\}$.
3. The distance matrix $\Delta = (\delta_{i,j})$ is extended to $\Delta' = (\delta'_{i,j})$ which is associated to E' such as:

$$\delta'_{i,j} = \begin{cases} \delta_{i,j} \ (i, j \in V), \\ \delta_{0,j} \ (i \in V_d, j \in V), \\ \delta_{i,0} \ (i \in V, j \in V_a), \\ \infty \ \ (i \in V', j \in V_d) \ or \ (i \in V_a, j \in V') \end{cases} \tag{15}$$



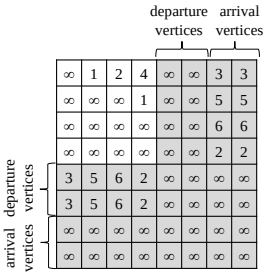
a) Illustration of an m-VRPTW instance with 4 customers and 2 vehicles

i	$[e_i,l_i]$	s_i
1	[2,3]	2
2	[6,6]	2
3	[7,7]	2
4	[9,10]	2
0	[0,14]	0

b) The customers data

from \ to	1	2	3	4	0
1	∞	1	2	4	3
2	∞	∞	∞	1	5
3	∞	∞	∞	∞	6
4	∞	∞	∞	∞	2
0	3	5	6	2	∞

c) The distance matrix Δ



∞	1	2	4	∞	∞	3	3
∞	∞	∞	1	∞	∞	5	5
∞	∞	∞	∞	∞	∞	6	6
∞	∞	∞	∞	∞	∞	2	2
3	5	6	2	∞	∞	∞	∞
3	5	6	2	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

d) The extended distance matrix Δ'

∞	**0**	1	3	∞	∞	2	2
∞	∞	∞	0	∞	∞	4	4
∞	∞	∞	∞	∞	∞	**0**	**0**
∞	∞	∞	∞	∞	∞	**0**	**0**
1	3	4	**0**	∞	∞	∞	∞
0	2	3	**0**	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

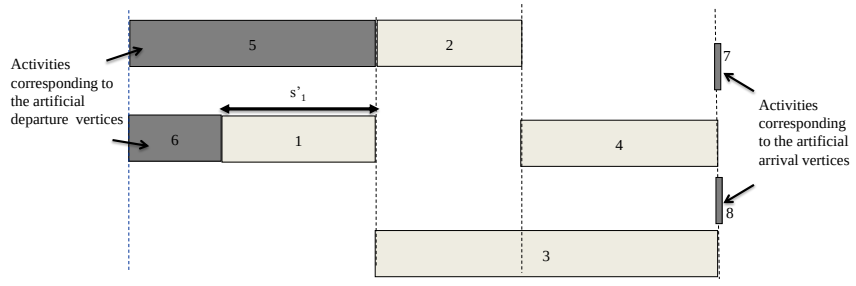
e) Reducing the rows

∞	**0**	**0**	3	∞	∞	2	2
∞	∞	∞	0	∞	∞	4	4
∞	∞	∞	∞	∞	∞	**0**	**0**
∞	∞	∞	∞	∞	∞	**0**	**0**
1	3	3	**0**	∞	∞	∞	∞
0	2	2	**0**	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

f) Reducing the columns

i'	$[e'_i,l'_i]$	s'_i
1	[2,3]	3
2	[6,6]	3
3	[6,6]	9
4	[9,10]	4
5	[0,0]	2
6	[0,0]	3
7	[0,14]	0
8	[0,14]	0

g) The activities data of the resulting PMSP



h) Illustration of the resulting PMSP

Fig. 1. An example of m-VRPTW instance relaxed to PMSP instance.

The set of vertices V' in G' denotes the $n + 2 \times m$ activities assigned to I'. The artificial departure activities corresponding to V_d have a time window equal to $[0, 0]$ and durations equal to the m smallest travel time from the depot to customers $\{\delta^{min}_{0,1}, ..., \delta^{min}_{0,m}\}$. This supposes that the vehicles must leave the depot immediately in order to visit the m first customers. The artificial arrival activities corresponding to V_a have the largest possible time window $[0, l_0]$ and no processing time. For the remaining activities, the range of the possible start dates is equal to the customer's time window $[e_i, l_i]$, while the processing time is equal to the sum of the service time s_i with the minimal travel time that the vehicle will necessary perform to reach the next customer.

$$[e'_i, l'_i] = \begin{cases} [0, 0] & i \in V_d, \\ [0, l_0] & i \in V_a, \\ [e_i, l_i] & i \in V \end{cases} \tag{16}$$

$$s'_i = \begin{cases} \delta^{min}_{0,i-n} & i \in V_d, \\ 0 & i \in V_a, \\ s_i + min_{j \in V'}\{\delta'_{i,j}\} & i \in V \end{cases} \tag{17}$$

Each row i , $i \in V'$ of the extended matrix Δ' is updated by subtracting the smallest element from the remaining ones (18). This means that the minimal travel time after serving customer i is subtracted from the total distance of any solution since every solution must include only one customer from this row. This process is called *reducing the rows*. It was introduced by [22] in order to solve the well known Traveling Salesman Problem.

$$\delta'_{i,j} = \begin{cases} \delta'_{i,j} - min_{j \in V'}\{\delta'_{i,j}\} & \forall i \in V j \in V', \\ \max(0, \delta'_{i,j} - \delta^{min}_{0,i-n}) & \forall i \in V_d j \in V' \end{cases} \tag{18}$$

Next, we apply the same argument to the resulting matrix, by considering the minimal travel time to arrive from any customer j to customer i (19). This time is added at the beginning of activity i. For this reason, the bounds of the corresponding time window are shifted (20) (21). After these reducing operations, the matrix Δ' contains at least one zero in each row and each column. The Figure 1 illustrates the relaxation of an m-VRPTW instance with 4 customers and 2 vehicles.

$$s'_i \leftarrow s'_i + min_{j \in V'}\{\delta'_{j,i}\} \qquad \forall i \in V' \tag{19}$$

$$e'_i \leftarrow \max(0, e'_i - min_{j \in V'}\{\delta'_{j,i}\}) \qquad \forall i \in V' \tag{20}$$

$$l'_i \leftarrow \max(0, l'_i - min_{j \in V'}\{\delta'_{j,i}\}) \qquad \forall i \in V' \tag{21}$$

According to the evaluation procedure of travel times, we distinguish two possible lower bounds $LB_{ER_{eval1}}$ and $LB_{ER_{eval2}}$. The former is obtained if the travel times to the successors are considered before the remaining travel times from the predecessors whereas the latter is obtained by reversing the order of the considered travels. LB_{ER} denotes the maximum between $LB_{ER_{eval1}}$ and $LB_{ER_{eval2}}$.

4.3 Bin-packing lower bounds and Energetic Reasoning

We extend Energetic Reasoning, using the Bin-Packing Problem with Conflicts (BPPC), to get tighter lower bounds for VRPTW. In each time-interval $[t_1, t_2]$, we compute the mandatory parts of activities and then we deduce an associated bin-packing instance. The decision version of BPPC that we use can be formulated as follows: given a set of items with different weights and a graph where the vertices represent the items and the edges represent the conflicts between the pairs of items; is there a packing of these items in less than m bins with a capacity T ?

We now state the link between a necessary condition for the existence of m-VRPTW solution and the existence of BPPC solution. Let $I'(V', G_{inc}, m)$ denote a relaxed instance of m-VRPTW where V' is the set of activities, m the number of available vehicles and G_{inc} the graph of incompatibilities between activities. Let $[t_1, t_2]$ be a time-interval, we assume that the corresponding mandatory parts of activities have been computed: $W(i, t_1, t_2), \forall i \in V'$. Then, $BPPC(I', G_{inc}, t_1, t_2)$ denotes the packing instance which is associated to the scheduling instance I' in the time-interval $[t_1, t_2]$. $BPPC(I', G_{inc}, t_1, t_2)$ is made of m bins and n' items of size $W_i = W(i, t_1, t_2), \forall i \in \{1, ..., n\}$. The size of the bin is equal to the length of the time-interval $T = t_2 - t_1$. Then, deciding whether all mandatory parts of the activities can be scheduled within $[t_1, t_2]$ in I' is equivalent to determine for $BPPC(I', G_{inc}, t_1, t_2)$ if all items can be packed into the available bins.

Property 1. If there exists a time-interval $[t_1, t_2]$, such that $BPPC(I', G_{inc}, t_1, t_2)$ has no solution, then there is no solution to the initial problem $I'(V, G_{inc}, m)$.

Since this lower bound is based on NP-hard relaxation of m-VRPTW, it is naturally much time consuming than LB_{ER}. Therefore, we did not lunch it on all the previously defined time intervals. An interval $[t_1, t_2]$ is selected if its conflict sub graph density is greater than or equal 80% or if the ratio of activities works to the available energy is close to 1 i.e. $W(t_1, t_2)/[m * (t_2 - t_1)] > 0.9$.

As stated in Section 4.2, Energetic Reasoning uses two procedures to determine the processing time of activities. The new obtained lower bound LB_{ERBPPC} represents the maximum between LB_{ERBPPC_eval1} and LB_{ERBPPC_eval2}. In the same way and by ignoring the conflict constraints, we can obtain a quicker lower bound LB_{ERBPP}.

The example in Figure 2 illustrates the contribution of bin packing lower bounds in the improvement of Energetic Reasoning results. We consider a VRPTW

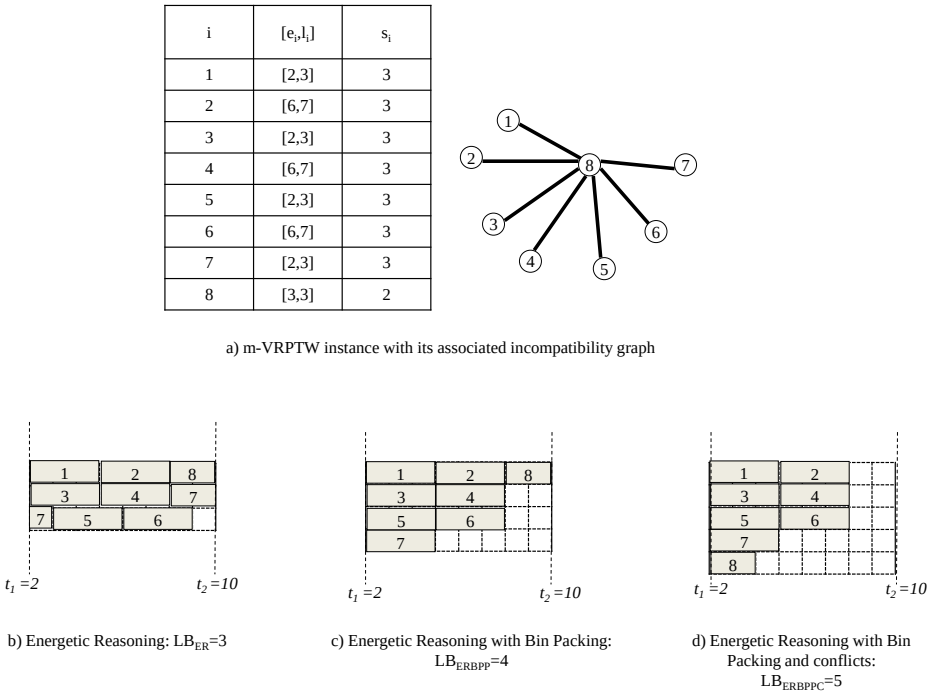


Fig. 2. Illustration of Energetic Reasoning lower bounds.

instance with 8 customers defined by their time windows and service times. We suppose that the vehicle capacity is large enough to satisfy all customers demands and that customer 8 cannot be served with any other customer. The results obtained by $LB_{Capacity}$ and LB_{Clique} are equal to 1 and 2 respectively. When analyzing the interval $[t_1, t_2]$, the Energetic Reasoning LB_{ER} gives 3. This result is improved by applying Bin Packing lower bound and taking into account the conflicts between customers ($LB_{ERBPP} = 4$ and $LB_{ERBPPC} = 5$).

5 Numerical results

We tested our algorithms on the well known instances of Solomon [25], Gehring and Homberger [11]. The benchmark comprises 6 sets ($R1$, $C1$, $RC1$, $R2$, $C2$, $RC2$). Each data set contains $25, 50, 100, 200, 400, 600, 800$ and 1000 customers who have specific euclidean coordinates. Customers' locations are determined using a random uniform distribution for the problem sets $R1$ and $R2$, but are restricted to be within clusters for the sets $C1$ and $C2$. Sets $RC1$ and $RC2$ have a combination of clustered and randomly placed customers. Sets $R1$, $C1$ and $RC1$ have a short scheduling horizon with tight time windows, while $R2$, $C2$ and $RC2$ are based on wide time windows. Our algorithms are coded in C++ and all experiments were conducted on an Intel(R) Core(TM) 2 Duo 2.93GHz.

Finding a clique with the greatest cardinality involves the use of an exact method with exponential worst case performance. Nevertheless, our experiments on the standard benchmarks show that the maximum clique can be identified in a fraction of a second using the exact method described in [18]. For the Bin Packing Problem, we use the heuristic algorithm developed by [14] to get good lower bounds in a reasonable computational times. When conflicts are considered in solving Bin Packing, we apply the approach proposed in [12]. For performance purpose, we launch this algorithm with a time out of 3 hours.

Table 1 and Table 2 compare the performance of our Energetic Reasoning bounds: LB_{ER}, LB_{ERBPP} and LB_{ERBPPC} to the elementary bounds present in the literature: LB_{Clique}, $LB_{Capacity}$ and LB_{BP}. The column $BestUB$ represents the overall best-published upper bounds. The maximum of the lower bounds is reported in column $BestLB$. In $AvgGAP$, we present the average gap between $BestUB$ and $BestLB$.

In general, the proposed techniques give the minimum number of vehicles of 339 instances among the 468 instances tested and give near optimal solution for the rest. The average performance of $LB_{Capacity}$ is consistently better than LB_{Clique}, but LB_{Clique} outperforms $LB_{Capcaity}$ in 5 instances in $C1$, 25 instances in $R1$, 4 instances in $RC1$ and 1 instance in $RC2$ by a margin of 128. This is due to the structure of the data sets which does not favor time and capacity incompatible pairs. On the other hand, the three new lower bounds: LB_{ER}, LB_{ERBPP} and LB_{ERBPPC} produced consistent results across all data sets. Compared to the classical lower bound techniques: LB_{Clique}, $LB_{Capacity}$ and LB_{BP}, they give better bounds for 23 instances.

When Energetic Reasoning is combined to BPP (LB_{ERBPP}) and BPPC (LB_{ERBPPC}), the results outperform the bounds produced by LB_{ER} in 3 instances. This is due to the fact that the incompatibilities are considered at each examined time interval. These results confirm that the association of ER and BPPC is very efficient for VRPTW. We believe that ERBPPC will clearly outperform ER on highly constrained data set with more incompatible pairs. To conclude, the overall performance of the new lower bounding procedures has been encouraging. The use of Energetic Reasoning improves many lower bounds and gives good results for both capacity constrained problems and time constrained problems.

6 Conclusion

In this paper, we introduced several combinatorial optimization methods which can be used to get lower bounds for the Vehicle Routing Problem with Time Windows (VRPTW). Investigating the concept of Energetic Reasoning, we were able to propose new lower bounding techniques based on the transformation of m-VRPTW instance to PMSP. The numerical results confirm the contribution brought by the new proposed techniques. With a very fast computing time, we were able to provide the exact number or a reasonable approximation of the minimum number of vehicles required to visit all the customers. This suggests that our lower bounds techniques can quickly produce a good estimation of the fleet size. A challenging area for future research is to develop an exact method using the proposed lower bound procedures.

Acknowledgments

This work is partially supported by the Regional Council of Picardie and the European Regional Development Fund (ERDF), under PRIMA project. It is also partially supported by the National Agency for Research (project ATHENA, Reference ANR-13-BS02-0006-01).

Data Set	n	*Classical*						*New*						*BestLB*	*BestUB*	*AvgGAP*
		Clique		*Capacity*		*BP*		*ER*		*ERBPP*		*ERBPPC*				
		LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU			
C1	25	1.89	0	3	0	2	0	3	0	3	0	3	0	3	3	0
C2	25	1	0	1	0	1	0	1.13	0	1.13	0	1.13	0	1.13	1.13	0
R1	25	3.58	0	2	0	3.25	0	3.92	0	3.92	0.01	4	0.26	4	4.75	0.75
R2	25	1	0	1	0	1	0	1.09	0	1.09	0	1.09	0.04	1.09	1.27	0.18
RC1	25	2.75	0	3	0	2.25	0	3.13	0	3.13	0	3.13	0.03	3.13	3.25	0.13
RC2	25	1	0	1	0	1	0	1	0	1	0.01	1	0.08	1	1.5	0.5
C1	50	3	0	5	0	4	0	5	0	5	0	5	0	5	5	0
C2	50	1	0	2	0	2	0	2	0	2	0	2	0	2	2	0
R1	50	5.25	0	4	0	5.17	0	6.33	0.02	6.33	0.09	6.42	76.41	6.42	7.42	1
R2	50	1	0	1	0	1.18	0	1.27	0.01	1.27	0.06	1.27	0.23	1.27	2	0.73
RC1	50	4.25	0	5	0	4.13	0	5.25	0.01	5.25	0.05	5.38	89.88	5.38	6.5	1.13
RC2	50	1.13	0	1	0	1	0	1.13	0.01	1.13	0.05	1.13	0.28	1.13	2	0.88
C1	100	4.89	0	10	0	8	0	10	0	10	0	10	0	10	10	0
C2	100	1.38	0	3	0	3	0	3	0	3	0	3	0	3	3	0
R1	100	7.92	0	8	0	8.33	0	10.42	0.11	10.42	0.45	10.42	364	10.42	11.92	1.5
R2	100	1.18	0	2	0	2	0	2.09	0.07	2.09	0.31	2.09	27.2	2.09	2.73	0.64
RC1	100	6.38	0	9	0	7.63	0	9.63	0.1	9.63	0.45	9.63	99.52	9.63	11.5	1.88
RC2	100	1.13	0	2	0	2	0	2.13	0.11	2.13	0.44	2.13	33.16	2.13	3.25	1.13

Table 1. Average lower bound results and CPU times for Solomon instances

Data Set	n	*Classical*						*New*						*BestLB*	*BestUB*	*AvgGAP*
		Clique		*Capacity*		*BP*		*ER*		*ERBPP*		*ERBPPC*				
		LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU			
C1	200	8.8	0	18	0	15	0	18.3	0.41	18.3	1.82	18.3	429.41	18.3	18.9	0.6
C2	200	2.1	0	6	0	6	0	6	0	6	0	6	0	6	6	0
R1	200	10.4	0	18	0	7.3	0	18.2	0	18.2	0	18.2	0	18.2	18.2	0
R2	200	1.6	0	4	0	2	0	4	0	4	0	4	0	4	4	0
RC1	200	6.8	0	18	0	6.5	0	18	0	18	0	18	0	18	18	0
RC2	200	1.9	0	4	0	2	0	4	0.15	4	0.75	4	140.56	4	4.3	0.3
C1	400	16.9	0.01	36	0.01	26.5	0.01	36.5	2.8	36.5	12.47	36.5	2880.12	36.5	37.6	1.1
C2	400	2.5	0.01	11	0.01	11	0.01	11.2	2.6	11.2	12.31	11.2	2880.09	11.2	11.6	0.4
R1	400	17.9	0.01	36	0.01	11.7	0.01	36.4	0	36.4	0	36.4	0	36.4	36.4	0
R2	400	2.4	0.01	8	0.01	2.7	0.01	8	0	8	0	8	0	8	8	0
RC1	400	12.8	0.01	36	0.01	10.8	0.01	36	0	36	0	36	0	36	36	0
RC2	400	3.1	0.01	8	0.01	2.8	0.01	8	1.11	8	5.55	8	1440.04	8	8.4	0.4
C1	600	24.2	0.03	56	0.02	40.4	0.02	56.4	5.35	56.4	31.25	56.4	2160.32	56.4	57.2	0.8
C2	600	4.3	0.02	17	0.01	15.9	0.02	17.1	5.92	17.1	28.72	17.1	2160.31	17.1	17.4	0.3
R1	600	28.1	0.03	54	0.01	13.9	0.02	54.5	0	54.5	0	54.5	0	54.5	54.5	0
R2	600	4.3	0.02	11	0.01	3.4	0.01	11	0	11	0	11	0	11	11	0
RC1	600	19.7	0.04	55	0.02	12.2	0.02	55	0	55	0	55	0	55	55	0
RC2	600	4.7	0.02	11	0.01	2.9	0.02	11	3.63	11	18.38	11	1440.16	11	11.4	0.4
C1	800	34.4	0.08	72	0.05	49.3	0.05	72.8	18.36	72.8	97.45	72.8	4322.5	72.8	75	2.2
C2	800	5.7	0.05	22	0.04	21	0.04	22.2	48.55	22.2	228.25	22.2	9722.18	22.2	23.3	1.1
R1	800	35.3	0.06	72	0.05	15.8	0.05	72.8	0	72.8	0	72.8	0	72.8	72.8	0
R2	800	5.3	0.05	15	0.04	3.5	0.04	15	0	15	0	15	0	15	15	0
RC1	800	27.5	0.41	72	0.05	14.9	0.04	72	0	72	0	72	0	72	72	0
RC2	800	6.6	0.05	15	0.04	3.5	0.04	15	8.96	15	45.7	15	2160.52	15	15.4	0.4
C1	1000	44.6	0.34	90	0.09	58	0.09	91	35.22	91	185.64	91	4324.8	91	93.9	2.9
C2	1000	7.9	0.11	28	0.09	25.1	0.07	28.1	67.81	28.1	284.21	28.1	6484.59	28.1	28.8	0.7
R1	1000	44.5	0.13	91	0.09	19.5	0.08	91.9	0	91.9	0	91.9	0	91.9	91.9	0
R2	1000	6.5	0.09	19	0.08	4	0.07	19	0	19	0	19	0	19	19	0
RC1	1000	31.5	0.79	90	0.08	17.7	0.08	90	0	90	0	90	0	90	90	0
RC2	1000	7.8	0.1	18	0.08	4.1	0.07	18	8.21	18	40.32	18	1080.57	18	18.2	0.2

Table 2. Average lower bound results and CPU times for Gehring and Homberger instances

Bibliography

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

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

[3] Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

[4] Baptiste, P., Le Pape, C., and Nuijten, W. (1999). Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92:305–333.

[5] Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

[6] Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139.

[7] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). The vehicle routing problem. chapter VRP with Time Windows, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[8] Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404.

[9] Erschler, J., Lopez, P., and Thuriot, C. (1991). Raisonnement temporel sous contraintes de ressources et problèmes d'ordonnancement. *Revue d'Intelligence Artificielle*, 5(3):7–36.

[10] Fisher, M. L., Jörnsten, K. O., and Madsen, O. B. (1997). Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):488–492.

[11] Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, volume 2, pages 57–64.

[12] Gendreau, M., Laporte, G., and Semet, F. (2004). Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31(3):347–358.

[13] Golden, B. L., Assad, A. A., and Wasil, E. A. (2002). Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia*, pages 245–286.

[14] Haouari, M. and Gharbi, A. (2005). Fast lifting procedures for the bin packing problem. *Discrete Optimization*, 2(3):201–218.

[15] Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

[16] Jung, S. and Moon, B. R. (2002). A hybrid genetic algorithm for the vehicle routing problem with time windows. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, pages 1309–1316. Morgan Kaufmann.

[17] Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307 – 2330.

[18] Konc, J. and Janezic, D. (2007). An improved branch and bound algorithm for the maximum clique problem. *proteins*, 58:569–590.

[19] Kontoravdis, G. and Bard, J. F. (1995). A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, 7(1):10–23.

[20] Labadi, N., Prins, C., and Reghioui, M. (2008). A memetic algorithm for the vehicle routing problem with time windows. *Rairo-operations Research*, 42:415–431.

[21] Lenstra, J. K. and Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

[22] Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989.

[23] Néron, E., Baptiste, P., and Gupta, J. N. (2001). Solving hybrid flow shop problem using energetic reasoning and global operations. *Omega*, 29(6):501–511.

[24] Ombuki, B., Ross, B. J., and Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24:17–30.

[25] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[26] Tan, K. C., Chew, Y., and Lee, L. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34(1):115–151.

[27] Ursani, Z., Essam, D., Cornforth, D., and Stocker, R. (2011). Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11(8):5375–5390.

[28] Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489.

B | Detailed results for Chapter 2

Table B.1 – Results on Solomon and Desrosiers [82] instances with 25 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
c101	3	3	2	3	3	3	3	3	3
c102	2	3	2	3	3	3	3	3	3
c103	2	3	2	3	3	3	3	3	3
c104	2	3	2	3	3	3	3	3	3
c105	2	3	2	3	3	3	3	3	3
c106	3	3	2	3	3	3	3	3	3
c107	1	3	2	3	3	3	3	3	3
c108	1	3	2	3	3	3	3	3	3
c109	1	3	2	3	3	3	3	3	3
c201	1	1	1	2	2	2	2	2	2
c202	1	1	1	1	1	1	1	1	1
c203	1	1	1	1	1	1	1	1	1
c204	1	1	1	1	1	1	1	1	1
c205	1	1	1	1	1	1	1	1	1
c206	1	1	1	1	1	1	1	1	1
c207	1	1	1	1	1	1	1	1	1
c208	1	1	1	1	1	1	1	1	1
r101	8	2	5	8	8	8	8	8	8
r102	6	2	3	6	6	7	7	7	7
r103	4	2	3	4	4	4	4	4	4
r104	4	2	3	4	4	4	4	4	4
r105	4	2	4	4	4	4	5	5	5
r106	3	2	3	3	3	3	4	4	4
r107	3	2	3	3	3	3	4	4	4
r108	3	2	3	3	3	3	3	3	4
r109	3	2	3	3	3	3	4	4	4
r110	1	2	3	3	3	3	4	4	4
r111	3	2	3	3	3	3	4	4	4
r112	1	2	3	3	3	3	3	3	4
r201	1	1	1	2	2	2	2	2	2
r202	1	1	1	1	1	1	2	2	2
r203	1	1	1	1	1	1	2	2	2
r204	1	1	1	1	1	1	1	1	1
r205	1	1	1	1	1	1	1	1	1
r206	1	1	1	1	1	1	1	1	1
r207	1	1	1	1	1	1	1	1	1
r208	1	1	1	1	1	1	1	1	1
r209	1	1	1	1	1	1	1	1	1
r210	1	1	1	1	1	1	1	1	1
r211	1	1	1	1	1	1	1	1	1
rc101	3	3	3	3	3	3	4	4	4
rc102	3	3	2	3	3	3	3	3	3
rc103	3	3	2	3	3	3	3	3	3
rc104	3	3	2	3	3	3	3	3	3
rc105	4	3	3	4	4	4	4	4	4
rc106	2	3	2	3	3	3	3	3	3
rc107	2	3	2	3	3	3	3	3	3
rc108	2	3	2	3	3	3	3	3	3
rc201	1	1	1	1	1	1	2	2	2
rc202	1	1	1	1	1	1	1	1	1
rc203	1	1	1	1	1	1	1	1	1
rc204	1	1	1	1	1	1	1	1	1
rc205	1	1	1	1	1	1	2	2	2
rc206	1	1	1	1	1	1	1	1	1
rc207	1	1	1	1	1	1	1	1	1
rc208	1	1	1	1	1	1	1	1	1

Table B.2 – Results on Solomon and Desrosiers [82] instances with 50 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
c101	5	5	4	5	5	5	5	5	5
c102	5	5	4	5	5	5	5	5	5
c103	4	5	4	5	5	5	5	5	5
c104	2	5	4	5	5	5	5	5	5
c105	3	5	4	5	5	5	5	5	5
c106	5	5	4	5	5	5	5	5	5
c107	1	5	4	5	5	5	5	5	5
c108	1	5	4	5	5	5	5	5	5
c109	1	5	4	5	5	5	5	5	5
c201	1	2	2	2	2	2	2	2	2
c202	1	2	2	2	2	2	2	2	2
c203	1	2	2	2	2	2	2	2	2
c204	1	2	2	2	2	2	2	2	2
c205	1	2	2	2	2	2	2	2	2
c206	1	2	2	2	2	2	2	2	2
c207	1	2	2	2	2	2	2	2	2
c208	1	2	2	2	2	2	2	2	2
r101	11	4	8	11	11	11	11	11	11
r102	9	4	5	9	9	10	10	10	10
r103	8	4	5	8	8	8	8	8	8
r104	5	4	5	5	5	5	5	5	6
r105	5	4	6	7	7	7	8	8	8
r106	5	4	5	6	6	6	7	7	7
r107	5	4	5	5	5	5	6	6	6
r108	4	4	4	5	5	5	5	5	6
r109	3	4	5	5	5	5	5	5	7
r110	3	4	5	5	5	5	5	5	7
r111	4	4	5	5	5	5	5	5	7
r112	1	4	4	5	5	5	5	5	6
r201	1	1	2	2	2	2	2	2	2
r202	1	1	1	2	2	2	2	2	2
r203	1	1	1	1	1	1	2	2	2
r204	1	1	1	1	1	1	2	2	2
r205	1	1	2	2	2	2	2	2	2
r206	1	1	1	1	1	1	1	1	2
r207	1	1	1	1	1	1	1	1	2
r208	1	1	1	1	1	1	1	1	2
r209	1	1	1	1	1	1	1	1	2
r210	1	1	1	1	1	1	2	2	2
r211	1	1	1	1	1	1	1	1	2
rc101	6	5	5	6	6	6	8	8	8
rc102	5	5	4	5	5	5	7	7	7
rc103	5	5	4	5	5	5	6	6	6
rc104	3	5	4	5	5	5	5	5	5
rc105	6	5	4	6	6	6	8	8	8
rc106	4	5	4	5	5	5	6	6	6
rc107	3	5	4	5	5	5	5	5	6
rc108	2	5	4	5	5	5	5	5	6
rc201	1	1	1	1	1	1	2	2	2
rc202	1	1	1	1	1	1	2	2	2
rc203	1	1	1	1	1	1	2	2	2
rc204	1	1	1	1	1	1	2	2	2
rc205	2	1	1	2	2	2	2	2	2
rc206	1	1	1	1	1	1	2	2	2
rc207	1	1	1	1	1	1	2	2	2
rc208	1	1	1	1	1	1	1	1	2

Table B.3 – Results on Solomon and Desrosiers [82] instances with 100 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
c101	10	10	8	10	10	10	10	10	10
c102	9	10	8	10	10	10	10	10	10
c103	7	10	8	10	10	10	10	10	10
c104	4	10	8	10	10	10	10	10	10
c105	5	10	8	10	10	10	10	10	10
c106	6	10	8	10	10	10	10	10	10
c107	1	10	8	10	10	10	10	10	10
c108	1	10	8	10	10	10	10	10	10
c109	1	10	8	10	10	10	10	10	10
c201	2	3	3	3	3	3	3	3	3
c202	2	3	3	3	3	3	3	3	3
c203	2	3	3	3	3	3	3	3	3
c204	1	3	3	3	3	3	3	3	3
c205	1	3	3	3	3	3	3	3	3
c206	1	3	3	3	3	3	3	3	3
c207	1	3	3	3	3	3	3	3	3
c208	1	3	3	3	3	3	3	3	3
r101	18	8	13	18	18	18	19	19	19
r102	17	8	9	17	17	17	17	17	17
r103	13	8	8	13	13	13	13	13	13
r104	8	8	7	8	8	8	8	8	9
r105	7	8	9	11	11	11	11	11	14
r106	7	8	8	9	9	9	11	11	12
r107	6	8	8	8	8	8	9	9	10
r108	5	8	7	8	8	8	8	8	9
r109	4	8	8	9	9	9	9	9	11
r110	3	8	8	8	8	8	8	8	10
r111	6	8	8	8	8	8	8	8	10
r112	1	8	7	8	8	8	8	8	9
r201	2	2	2	3	3	3	3	3	4
r202	2	2	2	2	2	2	3	3	3
r203	1	2	2	2	2	2	3	3	3
r204	1	2	2	2	2	2	2	2	2
r205	1	2	2	2	2	2	2	2	3
r206	1	2	2	2	2	2	2	2	3
r207	1	2	2	2	2	2	2	2	2
r208	1	2	2	2	2	2	2	2	2
r209	1	2	2	2	2	2	2	2	3
r210	1	2	2	2	2	2	2	2	3
r211	1	2	2	2	2	2	2	2	2
rc101	8	9	9	11	11	11	13	13	14
rc102	7	9	8	9	9	9	12	12	12
rc103	7	9	7	9	9	9	9	9	11
rc104	6	9	7	9	9	9	9	9	10
rc105	12	9	8	12	12	12	13	13	13
rc106	4	9	8	9	9	9	9	9	11
rc107	4	9	7	9	9	9	9	9	11
rc108	3	9	7	9	9	9	9	9	10
rc201	1	2	2	3	3	3	3	3	4
rc202	1	2	2	2	2	2	3	3	3
rc203	1	2	2	2	2	2	2	2	3
rc204	1	2	2	2	2	2	2	2	3
rc205	2	2	2	2	2	2	4	4	4
rc206	1	2	2	2	2	2	2	2	3
rc207	1	2	2	2	2	2	2	2	3
rc208	1	2	2	2	2	2	2	2	3

Table B.4 – Results on Homberger and Gehring [46] instances with 200 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
C1_2_1	20	18	15	20	20	20	20	20	20
C1_2_2	17	18	15	18	18	18	18	18	18
C1_2_3	13	18	15	18	18	18	18	18	18
C1_2_4	9	18	15	18	18	18	18	18	18
C1_2_5	10	18	15	19	19	19	20	20	20
C1_2_6	10	18	15	18	18	18	18	18	20
C1_2_7	4	18	15	18	18	18	18	18	20
C1_2_8	3	18	15	18	18	18	18	18	19
C1_2_9	1	18	15	18	18	18	18	18	18
C1_2_10	1	18	15	18	18	18	18	18	18
C2_2_1	3	6	6	6	6	6	6	6	6
C2_2_2	3	6	6	6	6	6	6	6	6
C2_2_3	3	6	6	6	6	6	6	6	6
C2_2_4	3	6	6	6	6	6	6	6	6
C2_2_5	1	6	6	6	6	6	6	6	6
C2_2_6	1	6	6	6	6	6	6	6	6
C2_2_7	3	6	6	6	6	6	6	6	6
C2_2_8	1	6	6	6	6	6	6	6	6
C2_2_9	2	6	6	6	6	6	6	6	6
C2_2_10	1	6	6	6	6	6	6	6	6
R1_2_1	20	18	11	20	20	20	20	20	20
R1_2_2	16	18	7	18	18	18	18	18	18
R1_2_3	14	18	6	18	18	18	18	18	18
R1_2_4	8	18	6	18	18	18	18	18	18
R1_2_5	11	18	9	18	18	18	18	18	18
R1_2_6	10	18	7	18	18	18	18	18	18
R1_2_7	9	18	6	18	18	18	18	18	18
R1_2_8	6	18	6	18	18	18	18	18	18
R1_2_9	6	18	8	18	18	18	18	18	18
R1_2_10	4	18	7	18	18	18	18	18	18
R2_2_1	3	4	2	4	4	4	4	4	4
R2_2_2	2	4	2	4	4	4	4	4	4
R2_2_3	2	4	2	4	4	4	4	4	4
R2_2_4	2	4	2	4	4	4	4	4	4
R2_2_5	1	4	2	4	4	4	4	4	4
R2_2_6	1	4	2	4	4	4	4	4	4
R2_2_7	1	4	2	4	4	4	4	4	4
R2_2_8	1	4	2	4	4	4	4	4	4
R2_2_9	2	4	2	4	4	4	4	4	4
R2_2_10	1	4	2	4	4	4	4	4	4
RC1_2_1	13	18	8	18	18	18	18	18	18
RC1_2_2	11	18	6	18	18	18	18	18	18
RC1_2_3	8	18	6	18	18	18	18	18	18
RC1_2_4	7	18	6	18	18	18	18	18	18
RC1_2_5	8	18	7	18	18	18	18	18	18
RC1_2_6	6	18	7	18	18	18	18	18	18
RC1_2_7	6	18	7	18	18	18	18	18	18
RC1_2_8	4	18	6	18	18	18	18	18	18
RC1_2_9	3	18	6	18	18	18	18	18	18
RC1_2_10	2	18	6	18	18	18	18	18	18
RC2_2_1	3	4	2	4	4	4	4	4	6
RC2_2_2	2	4	2	4	4	4	4	4	5
RC2_2_3	2	4	2	4	4	4	4	4	4
RC2_2_4	2	4	2	4	4	4	4	4	4
RC2_2_5	4	4	2	4	4	4	4	4	4
RC2_2_6	1	4	2	4	4	4	4	4	4
RC2_2_7	2	4	2	4	4	4	4	4	4
RC2_2_8	1	4	2	4	4	4	4	4	4
RC2_2_9	1	4	2	4	4	4	4	4	4
RC2_2_10	1	4	2	4	4	4	4	4	4

Table B.5 – Results on Homberger and Gehring [46] instances with 400 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
C1_4_1	40	36	28	40	40	40	40	40	40
C1_4_2	35	36	26	36	36	36	36	36	36
C1_4_3	25	36	26	36	36	36	36	36	36
C1_4_4	15	36	26	36	36	36	36	36	36
C1_4_5	21	36	27	37	37	37	39	39	40
C1_4_6	21	36	27	36	36	36	36	36	40
C1_4_7	5	36	27	36	36	36	36	36	39
C1_4_8	4	36	26	36	36	36	36	36	37
C1_4_9	1	36	26	36	36	36	36	36	36
C1_4_10	2	36	26	36	36	36	36	36	36
C2_4_1	5	11	11	12	12	12	12	12	12
C2_4_2	4	11	11	11	11	11	11	11	12
C2_4_3	4	11	11	11	11	11	11	11	11
C2_4_4	3	11	11	11	11	11	11	11	11
C2_4_5	1	11	11	12	12	12	12	12	12
C2_4_6	1	11	11	11	11	11	11	11	12
C2_4_7	3	11	11	11	11	11	11	11	12
C2_4_8	1	11	11	11	11	11	11	11	11
C2_4_9	2	11	11	11	11	11	11	11	12
C2_4_10	1	11	11	11	11	11	11	11	11
R1_4_1	40	36	19	40	40	40	40	40	40
R1_4_2	28	36	11	36	36	36	36	36	36
R1_4_3	21	36	10	36	36	36	36	36	36
R1_4_4	14	36	9	36	36	36	36	36	36
R1_4_5	18	36	15	36	36	36	36	36	36
R1_4_6	16	36	10	36	36	36	36	36	36
R1_4_7	14	36	10	36	36	36	36	36	36
R1_4_8	12	36	9	36	36	36	36	36	36
R1_4_9	11	36	13	36	36	36	36	36	36
R1_4_10	5	36	11	36	36	36	36	36	36
R2_4_1	5	8	4	8	8	8	8	8	8
R2_4_2	4	8	3	8	8	8	8	8	8
R2_4_3	4	8	2	8	8	8	8	8	8
R2_4_4	3	8	2	8	8	8	8	8	8
R2_4_5	1	8	3	8	8	8	8	8	8
R2_4_6	1	8	3	8	8	8	8	8	8
R2_4_7	1	8	2	8	8	8	8	8	8
R2_4_8	1	8	2	8	8	8	8	8	8
R2_4_9	3	8	3	8	8	8	8	8	8
R2_4_10	1	8	3	8	8	8	8	8	8
RC1_4_1	22	36	13	36	36	36	36	36	36
RC1_4_2	22	36	10	36	36	36	36	36	36
RC1_4_3	17	36	10	36	36	36	36	36	36
RC1_4_4	15	36	9	36	36	36	36	36	36
RC1_4_5	17	36	12	36	36	36	36	36	36
RC1_4_6	10	36	12	36	36	36	36	36	36
RC1_4_7	11	36	11	36	36	36	36	36	36
RC1_4_8	7	36	11	36	36	36	36	36	36
RC1_4_9	4	36	10	36	36	36	36	36	36
RC1_4_10	3	36	10	36	36	36	36	36	36
RC2_4_1	4	8	3	8	8	8	8	8	11
RC2_4_2	4	8	3	8	8	8	8	8	9
RC2_4_3	4	8	2	8	8	8	8	8	8
RC2_4_4	3	8	2	8	8	8	8	8	8
RC2_4_5	7	8	3	8	8	8	8	8	8
RC2_4_6	1	8	3	8	8	8	8	8	8
RC2_4_7	4	8	3	8	8	8	8	8	8
RC2_4_8	2	8	3	8	8	8	8	8	8
RC2_4_9	1	8	3	8	8	8	8	8	8
RC2_4_10	1	8	3	8	8	8	8	8	8

Table B.6 – Results on Homberger and Gehring [46] instances with 600 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
C1_6_1	60	56	43	60	60	60	60	60	60
C1_6_2	51	56	40	56	56	56	56	56	56
C1_6_3	38	56	39	56	56	56	56	56	56
C1_6_4	22	56	39	56	56	56	56	56	56
C1_6_5	27	56	42	56	56	56	56	56	60
C1_6_6	26	56	41	56	56	56	56	56	59
C1_6_7	8	56	41	56	56	56	56	56	57
C1_6_8	6	56	40	56	56	56	56	56	56
C1_6_9	2	56	40	56	56	56	56	56	56
C1_6_10	2	56	39	56	56	56	56	56	56
C2_6_1	8	17	16	18	18	18	18	18	18
C2_6_2	7	17	16	17	17	17	17	17	17
C2_6_3	7	17	16	17	17	17	17	17	17
C2_6_4	5	17	15	17	17	17	17	17	17
C2_6_5	3	17	16	17	17	17	17	17	18
C2_6_6	2	17	16	17	17	17	17	17	18
C2_6_7	4	17	16	17	17	17	17	17	18
C2_6_8	1	17	16	17	17	17	17	17	17
C2_6_9	5	17	16	17	17	17	17	17	17
C2_6_10	1	17	16	17	17	17	17	17	17
R1_6_1	59	54	25	59	59	59	59	59	59
R1_6_2	45	54	12	54	54	54	54	54	54
R1_6_3	37	54	11	54	54	54	54	54	54
R1_6_4	28	54	10	54	54	54	54	54	54
R1_6_5	26	54	19	54	54	54	54	54	54
R1_6_6	24	54	12	54	54	54	54	54	54
R1_6_7	21	54	10	54	54	54	54	54	54
R1_6_8	15	54	10	54	54	54	54	54	54
R1_6_9	17	54	16	54	54	54	54	54	54
R1_6_10	9	54	14	54	54	54	54	54	54
R2_6_1	7	11	5	11	11	11	11	11	11
R2_6_2	7	11	3	11	11	11	11	11	11
R2_6_3	6	11	3	11	11	11	11	11	11
R2_6_4	5	11	3	11	11	11	11	11	11
R2_6_5	4	11	4	11	11	11	11	11	11
R2_6_6	3	11	3	11	11	11	11	11	11
R2_6_7	3	11	3	11	11	11	11	11	11
R2_6_8	2	11	3	11	11	11	11	11	11
R2_6_9	5	11	4	11	11	11	11	11	11
R2_6_10	1	11	3	11	11	11	11	11	11
RC1_6_1	37	55	16	55	55	55	55	55	55
RC1_6_2	32	55	11	55	55	55	55	55	55
RC1_6_3	28	55	10	55	55	55	55	55	55
RC1_6_4	20	55	9	55	55	55	55	55	55
RC1_6_5	23	55	14	55	55	55	55	55	55
RC1_6_6	17	55	14	55	55	55	55	55	55
RC1_6_7	17	55	13	55	55	55	55	55	55
RC1_6_8	11	55	12	55	55	55	55	55	55
RC1_6_9	7	55	12	55	55	55	55	55	55
RC1_6_10	5	55	11	55	55	55	55	55	55
RC2_6_1	7	11	4	11	11	11	11	11	14
RC2_6_2	7	11	3	11	11	11	11	11	12
RC2_6_3	5	11	2	11	11	11	11	11	11
RC2_6_4	4	11	2	11	11	11	11	11	11
RC2_6_5	10	11	3	11	11	11	11	11	11
RC2_6_6	3	11	3	11	11	11	11	11	11
RC2_6_7	6	11	3	11	11	11	11	11	11
RC2_6_8	3	11	3	11	11	11	11	11	11
RC2_6_9	1	11	3	11	11	11	11	11	11
RC2_6_10	1	11	3	11	11	11	11	11	11

Table B.7 – Results on Homberger and Gehring [46] instances with 800 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
C1_8_1	80	72	54	80	80	80	80	80	80
C1_8_2	72	72	49	72	72	72	72	72	72
C1_8_3	54	72	48	72	72	72	72	72	72
C1_8_4	33	72	47	72	72	72	72	72	72
C1_8_5	36	72	51	72	72	72	72	72	80
C1_8_6	39	72	50	72	72	72	72	72	79
C1_8_7	14	72	50	72	72	72	72	72	77
C1_8_8	9	72	48	72	72	72	72	72	74
C1_8_9	4	72	48	72	72	72	72	72	72
C1_8_10	3	72	48	72	72	72	72	72	72
C2_8_1	12	22	22	24	24	24	24	24	24
C2_8_2	11	22	21	22	22	22	22	22	23
C2_8_3	9	22	21	22	22	22	22	22	23
C2_8_4	7	22	20	22	22	22	22	22	23
C2_8_5	3	22	21	22	22	22	22	22	24
C2_8_6	2	22	21	22	22	22	22	22	23
C2_8_7	7	22	21	22	22	22	22	22	24
C2_8_8	1	22	21	22	22	22	22	22	23
C2_8_9	4	22	21	22	22	22	22	22	23
C2_8_10	1	22	21	22	22	22	22	22	23
R1_8_1	80	72	29	80	80	80	80	80	80
R1_8_2	59	72	14	72	72	72	72	72	72
R1_8_3	42	72	11	72	72	72	72	72	72
R1_8_4	24	72	11	72	72	72	72	72	72
R1_8_5	36	72	23	72	72	72	72	72	72
R1_8_6	32	72	13	72	72	72	72	72	72
R1_8_7	26	72	11	72	72	72	72	72	72
R1_8_8	19	72	11	72	72	72	72	72	72
R1_8_9	22	72	19	72	72	72	72	72	72
R1_8_10	13	72	16	72	72	72	72	72	72
R2_8_1	10	15	5	15	15	15	15	15	15
R2_8_2	8	15	3	15	15	15	15	15	15
R2_8_3	7	15	3	15	15	15	15	15	15
R2_8_4	5	15	3	15	15	15	15	15	15
R2_8_5	4	15	4	15	15	15	15	15	15
R2_8_6	4	15	3	15	15	15	15	15	15
R2_8_7	4	15	3	15	15	15	15	15	15
R2_8_8	4	15	3	15	15	15	15	15	15
R2_8_9	5	15	4	15	15	15	15	15	15
R2_8_10	2	15	4	15	15	15	15	15	15
RC1_8_1	48	72	19	72	72	72	72	72	72
RC1_8_2	44	72	13	72	72	72	72	72	72
RC1_8_3	41	72	12	72	72	72	72	72	72
RC1_8_4	29	72	11	72	72	72	72	72	72
RC1_8_5	35	72	17	72	72	72	72	72	72
RC1_8_6	25	72	17	72	72	72	72	72	72
RC1_8_7	21	72	16	72	72	72	72	72	72
RC1_8_8	15	72	15	72	72	72	72	72	72
RC1_8_9	9	72	15	72	72	72	72	72	72
RC1_8_10	8	72	14	72	72	72	72	72	72
RC2_8_1	9	15	5	15	15	15	15	15	18
RC2_8_2	9	15	3	15	15	15	15	15	16
RC2_8_3	9	15	3	15	15	15	15	15	15
RC2_8_4	8	15	3	15	15	15	15	15	15
RC2_8_5	10	15	4	15	15	15	15	15	15
RC2_8_6	4	15	4	15	15	15	15	15	15
RC2_8_7	8	15	4	15	15	15	15	15	15
RC2_8_8	6	15	3	15	15	15	15	15	15
RC2_8_9	2	15	3	15	15	15	15	15	15
RC2_8_10	1	15	3	15	15	15	15	15	15

Table B.8 – Results on Homberger and Gehring [46] instances with 1000 customers.

Data Set	*Classical*			*New*				*BestLB*	*BestUB*
	Clique	*Capacity*	*BP*	*ER*	*ERBPP*	*ERBPPC*	*SubSets*		
C1_10_1	100	90	64	100	100	100	100	100	100
C1_10_2	88	90	57	90	90	90	90	90	90
C1_10_3	72	90	55	90	90	90	90	90	90
C1_10_4	45	90	54	90	90	90	90	90	90
C1_10_5	49	90	61	90	90	90	90	90	100
C1_10_6	49	90	59	90	90	90	90	90	100
C1_10_7	19	90	60	90	90	90	90	90	98
C1_10_8	14	90	57	90	90	90	90	90	93
C1_10_9	5	90	57	90	90	90	90	90	90
C1_10_10	5	90	56	90	90	90	90	90	90
C2_10_1	16	28	26	29	29	29	29	29	30
C2_10_2	15	28	25	28	28	28	28	28	29
C2_10_3	12	28	25	28	28	28	28	28	28
C2_10_4	10	28	25	28	28	28	28	28	28
C2_10_5	4	28	25	28	28	28	28	28	30
C2_10_6	4	28	25	28	28	28	28	28	29
C2_10_7	9	28	25	28	28	28	28	28	29
C2_10_8	2	28	25	28	28	28	28	28	28
C2_10_9	6	28	25	28	28	28	28	28	29
C2_10_10	1	28	25	28	28	28	28	28	28
R1_10_1	100	91	37	100	100	100	100	100	100
R1_10_2	78	91	16	91	91	91	91	91	91
R1_10_3	54	91	14	91	91	91	91	91	91
R1_10_4	27	91	13	91	91	91	91	91	91
R1_10_5	45	91	29	91	91	91	91	91	91
R1_10_6	40	91	16	91	91	91	91	91	91
R1_10_7	32	91	13	91	91	91	91	91	91
R1_10_8	22	91	12	91	91	91	91	91	91
R1_10_9	30	91	25	91	91	91	91	91	91
R1_10_10	17	91	20	91	91	91	91	91	91
R2_10_1	12	19	7	19	19	19	19	19	19
R2_10_2	11	19	4	19	19	19	19	19	19
R2_10_3	8	19	3	19	19	19	19	19	19
R2_10_4	7	19	3	19	19	19	19	19	19
R2_10_5	5	19	5	19	19	19	19	19	19
R2_10_6	5	19	3	19	19	19	19	19	19
R2_10_7	4	19	3	19	19	19	19	19	19
R2_10_8	4	19	3	19	19	19	19	19	19
R2_10_9	6	19	5	19	19	19	19	19	19
R2_10_10	3	19	4	19	19	19	19	19	19
RC1_10_1	56	90	24	90	90	90	90	90	90
RC1_10_2	51	90	15	90	90	90	90	90	90
RC1_10_3	44	90	13	90	90	90	90	90	90
RC1_10_4	30	90	12	90	90	90	90	90	90
RC1_10_5	39	90	21	90	90	90	90	90	90
RC1_10_6	29	90	21	90	90	90	90	90	90
RC1_10_7	27	90	19	90	90	90	90	90	90
RC1_10_8	16	90	18	90	90	90	90	90	90
RC1_10_9	12	90	17	90	90	90	90	90	90
RC1_10_10	11	90	17	90	90	90	90	90	90
RC2_10_1	11	18	6	18	18	18	18	18	20
RC2_10_2	10	18	3	18	18	18	18	18	18
RC2_10_3	10	18	3	18	18	18	18	18	18
RC2_10_4	9	18	3	18	18	18	18	18	18
RC2_10_5	12	18	5	18	18	18	18	18	18
RC2_10_6	6	18	5	18	18	18	18	18	18
RC2_10_7	9	18	4	18	18	18	18	18	18
RC2_10_8	6	18	4	18	18	18	18	18	18
RC2_10_9	3	18	4	18	18	18	18	18	18
RC2_10_10	2	18	4	18	18	18	18	18	18

C | Detailed results for Chapter 3

This document complements the results of Chapter 3: Particle Swarm Optimization for VRPTW with detailed experimental results. Tables Tables C.1 to C.6 present the results of all the instances.

Table C.1 – Results on Solomon and Desrosiers [82] instances.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	10	828.94	3	591.56	19	1 650.8	4	1 252.37	14	1 696.95	4	1 406.94
2	10	828.94	3	591.56	17	1 486.12	3	1 191.7	12	1 554.75	3	1 365.64
3	10	828.06	3	591.17	13	1 292.67	3	939.5	11	1 261.67	3	1 050.63
4	10	824.78	3	590.6	9	1 011.04	2	831.69	10	1 135.52	3	798.46
5	10	828.94	3	588.88	14	1 377.11	3	994.43	13	1 629.44	4	1 297.65
6	10	828.94	3	588.49	12	1 252.03	3	906.14	11	1 424.73	3	1 146.32
7	10	828.94	3	588.29	10	1 104.66	2	893.33	11	1 230.48	3	1 061.14
8	10	828.94	3	588.32	9	960.88	2	726.82	10	1 139.82	3	828.14
9	10	828.94	-	-	11	1 197.42	3	909.16	-	-	-	-
10	-	-	-	-	10	1 118.84	3	939.37	-	-	-	-
11	-	-	-	-	10	1 096.73	2	898.15	-	-	-	-
12	-	-	-	-	9	1 005.2	-	-	-	-	-	-

Table C.2 – Results on Homberger and Gehring [46] instances with 200 customers.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	20	2 704.57	6	1 931.44	20	4 796.9	4	4 526.01	18	3 976.13	6	3 099.76
2	18	2 976.45	6	1 863.15	18	4 136.45	4	3 649.42	18	3 384.15	5	2 825.24
3	18	2 720.57	6	1 775.07	18	3 400.87	4	2 881.46	18	3 061.6	4	2 604.09
4	18	2 647.99	6	1 719.95	18	3 086.48	4	1 981.3	18	2 892.76	4	2 063.76
5	20	2 702.05	6	1 878.85	18	4 195.22	4	3 367.53	18	3 599.81	4	2 911.85
6	20	2 701.03	6	1 857.35	18	3 672.23	4	2 913.81	18	3 473.34	4	2 920.34
7	20	2 701.03	6	1 849.46	18	3 188.93	4	2 451.14	18	3 270.23	4	2 528.62
8	19	2 782.86	6	1 820.53	18	2 979.27	4	1 849.87	18	3 156.5	4	2 315.07
9	18	2 690.16	6	1 830.05	18	3 829.86	4	3 099.11	18	3 118.25	4	2 179.09
10	18	2 646.16	6	1 808.21	18	3 357.66	4	2 654.97	18	3 014.46	4	2 015.61

Table C.3 – Results on Homberger and Gehring [46] instances with 400 customers.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	40	7 152.05	12	4 116.14	40	10 649.57	8	9 289.07	38	8 769.72	12	6 626.29
2	36	8 519.32	12	3 930.25	36	9 980.62	8	7 677.95	36	8 767.09	10	6 090.93
3	36	7 796.66	12	3 817.18	36	9 032.63	8	6 029.85	36	8 258.38	8	5 289.17
4	36	7 368.74	12	3 587.34	36	8 020.74	8	4 317.15	36	8 084.44	8	3 673.72
5	40	7 152.05	12	3 945.39	36	10 404	8	7 205.62	36	9 111.87	10	6 015.41
6	40	7 153.45	12	3 875.94	36	9 205.4	8	6 164.4	37	8 411.75	9	5 767.26
7	40	7 149.43	12	3 894.16	36	8 245.67	8	5 125.19	36	8 699.65	8	5 799.93
8	38	7 944.13	12	3 816.55	36	7 923.45	8	4 048.53	36	8 503.98	8	4 893.48
9	37	7 240.95	12	3 892.79	36	9 387.42	8	6 460.02	36	8 414.57	8	4 626.89
10	37	6 910.03	12	3 720.77	36	8 696.52	8	5 895.6	36	8 153.33	8	4 341.4

Table C.4 – Results on Homberger and Gehring [46] instances with 600 customers.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	60	14 095.64	18	7 774.15	59	22 749.7	11	19 682.94	56	18 717.9	16	13 124.78
2	58	14 383.26	18	7 498.37	54	26 080.74	11	15 566.82	55	19 808.89	14	11 182.53
3	56	15 688.49	18	7 331.46	54	21 336.03	11	11 949.05	55	19 288.63	11	10 575.2
4	56	14 637.59	17	9 499.32	54	19 670.21	11	8 565	55	17 257.47	11	7 859.38
5	60	14 085.72	18	7 575.19	55	21 375.16	11	15 796.01	55	19 682.65	13	12 911.47
6	60	14 089.66	18	7 628.96	55	19 983.26	11	13 242.92	55	19 194.46	12	12 230.97
7	60	14 119.26	18	9 407.6	54	19 808.42	11	10 613.56	55	18 980.45	11	12 385.37
8	59	14 523.48	18	7 416.4	54	18 336.63	11	8 039.71	55	18 170.06	11	11 194.13
9	56	14 580.85	18	7 470.11	55	20 577	11	14 163.34	55	18 257.3	11	10 478.34
10	56	14 477.33	17	9 484.11	54	20 946.79	11	12 902.38	55	18 084.65	11	9 776.6

Table C.5 – Results on Homberger and Gehring [46] instances with 800 customers.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	80	25 184.38	24	11 664.8	80	39 556.46	15	29 728.36	73	36 608.37	20	20 667.59
2	78	25 953.1	24	12 073.52	72	42 323.02	15	24 070.32	73	34 824.45	18	18 663.19
3	73	28 237.11	24	12 499.92	72	38 426.39	15	18 802.58	73	33 094.63	16	15 696.17
4	72	27 337.89	24	12 317.44	72	32 491.89	15	14 527.74	73	30 993.09	15	12 224.84
5	80	25 166.28	25	11 813.42	72	43 049.12	15	25 933.76	73	34 020.69	17	18 904.83
6	80	25 161.06	25	12 000.69	72	39 134.71	15	21 814.6	73	33 717.51	16	18 824.58
7	79	25 814.91	25	12 060.74	72	34 773.54	15	17 730.51	73	33 467.35	15	18 190.43
8	78	25 743.51	25	12 463.99	72	32 567.45	15	13 711.44	73	32 894.3	15	17 217.29
9	76	26 997.04	25	12 560.57	72	40 381.26	15	23 627.11	73	33 534.83	15	16 372.67
10	74	27 513.18	24	15 842.59	72	37 478.7	15	21 861.52	73	32 032.23	15	15 527.86

Table C.6 – Results on Homberger and Gehring [46] instances with 1000 customers.

	C1		C2		R1		R2		RC1		RC2	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
1	100	42 478.95	30	16 891.48	100	57 826.73	19	44 953.16	91	56 083.16	22	33 001.28
2	97	46 365.66	30	16 906.65	92	59 937.6	19	36 785.6	90	57 426.8	21	27 187.2
3	91	47 145.85	31	18 485.7	92	53 863.29	19	27 780	90	50 028.47	18	27 179.77
4	91	43 255.58	30	18 623.06	91	52 905.02	19	19 894.15	90	48 676.3	18	18 460.58
5	100	42 469.88	32	17 626.8	92	60 896.5	19	38 727.62	90	56 378.35	19	31 084.49
6	100	42 472.09	32	18 147.3	92	56 254.06	19	32 584.35	90	56 383.44	19	29 085.51
7	100	42 465.9	32	18 843.04	92	51 014.94	19	25 812.23	90	56 398.77	19	26 819.22
8	100	46 572.1	31	18 844.05	91	50 563.15	19	19 039.25	90	53 785.44	18	26 623.06
9	96	45 325.25	32	18 128.99	92	58 395.63	19	35 475.38	90	53 735.28	18	26 236.53
10	93	44 950.54	31	17 915.73	92	56 068.89	19	32 729.32	90	52 273.54	18	25 114.94

Bibliography

[1] Sohaib Afifi, Duc-Cuong Dang, and Aziz Moukrim. “A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints”. In: LION 7, Catania, Italy. Vol. 7997. Lecture Notes in Computer Science. Springer, 2013, pp. 259–265.

[2] Sohaib Afifi, Duc-Cuong Dang, and Aziz Moukrim. “Un algorithme de recuit simulé pour le problème de tournées de véhicules avec fenêtres de temps et contraintes de synchronisation”. (In french) ROADEF 14ième, Troyes, France. 2013.

[3] Sohaib Afifi, Rym Nesrine Guibadj, and Aziz Moukrim. “New Lower Bounds on the Number of Vehicles for the Vehicle Routing Problem with Time Windows”. In: CPAIOR 2014, Cork, Ireland. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014, pp. 422–437.

[4] Sohaib Afifi, Duc-Cuong Dang, and Aziz Moukrim. “Heuristic Solutions for the Vehicle Routing Problem with Time Windows and Synchronized Visits”. Submitted, 2014.

[5] Alfred V. Aho, Michael R. Garey, and Jeffrey D. Ullman. “The transitive reduction of a directed graph”. In: SIAM Journal on Computing 1.2 (1972), pp. 131–137.

[6] Guilherme Bastos Alvarenga, Geraldo Robson Mateus, and G De Tomi. “A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows”. In: Computers & Operations Research 34.6 (2007), pp. 1561–1584.

[7] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. “New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem”. In: Operations Research 59.5 (2011), pp. 1269–1283.

[8] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. “Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints”. In: European Journal of Operational Research 218.1 (2012), pp. 1–6.

[9] Raúl Baños, Julio Ortega, Consolacion Gil, Antonio Fernandez, and Francisco De Toro. “A Simulated Annealing-based parallel multi-objective approach to vehicle routing problems with time windows”. In: Expert Systems with Applications 40.5 (2013), pp. 1696–1707.

[10] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. “Satisfiability tests and time-bound adjustments for cumulative scheduling problems”. In: Annals of Operations Research 92 (1999), pp. 305–333.

[11] Jonathan F Bard, George Kontoravdis, and Gang Yu. “A branch-and-cut procedure for the vehicle routing problem with time windows”. In: Transportation Science 36.2 (2002), pp. 250–269.

[12] Richard Bellman. “On a routing problem”. In: Quarterly of Applied Mathematics 16 (1958), pp. 87–90.

[13] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. “A memetic algorithm for the team orienteering problem”. In: 4or 8.1 (2009), pp. 49–70.

[14] Olli Bräysy and Michel Gendreau. “Vehicle routing problem with time windows, Part I: Route construction and local search algorithms”. In: Transportation science 39.1 (2005), pp. 104–118.

[15] Olli Bräysy and Michel Gendreau. “Vehicle routing problem with time windows, Part II: Metaheuristics”. In: Transportation science 39.1 (2005), pp. 119–139.

[16] David Bredström and Mikael Rönnqvist. “A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints”. In: NHH Dept. of Finance and Management Science Discussion Paper (2007).

[17] David Bredström and Mikael Rönnqvist. “Combined vehicle routing and scheduling with temporal precedence and synchronization constraints”. In: European Journal of Operational Research 191.1 (2008), pp. 19–31.

[18] Jacques Carlier and Eric Pinson. “Adjustment of heads and tails for the job-shop problem”. In: European Journal of Operational Research 78.2 (1994), pp. 146–161.

[19] Wen-Chyuan Chiang and Robert A. Russell. “Simulated annealing metaheuristics for the vehicle routing problem with time windows”. In: Annals of Operations Research 63.1 (1996), pp. 3–27.

[20] Jean-Francois Cordeau, Guy Desaulniers, Jacques Desrosiers, Marius M Solomon, and François Soumis. “The vehicle routing problem”. In: ed. by Paolo Toth and Daniele Vigo. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001. Chap. VRP with Time Windows, pp. 157–193.

[21] Jean-François Cordeau, Michel Gendreau, Alain Hertz, Gilbert Laporte, and Jean-Sylvain Sormany. New heuristics for the vehicle routing problem. Springer, 2005.

[22] Zbigniew J Czech and Piotr Czarnas. “Parallel simulated annealing for the vehicle routing problem with time windows”. In: Proc. of 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (2002).

[23] Duc-cuong Dang, Rym Nesrine, and Aziz Moukrim. “Particle Swarm Optimization for the Team Orienteering Problem”. In: Computers & Operations Research (2011).

[24] George B Dantzig and John H Ramser. “The truck dispatching problem”. In: Management science 6.1 (1959), pp. 80–91.

[25] Guy Desaulniers, François Lessard, and Ahmed Hadjar. “Tabu Search, Partial Elementarity, and Generalized k-Path Inequalities for the Vehicle Routing Problem with Time Windows”. In: Transportation Science 42.3 (2008), pp. 387–404.

[26] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. “A new optimization algorithm for the vehicle routing problem with time windows”. In: Operations research 40.2 (1992), pp. 342–354.

[27] Jacques Desrosiers, François Soumis, and Martin Desrochers. “Routing with time windows by column generation”. In: Networks 14.4 (1984), pp. 545–565.

[28] Jacques Desrosiers, Yvan Dumas, and Francois Soumis. “A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows”. In: American Journal of Mathematical and Management Sciences 6.3-4 (1986), pp. 301–325.

[29] Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and François Soumis. “Time constrained routing and scheduling”. In: Handbooks in operations research and management science 8 (1995), pp. 35–139.

[30] Anders Dohn, Matias Sevel Rasmussen, and Jesper Larsen. “The Vehicle Routing Problem with Time Windows and Temporal Dependencies”. In: Networks 58.4 (2011), pp. 273–289.

[31] Michael Drexl. “Synchronization in vehicle routing-A survey of VRPs with multiple synchronization constraints”. In: Transportation Science 46.3 (2012), pp. 297–316.

[32] Nizar El Hachemi, Michel Gendreau, and Louis-Martin Rousseau. “A heuristic to solve the synchronized log-truck scheduling problem”. In: Computers & Operations Research 40.3 (2013), pp. 666–673.

[33] Jacques Erschler, Pierre Lopez, and Catherine Thuriot. “Raisonnement temporel sous contraintes de ressources et problèmes d’ordonnancement”. In: Revue d’Intelligence Artificielle 5.3 (1991), pp. 7–36.

[34] Matteo Fischetti and Andrea Lodi. “Local branching”. In: Mathematical Programming 98.1-3 (2003), pp. 23–47.

[35] Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. “Solving the orienteering problem through branch-and-cut”. In: INFORMS Journal on Computing 10.2 (1998), pp. 133–148.

[36] Marshall L Fisher, Kurt O Jörnsten, and Oli BG Madsen. “Vehicle Routing with Time Windows: Two Optimization Algorithms”. In: Operations Research 45.3 (1997), pp. 488–492.

[37] Hermann Gehring and Jörg Homberger. “A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows”. In: Proceedings of EUROGEN99. Vol. 2. 1999, pp. 57–64.

[38] Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. “Metaheuristics for the capacitated VRP”. In: The vehicle routing problem 9 (2002), pp. 129–154.

[39] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. “Heuristics and lower bounds for the bin packing problem with conflicts”. In: Computers & Operations Research 31.3 (2004), pp. 347–358.

[40] Michel Gendreau, Jean-Yves Potvin, Olli Bräumlaysy, Geir Hasle, and Arne Løkketangen. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. Springer, 2008.

[41] Bruce L Golden, Arjang A Assad, and Edward A Wasil. “Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries”. In: The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (2002), pp. 245–286.

[42] Rym Nesrine Guibadj. “Problèmes de tournées de véhicules et application industrielle pour la réduction de l’empreinte écologique”. PhD thesis. 2013. url: http://hal.archives-ouvertes.fr/tel-00966428/.

[43] Mohamed Haouari and Anis Gharbi. “Fast lifting procedures for the bin packing problem”. In: Discrete Optimization 2.3 (2005), pp. 201–218.

[44] Geir Hasle and Oddvar Kloster. “Industrial vehicle routing”. In: Geometric modelling, numerical simulation, and optimization. Springer, 2007, pp. 397–435.

[45] Karla L Hoffman and Manfred Padberg. “Solving airline crew scheduling problems by branch-and-cut”. In: Management Science 39.6 (1993), pp. 657–682.

[46] Jörg Homberger and Hermann Gehring. “A two-phase hybrid metaheuristic for the vehicle routing problem with time windows”. In: European Journal of Operational Research 162.1 (2005), pp. 220–238.

[47] Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014.

[48] Irina Ioachim, Jacques Desrosiers, François Soumis, and Nicolas Bélanger. “Fleet assignment and routing with schedule synchronization constraints”. In: European Journal of Operational Research 119.1 (1999), pp. 75–90.

[49] Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. “Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows”. In: Operations Research 56.2 (2008), pp. 497–511.

[50] Yan Jin, Jin-Kao Hao, and Jean-Philippe Hamiez. “A memetic algorithm for the Minimum Sum Coloring Problem”. In: Computers & Operations Research 43 (2014), pp. 318–327.

[51] Soonchul Jung and Byung Ro Moon. “A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows”. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002. Morgan Kaufmann, 2002, pp. 1309–1316.

[52] Brian Kallehauge. “Formulations and exact algorithms for the vehicle routing problem with time windows”. In: Computers & Operations Research 35.7 (2008), pp. 2307–2330.

[53] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: Proceedings of IEEE international conference on neural networks. Vol. 4. 2. Perth, Australia. 1995, pp. 1942–1948.

[54] Scott Kirkpatrick, MP Vecchi, et al. “Optimization by simmulated annealing”. In: science 220.4598 (1983), pp. 671–680.

[55] Janez Konc and D Janezic. “An improved branch and bound algorithm for the maximum clique problem”. In: proteins 58 (2007), pp. 569–590.

[56] George Kontoravdis and Jonathan F Bard. “A GRASP for the vehicle routing problem with time windows”. In: ORSA journal on Computing 7.1 (1995), pp. 10–23.

[57] Nacima Labadi, Christian Prins, and Mohamed Reghioui. “A memetic algorithm for the vehicle routing problem with time windows”. In: Rairo-operations Research 42 (3 2008), pp. 415–431.

[58] Nacima Labadie, Christian Prins, and Yanyan Yang. “Iterated Local Search for a Vehicle Routing Problem with Synchronization Constraints”. In: ICORES 2014 - Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems, Angers, Loire Valley, France, March 6-8, 2014. 2014, pp. 257–263.

[59] Philippe Laborie and Jerome Rogerie. “Reasoning with Conditional Time-Intervals.” In: FLAIRS conference. 2008, pp. 555–560.

[60] Gilbert Laporte and Yves Nobert. “Exact algorithms for the vehicle routing problem”. In: Surveys in Combinatorial Optimization 31 (1987), pp. 147–184.

[61] Learning and Intelligent Optimization - 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers. Vol. 7997. Lecture Notes in Computer Science. Springer, 2013.

[62] Jan Karel Lenstra and Rinnooy Alexander Kan. “Complexity of vehicle routing and scheduling problems”. In: Networks 11.2 (1981), pp. 221–227.

[63] Yanzhi Li, Andrew Lim, and Brian Rodrigues. “Manpower allocation with time windows and job-teaming constraints”. In: Naval Research Logistics (NRL) 52.4 (2005), pp. 302–311.

[64] Andrew Lim and Xingwen Zhang. “A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows”. In: INFORMS Journal on Computing 19.3 (2007), pp. 443–457.

[65] Tsung-Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao, Yuan-Hsin Chen, Ray-Shine Run, Rong-Jian Chen, Jui-Lin Lai, I Kuo, et al. “An efficient job-shop scheduling algorithm based on particle swarm optimization”. In: Expert Systems with Applications 37.3 (2010), pp. 2629–2636.

[66] John DC Little, Katta G Murty, Dura W Sweeney, and Caroline Karel. “An Algorithm for the Traveling Salesman Problem”. In: Operations Research 11.6 (1963), pp. 972–989.

[67] Denis Naddef and Giovanni Rinaldi. “Branch-and-cut algorithms for the capacitated VRP”. In: The vehicle routing problem 9 (2002), pp. 53–81.

[68] Yuichi Nagata, Olli Bräysy, and Wout Dullaert. “A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows”. In: Computers & Operations Research 37.4 (2010), pp. 724–737.

[69] Emmanuel Néron, Philippe Baptiste, and Jatinder ND Gupta. “Solving hybrid flow shop problem using energetic reasoning and global operations”. In: Omega 29.6 (2001), pp. 501–511.

[70] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. “MiniZinc: Towards a Standard CP Modelling Language.” In: 13th International Conference on Principles and Practice of Constraint Programming (CP2007). Providence, Rhode Island, USA, 2007, pp. 23–27.

[71] Clara Novoa and Robert Storer. “An approximate dynamic programming approach for the vehicle routing problem with stochastic demands”. In: European Journal of Operational Research 196.2 (2009), pp. 509–515.

[72] Beatrice Ombuki, Brian J Ross, and Franklin Hanshar. “Multi-objective Genetic Algorithms for Vehicle Routing Problem with Time Windows”. In: Applied Intelligence 24 (2006), pp. 17–30.

[73] David Pisinger and Stefan Ropke. “A general heuristic for vehicle routing problems”. In: Computers & operations research 34.8 (2007), pp. 2403–2435.

[74] Jean-Yves Potvin, Tanguy Kervahut, Bruno-Laurent Garcia, and Jean-Marc Rousseau. “The vehicle routing problem with time windows part I: tabu search”. In: INFORMS Journal on Computing 8.2 (1996), pp. 158–164.

[75] Eric Prescott-Gagnon, Guy Desaulniers, and Louis-Martin Rousseau. “A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows”. In: Networks 54.4 (2009), pp. 190–204.

[76] Christian Prins. “A simple and effective evolutionary algorithm for the vehicle routing problem”. In: Computers & Operations Research 31.12 (2004), pp. 1985–2002.

[77] Matias Sevel Rasmussen, Tor Justesen, Anders Dohn, and Jesper Larsen. "The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies". In: European Journal of Operational Research 219.3 (2012), pp. 598–610.

[78] Panagiotis P Repoussis, Christos D Tarantilis, and George Ioannou. "Arc-guided evolutionary algorithm for the vehicle routing problem with time windows". In: Evolutionary Computation, IEEE Transactions on 13.3 (2009), pp. 624–647.

[79] Celso C Ribeiro and François Soumis. "A column generation approach to the multiple-depot vehicle scheduling problem". In: Operations research 42.1 (1994), pp. 41–52.

[80] Nicola Secomandi. "Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands". In: Computers & Operations Research 27.11 (2000), pp. 1201–1225.

[81] Marius M Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: Operations Research 35.2 (1987), pp. 254–265.

[82] Marius M Solomon and Jacques Desrosiers. "Time window constrained routing and scheduling problems". In: Transportation science 22 (1988), pp. 1–13.

[83] Kay Chen Tan, YH Chew, and LH Lee. "A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows". In: Computational Optimization and Applications 34.1 (2006), pp. 115–151.

[84] Robert Tarjan. "Depth-first search and linear graph algorithms". In: SIAM journal on computing 1.2 (1972), pp. 146–160.

[89] Paolo Toth and Daniele Vigo. The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002.

[90] Alan Mathison Turing. “On computable numbers, with an application to the Entscheidungsproblem”. In: J. of Math 58 (1936), pp. 345–363.

[91] Ziauddin Ursani, Daryl Essam, David Cornforth, and Robert Stocker. “Localized genetic algorithm for vehicle routing problem with time windows”. In: Applied Soft Computing 11.8 (2011), pp. 5375–5390.

[92] Alex Van Breedam. “Improvement heuristics for the vehicle routing problem based on simulated annealing”. In: European Journal of Operational Research 86.3 (1995), pp. 480–490.

[93] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. “A Hybrid Genetic Algorithm with Adaptive Diversity Management for a Large Class of Vehicle Routing Problems with Time-windows”. In: Computers & Operations Research 40.1 (2013), pp. 475–489.

[94] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Time-Window Relaxations in Vehicle Routing Heuristics. Tech. rep., CIRRELT, Montréal, 2013.

[95] Min Wen, Jesper Larsen, Jens Clausen, Jean-François Cordeau, and Gilbert Laporte. “Vehicle routing with cross-docking”. In: Journal of the Operational Research Society 60.12 (2009), pp. 1708–1718.