DOCTORAT BRETAGNE LOIRE / MATHSTIC

université angers

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ D'ANGERS

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Pengfei HE

Hybrid genetic algorithm for routing problems

Thèse présentée et soutenue à Angers, le 27 Octobre 2022
Unité de recherche : Laboratoire d'Étude et de Recherche en Informatique d'Angers
Thèse N° :

Rapporteurs avant soutenance :

Mme. Caroline PRODHON	Professeur à Université de Technologie de Troyes
M. Philippe LACOMME	Professeur à Université Clermont Auvergne

Composition du Jury :

Président :	Mme. Béatrice DUVAL	Professeur à Université d'Angers
Examinateurs :	Mme. Caroline PRODHON	Professeur à Université de Technologie de Troyes
	M. Philippe LACOMME	Professeur à Université Clermont Auvergne
	Mme. Béatrice DUVAL	Professeur à Université d'Angers
	M. Jin-Kao HAO	Professeur à Université d'Angers
	M. Qinghua WU	Professeur à Huazhong University of Science and Technology
Directeur de thèse :	M. Jin-Kao HAO	Professeur à Université d'Angers

Acknowledgement

It has been a wonderful three-year study at Université d'Angers. In this fruitful journey, I came to know so many knowledgeable and respectable people. Firstly, I would like to express my deepest gratitude to my advisor Professor Jin-Kao Hao, who has been extremely nice to offer my countless guidance and encouragement in research. Working with him over the past few years has been a real privilege, which helps to make up my mind to pursue an academic career and become a dedicated researcher like him. I am sincerely grateful to him for numerous meaningful discussions.

I would like to give special thanks to Professor Béatrice Duval and Professor Qinghua Wu in my thesis committee for providing me help during my study. I would also like to thank Professor Caroline Prodhon and Professor Philippe Lacomme who reviewed my thesis.

I'd like to acknowledge the assistance of our technicians Jean Mathieu Chantrein, Benjamin Jeanneau and Jerome Chalain, our nice secretaries Catherine Pawlonski and Christine Bardarine and all other lab members. Thanks to them, I have enjoyed my life in Angers and had much fun in the daily life.

I am particularly grateful to my parents and my brother for their unconditional love and support, which help me stay determined and go through those hard times. Moreover, I am grateful to my friends Zhengrong Jiang, Yu Wang, Xu Yang, Ke Zhang, Zhiqiang Wei, Hao Guo and Tingyu Jia for their long lasting friendship.

This research has been financially supported by China Scholarship Council (CSC).

Table of Contents

General Introduction

Context

The traveling salesman problem (TSP) and the vehicle routing problem (VRP) have been studied in many fields as canonical combinatorial optimization problems but also due to their practical relevance. The TSP is easy to state: given a finite number of "cities" along with the travel cost between each pair of them, find the shortest way of visiting all the cities and returning to the starting city. Compared with the obscure origins of the TSP, the VRP [DR59] was introduced by Dantzig and Ramser when studying the truck dispatching problem in 1959. Since then, many works have been devoted to the VRP and its variants. In this thesis, we focus on four representative routing problems: the colored traveling salesman problem (CTSP) [Li+14], the minmax multiple traveling salesmen problem (minmax mTSP) [Fra+95], the traveling salesman problems with profits [FDG05] and the split delivery vehicle routing problem (SDVRP) [DT89]. The CTSP is a node routing problem with multiple salesmen, where the cities are divided into m exclusive city sets and one shared city set. The objective is to minimize the total traveling distance of m Hamiltonian circuits (routes) under the following constraints: each exclusive city is to be visited by the corresponding salesman, while each shared city can be visited by any salesman. The minmax mTSP is a generation of the TSP and aims to minimize the longest tour among a set of tours. The TSPs with profits visit some cities (vertices) to optimize the collected profit and the travel costs. The SDVRP extends the classical capacitated VRP and each customer can be visited by more than one vehicle. These routing problems concern various relevant applications such as flexible manufacturing schedule [Men+17], transportation [DT90], robotics [CK21] and unmanned aerial vehicles [MR20].

Given their theoretical and practical significance, a large number of solution approaches including exact and metaheuristic algorithms have been presented to solve these problems. In this thesis, we aim to advance the state-of-the-art of solving large instances of the four representative routing problems with effective metaheuristic algorithms.

Objectives

This thesis focuses on working on an efficient and effective hybrid genetic algorithmic framework for the four routing problems. The main objectives of this thesis can be summarized as follows.

— Investigate the edge assembly crossover operator and related crossover operators applied to routing problems.
— Generalize the edge assembly crossover operator to rich routing problems.
— Integrate powerful TSP heuristics into local search for reducing the length of single route.
— Analyze and compare the performances of different crossover operators.
— Evaluate the performances of the proposed algorithms on commonly used benchmark instances in comparison with state-of-the-art algorithms.
— Analyze the ingredients of the proposed methods to get useful insights about their impacts on the performances of the algorithms.

Contributions

The main contributions of this thesis are summarized as follows.

— For the CTSP, we present the first grouping memetic algorithm for solving this challenging problem. The algorithm includes three main components: (i) a greedy randomized heuristic for population initialization; (ii) a dedicated local search procedure for local optima exploration; (iii) a backbone-based crossover operator for solution recombination. We show computational results on three sets of 65 popular benchmark instances to demonstrate the competitiveness of our algorithm. We especially report improved upper bounds for 38 instances (for more than 58% cases). This work has been published in *Information Sciences* [HHW21].
— For the minmax mTSP, the proposed algorithm combines a generalized edge assembly crossover to generate new solutions, an efficient variable neighborhood descent to ensure local optimization as well as an aggressive post-optimization for additional solution improvements. Extensive experimental results on 77 minmax mTSP benchmark instances and 43 minmax multidepot mTSP instances commonly used in the literature indicate a high performance of the algorithm compared to the leading state-of-the-art algorithms. This work has been submitted to *European Journal*

of Operational Research.

— For the TSPs with profits, since two problems, orienteering problem (OP) and prize-collecting traveling salesman problem (PCTSP), are representative, we introduce a hybrid genetic algorithm that addresses these two problems under a unified framework. The algorithm combines an extended edge assembling crossover operator to produce promising offspring solutions and an effective local search to ameliorate each offspring solution. The algorithm is further enforced by a diversification-oriented mutation and a population-diversity management. Extensive experiments show that the method competes favorably with the best existing methods both in terms of solution quality and computational efficiency. This work has been submitted to *Networks*.

— For the SDVRP, we present an effective memetic algorithm for solving the problem with a fleet of limited or unlimited vehicles. The algorithm features a general edge assembly crossover to generate promising offspring solutions from the perspective of assembling suitable edges and an effective local search to improve each offspring solution. The algorithm is further reinforced by a feasibility-restoring procedure, a diversification-oriented mutation and a quality-and-distance pool updating technique. Extensive experiments on 324 benchmark instances indicate that our algorithm is able to update 143 best upper bounds in the literature and match the best results for 156 other instances. This work has been submitted to *Transportation Science*.

Furthermore, a work with presenting a two phase iterated local search algorithm to solving the CTSP was launched in the beginning of my study period and published in *Engineering Applications of Artifcial Intelligence* [HH21]. Before presenting MA to addressing the minmax mTSP, a hybrid search with neighborhood reduction for the multiple traveling salesmen problem is also investigated and published in *Computers & Operations Research* [HH22].

Finally, I also worked on the Hamiltonian p-median problem and a hybird genetic algorithm was launched, and this paper has been submitted to *IEEE Transactions on Evolutionary Computation*.

Organization

The thesis is organized in the following way:

— In the first chapter, we introduce the TSP, VRP and its variants first. Then the four studied problems are described in terms of definition and mathematical models. Furthermore, representative solution approaches, including heuristic, metaheuristic and exact algorithms are reviewed, as well as commonly used benchmark instances. Finally, hybrid genetic algorithms for solving rich routing problems are investigated and summarized.

— In the second chapter, an effective grouping memetic algorithm (GMA) is presented for the CTSP. We introduce its algorithmic components in detail. Extensive experiments indicate the competitiveness of our algorithm. Moreover, we also shed lights on the impacts of the key components of the algorithm.

— In the third chapter, a new memetic algorithm (MA) is addressed for the minmax mTSP with single depot (the minmax mTSP) and multiple depots (the minmax multidepot mTSP). The framework and detail components of the algorithm are introduced sequentially. Experiments to assess the performance of our algorithm and additional experimental investigations for analyzing the impacts of key components are also carried out.

— In the fourth chapter, a hybrid genetic algorithm (HGA) is presented for TSPs with profits. Two representative problems are concerned, that is the orienteering problem (OP) and the prize-collecting traveling salesman problem (PCTSP), under a unified framework. Extensive experiments show that the method competes favorably with the best existing methods both in terms of solution quality and computational efficiency.

— In the fifth chapter, we study the split delivery vehicle routing problem. We present an effective memetic algorithm (SplitMA) for solving the problem with a fleet of limited or unlimited vehicles. Extensive experiments on well known benchmark instances indicate that our algorithm is favourably competitive.

— In the last chapter, we summarize the contributions of this thesis and provide some perspectives for future research.

Part I

Introduction

Chapter 1

Introduction

1.1 Traveling salesman problem and vehicle routing problems

The traveling salesman problem (TSP) is one of the most famous combinatorial optimization problems. Numerous studies have been devoted to the problem from different perspectives, such as mathematical programming [DFJ54; Mil78; MTZ60] and heuristics [Lin65; LK73; ML55]. Let x_{ij} be a binary variable and $x_{ij} = 1$ means the path goes from city i to city j, otherwise $x_{ij} = 0$. The symmetric TSP on n cities can be expressed as an integer linear program (ILP) on the binary variables x_{ij}, $i < j$, $i = 1, \cdots, n-1$; $j = i+1, \cdots, n$:

$$(TSP) \quad \text{Minimize} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \tag{1.1}$$

$$\text{subject to} \quad x_{ij} \geq 0 \tag{1.2}$$

$$x_{ij} \leq 1 \tag{1.3}$$

$$\sum_{i=1}^{k-1} x_{ik} + \sum_{j=k+1}^{n} x_{kj} = 2 \tag{1.4}$$

$$\sum_{i \in \mathcal{S}, j \in \mathcal{S}} x_{ij} \leq |\mathcal{S}| - 1 \tag{1.5}$$

Constraints (1.4) are referred to as the vertex constraints and ensure that each city is visited. Constraints 1.5 correspond to subtour elimination constraints [DFJ54].

The vehicle routing problem (VRP) concerns multiple routes starting from the depot and ending at the depot. Without loss generality, we use *set partition formulation* [BQ64] to model the VRP.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V} = \{0, 1, \cdots, n\}$ is the vertex set with 0 being the depot and $\mathcal{N} = \{1, \cdots, n\}$ representing n customers and $\mathcal{E}$ is the edge set. Let Ω be the set of routes, each route being given by a sequence of edges that describe a path from the depot to some customers. The travel cost c_r of a route $r \in \Omega$ is given by the sum of the cost of the edges in its path. Let a_{ir} state the number of times customer i is visited by route r. Let λ_r be a binary variable. $\lambda_r = 1$ if the route r is performed, otherwise not. The *set partition formulation* is as follows:

$$(VRP) \quad \text{Minimize} \quad \sum_{r \in \Omega} c_r \lambda_r \tag{1.6}$$

$$\text{subject to} \quad \sum_{r\in\Omega} a_{ir}\lambda_r = 1 \qquad \forall i \in \mathcal{N} \tag{1.7}$$

$$\sum_{r\in\Omega} \lambda_r = |\mathcal{K}|, \tag{1.8}$$

The objective function (1.6) minimizes the overall cost of the selected routes. Constraints (1.7) guarantee that each customer is visited by exactly one route. Constraint (1.8) imposes the use of $|\mathcal{K}|$ vehicles. There are many models and algorithms for the optimal and approximate solution of different versions of the VRP.

Various TSP and VRP variants exist in real-life applications. In this thesis, we investigate four problems: colored traveling salesman problem (CTSP), minmax multiple traveling salesmen problem (minmax mTSP), traveling salesman problems with profits (TSPs with profits) and split delivery vehicle routing problems (SDVRP), which are introduced in the next sections.

1.2 Colored traveling salesmen problem

1.2.1 Problem introduction

Let $\mathcal{G}$=($\mathcal{V}, \mathcal{E}$) be a complete undirected graph, where $\mathcal{V} = \{0, 1, 2, \cdots, n-1\}$ is the set of nodes (or cities) and $\mathcal{E} = \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ is the set of edges. Each edge $\{i, j\} \in \mathcal{E}$ has a non-negative weight c_{ij} representing the traveling distance between cities i and j. All cities are divided into $m + 1$ disjoint sets: m exclusive city sets $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$, and one shared city set $\mathcal{S}$ such that $\cup_{k=1}^{m}\mathcal{C}_k \cup \mathcal{S} = \mathcal{V}$ and $\cap_{k=1}^{m}\mathcal{C}_i \cap \mathcal{S} = \emptyset$. Let $\mathcal{K} = \{1, 2, \cdots, m\}$ be a set of salesmen. The cities of an exclusive set $\mathcal{C}_k$ ($k \in \mathcal{K}$) are to be visited by salesman k only, while the shared cities can be visited by any of the m salesmen. Besides, city 0 (the depot) belongs to the shared city set $\mathcal{S}$ and is visited by all salesmen. The CTSP is to determine m shortest Hamiltonian tours (routes) starting from the depot and ending at the depot such that each exclusive city in $\mathcal{C}_k$ is visited exactly once by salesman k and each shared city is visited exactly once by one of the m salesmen.

$$(CTSP) \quad \text{Minimize} \quad \sum_{k=1}^{m}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} c_{ij}x_{ijk} \tag{1.9}$$

$$\sum_{i=1}^{n-1} x_{0ik} = 1, \forall k \in \mathcal{K} \tag{1.10}$$

$$\sum_{i=1}^{n-1} x_{i0k} = 1, \forall k \in \mathcal{K} \tag{1.11}$$

$$\sum_{i \in \mathcal{C}_k \cup \mathcal{S}} \sum_{j \in \mathcal{V} \backslash (\mathcal{C}_k \cup \mathcal{S})} x_{ijk} = 0, \forall k \in \mathcal{K} \tag{1.12}$$

$$\sum_{j=0}^{n-1} \sum_{k=1}^{m} x_{jik} = 1, j \neq i, i \in \mathcal{V} \backslash \{0\} \tag{1.13}$$

$$\sum_{l} x_{jlk} = \sum_{i} x_{ijk}, i \neq j \neq l, i, j, l \in \mathcal{C}_k \cup \mathcal{S}, \forall k \in \mathcal{K} \tag{1.14}$$

$$u_{ik} - u_{jk} + n \times x_{ijk} \leq n - 1, j \neq i, i, j \in \mathcal{V} \backslash \{0\}, \forall k \in \mathcal{K} \tag{1.15}$$

The binary variable $x_{ijk} = 1$ indicates that the k-th salesman passes through edge $\{i, j\}$, and otherwise $x_{ijk} = 0$. u_{ik} is the number of cities visited on the k-th route from the depot up to city i. The objective function of CTSP is given by objective (1.9) and Constraints (1.10-1.15) are the constraints of the problem. Constraints (1.10) and (1.11) require that each salesman starts from the depot and returns to the depot. Constraint (1.12) indicates that each salesman can only visit its own exclusive cities and some shared cities. Constraint (1.13) means that each city except the depot can only be visited exactly once. Constraint (1.14) indicates that a salesman can only arrive at its exclusive and shared cities to continue its route. Constraints (1.14) and (1.15) are employed to eliminate the subtours for each salesman. One notices that the Miller-Tucker-Zemlin subtour elimination constraints [MTZ60] are presented to eliminate all subtours. Although the subtour elimination constraints of Dantzig-Fulkerson-Johnson [DFJ54] concern less decision variables, it becomes impossible when using ILP solvers directly.

The CTSP generalizes a variant of the classical traveling salesman problem (TSP), known as the multiple traveling salesmen problem (mTSP) where all cities are shared [Bek06; CR06]. Besides, if there is only one salesman ($m = 1$), the CTSP becomes the TSP [App+06]. Given that the CTSP generalizes these $\mathcal{NP}$-hard problems, solving the CTSP is computationally challenging.

1.2.2 Related work

Given the theoretical and practical significance, the CTSP has attracted considerable attention in recent years [He+20; He+18; Li+14] and several heuristic methods have been presented. In this section, we review the existing solution approaches for the CTSP and related works.

In 2014, Li et al. [Li+14] introduced the colored traveling salesmen problem to optimize routes of a dual-bridge waterjet cutting machine tool. As solution methods, they presented four genetic algorithms (basic genetic algorithm (GA), GA with greedy initialization, hill-climbing GA and simulated annealing GA), where the *dual-chromosome* encoding was used to represent the candidate solutions. The first chromosome is a permutation of all cities except depot 0, while the second chromosome assigns a salesman to each of the shared and exclusive cities in the corresponding position of the first chromosome. They presented computational results on 20 small scale benchmarks created from existing symmetric TSP instances (with up to 101 vertices). They showed that the hybrid algorithm combining simulated annealing and GA dominated the three other algorithms and their algorithms performed better than the general mixed integer programming tool Lingo.

Then, in 2017, Meng et al. [Men+17] proposed a variable neighborhood search (VNS) which employs a *direct-route* encoding to represent the solutions. VNS consists of two phases. The first phase perturbs the current solution by two shaking operations (*Interchange* and *Relocation*), while the second phase improves the perturbed solution by applying a local search based on two search operations (neighborhood change and 2-opt). Compared with the four GAs [Li+14], VNS showed its competitiveness on the 20 benchmark instances.

Later, in 2018, Pandiri and Singh [PS18] presented an artificial bee colony (ABC) based on the *m-tour* encoding. This encoding uses m arrays, and each array includes all the cities visited by the corresponding salesman. They provided a proof that the size of the solution space of the CTSP with the *m-tour* encoding is smaller than that of the *dual-chromosome* encoding. They showed that ABC could match or update the best results reported in [Li+14; Men+17] on the 20 small scale benchmark instances with very short cutoff times. Besides, they introduced 8 new medium scale instances (with 229-666 cities) and reported computational results.

Also in 2018, Dong et al. [DDC18] employed an ant colony optimization (ACO) with multi-tasks learning for solving the CTSP. The multi-task cooperative learning was proposed to improve the efficiency of ACO. To assess their algorithm, they introduced 6 medium (with 202-431 cities) and 5 large instances (with 1002 cities) and showed the competitiveness of ACO compared with the four GAs [Li+14]. Nevertheless, this algorithm did not compete well with VNS [Men+17] on the 20 small scale instances in terms of the best and average results.

In 2019, Dong et al. [Don+19] presented another artificial bee colony algorithm (ABC) and reported computational results on 26 new large instances (with 2461-7397 cities). These new large scale instances could be used by subsequent studies to evaluate their algorithms. However, this ABC algorithm performed worse than the ABC algorithm of [PS18] on the 20 small scale instances.

He and Hao [HH21] proposed an iterated two-phase local search (ITPLS), which is based on a new *adjacency representation* of the candidate solutions. This representation relies on an array $A[m, n+1]$ such that for each route r $(r = 1, \ldots, m)$, $A[r, i] = j$ $(i, j = 0, \ldots, n, i \neq j)$ if and only if the route goes from city i to city j. ITPLS applies jointly inter-route optimization and intra-route optimization for solution improvement, reinforced by a probabilistic greedy perturbation strategy to diversify the search. Extensive computational results were reported on all the benchmark instances available in the literature (a total of 65 instances), including 29 improved best known results.

After He et al. [HHW21], Zhou et al. [Zho+22] proposed multi-neighborhood simulated annealing-based iterated local search to solve the CTSP and experimental results indicated it performs significantly better than the grouping memetic algorithm [HHW21]. Lastly, Zheng et al. [Zhe+22a] addressed the CTSP by a reinforced Lin-Kernighan-Helsgaun Algorithm and many upper bounds are updated again.

According to the computational results reported in the literature, we identify ABC [PS18] and ITPLS [HH21] as the current state-of-the-art CTSP algorithms.

The CTSP generalizes the popular multiple traveling salesmen problem (mTSP), which has attracted much interest in the last decades. For instance, Wang et al. [WCL17] introduced a memetic algorithm for solving mTSP, which includes a variable neighborhood descent to search local optima. Another evolutionary algorithm was proposed by Kashan et al. [KAO15] for solving mTSP from the perspective of grouping problems. Other representative studies were reported in [CR06; SB09; Soy15; Yua+13]. However, these methods are not suitable for the CTSP, because of the presence of exclusive cities.

In this work, we are interested in designing a practically effective algorithm for solving the CTSP with the memetic framework. This is motivated by two considerations. First, one notices that the route of each salesman can be considered as a TSP solution. Therefore, the optimization of each individual route can naturally benefit from existing powerful TSP methods. Second, we can consider the CTSP from the perspective of grouping problems in the sense that the shared cities are to be dispatched into m groups (m being the number of salesmen). As such, the population-based memetic framework with a meaningful crossover

represents an attractive approach given that it has been applied with great success to several difficult grouping problems (e.g., [Fal98; GBF11; ZHG18]).

1.2.3 Benchmark instances

We employ three sets of 65 benchmark instances, which were commonly used in previous studies on the CTSP. The first set (Set I) contains 20 small instances which were introduced in [Li+14], and the number of cities is between 21 to 101 while the number of salesmen m is between 2 and 7. The second set (Set II), introduced in [DDC18; PS18], contains 14 medium size instances with 202, 229, 431, 666 cities, and $10-40$ salesmen. The last set (Set III), presented in [DDC18; Don+19], contains 31 large instances with $1002-7397$ cities and $3-60$ salesmen. These benchmark instances and the solution certificates for them obtained by the GMA algorithm are available online [1].

1.3 Minmax multiple traveling salesmen problem

1.3.1 Problem introduction

Let $\mathcal{G}=(\mathcal{V},\mathcal{E})$ be an edge-weighted graph, where $\mathcal{V}=\{0,1,\ldots,n\}$ is the vertex set with 0 being the starting-ending city (depot) and $\mathcal{N}=\{1,\cdots,n\}$ representing n cities and $\mathcal{E}$ is the set of arcs (edges). Let $\mathcal{C}=(c_{ij})$ be a non-negative cost (distance) matrix associated with $\mathcal{E}$, which satisfies the triangle inequality ($c_{ij}+c_{jk}>c_{ik}$ for all $i,j,k\in\mathcal{V}$ and $i\neq j\neq k$). The matrix $\mathcal{C}$ is said to be symmetric when $c_{ij}=c_{ji},(i,j)\in\mathcal{E}$ and asymmetric otherwise. The basic mTSP is to partitioning the set of the cities ($\mathcal{N}$) into m distinct Hamiltonian tours $\{r_1,r_2,\ldots,r_m\}$ starting at the depot (vertex 0), such that 1) each tour r_k ($k\in\{1,2,\ldots,m\}$) includes at least two vertices, and 2) an objective function is minimized. From an application perspective, one of the following minimization objectives is considered in the literature: 1) the minsum mTSP which minimizes the total traveling distance of the m tours [SH73], and 2) the minmax mTSP which minimizes the longest tour among the m tours [Fra+95].

It is known for a long time that the minsum mTSP can be conveniently transformed to the TSP [HP77; Rao80]. Recently, it was shown that this transformation approach is quite powerful and able to effectively solve the existing minsum mTSP benchmark instances by leading TSP methods [HH22]. On the other hand, the situation is different for the minmax

1. https://leria-info.univ-angers.fr/ jinkao.hao/GMACTSP.html

mTSP for which a number of dedicated methods have been developed. In this work, we focus on the minmax mTSP including both the cases of single depot and multiple depots.

The minmax mTSP with single depot can be used to formulate many applications. Meanwhile, there are other situations where multiple depots need to be considered. For example, in humanitarian logistics, several depots are deployed in different locations to ensure an efficient delivery of relief supplies to specific places [CVH08]. The minmax multidepot vehicle routing problem was first proposed to formulate such applications, where the objective is to minimize the longest tour [Car+09]. If the capacity constraint is ignored, the problem becomes the minmax multidepot mTSP [Car+09; WGW15]. Clearly, the minmax multidepot mTSP generalizes the minmax mTSP and has interesting applications such as allocating targets to unmanned vehicles [Ras+03] and allocating computer networks resources where the objective is to minimize the maximum latency between a server and a client [WGW15].

1.3.2 Related work

We briefly review the state-of-the-art heuristic algorithms for the minmax mTSP and the minmax multidepot mTSP. Given that the minmax mTSP was introduced much earlier than the minmax multidepot mTSP (1995 vs. 2009), there are more studies on the minmax mTSP than on the minmax multidepot mTSP.

The minmax mTSP

The minmax mTSP was introduced in 1995 by França et al. [Fra+95]. Since then many studies have been devoted to the problem. Comprehensive surveys about the applications, solution approaches and taxonomy are available in [Bek06; CK21]. In this section, we focus on the most recent and representative heuristics for the problem.

Population-based metaheuristics have been presented for solving the minmax mTSP. Carter and Ragsdale [CR06] proposed a genetic algorithm in 2006. The algorithm was based on a two-part chromosome representation and applied classic crossover operators for the TSP to generate offspring solutions. Similarly, in 2007, Brown et al. [BRC07] introduced another genetic algorithm, which adopted a two-part chromosome representation with real values. Subsequently, in 2009, Singh and Baghel [SB09] presented a grouping genetic algorithm, which features a new chromosome representation and a concise crossover operator such that the most promising tour (the shortest) from the parents was

inherited by the offspring. In 2013, Yuan et al. [Yua+13] presented a specific crossover operator based on the two-part chromosome representation of [CR06]. In 2017, Wang et al. [WCL17] investigated a memetic algorithm based on sequential variable neighborhood descent (MASVND) and the crossover operator of [SB09]. Computational experiments on 31 instances with 51-1173 cities and 3-20 tours indicated MASVND was competitive compared to other algorithms, especially for instances with a large number of cities. In 2021, Karabulut et al. [Kar+21] proposed an evolution strategy algorithm (ES), where a self-adaptive Ruin and Recreate heuristic was employed to generate offspring solutions. ES reported excellent results by improving 14 best-known solutions with 51-1173 cities and 3-30 tours among 51 minmax mTSP instances. One notices that these algorithms are based on crossover operators that focus on cities and tours, contrary to powerful TSP crossovers such as EAX [NK97; NK13] where the focus is on how to inherit set of edges (subtours) from parents to offspring solutions.

Swarm intelligence algorithms have been studied for solving the minmax mTSP, which showed good performances. In 2015, Pandiri and Singh [PS15] presented two bio-inspired algorithms (ABC and IWO). The IWO algorithm delivered excellent results and updated 12 best results reported in [BRC07; CR06; SB09; Yua+13] for the 25 tested instances. Additional studies on swarm intelligence algorithms for the minmax mTSP were presented in [LY19; ZSP18]. However, they are less competitive compared to the best algorithms such as ES [Kar+21] and IWO [PS15].

Neighborhood-based local optimization has also been investigated for solving the minmax mTSP. In 2015, Soylu [Soy15] presented a general variable neighborhood search algorithm based on several move operators including 2-opt and or-opt moves. Experimental results indicated a good performance of the algorithm, though it is less competitive compared to the IWO algorithm [PS15]. In 2022, He and Hao [HH22] introduced a hybrid search algorithm with neighborhood reduction (HSNR), where two tabu search procedures based on different neighborhoods were alternatively used in combination with the leading TSP heuristic EAX [NK13]. HSNR achieved a remarkable performance by updating the best-known solutions for 15 out of the 41 popular benchmark instances (with 51-1173 cities and 3-30 tours). Additional results were reported on a new set of 36 large instances with 1379-5915 cities and 3-20 tours. Also in 2022, Zheng et al. [Zhe+22b] proposed an effective iterated two-stage heuristic algorithm (ITSHA), which combines a clustering-based random greedy initialization procedure and a variable neighborhood search with three move operators (2-opt, Insert and Swap). Experimental results indicated that ITSHA obtained

a good performance by improving 22 upper bounds among 44 instances.

Among the reviewed studies, five algorithms (IWO [PS15], MASVND [WCL17], ES [Kar+21], HSNR [HH22] and ITSHA [Zhe+22b]) hold the best-known results for the minmax mTSP on the 77 benchmark instances. Thus, these methods serve as the main reference algorithms for our comparative study.

The minmax multidepot mTSP

In 2009, Carlsson et al. [Car+09] introduced the minmax multidepot vehicle routing problem with unbounded vehicle capacity. Interestingly, this problem is strictly equivalent to the minmax multidepot mTSP studied in this work. To solve the problem, the authors presented a LP-based heuristic based on the linear programing technique. In 2013, Narasimha et al. [Nar+13] presented an ant colony optimization algorithm for the problem and showed interesting computation results on 11 test instances. Later in 2015, Wang et al. [WGW15] proposed two highly effective heuristics (MD and VNS) for solving the problem. The MD algorithm consists of three stages: (1) the multidepot problem is transformed to a single depot problem, which is then solved; (2) the longest tour is improved with TSP heuristics; (3) all tours are improved by exchanging cities between tours. The VNS algorithm combines variable neighborhood search with the powerful LKH solver [Hel00]. Computational results on a new set of 43 instances with 10-500 cities and 3-20 tours indicated a high performance of these heuristics. Among the reviewed studies, the latest MD and VNS algorithms in [WGW15] represent the state-of-the-art for solving the minmax multidepot mTSP (i.e., the minmax multidepot vehicle routing problem with unbounded capacity).

One notices that until now, the minmax mTSP and the minmax multidepot mTSP have been studied separately, even if they are tightly related. We present below a unified approach to handle both problems.

1.3.3 Benchmarks

Three sets of benchmark instances are used in our experiments: Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP and Set $\mathbb{M}$ for the minmax multidepot mTSP.

Set $\mathbb{S}$: This set includes 41 small and medium-sized instances with 51-1173 cities and 3-30 tours. These instances were introduced in [BRC07; CR06; WCL17] and used in [HHW21; Kar+21; PS15; WCL17; Zhe+22b].

Set $\mathbb{L}$: This set consists of 36 large-sized instances with 1379-5915 cities and 3-20 tours, which were introduced in [HHW21].

Set $\mathbb{M}$: This set includes 43 instances with 10-500 cities and 3-20 tours, which were introduced in [WGW15] [2].

These benchmark instances and the solution certificates for them obtained by the MA algorithm are available online [3].

1.4 Traveling salesman problems with profits

1.4.1 Problem introduction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V} = \{v_0, v_1, \cdots, v_n\}$ is the vertex set with v_0 being the depot and $\mathcal{N} = \{v_1, \cdots, v_n\}$ representing n vertices (customers) and $\mathcal{E}$ is the edge set. Let p_i be a non-negative profit associated with each vertex $v_i \in \mathcal{V}$ ($p_0 = 0$). Let $\mathcal{C} = (c_{ij})$ be a non-negative cost (distance) matrix associated with $\mathcal{E}$ satisfying the triangle inequality ($c_{ij} + c_{jk} > c_{ik}$ for $v_i, v_j, v_k \in \mathcal{V}$ and $v_i \neq v_j \neq v_k$). Traveling salesman problems with profits seek to find an elementary circuit starting and ending at the depot, and visiting some customers to optimize the collected profit and the travel costs. According to the way the profit objective and the travel cost objective are considered, three different TSPs with profits can be identified [FDG05].

The first problem is the profitable tour problem (PTP), where the two objectives are combined into a single objective function which seeks to minimize the travel costs minus the collected profit. The second problem is the orienteering problem (OP) [GLV87; VSV11], which aims to maximize the collected profit under the constraint that the travel costs do not exceed a given value c_{max}. The OP is also known as the selective traveling salesperson problem [GLS98b; LM90] in the literature. The third problem is the prize-collecting TSP (PCTSP) [Bal89; Bie+93], which aims to minimize the travel costs under the constraint that the collected profit must reach a given minimum value p_{min}. As it is indicated in [FDG05], these problems are NP-hard and thus computationally challenging. Also according to [FDG05], among these three problems, the OP and the PCTSP attracted much more attention than the PTP. These TSPs with profits are useful models for a broad range of applications [BM85; FDG05; FT88; GLV16; RB91; VSV11]. In this work,

2. The benchmark instances of the minmax multidepot mTSP is available at https://drum.lib.umd.edu/handle/1903/18710

3. https://github.com/pengfeihe-angers/minmax-mTSP.git

we follow this trend and focus on effective solving of the OP and the PCTSP.

1.4.2 Related work

We provide a literature review of the studies on TSPs with profits according to [FDG05] and [VG19; VSV11].

For the OP, Table 1.1 summarizes the existing heuristic algorithms. A comprehensive review on earlier heuristics up to 2010 is provided in [VSV11]. Our review focuses on more recent studies posterior to that date. In 2010, Silberholz and Golden [SG10] studied the generalized orienteering problem and presented an iterated local search, where routes were improved by 2-opt while unrouted vertices were inserted into the route when the travel costs became less than c_{max}. In 2014, Campos et al. [Cam+14] introduced a GRASP algorithm, which combines the general greedy randomized adaptive search procedure, path relinking and local search with three neighborhoods. Experimental results indicated that the algorithm obtained high-quality solutions within a short running time. In 2015, Marinakis et al. [Mar+15] used a GRASP procedure to construct a population of solutions, which was evolved by applying the simple 1-point crossover and local search. In 2016, Keshtkaran and Ziarati [KZ16] developed another GRASP, where new solutions were generated by a segment removing strategy. Computational results showed the competitiveness of the algorithm on two standard benchmark instances. In 2017, Ostrowski et al. [Ost+17] implemented a specific crossover, where common vertices involved in two routes were considered to produce offspring solutions by changing fragments of the two routes. In this algorithm, feasible and infeasible routes were allowed to be cross-overed, while the fitness function was redefined with respect to the travel costs.

In 2018, Kobeaga et al. [KML18] proposed an evolutionary algorithm for the orienteering problem (EA4OP), which features an interesting edge recombination operator to produce offspring individuals. This recombination operator inherits two main characteristics from parent solutions with respect to vertices and edges. All vertices that are common to both parents are maintained, while vertices that belong to only one parent are included with a probability and all vertices that did not belong any parent are excluded. Edges of the parents were inherited as many as possible in order to pass on the maximum amount of information and decrease length quality losses in offspring solutions. Experimental results indicated that EA4OP was very effective and efficient. In 2019, Santini [San19] presented an adaptive large neighborhood search algorithm (ALNS) including various destroy and repair methods. Experiments on four sets of benchmark instances showed that the algo-

rithm was competitive by producing a number of new best results.

In addition to these heuristic algorithms, we mention the recent revisited branch and cut exact algorithm (RB&C) [KML20], which proved many optimal solutions and updated numerous lower bounds for the benchmark instances.

This review reveals that for the OP, the two heuristic algorithms presented in [KML18; San19] and the exact algorithm of [KML20] represent the current state-of-the-art for solving the OP. They hold the best-known results on the four sets of benchmark instances commonly tested in the literature.

Table 1.1 – Summary of the taxonomy of representative heuristic algorithms for the OP

Literature	Year	Framework
Tsiligirides [Tsi84]	1984	Stochastic algorithm
Golden et al. [GLV87]	1987	Centre of gravity heuristic
Ramesh and Brown [RB91]	1991	Tabu search
Wang et al. [Wan+95]	1995	Artificial neural network
Chao et al. [CGW96]	1996	Record-to-record
Gendreau et al. [GLS98a]	1998	Tabu search
Tasgetiren and Smith [TS00]	2000	Genetic algorithm
Liang et al. [LS06]	2006	Ant colony optimzation
Silberholz and Golden [SG10]	2010	Iterated local search
Campos et al. [Cam+14]	2014	GRASP with path relinking
Marinakis et al. [Mar+15]	2015	Memetic-GRASP
Keshtkaran and Ziarati [KZ16]	2016	GRASP
Ostrowski et al. [Ost+17]	2017	Evolution-inspired local improvement algorithm
Kobeaga et al. [KML18]	2018	Evolutionary algorithm
Santini [San19]	2019	Adaptive large neighborhood search

For the PCTSP, even though several studies have been reported under the name "PCTSP", such as those of [CL08; CSR21; GDM00; PSC13], they deal with in reality a different objective that aims to the minimize the sum of the travel costs and penalties paid for each unrouted vertex. According to the terminology introduced [FDG05], these studies concern thus the profitable tour problem PTP, rather than the PCTSP considered in this work. Meanwhile, Bérubé et al. proposed a branch & cut exact algorithm (B&C) [BGP09] and reported results on medium-sized instances with up to 532 vertices.

1.4.3 Benchmarks

For the OP, there are four sets of instances used in literature, and all of them were introduced by Kobeaga et al. [KML18]. Each set includes 86 instances which are split into two groups: medium-sized instances with up to 400 vertices and large-sized instances with up to 7397 vertices. For the first three sets, the maximum travel cost $c_{max} = \lceil \alpha \cdot v(TSP) \rceil$, where $v(TSP)$ is the length of the shortest Hamiltonian route visiting all vertices and

$\alpha = 0.5$. The profit of each vertex is generated by three methods given by Fischetti et al. [FGT98]. In the last set, α takes different values while all vertices have the same profits as the second set. Furthermore, Vansteenwegen and Gunawan [VG19] also collected a number of OP benchmark instances available online [4], including many small-sized instances. Since four sets of 344 instances in [KML18] are representative, we ignore these small-sized instances mentioned in [VG19].

Since there are no unified instances for the PCTSP, we follow the study [BGP09] and use the same method in [FGT98] to generate three sets of 240 instances with up to 7397 vertices, where each set includes 80 instances and is further split into two groups: medium-sized instances with up to 532 vertices and large-sized instances with up to 7397 vertices. The profit of each vertex is the same as in [BGP09]. Furthermore, Vansteenwegen [Van09] stated that the most difficult OP instances are those where the selected number of vertices is a little more than half of the total number of vertices, we set $p_{min}=\lfloor 0.5 \cdot \sum_{i \in \mathcal{N}} p_i \rfloor$.

All these 344 instances for the OP and 240 instances for the PCTSP are used in our experiments and are available online [5].

1.5 Split delivery vehicle routing problem

1.5.1 Problem introduction

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V} = \{0, 1, \cdots, n\}$ is the vertex set with 0 being the depot and $\mathcal{N} = \{1, \cdots, n\}$ representing n customers and $\mathcal{E}$ is the edge set. Each customer $i \in \mathcal{N}$ is associated with an integer demand $d_i \in \mathcal{Z}^+$. Let $\mathcal{C} = (c_{ij})$ be a non-negative cost (distance) matrix associated with $\mathcal{E}$ satisfying the triangle inequality ($c_{ij} + c_{jk} > c_{ik}$ for all $i, j, k \in \mathcal{V}$ and $i \neq j \neq k$). Given a set of K identical vehicles with capacity Q available at the depot, the SDVRP is to find K routes (K can be limited or unlimited) such that 1) each route starts at the depot to serve a number of customers and ends at the depot without exceeding the vehicle capacity Q, 2) the demand d_i of customer $i \in \mathcal{N}$ can be split and served by more than one vehicle, and 3) the total traveling distance of the K routes is minimized. According to the number K of the available vehicles (fleet size), the problem is called the SDVRP-LF (for limited fleet size) if K is fixed or the SDVRP-UF (for unlimited fleet size) otherwise. For the

4. https://www.mech.kuleuven.be/en/cib/op
5. https://github.com/pengfeihe-angers/tsps-with-profits.git

SDVRP-LF, K is fixed to $K_{min} = \lceil (\sum_{i=1}^{n} d_i/Q) \rceil$ to ensure the feasibility of the solution. A mathematical formulation of both problems is shown as follows.

Given a undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the vertex set $\mathcal{V} = \{0, 1, \cdots, n\}$ where 0 is the depot and $\mathcal{N} = \{1, \cdots, n\}$ represents n customers, and the edge set $\mathcal{E}$. Let $d_i \in \mathcal{Z}^+$ be the demand of customer $i \in \mathcal{N}$ and $\mathcal{C} = (c_{ij})$ a non-negative cost (distance) matrix associated with $\mathcal{E}$ satisfying the triangle inequality ($c_{ij} + c_{jk} > c_{ik}$ for all $i, j, k \in \mathcal{V}$ and $i \neq j \neq k$). Let Q be the capacity of K identical vehicles. The formulation of the SDVRP is based on two decision variables. Binary variable x_{ij}^k takes the value of 1 if vehicle k traverses edge (i, j), and it takes the value of 0 otherwise. Variable y_{ik} is the quantity of the demand of customer i delivered by the kth vehicle. The mathematical model for the SDVRP-UF is described as follows.

$$(SDVRP) \quad \text{Minimize} = \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k \tag{1.16}$$

$$\text{subject to} \quad \sum_{k=1}^{K} \sum_{i=0}^{n} x_{ij}^k \geq 1 \quad j = 0, \cdots, n \tag{1.17}$$

$$\sum_{i=0}^{n} x_{ip}^k - \sum_{j=0}^{n} x_{pj}^k = 0 \quad p = 0, \cdots, n; \; k = 1, \cdots, K \tag{1.18}$$

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{ij}^k \leq |\mathcal{S}| - 1 \quad k = 1, \cdots, K; \; \mathcal{S} \subseteq \mathcal{N} \tag{1.19}$$

$$y_{ik} \leq d_i \sum_{j=0}^{n} x_{ij}^k \quad k = 1, \cdots, K; \; i = 1, \cdots, n \tag{1.20}$$

$$\sum_{k=1}^{K} y_{ik} = d_i \quad i = 1, \cdots, n \tag{1.21}$$

$$\sum_{i=1}^{n} y_{ik} \leq Q \quad k = 1, \cdots, K \tag{1.22}$$

$$x_{ij}^k \in \{0, 1\} \quad i = 0, \cdots, n; \; i = 0, \cdots, n; \; k = 1, \cdots, K \tag{1.23}$$

$$y_{ik} \geq 0 \quad i = 1, \cdots, n; k = 1, \cdots, K \tag{1.24}$$

Constraint (1.17) imposes that each vertex has to be visited at least once. Constraint (1.18) is the flow conservation constraint while constraint 1.19 is used to eliminate subtours. The first three constraints are classical constraints used in routing problems. Constraints (1.20)–(1.22) are related to the allocation of the demands of customers among

vehicles. Constraint (1.20) indicates that customer i can be served by vehicle k only when k visits it. Constraint (1.21) guarantees that the total demand of each customer must be met. Constraint (1.22) imposes that the capacity for each vehicle cannot be exceeded.

Finally, since the SDVRP-LF limits the number of vehicles K to the minimum possible $K_{min} = \lceil (\sum_{i=1}^{n} d_i/Q) \rceil$, this extra constraint ($K = K_{min}$) needs to be added into the model.

1.5.2 Related work

A comprehensive review of exact and heuristic solution approaches untill 2012 can be found in [AS12]. In this section, we focus on a literature review on heuristic approaches, while mentioning some representative studies on exact approaches developed since 2014. Table 1.2 summarizes the methods discussed in this section.

[ABS14] presented two branch-and-cut (B&C) algorithms, where the first uses the formulation of [BMM00] and the other adopts a commodity-flow formulation. The methods solved 17 instances to optimality (one instance with 100 customers). [OKY18] created a compact vehicle-indexed flow formulation and presented computational results including optimal solutions for instances with 76 customers. [MS22] proposed three compact formulations and developed a B&C algorithm, which solved 91 instances to proven optimality (with up to 80 customers). For larger instances, heuristics/metaheuristics such as neighborhood-based local search and population-based search are used to find suboptimal solutions with a reasonable time.

The first local search algorithm for solving the SDVRP was presented by [DT89; DT90]. Two neighborhood operators, namely *k-Split* and *RouteAddition*, were combined into the local search. The *k-Split* operator divides the demand of a customer and inserts the divided demand into different routes with an enough residual capacity. On the contrary, the *RouteAddition* operator tries to remove a split customer from all routes and create a new route to serve the customer. These two operators were widely used in follow-up studies. To better handle the problem and cope with the complexity of the SDVRP, other neighborhood operators were presented. [BPR07] proposed two new operators where two or three customers in two routes are swapped with the possibility of splitting their demands. [DLV10] introduced a new relocation operator where three routes were manipulated to explore neighboring solutions.

The tabu search metaheuristic was adapted to the SDVRP by [ASH06] for the first time, where a neighboring solution was obtained by removing a customer from a set of

Table 1.2 – Summary of the taxonomy of representative algorithms

Literature	Framework	Problem Solved
Exact algorithms		
Archetti et al. [ABS14]	Branch-and-cut	Both
Ozbaygin et al. [OKY18]	Vehicle indexed flow formulation	Both
Munari and Savelsbergh [MS22]	Branch-and-cut	Both
Heuristic methods		
Dror and Trudeau [DT89; DT90]	Local search	SDVRP-UF
Derigs et al. [DLV10]	Local search	SDVRP-UF
Archetti et al. [ASH06]	Tabu search	SDVRP-UF
Aleman and Hill [AH10]	Tabu search	SDVRP-UF
Berbotto et al. [BGN14]	Tabu search	SDVRP-LF
Zhang et al. [Zha+15]	Tabu search	SDVRP-UF
Chen et al. [Che+17]	Priori split strategy	SDVRP-UF
Aleman et al. [AZH10]	Variable neighborhood descent	SDVRP-LF
Han and Chu et al. [HC16]	Variable neighborhood descent	SDVRP-UF
Silva et al. [SSO15]	Iterated local search	Both
Mota et al. [MCC07]	Scatter search algorithm	SDVRP-LF
Campos et al. [CCM08]	Scatter search algorithm	SDVRP-UF
Shi et al. [Shi+18]	Particle swarm optimization	SDVRP-UF
Chen et al. [CGW07]	Hybrid algorithm/matheuristic	SDVRP-UF
Archetti et al. [ASS08]	Hybrid algorithm/matheuristic	SDVRP-UF
Jin et al. [JLE08]	Hybrid algorithm/matheuristic	SDVRP-UF
Boudia et al. [BPR07]	Memetic algorithm	SDVRP-UF
Wilck and Cavalier [WC12]	Genetic algorithm	SDVRP-LF

routes in which it was currently visited and inserting it either into a new route or into an existing route with an enough residual capacity. This algorithm outperformed significantly Dror and Trudeau's algorithms [DT89; DT90]. Then, [AH10] proposed a so-called tabu search with vocabulary building approach (TSVBA). An initial set of solutions was constructed firstly and attractive solution attributes were summarized to explore new solutions. Solutions in the set evolved along with the searching progress. The random granular tabu search (RGTS) was proposed by [BGN14], where a heuristic pruning technique is used to filter non-promising neighborhood solutions and speed up the neighborhood search. Another tabu search algorithm, namely forest-based tabu search (FBTS), was introduced by [Zha+15], where the forest structure is used to represent each solution. Several dedicated operators based on the forest structure were also designed, and the experimental results showed that the FBTS algorithm was competitive with existing algorithms.

[MCC07] proposed a scatter search heuristic to address the SDVRP-LF for the first time. [CCM08] introduced another scatter search for the SDVRP-LF with two distinct procedures for generating initial populations. [HC16] presented a multi-start solution approach for solving the SDVRP-UF. [AZH10] proposed an adaptive memory algorithm for the SDVRP-LF, which uses a constructive procedure for initial solution generation and a variable neighborhood descent for solution improvement. The constructive procedure

builds an initial solution by greedily inserting customers with a mechanism called route angle control. The VND procedure follows to seek improved solutions by exploring three commonly used neighborhoods. [SSO15] presented a multi-start iterated local search (SplitILS) for both cases of limited and unlimited fleet. SplitILS is composed of an efficient perturbation procedure and a randomized variable neighborhood descent which included numerous VRP neighborhood operators and SDVRP neighborhood operators. Extensive experiments indicated that SplitILS dominated previous algorithms. [Che+17] introduced a novel and efficient approach to solve the SDVRP-UF, where each customer's demand was split into small pieces in advance and then the SDVRP was solved by applying leading VRP algorithms [GGW10]. [Shi+18] proposed the first particle swarm optimization for the SDVRP-UF and reported some new upper bounds, although its performance is generally worse than SplitILS [SSO15].

In addition to these local search approaches, two hybrid population-based approaches were investigated. [BPR07] presented the memetic algorithm with population management, which used the crossover operator from [Pri04] and a local search procedure including two new swap moves. The algorithm performed competitively with the tabu search of [ASH06] on a number of benchmark instances. [WC12] proposed another hybrid genetic algorithm that reproduced offspring solutions using route-by-route methods and reported competitive results with previous algorithms, although its results were significantly improved by SplitILS [SSO15] later.

Our review shows that the algorithms in [AH10; AZH09; BGN14; BPR07; CCM08; DLV10; SSO15; WC12; Zha+15] hold the best-known results for the SDVRP-LF and SDVRP-UF. Thus, we use these approaches as our reference algorithms for the comparative study.

1.5.3 Benchmark instances

Four sets of commonly tested instances are used in the experiments.

— Set I. It was proposed by [BMM00] and consists of 25 instances with 22–101 customers. The set has been widely tested by almost all SDVRP algorithms. This set considers two cost matrices (i.e., unrounded and rounded costs), leading to 50 distinct instances.

— Set II. This set was generated by [CCM08] following the procedure provided by [ASH06]. It includes 49 test-instances with up to 199 customers. These instances are divided into 7 groups such that the instances of a group have the same cost

matrix and distinct demands. This set was also used to evaluate some algorithms' performances, such as SplitILS silva2015iterated, [AH10] and [AZH09].

— Set III. The set was presented by [ASS08] following the same approach of [ASH06]. The set is composed of 6 groups including 42 instances with 50–199 customers, and the instances in each group have the same cost matrix and distinct demands.

— Set IV. This set was provided by [CGW07]. It includes 21 instances with 8–288 customers. These instances have the particularity that customers are concentrically distributed around the depot.

All these 162 instances are used in our experiments to evaluate the performance of the proposed SplitMA algorithm. The instances and the best solutions obtained by SplitMA are available online at `https://github.com/pengfeihe-angers/SplitMA`.

1.6 An overview of hybrid genetic algorithms for routing problems

Potvin [Pot09] presented a review of evolutionary algorithms for vehicle routing problems in 2009. We provide a brief review for crossover operators applied on routing problems.

In the beginning, the *path representation* is naturally adopted to state each solution when using genetic algorithms to solve the TSP. Then, the classical one-point crossover proved inadequate, since offspring solutions are usually invalid with missing and duplicated cities. Subsequently, a well-known crossover operator for the TSP was proposed by Oliver et al. [OSH87], namely order crossover (OX). Indeed, many order-based operators are presented to deal with sequencing problems, where the solutions differ only by the ordering of their elements [Pot96]. For example, Tasgetiren and Smith [TS00] and Ostrowski et al. [Ost+17] presented similar crossover operators for orienteering problems. Indeed, such crossover operators are limited for solving the TSP since their performances are not as good as the powerful TSP heuristic LKH [Hel00]. Nagata et al. [NK97; NK13] proposed the most powerful genetic algorithm for solving the TSP, where the key part is the edge assembly crossover (EAX). The algorithm assembles suitable edges from elite solutions to produce promising offspring solutions, not arranging cities in the path. The edge assembly crossover produces offspring solutions from edge perspective, which is distinct with other crossover operators. The crossover firstly constructs a multigraph where all edges from parent solutions are included. Then, all edges in the multigraph are parti-

tioned into some *AB-cycles*, where edges are linked alternatively from father and mother solution. Subsequently, these *AB-cycles* are grouped into several *E-sets*, where each *E-set* may include one or more *AB-cycles*. During the fourth step, each intermediate solution is constructed with a basic solution and an *E-set*. If an edge occurs in both the basic solution and the *E-set*, it will be discarded. Remaining edges are combined to form an intermediate solution. Finally, offspring solutions are produced by removing subtours. It is worth mentioning that this genetic algorithm discards local search operators such as 2-opt and matches the state-of-the-art TSP algorithms, such as LKH [Hel00].

For the VRP, many interesting crossover operators were designed and developed. An adaption of natural crossover is reported in Jung and Moon [JM02] for the VRP with time windows. The operator works on a two dimensional graphical representation. A solution is partitioned into two different classes by drawing one or more curves or geometric figures, like rectangles and ellipses. Then, arcs in one class are kept and transferred to the offspring solutions. After this stage, a repair algorithm is applied to restore feasibility of all routes. The edge recombination crossover (ER) is originally designed for the TSP and extended to the VRP [Kra+95; SFK97]. This operator is also applied to the OP [KML18] by progressively extending a tour with adding edges from parent solutions. There are many similar crossover operators such as matrix-based crossover operator [FM91] and extended order-based crossover operator [TSZ06].

The most successfully hybrid genetic algorithm adopts an alternative paradigm, that is route-first, cluster second. The path representation encodes a unique giant tour that covers all customers. Given such a representation, classical order-based crossover for the TSP can be used. The giant tour then needs to be partitioned into individual feasible routes [Bea83]. For a long time, the difficult issue is how to partition giant tours efficiently. In 2004, Prins [Pri04] presented a polynomial-time algorithm to partition the giant tour into individual routes in an optimal way under corresponding constraints, such as capacity. The algorithm, namely SPLIT, is further improved by Vidal et al. [Vid+12] to solve many vehicle routing problems [Vid17; Vid22; Vid+13; Vid+14]. The hybrid genetic algorithmic framework has been widely used to solve various routing problems, such as team orienteering problem [BDM10] and multi-trip vehicle routing problem [Cat+14]. Furthermore, the EAX crossover has been extended to the capacitated VRP [NB09] and the VRP with time windows [NBD10] with slight revision. These two papers have achieved remarkable results compared with state-of-the-art methods.

Indeed, we notice that hybrid genetic algorithms with giant tour crossover operators

are associated with an obvious shortcoming, that is they require an efficient SPLIT algorithm. For example, for the SDVRP, the SPLIT algorithm is not easy to be implemented since each customer may be visited by more than one vehicle. The similar problem also exists on the minmax mTSP. On the other hand, the EAX crossover has also a limitation since it requires that each vertex in parent solutions should have the same degree in the associated graph. However, for the SDVRP and TSPs with profits, this condition can not be satisfied since each vertex may have different degrees in distinct solutions. In this thesis, one of our objectives is to present a general and effective edge assembly crossover operator for rich routing problems.

1.7 Algorithm assessment

1.7.1 Evaluation indicators

In this thesis, in addition to classical comparison, we use the performance profile [DM02], a visual and popular benchmarking tool, to show a more intuitive performance assessment. For assessment, we focus on a comparison of our algorithm with the state-of-the-art algorithms. Given a set of algorithms (results) $\mathcal{S} = \{s_1, s_2, \cdots, s_k\}$ and a set of instances $\mathcal{Q}$, the performance ratio $r_{s,q}$ of algorithm s on instance q with respect to the best approach for the minimization objective f is given by $r_{s,q} = \frac{f_{s,q}}{min\{f_{a,q:a\in\mathcal{S}}\}}$. The overall performance of approach s is determined by $\mathcal{Q}_s(\tau) = \frac{|q\in\mathcal{Q}|r_{s,q}\leq\tau|}{|\mathcal{Q}|}$, which is the probability for algorithm s that its performance ratio $r_{s,q}$ is within a factor τ. $\mathcal{Q}_s(\tau)$ represents the (cumulative) distribution function for the performance ratio. $\mathcal{Q}_s(\tau = 1)$ is the percentage of instances on which algorithm s performs the best compared to all other algorithms.

1.7.2 Statistical methods

For the experimental studies in this thesis, we constantly apply the Wilcoxon signed-rank test with a confidence level of 0.05 to access our algorithm and each reference state-of-the-art algorithm. If the *p-value* is less than 0.05, the null hypothesis is rejected.

1.8 Chapter conclusion

In this chapter, we presented a brief overview of the well-known traveling salesman problem and vehicle routing problem. Four well-known variants of the TSP and VRP are

considered and related metaheuristic algorithms are summarized. The commonly used benchmark instances are presented subsequently. Finally, hybrid genetic algorithms for rich routing problems are also discussed, including various crossover operators.

Part II

Contributions

Chapter 2

Grouping memetic algorithm for colored traveling salesmen problem

In this chapter, we present the first grouping memetic algorithm for solving the CTSP. The algorithm includes three main components: (i) a greedy randomized heuristic for population initialization; (ii) a dedicated local search procedure for local optima exploration; (iii) a backbone-based crossover operator for solution recombination. Computational results on three sets of 65 popular benchmark instances demonstrate the competitiveness of our algorithm. We especially report improved upper bounds for 38 instances (for more than 58% cases). First computational results with the general CPLEX solver are presented, including 10 proven optimal solutions. Finally, we shed lights on the impacts of the key components of the algorithm. The content of this chapter is based on an article published in *Information Sciences*.

2.1 Introduction

The CTSP is a useful model for a number of practical problems [He+20; He+18; Li+14]. Given its theoretical and practical significance, it has received more and more attention. As shown in Chapter 1.2.2, the review reveals that many metaheuristic algorithms have been presented aiming to obtain the better solutions. Although these algorithms have reported valuable computational results on various benchmark instances, they lack robustness and stability in particular when they are applied to solve large scale instances.

In this chapter, we investigate for the first time the powerful memetic algorithm (MA) framework for solving the CTSP and present a competitive grouping memetic algorithm (GMA) dedicated to the problem. Indeed, effective MAs have been proposed to solve the related mTSP [KAO15; KFT18; LHW19; WCL17] and several vehicle routing problems [Cat+14; NB09; Pri04; Vid+12]. However, most of these MAs are based on the so-called giant tours and split algorithms, which are not suitable for the CTSP due to the presence of exclusive cities. We consider the CTSP from the perspective of grouping problems [Fal98] and introduce an effective grouping memetic algorithm. The proposed algorithm integrates two complementary key components: an original local optima exploration procedure (to find high quality local optima, Section 2.2.3) and a dedicated backbone-based crossover operator (to generate promising new offspring, Section 2.2.4). As demonstrated by the computational results shown in Section 2.3, the proposed algorithm competes very favorably with the state-of-the-art CTSP algorithms on three sets of benchmark instances.

The rest of this chapter is organized as follows. The proposed grouping memetic algorithm is presented in Section 2.2. Computational results and comparisons with state-of-the-art algorithms are presented in Section 2.3. In Section 2.4, the impacts of key components of the algorithm are discussed. Section 2.5 presents conclusions.

2.2 Grouping memetic algorithm

Given a CTSP instance, the search space explored by the CTSP is a multi-route problem whose candidate solutions consist of m tours where the k-th tour includes city 0, the exclusive cities of $\mathcal{C}_k$ and some shared cities of $\mathcal{S}$.

In this section, we present the grouping memetic algorithm for solving the CTSP. For a CTSP instance, GMA explores a search space Ω composed of all candidate feasible solutions, where a candidate solution φ consists of m tours $\{r_1, r_2, \ldots, r_m\}$ with r_k $(k =$

$1, 2, \ldots, m$) being the k-th route visited by the k-th salesman. Given a solution $\varphi \in \Omega$, its objective value $f(\varphi)$ is given by the total distance of its m routes. The goal of GMA is thus to find a solution with the smallest objective value as possible.

In the literature, three common methods were used to represent solutions of the CTSP: dual chromosome encoding [Li+14], *m-tour* encoding [PS18] and *adjacency representation* encoding [HH21]. In this chapter, we adopt the *adjacency representation* encoding, which has the advantage of encoding each route (group of cities) independently to facilitate inter-routes operations. The interested reader is referred to [HH21] for more details and an illustrative example.

2.2.1 General scheme

Algorithm 1 Pseudo-code of GMA algorithm

Input: Instance I, population size p, number of the nearest cities N_n, parameter α;
Output: The best solution φ^* found;
$\mathcal{P} = \{\varphi_1, \varphi_2, \cdots, \varphi_p\} \leftarrow$ PopInitilize (I, p);/*Build an initial population of p elite solutions, Section 2.2.2*/
$\varphi^* \leftarrow \arg\min \{f(\varphi_i) : i = 1, 2, \cdots, p\}$;
while *Stopping condition is not met* **do**
    randomly and uniformly select two parents φ_F and φ_M from $\mathcal{P}$;
    $\varphi_O \leftarrow$ *Backbone_Crossover*(φ_F, φ_M);/*Generate an offspring solution by backbone-based crossover, Section 2.2.4*/
    $\varphi_O \leftarrow$ LOE (φ_O, N_n, α);/*Improve the new solution by local optima exploration, Section 2.2.3*/
    if $f(\varphi_O) < f(\varphi^*)$ **then**
        $\varphi^* \leftarrow \varphi_O$;
    end if
    $\mathcal{P} \leftarrow$ *PoolUpdating*$(\mathcal{P}, \varphi_O)$;/*Update the population with the new solution, Section 2.2.5 */
end while
return φ^*;

The proposed GMA algorithm consists of four main components: population initialization, local optima exploration, backbone-based crossover and population updating. GMA starts with an initial population $\mathcal{P}$ of p solutions generated by the population initialization procedure (Section 2.2.2). It then repeats a number of generations during which new candidate solutions are sampled. At each generation, the backbone-based crossover combines two randomly and uniformly selected parent solutions to generate a promising offspring solution (Section 2.2.4). The local optima exploration (LOE) is then applied to improve the offspring solution (Section 2.2.3), followed by population update (Section 2.2.5). This evolution process is terminated when a predefined stopping condition (e.g.,

an allowed number of generations, an allotted cutoff time limit) is reached. In this work, we use a cutoff time limit. The pseudo-code of GMA is shown in Algorithm 1.

2.2.2 Population initialization

The GMA starts its search with an initial population $\mathcal{P}$ of p high-quality (elite) solutions. To construct a population, we use a greedy randomized heuristic to generate a feasible solution, which is further improved by LOE described in Section 2.2.3. The improved solution is then inserted into $\mathcal{P}$ if the solution is different from any existing solution in $\mathcal{P}$; otherwise, this solution is discarded. This process is repeated until p different solutions are generated. Thanks to the greedy randomized heuristic and subsequent LOE improvement step, we obtain a diverse and high-quality population.

Algorithm 2 Pseudo-code of the greedy randomized heuristic

1: **Input**: Instance I (exclusive city sets $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$, shared city set $\mathcal{S}$) and distance matrix;
2: **Output**: A feasible solution φ;
3: $\varphi \leftarrow \emptyset$;/*First step: build m partial routes with exclusive cities*/
4: **for** $k = 1$ to m **do**
5: $r_k \leftarrow \{0\}$;/*Initiate the route with the city 0*/
6: **while** $\mathcal{C}_k \neq \emptyset$ **do**
7: Select randomly and uniformly a city i from set $\mathcal{C}_k$;
8: Insert city i in route r_k such that the route distance increase is minimal;
9: Remove city i from set $\mathcal{C}_k$;
10: **end while**
11: $\varphi \leftarrow \varphi \cup \{r_k\}$;
12: **end for**
/*Second step: dispatch the shared cities $\mathcal{S} \setminus \{0\}$ among m partial routes*/
13: $\mathcal{S}' \leftarrow \mathcal{S} \setminus \{0\}$;
14: **while** $\mathcal{S}' \neq \emptyset$ **do**
15: Select randomly and uniformly a city j from set $\mathcal{S}'$;
16: Insert city j into a route of φ such that the total distance increase is minimal;
17: Remove city j from set $\mathcal{S}'$;
18: **end while**
19: **return** φ;

A feasible solution is constructed by the greedy randomized heuristic according to the following steps: 1) build a partial route for each of the m salesmen by using the corresponding exclusive cities; 2) dispatch the shared cities among the m partial routes to obtain a complete solution. The pseudo-code of the greedy randomized heuristic is shown in Algorithm 2. During the first step (lines 4-12), to create the k-th partial route r_k, one first initiates the route with the city 0. Then, the exclusive cities in $\mathcal{C}_k$ are selected randomly and uniformly, and inserted one by one into r_k such that the insertion gives the

smallest increase of the route distance. When all exclusive cities of every salesman are inserted into the corresponding route, the first step stops, leading to a partial solution φ composed of m partial routes. During the second step (lines 14-19), the shared cities j from $\mathcal{S} \setminus \{0\}$ are processed randomly and uniformly, and inserted one by one into a route of the partial solution φ such that its total distance increase is minimal. When all shared cities are inserted, an initial solution is obtained. The first step has a time complexity of $\mathcal{O}(|\mathcal{C}_m|^2 \times m)$, while the second step is bounded by $\mathcal{O}(|\mathcal{S}| \times n)$. Therefore, the time complexity of the greedy randomized heuristic is $\mathcal{O}(|\mathcal{S}| \times n)$.

2.2.3 Local optima exploration

Local optimization plays a key role in a memetic algorithm and constitutes one of the driving forces for finding solutions of increasing quality. For an effective examination of local optima, GMA employs a specific strategy that combines an inter-route optimization and an intra-route optimization procedure heuristic. Specifically, our local optima exploration procedure (LOE) iterates two complementary search components: the constrained cross-exchange operator (CCE) (Section 2.2.3) and a TSP heuristic called Edge Assembly Crossover (EAX) [NK13] (Section 2.2.3). CCE improves solutions by exchanging two substrings (sub-routes) from two routes. The routes modified by CCE are indicated by a binary vector RT of length m ($RT[i] = 1$ if route i is changed by CCE, $RT[i] = 0$ otherwise). Then each modified route is further optimized by EAX. CCE and EAX are repeated until the current solution φ can not be further improved. Algorithm 3 shows the pseudo-code of the local optima exploration procedure integrating the CCE operator and the EAX heuristic.

Constrained cross-exchange

The conventional cross-exchange was initially designed for vehicle routing problems [AS19a; CHD10; Tai+97]. It is a generic local search operator which performs exchanges of two consecutive substrings (sub-routes) $\hat{r_i}$ and $\hat{r_j}$ from two different routes r_i and r_j. However, given the particularity of exclusive cities in the CTSP, the cross-exchange can not be used directly in our context. For this reason, we propose a constrained cross-exchange (CCE) in this work. Moreover, it is known that the cross-exchange has a high time complexity [AS19a; Tai+97]. CCE uses a suitable pruning technique to reduce this complexity.

Algorithm 3 Pseudo-code of local optima exploration

Input: Solution φ, number of the nearest cities N_n, parameter α;
Output: Improved solution φ_B;
$\varphi_B \leftarrow \varphi$;
$Flag \leftarrow$ true;
$RT[k] \leftarrow$ false $\forall k \in \{1, \ldots, m\}$; /*$RT$ is a binary vector, indicating the routes modified by CCE*/
while $Flag$ **do**
  $< \varphi, Flag, RT > \leftarrow CCE(\varphi, N_n, \alpha)$;/*Solution improvement by CCE, Section 2.2.3 */
  for $k = 1, \ldots, m$ **do**
    if $RT[k] =$ true **then**
      $\varphi \leftarrow \varphi \setminus \{r_k\}$;
      $r_k \leftarrow EAX(r_k)$;/*Intra-route improvement by EAX, section 2.2.3*/
      $\varphi \leftarrow \varphi \cup \{r_k\}$;
    end if
  end for
  if $f(\varphi) < f(\varphi_B)$ **then**
    $\varphi_B \leftarrow \varphi$;
  end if
end while
return φ_B;

The evaluation of a CCE move for the CTSP is summarized in two steps. The first step is to determine the start of two substrings and the second step is to identify the suitable length of both substrings ($\hat{r_{k_1}}$ and $\hat{r_{k_2}}$). For the start of substring $\hat{r_{k_1}}$, we first need to find an edge which will break route r_{k_1}. Suppose the edge is $\{I_1, I_2\}$. Then, a suitable new neighbor of city I_1 needs to be determined. To limit the number of candidate moves, CCE uses the following heuristic pruning technique that only considers the neighbors among the N_n nearest cities. Suppose that city J_3 is such a neighbor, and city J_3 belongs to route r_{k_2}. If edge $\{I_1, J_3\}$ is added as a new edge, edge $\{J_2, J_3\}$ or edge $\{J_3, J_4\}$ should be removed. Once the starts of two substrings (I_2 and J_3) are determined, we need to identify the length of each substring. It is worth noting that each substring should not include any exclusive cities because these cities are only visited by the corresponding salesman.

Because the number of cities of each substring can vary from 0 to α (a parameter), all feasible combinations of the two substrings with their given starts can be listed, and the move gain δ for each combination can also be calculated. There are at most $(\alpha + 1)^2$ combinations of two substrings. When a substring is empty and the other is non-empty ($\hat{r_{k_1}} = \emptyset$ or $\hat{r_{k_2}} = \emptyset$), these two cases are Or-opt [Or76; Tai+97]. However, both substrings can not be empty simultaneously. Therefore, at most $(\alpha + 1)^2 - 1$ combinations of two substrings could be listed for two given starts. Then, we need to identify the best move (i.e., with the largest gain δ_l) in these combinations. So far, a CCE move $< \hat{r_{k_1}}, \hat{r_{k_2}} >$ is acquired

and the lengths of two substrings are determined. For all combinations of the two starts, the global minimal move gain δ_b can be identified. If $\delta_b < 0$, $Flag \leftarrow true$, $RT[k_1] \leftarrow 1$ and $RT[k_2] \leftarrow 1$; then, solution φ is updated by swapping two substrings ($r\hat{}_{k_1}$ and $r\hat{}_{k_2}$); otherwise, solution φ, $Flag$ and matrix RT are returned, because the stopping condition ($\delta_b \geq 0$) of CCE is met. As for the time complexity of CCE, there are $\mathcal{O}(|S| \times (\alpha + 1))$ ways to select the first substring in any given route, while $\mathcal{O}(N_n \times (\alpha + 1))$ ways exist to select the second substring in another route. Therefore, the time complexity of CCE is $\mathcal{O}(|S| \times N_n \times ((\alpha + 1)^2 - 1))$.

For example, Fig. 2.1 illustrates two cases of determining the starts of two substrings. Then two complete CCE moves ($r\hat{}_{k_1} = \{I_2\}$ and $r\hat{}_{k_2} = \{J_3, J_4\}$ or $r\hat{}_{k_2} = \{J_3, J_2\}$) are illustrated in Fig. 2.2, where cities $\{I_2, J_2, J_3\}$ are shared. If edge $\{J_2, J_3\}$ is broken in the first step, the substring $r\hat{}_{k_2} = \{J_3, J_4\}$ is serial and in order. However, if edge $\{J_3, J_4\}$ is broken in the first step, the substring $r\hat{}_{k_2} = \{J_3, J_2\}$ is serial and reverse.

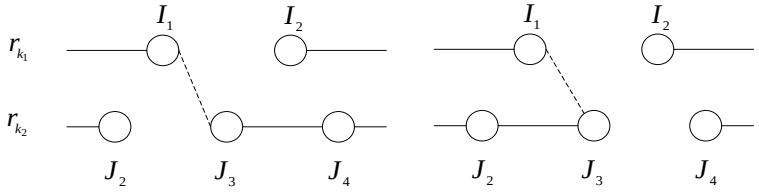


Figure 2.1 – Illustrative example of starts for a CCE move.

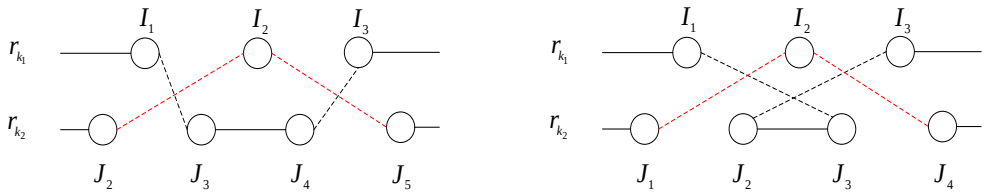


Figure 2.2 – Illustrative example of complete CCE moves.

One may note the following differences between CCE and cross-exchange [AS19a]. First, the cross-exchange operator used in [AS19a] does not limit the length of the substrings to be exchanged; however, in CCE, the length of the substrings must be less than or equal to the value fixed by the parameter α. Second, in CCE, exclusive cities are constrained to stay in a route and can not be moved to other routes. Therefore, the two substrings to be exchanged should not include any exclusive cities. Finally, unlike vehicle routing for which cross-exchange was designed, there is no capacity limitation for each salesman in the CTSP. So CCE does not consider this capacity constraint.

Edge assembly crossover for TSP

For the optimization of each individual route, the constraint of exclusive cities can be ignored. Thus optimizing each route corresponds to solving a TSP. There are several sophisticated and powerful heuristics designed for solving TSP. For example, the well-known fast 2-opt heuristic or LK algorithm can be used to improve each route [AS19a; Hel00; Soy15]. In this work, we adopt the EAX heuristic [NK13] [1], which is among the best TSP heuristics. In our case, EAX helps to keep each route to being optimal or near-optimal in the iterative process of LOE.

2.2.4 Backbone-based crossover

Crossover is another important ingredient of a memetic algorithm and should be designed with care in order to favor transmissions of useful information from parents to offspring [Hao12], while respecting the problem specific structure. One popular way of designing meaningful crossover for grouping problems such as the CTSP is to explore the so-called backbone information, which typically corresponds to solution attributes shared by elite solutions [GBF11; GH99; Sun+20; ZHG18]. In this work, we follow this idea and design a dedicated backbone crossover for the CTSP.

Let φ_F and φ_M be two parent solutions in the population. Based on φ_F and φ_M, we divide the set of shared cities except the depot ($\mathcal{S} \setminus \{0\}$) into two categories, i.e., common elements and non-common elements.

Definition 1: Given two parent solutions $\varphi_F = \{r_1^F, r_2^F, \ldots, r_m^F\}$ and $\varphi_M = \{r_1^M, r_2^M, \ldots, r_m^M\}$, a city $i \in \mathcal{S} \setminus \{0\}$ with respect to φ_F and φ_M is a common element if there exists a $k \in \{1, \ldots, m\}$ such that i appears in both r_k^F and r_k^M (i.e., $i \in r_k^F \cap r_k^M$). If i appears in r_k^F and r_l^M $(k \neq l)$, city i is a non-common element.

Then, an offspring solution φ_O is constructed in two steps. In the first step, a donor parent is first chosen randomly and uniformly between φ_F and φ_M. A partial offspring solution φ_O is then created by inheriting all m routes of the donor parent without the shared cities. In the second step, for each city $i \in \mathcal{S} \setminus \{0\}$, if it is a common element appearing in r_k^F and r_k^m, then city i is greedily inserted into route r_k^O of the partial offspring solution. If city i is a non-common element ($i \in r_k^F$, $i \in r_l^M$ and $k \neq l$), we randomly and uniformly select one route of the partial offspring solution and then greedily insert i into the selected route such that the insertion leads to the smallest increase of the total

1. The code of EAX is available at: https://github.com/sugia/GA-for-TSP

distance.

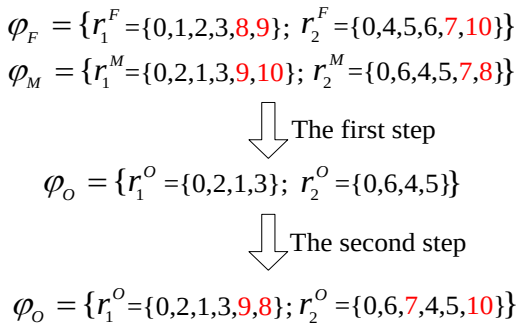


Figure 2.3 – Illustrative example of the backbone-based crossover

Fig. 2.3 shows an example of the crossover operator for a CTSP instance with 11 cities $\{0,1,\ldots,10\}$ and $m = 2$ salesmen with their sets of exclusive cites $\mathcal{C}_1 = \{1,2,3\}$, $\mathcal{C}_2 = \{4,5,6\}$, and the set of shared cities $\mathcal{S} \setminus \{0\} = \{7,8,9,10\}$ (marked in red color). Let $\varphi_F = \{r_1^F = \{0,1,2,3,8,9\}; r_2^F = \{0,4,5,6,7,10\}\}$ and $\varphi_M = \{r_1^M = \{0,2,1,3,9,10\}; r_2^M = \{0,6,4,5,7,8\}\}$ be the parent solutions. By *Definition* 1, cities 7 and 9 are common elements, while 8 and 9 are non-common elements. First, suppose that φ_M is the donor parent. Then offspring φ_O inherits the routes r_1^M and r_1^M by deleting the four shared cities, leading to $\varphi_O \leftarrow \{r_1^O = \{0,2,1,3\}; r_2^O = \{0,6,4,5\}\}$. Then the shared cities $\{7,8,9,10\}$ are successively considered until they are all inserted. City 7 is a common element of the second routes of the parent solutions, it is thus greedily inserted into the partial route r_2^O, supposing this is the cheapest insertion that increases the least the total distance. City 8 is a non-common element, it is greedily inserted into the partial route r_1^O or r_2^O with equal probability. Suppose that route r_1^O is selected, and city 8 is inserted into route r_1^O at the cheapest place leading to the smallest increase of the route distance. Cites 9 and 10 are processed in the same way. When all shared cities $\{7,8,9,10\}$ are inserted into φ_O, a feasible offspring solution is constructed successfully, which is then submitted to LOE for further improvement.

The time complexity of the crossover can be estimated as follows. The first step needs to scan all the cities of the donor parent to allow its m routes to be partially inherited. This is achieved in $\mathcal{O}(n)$ time. In the second step, the shared cities in $\mathcal{S} \setminus \{0\}$ are inserted into the partial offspring at the most suitable places, while the time complexity of evaluating each move gain is $\mathcal{O}(1)$. The second step can be performed in $\mathcal{O}(|\mathcal{S}| \times n)$ time. As the result, the time complexity of the crossover is $\mathcal{O}(|\mathcal{S}| \times n)$.

2.2.5 Pool updating strategy

For each new offspring solution φ_O improved by LOE in Section 2.2.3, the pool updating strategy uses φ_O to update the population $\mathcal{P}$ as follows. If the offspring φ_O is different from any existing solutions and better than the worst solution in $\mathcal{P}$, φ_O replaces the worst solution; otherwise φ_O is discarded.

2.3 Experimental results and comparisons

This section presents a performance assessment of the GMA algorithm. We show computational studies on well-known benchmark instances (see Section 1.2.3) from the literature, and comparisons with existing state-of-the-art algorithms for the CTSP.

2.3.1 Experimental protocol

GMA was coded in C++ and complied with a g++ compiler with the -O3 option [2]. All experiments were conducted on a computer with an AMD-6134 processor (2.3GHz and 6G RAM) running Linux.

To assess the performance of GMA, we show comparisons with the following algorithms: artificial bee colony (ABC) [PS18] (2018), ant colony optimization (ACO) [DDC18] (2018) and iterated two phase local search (ITPLS) [HH21] (2021). Indeed, computational results reported in the literature indicate that these three algorithms represent the state-of-the-art of solving the above benchmark instances, while ABC [PS18] and ITPLS [HH21] are clearly two dominating algorithms. So we use ABC (source code unavailable) and ITPLS (source code available) as the main reference algorithms and cite ACO (source code unavailable) when it is appropriate.

To make the comparisons as fair as possible, we faithfully re-implemented the best ABC algorithm of [PS18] [3]. We verified that our implementation (denoted as re-ABC) was able to reproduce the results reported in [PS18] (and in fact, our ABC implementation even obtained some better results than those reported in [PS18]).

In order to assess the gaps between the heuristic solutions (from GMA and the reference algorithms) and the optimal solutions, we also investigated the general mixed integer programming solver CPLEX (version 12.7) based on the mathematical model from [Li+14]

2. The code of our algorithm will be available at http://www.info.univ-angers.fr/pub/hao/CTSP.html
3. Our implementation of ABC [PS18] is available from the page given in footnote 2.

(see Section 1.2.1). Our experiment indicated that CPLEX with this model can only solve optimally 10 smallest instances of Set I within 7200 seconds, but it fails on Sets II and III due to memory overflow (even on a computer with 20G RAM). The results of CPLEX could be improved by investigating more sophisticated mathematical models.

2.3.2 Parameter tuning

GMA requires 3 parameters: population size p, number of nearest cities N_n and parameter α. In order to identify a set of suitable parameters, we used the popular 'Irace' package [Lóp+16] for automatic parameters tuning. The tuning was performed on 8 benchmark instances with 202 to 5397 cities. For the experiment, the tuning budget was set to 500 runs, each with a time limit of half of the cutoff time. The studied and final values (suggested by Irace) of these parameters are shown in Table 2.1.

Table 2.1 – Parameters tuning results

Parameters	Section	Description	Considered values	Final value
p	2.2.1	population size	{10,15,20,25,30}	20
N_n	2.2.3	number of nearest cities	{30,40,50,60,70,80,90}	50
α	2.2.3	maximum length of substring	{1,2,3,4,5,6,7}	7

2.3.3 Computational results and comparisons with existing algorithms

Computational results of GMA and the reference algorithms on set I are shown in Table 2.2. For CPLEX, we report for each instance the upper bound (UB), the lower bound (LB) and the *Gap* given by $(UB - LB)/LB \times 100$. So $Gap = 0$ implies that an optimal solution is found. Columns $6 - 17$ report respectively the results of re-ABC, ITPLS and GMA in terms of the best objective value f_{best} (over 20 runs), the average objective value f_{avg} , standard deviation σ and the average time in seconds to reach the best objective value (Time(s)). For the f_{best} and f_{avg} indicators, equally best values are shown in italic font.

Given that both the upper bounds and lower bounds are available for these instances, we include the geometric mean of each algorithm for a global assessment (row Geomean). For CPLEX, the geometric mean is calculated with the gaps between UB and LB by $(\prod_{i=1}^{h} \frac{UB_i}{LB_i})^{\frac{1}{h}}$ where UB_i and LB_i are the upper and lower bound of the ith instance, respectively. Similarly, for the other algorithms (re-ABC, ITPLS, and GMA), we calculate their geometric means for the best and average objective values by replacing UB_i with the f_{best} and f_{avg} values, respectively.

Finally, to assess the statistically significant difference between GMA and each main compared algorithm, Table 2.5 shows the *p-values* from the Wilcoxon signed-rank test.

With a confidence level of 95%, a *p-value* smaller than 0.05 indicates a statistically significant difference between the pair of compared results.

From Table 2.2 on the 20 small instances of Set I, the following observations can be made. First, CPLEX is able to solve optimally the 10 smallest instances with $21 - 51$ cities and $2 - 4$ salesmen. For the remaining instances, the gap between UB and LB remains reasonable and tends to increase with the size of the instance. For the three heuristic algorithms, they perform equally well in terms of solution quality by reaching their best solutions consistently including the 10 optimal values. The geometric means indicates that the three heuristic algorithms can reach the same results in terms of both the best and average results. Meanwhile, the heuristic algorithms have smaller geometric means compared with CPLEX and thus perform better for this set of instances. In terms of computational efficiency, GMA and re-ABC perform better than ITPLS since they require significantly less computation times to reach the same results. It is worth mentioning that none of the other algorithms in the literature, such as GA [Li+14] (2014), VNS [Men+17] [4] (2017), ACO [DDC18] (2018), and ABC [Don+19] (2019) can reach such a performance (they report worse results for some instances or their best results cannot be reached consistently).

Table 2.3 presents the results of the compared algorithms (re-ABC, ITPLS and GMA) on the 14 medium instances of Set II with $202 - 666$ cities and $10 - 40$ salesmen. In addition to the main reference algorithms re-ABC and ITPLS, we also include in this comparison ACO [DDC18] for indicative purposes, which only reported results for six instances. The 'BKS' values show the best-known results compiled from the literature [DDC18; HH21; Zhe+22a; Zho+22]. For each algorithm except ACO, we show the best and average objective values (f_{best} and f_{avg}), the standard deviation (σ) and the average time to reach the best objective value (Time(s)). Equally best values are indicated in italic font, while strictly best values are highlighted in boldface. Moreover, the last column $Imp(\%)$ provides the percentage improvement of GMA's best result f_{best} over the best objective value f_{bk} of the reference algorithms, computed as $(f_{best} - f_{bk})/f_{bk} \times 100$. Thus a negative $Imp(\%)$ value indicates that GMA improved the best results of the reference algorithms. For Set II, we ignored the geometric means given that the lower bounds needed for their calculations are unavailable. In fact, we tried to obtain LB for these instances by solving, with CPLEX, the linear programming relaxation of the model presented in the Appendix. But CPLEX terminates abnormally due to memory overflow.

Table 2.3 indicates that GMA finds better results for 7 out of the 14 instances, and matches the best results of two reference algorithms for 3 other instances. The Wilcoxon signed-rank test on the f_{best} and f_{avg} values in Table 2.5 also confirms that GMA significantly outperforms the two main reference algorithms. We do not insist on computation time because the main compared algorithms report solutions of different quality. Nevertheless, the three main compared algorithms (re-ABC, ITPLS and GMA) require comparable computation times to reach their best solutions. Note that the results of ACO [DDC18]

4. VNS reports a wrong result 465.28 for eil51-2 because it is smaller than the proven optimal value 478.08 from CPLEX.

Table 2.2 – Comparative results of GMA and reference algorithms on Set I. The equally best values are indicated in italic.

	CPLEX				re-ABC				ITPLS				GMA (this work)			
Instance	UB	LB	t(s)	*Gap*(%)	f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)
eil21-2	*144.92*	144.92	1	0.00	*144.92*	144.92	0.00	1.00	*144.92*	144.92	0.0	18.32	*144.92*	144.92	0.0	1.00
eil21-3	*157.48*	157.48	1	0.00	*157.48*	157.48	0.00	1.00	*157.48*	157.48	0.0	13.15	*157.48*	157.48	0.0	1.00
eil31-2	*259.36*	259.36	2	0.00	*259.36*	259.36	0.00	1.00	*259.36*	259.36	0.0	12.70	*259.36*	259.36	0.0	1.00
eil31-3	*295.31*	295.31	20	0.00	*295.31*	295.31	0.00	1.00	*295.31*	295.31	0.0	12.85	*295.31*	295.31	0.0	1.00
eil31-4	*315.97*	315.97	61	0.00	*315.97*	315.97	0.00	1.00	*315.97*	315.97	0.0	16.90	*315.97*	315.97	0.0	1.00
eil41-2	*346.24*	346.24	7	0.00	*346.24*	346.24	0.00	1.00	*346.24*	346.24	0.0	14.45	*346.24*	346.24	0.0	1.00
eil41-3	*367.84*	367.84	46	0.00	*367.84*	367.84	0.00	1.00	*367.84*	367.84	0.0	22.05	*367.84*	367.84	0.0	1.00
eil41-4	*392.14*	392.14	120	0.00	*392.14*	392.14	0.00	1.00	*392.14*	392.14	0.0	11.55	*392.14*	392.14	0.0	1.00
eil51-2	*478.08*	478.08	126	0.00	*478.08*	478.08	0.00	1.05	*478.08*	478.08	0.0	21.55	*478.08*	478.08	0.0	1.00
eil51-3	*469.50*	469.50	773	0.00	*469.50*	469.50	0.00	1.00	*469.50*	469.50	0.0	20.40	*469.50*	469.50	0.0	1.00
eil51-4	*489.99*	485.88	7201	0.85	*489.99*	489.99	0.00	1.00	*489.99*	489.99	0.0	28.55	*489.99*	489.99	0.0	1.40
eil51-5	*525.98*	503.84	7212	4.39	*525.98*	525.98	0.00	1.10	*525.98*	525.98	0.0	14.35	*525.98*	525.98	0.0	1.00
eil76-3	596.18	583.41	7211	2.19	*593.28*	593.28	0.00	1.00	*593.28*	593.28	0.0	16.40	*593.28*	593.28	0.0	1.00
eil76-4	*603.79*	585.69	7202	3.09	*603.79*	603.79	0.00	1.50	*603.79*	603.79	0.0	17.75	*603.79*	603.79	0.0	1.60
eil76-5	656.56	620.25	7206	5.85	*651.99*	651.99	0.00	1.00	*651.99*	651.99	0.0	6.75	*651.99*	651.99	0.0	1.00
eil76-6	687.43	624.25	7202	10.12	*672.73*	672.73	0.00	2.10	*672.73*	672.73	0.0	31.40	*672.73*	672.73	0.0	1.00
eil101-4	746.93	697.83	7204	7.04	*726.82*	726.82	0.00	1.30	*726.82*	726.82	0.0	18.45	*726.82*	726.82	0.0	1.00
eil101-5	854.23	750.92	7203	13.76	*779.15*	779.15	0.00	1.05	*779.15*	779.15	0.0	10.80	*779.15*	779.15	0.0	1.00
eil101-6	783.08	706.47	7209	10.84	*759.55*	759.55	0.00	1.25	*759.55*	759.55	0.0	12.75	*759.55*	759.55	0.0	1.70
eil101-7	840.60	729.92	7201	15.16	*798.85*	798.85	0.00	1.20	*798.85*	798.85	0.0	12.56	*798.85*	798.85	0.0	1.80
Geomean	1.0335	-	-	-	1.0235	1.0235	-	-	1.0235	1.0235	-	-	1.0235	1.0235	-	-

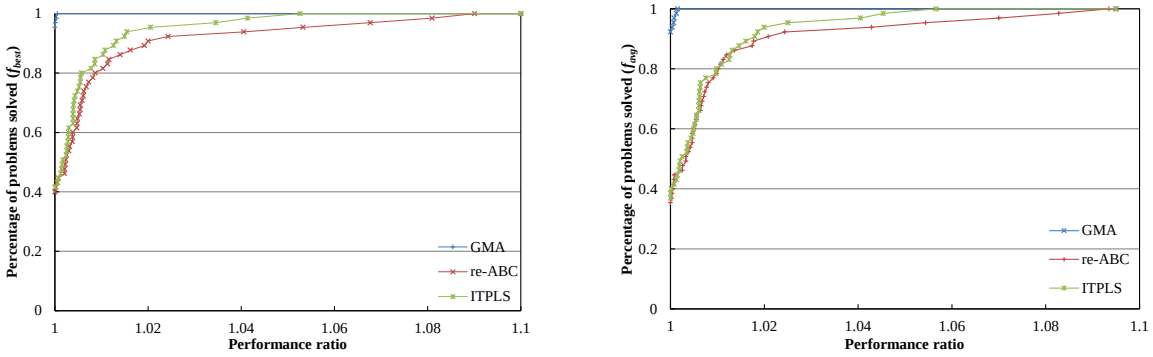
Table 2.3 – Comparative results of GMA and reference algorithms on Set II. Equally best values are indicated in italic. The strictly best values are indicated in boldface.

Instance	BKS	ACO [DDC18]	re-ABC [HH21]				ITPLS [HH21]				GMA (this work)				
			f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)	$Imp(\%)$
gr202-12	71924	71924	99871	100033.20	110.54	329.45	99871	100009.50	112.58	93.55	99871	100162.50	185.46	396.65	39.00
gr202-25	99606	99606	173547	173596.80	54.01	348.85	173418	173523.80	46.77	141.25	173477	173594.65	75.72	211.80	74.00
gr202-35	118495	118495	233749	233817.85	70.16	316.15	233749	233857.80	73.17	156.85	233871	234003.35	72.81	93.85	97.00
gr229-10	222167	-	222167	222354.85	164.08	244.60	222167	222347.65	103.50	226.70	222167	222173.75	30.19	201.90	0.00
gr229-15	264146	-	264146	264146.00	0.00	69.60	264146	264146.00	0.00	67.00	264146	264146.00	0.00	154.85	0.00
gr229-20	319669	-	319669	319669.00	0.00	303.05	319669	319671.90	12.97	128.20	319669	319880.15	547.77	83.20	0.00
gr229-30	406664	-	406664	407194.85	375.21	301.35	406664	406884.00	225.72	186.20	406701	407389.75	279.37	73.10	0.01
gr431-12	248447	330554	249031	249682.25	293.07	300.25	249421	250036.95	613.23	221.45	248447	248447.00	0.00	29.50	0.00
gr431-25	347335	464298	348056	348431.10	203.82	333.30	348181	349238.10	417.38	253.70	347335	347559.80	420.13	394.95	0.00
gr431-40	415169	483977	416189	416758.40	249.58	355.20	416552	417963.75	958.14	296.90	415314	415387.45	88.31	159.45	0.03
gr666-10	386157	-	390188	392234.00	971.38	159.45	389583	396841.55	2716.00	485.90	387562	389594.80	3417.34	473.55	0.36
gr666-15	445849	-	448604	449997.35	716.97	248.15	448257	449635.25	800.17	223.60	446475	447123.60	328.49	506.20	0.14
gr666-20	517842	-	522157	523583.15	937.90	177.55	521149	522650.90	1006.57	249.50	519121	519773.45	397.47	512.25	0.25
gr666-30	649479	-	652587	654001.50	633.57	224.80	651801	653318.10	927.19	255.05	650116	650974.90	417.87	535.70	0.10

are somewhat inconsistent. Among the six instances tested by ACO, even if it reports three better results than the other algorithms (indicated with a star), its results for the three other instances as well as for most of the 20 small instances of Set I are considerably worse than algorithms, such as ABC [PS18] and ITPLS [HH21].

Table 2.4 presents the computational results of the compared algorithms for the 31 large instances of Set III (1002−7397 cities and 3−60 salesmen) with the same information as in Table 2.3. These results clearly show the dominance of the proposed GMA algorithm over two algorithms for these large instances, by systematically reporting better results in terms of the best and the average objective values. Moreover, GMA requires the shortest computation times to reach its solutions for this set of large instances, demonstrating its remarkable search capacity and high computational efficiency. According to the *p-values* (less than 0.05) from the Wilcoxon signed-rank test shown in Table 2.5, the difference between GMA and each compared algorithm is statistically significant.

Tables 2.2-2.4 demonstrate the high competitiveness of the proposed GMA algorithm compared with the state-of-the-art algorithms for solving the existing CTSP benchmark instances. Its superiority becomes more evident when medium and large instances are solved. GMA reports improved best-known results (new upper bounds) for 7 medium instances of Set II and all 31 large instances of Set III, which are useful for future research on the CTSP.



(a) Best **(b)** Avg.

Figure 2.4 – Performance profiles of GMA and two reference algorithms on the 65 instances of sets I, II, and III.

For a more intuitive illustration of the performance assessment of the algorithms, we use the *performance profile* [DM02], which is a popular benchmarking tool for rigorous comparison of different algorithms (see Section 1.7.1).

Fig. 2.4 shows the performance profiles of GMA, ITPLS and re-ABC. We observe that GMA dominates the reference algorithms in terms of the best and average values. Indeed, GMA has a much higher $\mathcal{Q}_x(1)$ value compared with the reference algorithms, indicating that GMA can find better or equal results for all instances. Furthermore, GMA reaches $\mathcal{Q}_x(r_f)$ firstly, indicating a high robustness of our algorithm.

Table 2.4 – Comparative results of GMA and reference algorithms on Set III. The strictly best values are indicated in boldface.

		re-ABC [HH21]				ITPLS [HH21]				GMA (this work)				
Instance	BKS	f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)	f_{best}	f_{avg}	σ	Time(s)	$Imp(\%)$
pr1002-5	313848	316437	317425.40	479.59	1121.95	318587	320348.80	1058.72	2053.45	313885	314083.20	128.79	1868.40	0.01
pr1002-10	379627	382201	382844.90	423.48	843.40	383112	384908.55	936.35	1615.60	379846	379911.00	162.49	1361.65	0.06
pr1002-20	513415	516256	517481.55	452.81	1606.00	517917	519664.85	794.31	1683.05	514968	515784.55	498.49	1815.70	0.30
pr1002-30	660999	664648	665676.30	559.24	2213.60	664308	666702.20	929.80	1753.65	661540	662613.10	958.81	1792.00	0.08
pr1002-40	803365	806022	807838.65	786.79	1757.50	805967	808503.35	1444.57	1946.35	803624	803642.45	74.24	953.10	0.03
fnl2461-3	105211	114188	114509.80	145.77	3562.50	110007	110553.50	413.84	2773.75	105637	105754.90	43.78	1092.55	0.40
fnl2461-6	115340	122312	122612.30	188.81	3593.50	118513	119199.15	387.44	2273.70	116128	116287.05	77.36	1882.75	0.68
fnl2461-12	142211	145800	146374.85	220.68	3597.65	145023	145688.60	284.37	3128.30	143477	143866.10	214.32	3410.40	0.89
fnl2461-24	220633	222465	223335.30	456.26	2231.65	221494	221739.80	163.09	3039.95	221116	221317.35	121.79	3376.65	0.22
fnl2461-30	266848	268431	269140.30	535.88	2188.65	267355	267593.85	169.31	2842.00	267017	267249.45	116.69	3206.00	0.06
fnl3461-3	148602	162335	162909.20	177.47	3504.05	156753	157420.50	391.84	2900.75	148917	148979.55	33.13	1744.75	0.21
fnl3461-12	159388	170762	171243.80	238.70	3612.85	165455	166525.25	512.43	3034.60	159934	160040.70	63.09	1928.10	0.34
fnl3461-12	185195	192874	193582.75	336.98	3612.90	188223	188963.25	371.47	3293.00	185363	185621.60	143.67	3188.80	0.09
fnl3461-24	263073	266686	267130.75	187.32	3504.60	265078	265672.70	342.83	3045.55	263631	263980.40	177.39	3405.60	0.21
fnl3461-30	306639	308742	308963.10	95.74	2526.10	307562	308018.40	208.49	3014.70	307071	307252.30	113.62	3268.00	0.14
fnl3461-40	384329	385443	385727.50	120.94	2313.95	385122	385296.60	113.73	2827.00	384573	384722.95	77.96	3310.50	0.06
pla5397-20	38005790	38335000	38392330	26980.76	3504.15	38331500	38494950	73265.21	3483.15	38006100	38018450	25354.72	3059.70	0.00
pla5397-30	51132597	51299400	51340355	22848.87	3205.35	51339600	51451470	82834.36	3403.30	51138000	51143260	3180.10	3174.00	0.01
pla5397-40	64079810	64408200	64476755	30612.77	3480.90	64285900	64404060	61216.52	3437.45	64097900	64121760	18498.09	2854.40	0.03
pla5397-50	73993599	74008700	74019335	5148.66	2359.50	74051200	74145910	44659.82	3407.50	73993600	73993610	30.78	2709.10	0.00
pla5397-60	85266247	85303100	85324645	10454.99	2161.00	85323100	85424400	60048.45	3046.95	**85266200**	85266750	235.08	3116.55	0.00
pla6397-20	35862883	36672000	36748165	37220.64	3608.95	36404600	36575675	103990.10	3337.10	35951800	35997920	19571.59	3062.95	0.25
pla6397-30	47322900	47689800	47750055	24229.16	3457.40	47551800	47832460	98829.28	3175.25	47346400	47368155	12197.91	3141.80	0.05
pla6397-40	56611300	56948400	57022520	31690.54	3400.30	56860500	56945530	54061.04	3040.75	56638000	56653280	11345.84	3096.10	0.05
pla6397-50	67151700	67415000	67485965	27014.35	3077.95	67347700	67419380	40122.95	2983.45	67161500	67171190	7667.49	3018.50	0.01
pla6397-60	74774800	75077200	75118385	16341.50	3314.70	74983600	75063660	52339.27	3099.85	74791200	74803075	10330.12	3235.45	0.02
pla7397-20	40936853	42262900	42432435	78855.53	14431.45	41804200	42027405	138563.86	13750.30	41260500	41422195	70887.70	12204.65	0.79
pla7397-30	52502160	53648400	53717345	45595.94	14017.35	53183700	53358655	113523.08	13778.95	52636900	52780890	88332.83	12932.47	0.26
pla7397-40	64742100	65847100	65919250	42106.65	14072.90	65441600	65662845	142232.68	13643.15	64937200	65029520	58826.59	12906.25	0.30
pla7397-50	76214793	77194500	77265730	38981.29	14064.05	76701700	76784335	74134.09	13748.85	76331100	76406770	47628.33	12635.55	0.15
pla7397-60	86088267	87041500	87103321	35045.55	13562.42	86628200	86749490	77747.32	13763.90	86153700	86224380	48429.67	13182.35	0.08

Table 2.5 – Summary of comparative results between GMA and two reference algorithms

Algorithm pair	Set/Instance	Indicator	Better	Equal	Worse	$p-value$
GMA vs. ITPLS	I/20	f_{best}	0	20	0	0.00E+00
		f_{avg}	0	20	0	0.00E+00
	II/14	f_{best}	7	4	3	2.44E-04
		f_{avg}	8	1	5	4.80E-02
	III/31	f_{best}	31	0	0	1.17E-06
		f_{avg}	31	0	0	1.17E-06
GMA vs. re-ABC	I/20	f_{best}	0	20	0	0.00E+00
		f_{avg}	0	20	0	0.00E+00
	II/14	f_{best}	8	4	2	1.37E-02
		f_{avg}	9	1	4	4.79E-02
	III/31	f_{best}	31	0	0	1.17E-06
		f_{avg}	31	0	0	1.17E-06

Finally, Table 2.5 summarizes the results reported by the compared algorithms on the three sets of 65 instances. Column 2 gives the set name and the number of instances in the set. Column 3 shows the quality indicators (f_{best} and f_{avg}). Columns 4-6 count the number of instances for which GMA achieves a better, equal or worse value compared with each reference algorithm. The last column presents the *p-values* from the Wilcoxon signed-rank test. Table 2.5 reveals large performance gaps between GMA and each reference algorithm on Sets II and III. We conclude that GMA is very competitive for solving the CTSP and this is particularly true for large instances.

2.4 Discussion and analysis

2.4.1 Benefit of the key components

In this section, we justify the design choices behind the proposed GMA algorithm. For this, we investigate the impacts of its key components: Constrained Cross-exchange, EAX as well as backbone-based crossover. For our experiments, we used the 45 instances of Sets II and III and ignored the instances of Set I. Indeed, for the instances of Set I, their best-known results can be consistently reached by the state-of-the-art algorithms including ABC, ITPLS and GMA within a very short time. As such, these instances are too easy to be used to compare algorithm variants.

Table 2.6 – Comparative results on Set II between GMA and its three variants. Strictly best values are shown in boldface.

	GMA		GMA_0		GMA_1		GMA_2	
Instance	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
gr202-12	**99871.00**	100162.50	100292.00	100722.85	100196.00	100573.25	**99871.00**	100217.90
gr202-25	173477.00	173594.65	173782.00	173828.55	**173394.00**	173681.40	173511.00	173643.90
gr202-35	233871.00	234003.35	234126.00	234226.30	233907.00	234074.20	**233749.00**	233948.20
gr229-10	**222167.00**	222173.75	**222167.00**	222255.75	223266.00	224262.85	**222167.00**	222167.00
gr229-15	**264146.00**	264146.00	264224.00	265715.20	265537.00	266727.75	264183.00	264186.90
gr229-20	**319669.00**	319880.15	320976.00	322435.20	**319669.00**	320910.90	**319669.00**	320424.30
gr229-30	**406701.00**	407389.75	407692.00	408434.25	407962.00	408942.80	407226.00	407648.55
gr431-12	**248447.00**	248447.00	**248447.00**	248462.10	252253.00	254230.25	**248447.00**	248447.00
gr431-25	**347335.00**	347559.80	347545.00	348599.25	350446.00	351721.50	347459.00	347800.20
gr431-40	415314.00	415387.45	**415280.00**	415560.35	416983.00	419148.30	415280.00	415342.00
gr666-10	387562.00	389594.80	392586.00	395300.20	399415.00	404852.60	**387321.00**	388644.95
gr666-15	**446475.00**	447123.60	449908.00	452081.65	453684.00	459695.45	446839.00	447329.35
gr666-20	519121.00	519773.45	523090.00	525733.00	523178.00	526787.50	**519071.00**	520190.70
gr666-30	**650116.00**	650974.90	653524.00	656015.75	652608.00	655150.05	651330.00	652005.05

Table 2.7 – Comparative results on Set III between GMA and its three variants. Strictly best values are indicated in boldface.

Instance	GMA		GMA_0		GMA_1		GMA_2	
	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
pr1002-5	313885.00	314083.20	314065.00	314495.30	324126.00	327673.20	**313867.00**	313946.15
pr1002-10	**379846.00**	379911.00	380489.00	380656.65	388221.00	391193.95	**379846.00**	379920.85
pr1002-20	514968.00	515784.55	515927.00	516825.70	522573.00	524484.30	**513814.00**	515288.60
pr1002-30	**661540.00**	662613.10	663314.00	664050.80	666270.00	669233.55	**661540.00**	662284.40
pr1002-40	**803624.00**	803642.45	804971.00	805937.95	808207.00	810596.10	**803624.00**	804251.20
fnl2461-3	**105637.00**	105754.90	105793.00	105943.85	112896.00	113202.90	**105637.00**	105753.75
fnl2461-6	**116128.00**	116287.05	116531.00	116761.30	120621.00	121369.65	116173.00	116273.20
fnl2461-12	**143477.00**	143866.10	146060.00	146259.00	145953.00	146626.45	143739.00	144100.30
fnl2461-24	**221116.00**	221317.35	225527.00	226078.10	222280.00	222706.05	221167.00	221439.10
fnl2461-30	**267017.00**	267249.45	271199.00	271586.55	267799.00	268144.30	267296.00	267441.70
fnl3461-3	**148917.00**	148979.55	148957.00	149065.90	160427.00	161053.00	**148917.00**	148978.25
fnl3461-6	159934.00	160040.70	160181.00	160340.95	169106.00	169983.65	**159906.00**	160052.60
fnl3461-12	**185363.00**	185621.60	186394.00	186910.25	192212.00	192988.60	185652.00	185814.80
fnl3461-24	**263631.00**	263980.40	267515.00	267723.05	267007.00	267684.65	263763.00	264050.00
fnl3461-30	307071.00	307252.30	310275.00	310787.45	308936.00	309526.70	**306991.00**	307148.20
fnl3461-40	**384573.00**	384722.95	387335.00	387810.05	385794.00	385981.80	384611.00	384752.45
pla5397-20	38006100	38018450	38049400	38084660	38527700	38625835	**38006000**	38013320
pla5397-30	**51138000**	51143260	51297400	51353685	51294600	51385805	51141700	51148700
pla5397-40	**64097900**	64121760	64337200	64420905	64192200	64258245	64100100	64148815
pla5397-50	**73993600**	73993610	73993700	73993870	74048200	74119855	**73993600**	73993615
pla5397-60	**85266200**	85266750	85269600	85281395	85347400	85397905	85266900	85267150
pla6397-20	35951800	35997920	36298800	36344680	36502300	36650005	**35945200**	36020685
pla6397-30	**47346400**	47368155	47646200	47680965	47555300	47699290	47390500	47418730
pla6397-40	**56638000**	56653280	56881800	56921780	56828400	56884080	56661500	56688900
pla6397-50	**67161500**	67171190	67297700	67370130	67293800	67359825	67181100	67207485
pla6397-60	**74791200**	74803075	74941900	74988450	74935800	74986940	74814700	74845785
pla7397-20	**41260500**	41422195	42063000	42217895	41653000	41817975	41446600	41501520
pla7397-30	**52636900**	52780890	53655100	53801700	52953600	53082005	52810400	52893300
pla7397-40	**64937200**	65029520	66045400	66163565	65151400	65267450	64993500	65070450
pla7397-50	**76331100**	76406770	77166800	77274005	76467400	76569390	76344200	76422895
pla7397-60	**86153700**	86224380	87017600	87097055	86243600	86382840	86171600	86233840

Benefit of constrained cross-exchange

To highlight the benefit of the constrained cross-exchange (CCE, Section 2.2.3), we compared GMA with a variant GMA_0, where CCE is removed from LOE. In other words, only EAX is employed in GMA_0 in the local optima exploration component.

Computational results of GMA and GMA_0 are shown in Tables 2.6 and 2.7 and summarized in Table 2.8 and Fig. 2.5. The results indicate that GMA performs significantly better than GMA_0 in terms of f_{best} and f_{avg}. For f_{best}, GMA dominates GMA_0 by getting 42 better results out of the 45 tested instances and reporting only one worse result. Furthermore, the statistically significant difference between GMA and GMA_0 is verified by the Wilcoxon singed-rank test with a 95% level of confidence in Table 2.8. Therefore, this experiment confirms the usefulness of CCE for the GMA algorithm.

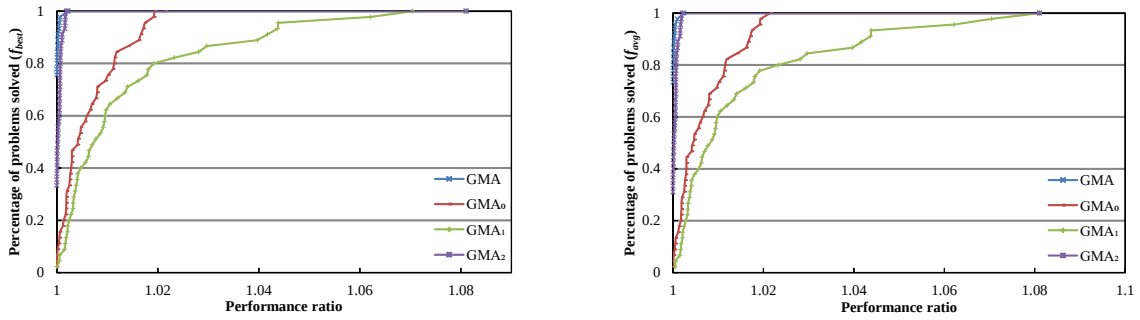
Benefit of EAX

To assess the benefit of EAX in LOE, we created another variant GMA_1 in which EAX is replaced by 2-opt [Hel00] for individual route optimization. GMA_1 shares the other ingredients of GMA.

From the results in Tables 2.6 and 2.7, we observe that GMA significantly outperforms GMA_1 on all instances, except gr202-25. For gr202-25, the best result of GMA_1 is slightly better than GMA. Furthermore, the small *p-value* (less than 0.05) in Table 2.8 from the Wilcoxon singed-rank test attests the significant difference between GMA and GMA_1. Moreover, the performance profiles of Fig. 2.5 indicate that GMA surpasses GMA_1 in

terms of f_{best} and f_{avg}. Indeed, GMA arrives at $\mathcal{Q}_x(r_f)$ firstly, much earlier than GMA_1. These observations illustrate the benefit of EAX in GMA.

Benefit of backbone-based crossover



(a) Best **(b)** Avg.

Figure 2.5 – Performance profiles of GMA and its three variants on the set of 45 selected instances.

To study the effectiveness of the backbone-based crossover, we compared GMA with a third variant GMA_2. In GMA_2, the backbone-based crossover is replaced by a crossover proposed by Singh and Baghel [SB09], which was designed for the related mTSP problem. This crossover selects one of the two parents uniformly at random and copies, from the parent to the offspring, the most promising route (i.e., the route having the smallest ratio of route length to the number of cities in that route). Then all the cities belonging to the route are deleted from both parents by connecting the predecessor of each city to its successor, and the length of the route is updated accordingly.

From the comparative results (f_{best} and f_{avg}) of GMA and GMA_2 in Tables 2.6 and 2.7, we observe that in terms of f_{best}, GMA dominates GMA_2 by acquiring 25 better results, 10 equal results and 10 worse results. The Wilcoxon signed-rank test, shown in Table 2.8, also confirms that GMA outperforms significantly GMA_2 on the large instances (set III). This experiment demonstrates that the backbone-based crossover operator contributes positively to the performance of GMA, in particular for solving large instances.

Finally, Fig. 2.5 provides other useful information for the importance of each ingredient of GMA. For example, GMA_1 performs the worst because it has the worst (smallest) $\mathcal{Q}_x(1)$ value and reaches $\mathcal{Q}_x(r_f)$ lastly. Therefore, we can summarize that EAX is the most important component of GMA, followed by CCE, finally the backbone-based crossover operator.

2.4.2 Influences of selection, pool updating and mutation

In addition to the local optimization and crossover components, the performance of a

Table 2.8 – Summary of comparative results between GMA and and its three variants

Algorithm pair	Set/Instance	Indicator	Better	Equal	Worse	$p-value$
GMA vs. GMA_0	II/14	f_{best}	11	2	1	9.77E-04
		f_{avg}	14	0	0	1.22E-04
	III/31	f_{best}	31	0	0	1.17E-06
		f_{avg}	31	0	0	1.17E-06
GMA vs. GMA_1	II/14	f_{best}	12	1	1	7.32E-04
		f_{avg}	14	0	0	1.22E-04
	III/31	f_{best}	31	0	0	1.17E-06
		f_{avg}	31	0	0	1.17E-06
GMA vs. GMA_2	II/14	f_{best}	6	4	4	3.34E-01
		f_{avg}	9	1	4	9.42E-02
	III/31	f_{best}	19	6	6	1.60E-03
		f_{avg}	23	0	8	9.94E-04

memetic algorithm such as GMA could be influenced by other factors such as parent selection, pool updating and mutation. According to our experiments with the roulette-wheel selection strategy and the rank-pool updating strategy [ZHG18], no significant changes were observed regarding the performance of the GMA algorithm. In this subsection, we focus on studying the influence of mutation. Specifically, when the best solution φ^* is not improved for $maxNoImpor$ consecutive iterations (we empirically set $maxNoImpor = 50$), the search is judged to be stagnating. Then a mutation operator is triggered to modify one third of the solutions in the population (i.e., each solution is mutated with an equal probability of 1/3). The mutation consists of displacing a certain number of randomly and uniformly chosen shared cities. To be specific, for a solution to be mutated, $|\mathcal{S}| \times 0.3$ shared cities are first removed, leading to a partial solution. Then these removed shared cities are inserted into the partial solution one by one, using the second step of the greedy randomized heuristic to minimize the distance increase. After that, each mutated solution is optimized by the local optima exploration procedure of Section 2.2.3. Comparative results of GMA and the GMA variant extended with the mutation (called GMA_3) are shown in Tables 2.9 and 2.10.

Table 2.9 – Comparative results on Set II between GMA and GMA_3 (with mutation). Strictly best values are indicated in boldface.

	GMA			GMA_3		
Instance	f_{best}	f_{avg}	σ	f_{best}	f_{avg}	σ
gr202-12	99871.00	100162.50	185.46	99871.00	100136.95	201.03
gr202-25	173477.00	173594.65	75.72	**173358.00**	173569.15	100.05
gr202-35	233871.00	234003.35	72.81	233871.00	233973.20	91.05
gr229-10	222167.00	222173.75	30.19	222167.00	222167.00	0.00
gr229-15	264146.00	264146.00	0.00	264146.00	264147.85	8.27
gr229-20	319669.00	319880.15	547.77	319669.00	319776.90	332.11
gr229-30	406701.00	407389.75	279.37	**406664.00**	407333.55	320.53
gr431-12	248447.00	248447.00	0.00	248447.00	248447.00	0.00
gr431-25	347335.00	347559.80	420.13	347335.00	347565.85	441.71
gr431-40	415314.00	415387.45	88.31	415314.00	415364.05	73.31
gr666-10	387562.00	389594.80	3417.34	**387459.00**	389350.40	2939.49
gr666-15	446475.00	447123.60	328.49	**446322.00**	447109.50	345.89
gr666-20	519121.00	519773.45	397.47	519121.00	519664.20	360.33
gr666-30	650116.00	650974.90	417.87	650116.00	650894.35	369.63
Best/All	0/14	2/14	-	**4/14**	**11/14**	-
p-value	-	-	-	1.25E-01	1.20E-03	-

The results indicate that in terms of f_{best}, GMA_3 outperforms GMA by obtaining 13 better results, 28 equal results and 4 worse results. However, the Wilcoxon signed-rank test shows that there is no statistically significant difference. On the contrary, in terms of f_{avg}, GMA_3 dominates GMA by acquiring 34 better results, 2 equal results and 9 worse results.

Table 2.10 – Comparative results on Set III between GMA and GMA_3 (with mutation). Strictly best values are indicated in boldface.

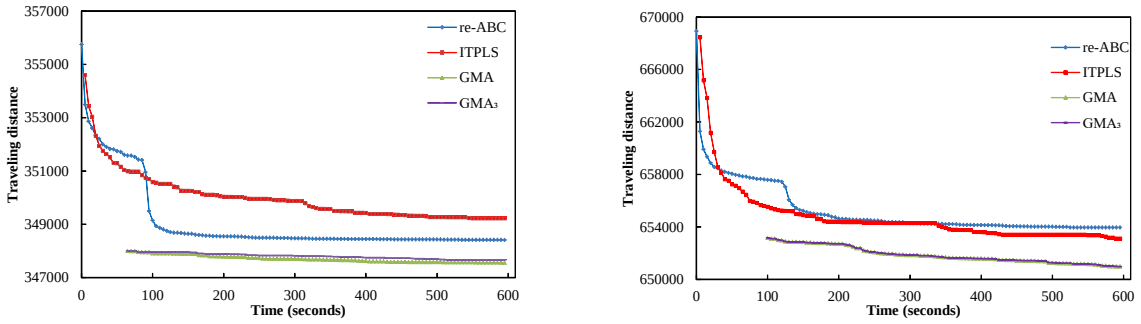
	GMA			GMA_3		
Instance	f_{best}	f_{avg}	σ	f_{best}	f_{avg}	σ
pr1002-5	313885.00	314083.20	128.79	313885.00	314106.55	127.60
pr1002-10	379846.00	379911.00	162.49	379846.00	379902.00	140.98
pr1002-20	514968.00	515784.55	498.49	**514244.00**	515655.65	586.90
pr1002-30	661540.00	662613.10	958.81	661540.00	662422.80	710.68
pr1002-40	803624.00	803642.45	74.24	803624.00	803624.00	0.00
fnl2461-3	105637.00	105754.90	43.78	105637.00	105755.90	43.10
fnl2461-6	116128.00	116287.05	77.36	116128.00	116278.25	75.96
fnl2461-12	143477.00	143866.10	214.32	143477.00	143811.60	173.59
fnl2461-24	221116.00	221317.35	121.79	**221105.00**	221299.40	110.59
fnl2461-30	267017.00	267249.45	116.69	267017.00	267230.20	102.23
fnl3461-3	148917.00	148979.55	33.13	148917.00	148979.55	33.13
fnl3461-12	159934.00	160040.70	63.09	159934.00	160035.25	65.67
fnl3461-12	185363.00	185621.60	143.67	185363.00	185605.90	136.94
fnl3461-24	**263631.00**	263980.40	177.39	263672.00	263972.35	162.72
fnl3461-30	307071.00	307252.30	113.62	**307026.00**	307233.95	122.99
fnl3461-40	384573.00	384722.95	77.96	384573.00	384720.40	79.81
pla5397-20	38006100	38018450	25354.72	38006100	38018355	25426.04
pla5397-30	51138000	51143260	3180.10	51138000	51142525	2790.75
pla5397-40	64097900	64121760	18498.09	64097900	64120030	16710.07
pla5397-50	73993600	73993610	30.78	73993600	73993605	22.36
pla5397-60	**85266200**	85266750	235.08	85266300	85266710	202.35
pla6397-20	35951800	35997920	19571.59	35951800	36000990	20031.03
pla6397-30	47346400	47368155	12197.91	47346400	47370840	12919.44
pla6397-40	56638000	56653280	11345.84	**56635600**	56653405	12491.62
pla6397-50	67161500	67171190	7667.49	**67158800**	67170200	7372.00
pla6397-60	74791200	74803075	10330.12	**74788500**	74801030	9630.71
pla7397-20	**41260500**	41422195	70887.70	41311800	41425215	57057.91
pla7397-30	**52636900**	52780890	88332.83	52672800	52781955	71085.17
pla7397-40	64937200	65029520	58826.59	**64926700**	65019290	57437.00
pla7397-50	76331100	76406770	47628.33	**76306200**	76391380	47713.59
pla7397-60	86153700	86224380	48429.67	**86121900**	86210057.89	53419.75
Best/All	4/31	7/31	-	**9/31**	**23/31**	-
p-value	-	-	-	3.31E-01	3.68E-02	-

The Wilcoxon signed-rank test confirms that GMA_3 significantly outperforms GMA. This experiment indicates that the mutation strategy can indeed improve the performance of GMA. Especially, it significantly reinforces the stability of the algorithm.

2.4.3 Convergence analysis

Finally, we investigate the convergence behaviors of GMA (and the GMA_3 variant with mutation) and two key reference algorithms (re-ABC and ITPLS). For this study, we acquired the running profiles of these algorithms on two representative instances of Set II (gr431-25, gr666-30). We ran each algorithm 20 times with the cutoff time of 600 seconds per run and recorded the best objective values during the process. The results of this experiment are shown in Fig. 2.6.

One notices first that the curves of the population-based GMA and GMA_3 do not start at time 0. This is because that these algorithms spent a non-negligible portion of the time on generating the initial population (around 60 and 100 seconds for gr431-25 and gr666-30, respectively). From Fig. 2.6, one observes that re-ABC and ITPLS improve their solution quality quickly at the beginning of the search, and slow down or even stagnate as the time going. For GMA and GMA_3, the population initialization step allowed them to start the search with high-quality solutions. The best solution in the population is continually updated when the time goes on, implying that GMA and GMA_3 can better benefit from the allowed time to improve their solutions.



(a) **(b)**

Figure 2.6 – Convergence charts (running profiles) of re-ABC, ITPLS, GMA and GMA_3 for solving two representative instances (gr431-25 and gr666-30). The results were obtained from 20 independent executions of each compared algorithms

2.5 Chapter conclusion

In this chapter, we presented the first grouping memetic algorithm for solving the CTSP. The algorithm relies on a specific backbone-based crossover to generate promising offspring solutions by solution recombination and a powerful local optima exploration for offspring improvement. Extensive computational results on three sets of 65 benchmark instances in the literature indicate that our algorithm is very competitive compared with existing leading algorithms. In particular, it reports 38 new upper bounds while matching 24 best-known results. We also investigated the interest of CPLEX for solving the CTSP and reported 10 proven optimal solutions for the first time. Furthermore, we analyzed the impacts of the main components of the algorithm on its performance.

In the next chapter, we will introduce a memetic algorithm for solving the minmax multiple traveling salesmen problem.

Chapter 3

Memetic search for the minmax mTSP with single and multiple depots

In this chapter, we propose a unified memetic approach to solving both cases of the minmax mTSP and the minmax multidepot mTSP. The proposed algorithm features a generalized edge assembly crossover to generate offspring solutions, an efficient variable neighborhood descent to ensure local optimization as well as an aggressive post-optimization for additional solution improvement. Extensive experimental results on 77 minmax mTSP benchmark instances and 43 minmax multidepot mTSP instances commonly used in the literature indicate a high performance of the algorithm compared to the leading algorithms. Additional experimental investigations are conducted to shed light on the rationality of the key algorithmic ingredients. The content of this chapter is based on an article submitted to *European Journal of Operational Research*.

3.1 Introduction

Due to the practical relevance and computational challenge of these mTSP problems, a number of solution methods have been developed. According to the review of Section 1.3.2, the existing methods are based on general frameworks such as evolutionary algorithms, bio-inspired methods and local searches. These methods have contributed to continually improve the state-of-the-art of solving these problems. Meanwhile, their performances vary typically according to the difficulty and scale of the problem instances. Moreover, existing methods have been developed for either the minmax mTSP or the minmax multidepot mTSP. In this chapter, we present a unified population-based memetic algorithm (MA) able to effectively deal with both the minmax mTSP and the minmax multidepot mTSP. The contributions of this work are summarized as follows.

1. The proposed MA algorithm features several complementary search components. First, it integrates a generalized edge assembly crossover to generate offspring solutions, which is inspired by the well-known EAX crossover for the TSP [NK97; NK13]. Second, MA uses an efficient variable neighborhood descent (with streamlining techniques) to improve offspring solutions. Third, the algorithm adopts an aggressive post-optimization procedure to further optimize some particularly promising offspring solutions.
2. The MA algorithm reports record-breaking best results (new upper bounds) for a number of benchmark instances commonly used in the literature. These new results are useful for future research on these problems and performance assessments of new algorithms.
3. We provide the code of the proposed algorithm, which can be used by researchers and practitioners to solve various problems that can be recast to the minmax mTSP or the minmax multidepot mTSP.

Next section provides detailed description of the MA algorithm. Section 3.3 is dedicated to computational results on benchmark instances and comparisons with the literature. Key components of the algorithm are investigated in Section 3.4. Section 3.5 draws conclusions.

3.2 Problem solving methodology

Memetic search is a general hybrid search framework based on population-based genetic search and neighborhood-based local optimization [NCM12]. The basic rationale behind this approach is take advantage of these two complementary search strategies [Hao12]. Indeed, population-based search offers more facilities for exploration while local optimization provides convenient means for exploitation. A suitable combination of these two types of methods would lead to a good balance between exploitation and exploration of the search process.

Population-based evolutionary algorithms have been successfully applied to the TSP [NK97; NK13], capacitated vehicle routing problem (CVRP) [NB09; Vid22] and its variants [NBD10; Vid+13; Vid+14]. In this work, we present an original memetic algorithm (MA) for solving both the minmax mTSP and the minmax multidepot mTSP. The algorithm integrates a population initialization procedure, a generalized edge assembly crossover (mEAX), a variable neighborhood descent (VND), a post-optimization and a population management procedure. Among these search components, we identify the mEAX crossover and the post-optimization as the most innovative while VND features a streamlining techniques to accelerate its search.

The general scheme of the MA algorithm is illustrated in Algorithm 4. The algorithm starts with a population of initial solutions (or individuals) generated by the population initialization procedure (Line 3, Algorithm 4). After recording the best solution φ^* found so far (Line 4), the algorithm performs a number of generations to evolve the population (Lines 5-16). For this, it applies the dedicated mEAX crossover (Line 7) to combine two random parent solutions, yielding γ (a parameter) new offspring solutions. Then each offspring solution is first improved by the VND procedure (Line 9) and then conditionally further improved by the post-optimization (Lines 10-13). The post-optimization is applied only to elite offspring solutions with a quality better than the best recorded solution φ^*. Finally, each improved offspring solution is considered by the population management procedure to update the population (Line 14). The algorithm stops and returns the best solution found φ^* when a predefined stopping condition is reached, which is either a maximum cutoff time or a maximum number of iterations. In the later case, one iteration corresponds to one call to the (expensive) VND procedure at Line 9 of Algorithm 4.

Algorithm 4 Pseudo code of the memetic algorithm

1: **Input**: Problem instance I with a minimization objective f, population size μ, number of offspring γ;
2: **Output**: The best solution φ^* found;
3: $\mathcal{P} = \{\varphi_1, \varphi_2, \cdots, \varphi_\mu\} \leftarrow$ *PopulationInitial* (I);/*Section 3.2.1*/
4: $\varphi^* \leftarrow \arg\min\{f(\varphi_i) : i = 1, 2, \cdots, \mu\}$;/*$\varphi^*$ records the best solution found*/
5: **while** *Stopping condition is not met* **do**
6: $\quad\{\varphi_A, \varphi_B\} \leftarrow$ *ParentSelection*$(\mathcal{P})$;/*Random parent selection*/
7: $\quad\{\varphi_O^1, \varphi_O^2, \cdots, \varphi_O^\gamma\} \leftarrow mEAX(\varphi_A, \varphi_B, \gamma)$;/*To generate γ offspring solutions, Section 3.2.2*/
8: $\quad$**for** $i = 1$ to γ **do**
9: $\quad\quad\varphi_O^i \leftarrow VND(\varphi_O^i)$;/*To improve each offspring solution, Section 3.2.3*/
10: $\quad\quad$**if** $f(\varphi_O^i) < f(\varphi^*)$ **then**
11: $\quad\quad\quad\varphi_O^i \leftarrow$ *PostOptimization*(φ_O^i);/*To further improve each elite offspring solution, Section 3.2.4*/
12: $\quad\quad\quad\varphi^* \leftarrow \varphi_O^i$;
13: $\quad\quad$**end if**
14: $\quad\quad\mathcal{P} \leftarrow$ *PoolUpdating*$(\mathcal{P}, \varphi_O^i)$;/*Section 3.2.5*/
15: $\quad$**end for**
16: **end while**
17: **return** φ^*;

3.2.1 Generation of the initial population

The MA algorithm starts its search from a population $\mathcal{P}$ of μ initial solutions. The construction process of each solution is composed of three steps. First, m tours are initialized with the depot. For the minmax multidepot mTSP, each salesman is located at one of the depots, and the tour is initialized by its corresponding depot. Second, a random unassigned city is selected and inserted into the shortest tour at the position with the least length increase of this tour. When all cities are assigned, a feasible solution is obtained. Finally, the solution is improved by the VND procedure (Section 3.2.3) and then added into the population. The initialization procedure stops when μ solutions are obtained.

3.2.2 The mEAX crossover based on edge assembly

The conventional edge assembly crossover operator (EAX) was first presented for solving the TSP [NK97; NK13]. It was subsequently applied to the CVRP [NB09] and the vehicle routing problem with time windows (VRPTW) [NBD10]. In this work, we introduce mEAX, which generalizes the idea of EAX to the minmax mTSP and the minmax multidepot mTSP. It is worth noting that these mTSPs are different from the TSP, CVRP and VRPTW. As such, the proposed mEAX crossover must consider the specific features of the minmax mTSP problems.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a candidate solution φ for the minmax mTSP or minmax multidepot mTSP corresponds to a partial graph $\mathcal{G}_\varphi = (\mathcal{V}, \mathcal{E}_\varphi)$, where $\mathcal{E}_\varphi$ is the set of edges traversed by φ. Let φ_A and φ_B be two parent solutions. Let $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ be the corresponding partial graphs. The proposed mEAX crossover consists of the following six steps (see Algorithm 5 for the general procedure and Fig. 3.1 for an illustrative example).

Algorithm 5 Procedures of mEAX for the minmax mTSP

1: **Input**: Parent φ_A and φ_B, parameter γ;
2: **Output**: γ offspring solutions;
3: Construct $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}) \backslash (\mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}))$;
4: Generate *AB-cycles* by decomposing $\mathcal{G}_{\mathcal{AB}}$;
5: Construct *E-sets* from *AB-cycles* with the block strategy;
6: Generate intermediate solutions by removing edges in $\varphi_A \cap$ *E-sets* from φ_A and adding edges in $\varphi_B \cap$ *E-sets*;
7: Split giant tours in intermediate solutions for the minmax multidepot mTSP;
8: Eliminating isolated subtours in intermediate solutions to generate feasible solutions;
9: Select γ best feasible solutions;
10: **return** γ *offspring solutions*;

1. Creation of a joint graph $\mathcal{G}_{\mathcal{AB}}$. From the parent solutions φ_A and φ_B, the joint graph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}) \backslash (\mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}))$ is constructed by the symmetric difference of $\mathcal{E}_\mathcal{A}$ and $\mathcal{E}_\mathcal{B}$.

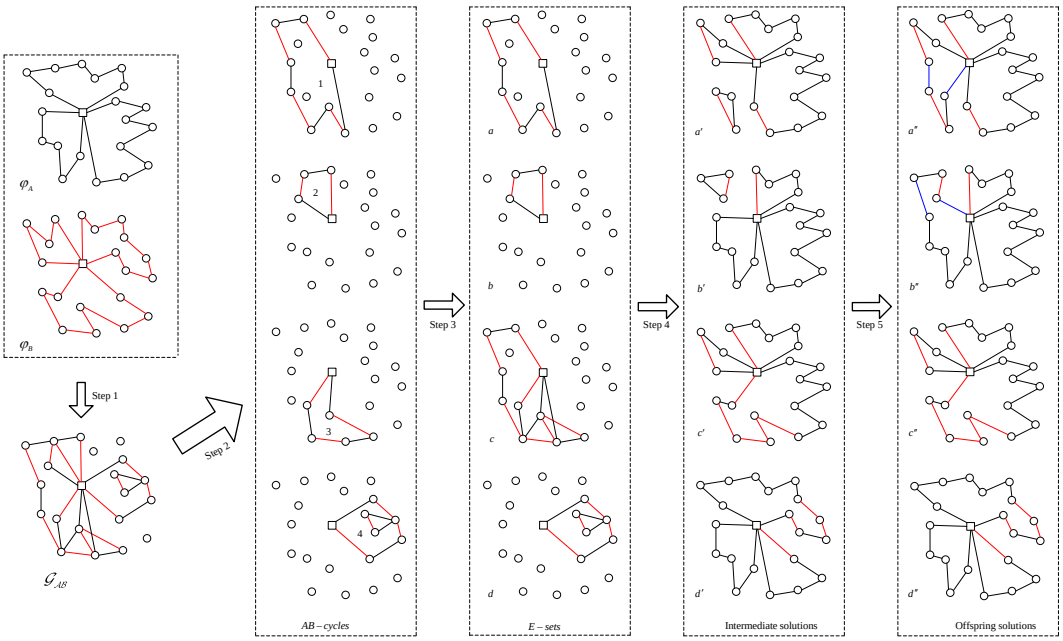


Figure 3.1 – Illustration of the mEAX crossover steps of the minmax mTSP

2. Generation of *AB-cycles*. Given the joint graph $\mathcal{G}_{\mathcal{AB}}$, a number of *AB-cycles* are generated where each new *AB-cycle* is constructed as follows. A random vertex associated with its edges from $\mathcal{G}_{\mathcal{AB}}$ is selected to initialize an *AB-cycle*. Then the edges of $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$ are traced alternatively to extend the ongoing *AB-cycle*. When the add of a new edge leads to a closed cycle and the number of edges is even, the *AB-cycle* is formed successfully. All the edges belonging to the *AB-cycle* are removed from $\mathcal{G}_{\mathcal{AB}}$ before building the next *AB-cycle*. This process continues until $\mathcal{G}_{\mathcal{AB}} = \emptyset$ and returns all *AB-cycles* obtained.

3. Generation of *E-sets*. From the set of *AB-cycles*, the block strategy is used to generate the so-called *E-sets*. If two *AB-cycles* share at least one vertex (e.g., *AB-cycles* 1 and 3 in Fig. 3.1), these two cycles are combined to generate the *E-set*. In the example of Fig. 3.1, the four *AB-cycles* should be combined to form one single *E-set* since the depot is shared. However, for illustrative purpose of steps 4 and 5 blow, we suppose there are four *E-sets* as showed in Fig. 3.1.

4. Generation of intermediate solutions. For each *E-set* (say $\mathcal{E}_i$), an intermediate solution φ_i' is created based on φ_A by removing from it the edges of $\mathcal{E}_{\mathcal{A}}$ shared with $\mathcal{E}_i$ and adding the edges of $\mathcal{E}_{\mathcal{B}}$ shared with $\mathcal{E}_i$, i.e., $\varphi_i' = (\mathcal{E}_{\mathcal{A}} \setminus (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{A}})) \cup (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{B}})$. This strategy ensures the preservation of all common edges of φ_A and φ_B in the intermediate solution. Furthermore, all edges in an intermediate solution exclusively come from the parent solutions.

5. Elimination of giant tours. For the minmax multidepot mTSP, giant tours that visit more than one depot, may occur in intermediate solutions (e.g., the tour in the

intermediate solution in Fig. 3.2 includes two depots represented by squares). These giant tours are split by the 2-opt* operator [PR95]. If a giant tour visits k depots, two new Hamiltonian tours are first generated by the 2-opt* operator, where one of the two new tours only visits one single depot while the other tour visits the remaining k-1 depots. We repeat this split operations k-1 times until k new Hamiltonian tours are generated. During the split process, the objective is to make the new tours have similar length and avoid too longer tours. For the giant tour with two depots in Fig. 3.2 (lower part of the intermediate solution), it includes two segments (each segment refers to the set of cities between two depots). The 2-opt* operator works as follows. Two edges (dash lines) from the two segments based on the α-nearness technique (Section 3.2.3) are replaced to create two new single depot tours such that the length of the new shorter tour is as close as possible half of the giant tour. We thus obtain two feasible tours which have similar lengths.

6. Elimination of isolated subtours. Isolated subtours may appear in intermediate solutions (e.g., the two triangle tours in intermediate solutions a' and b' in Fig. 3.1). We apply the 2-opt* approach to accommodate the particular feature of our problem as follows. For each isolated subtour, its adjacency tours are defined if a vertex u is an α neighbor (Section 3.2.3) of vertex v visited by the subtour. Then, the merges of the subtour into its adjacency tours are evaluated by 2-opt* and the best merge leading to the shortest tour is performed. Once all isolated subtours are eliminated, a feasible offspring solution composed of m distinct Hamiltonian tours is obtained (see the last sub-figure in Fig. 3.1).

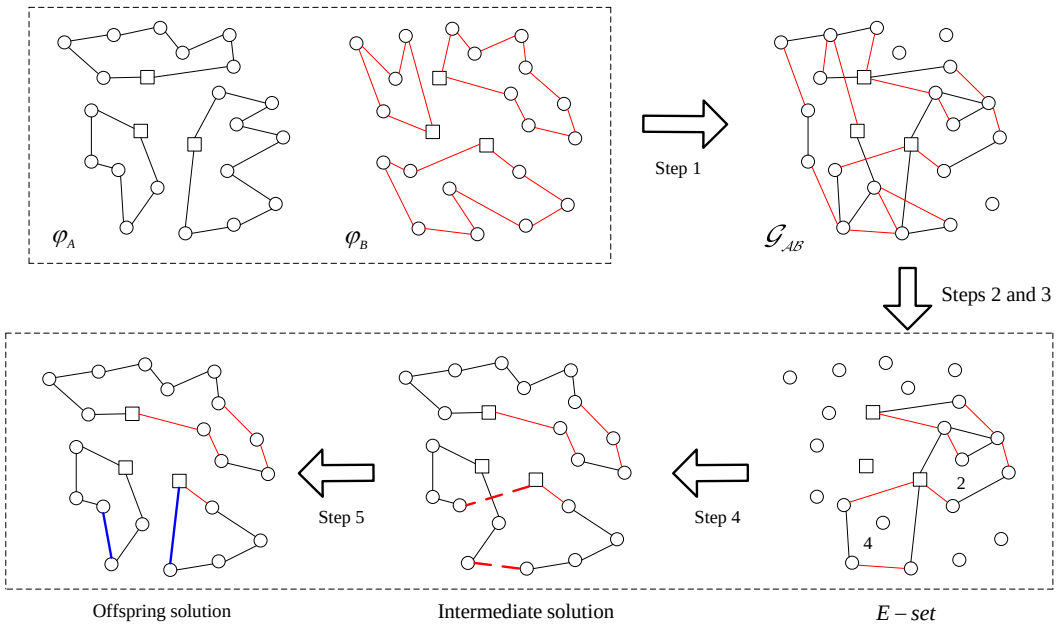


Figure 3.2 – Illustration of the mEAX crossover steps of the minmax multidepot mTSP

One notes that mEAX differs from EAX by the last two steps because contrary to

the TSP and the CVRP, giant tours may appear in the case of the minmax multidepot mTSP.

The above mEAX process typically generates numerous offspring solutions, many of them being of bad quality and thus uninteresting. Given that the subsequent VND local optimization (Section 3.2.3) is time consuming, we filter out non-promising offspring solutions with a mediocre quality to retain only the γ (a parameter) best offspring solutions for solution improvement.

The mEAX crossover for the minmax mTSP and minmax multidepot mTSP follows the idea of the EAX operator initially designed for the TSP [NK97; NK13]. Meanwhile, adaptations are necessary to take into account the particular features concerning the minmax objective and the presence of possible multiple depots. The main adaptations concern the processing of giant tours and isolated subtours in intermediate solutions (steps (5) and (6)).

Our way of handling isolated subtours is similar to the technique presented in [NB09] where EAX is adapted to the CVRP. In [NB09], isolated subtours are eliminated by testing all possible combinations with the 2-opt* heuristic [PR95] and performing the best combination which minimizes the total traveling distance. In mEAX, since the objective is to minimize the longest tour instead of total traveling distance, the 2-opt* heuristic is applied with this specific minimization objective. Furthermore, for the minmax multidepot mTSP, giant tours which include two or more depots may occur in intermediate solutions due to the presence of multiple depots. This feature cannot be resolved by the conventional EAX operator. In mEAX, the 2-opt* based spitting strategy is introduced to split each giant tour into feasible tours while keeping the longest tour as short as possible. In sum, the mEAX crossover renders the idea of the EAX operator applicable to routing problems with the minmax objective.

Since a minmax mTSP solution contains $n + m - 1$ edges, the space complexity of mEAX is bounded by $\mathcal{O}(n + m)$. During the first four steps, $2 \times (n + m - 1)$ edges are involved, and the time complexity is bounded by $\mathcal{O}(n+m)$. In step 5, suppose that there are g giant tours and the cycle with the largest number of edges includes $|\mathcal{E}_g|$ edges, the time complexity is bounded by $\mathcal{O}(|\mathcal{E}_g| \times \alpha)$. Furthermore, suppose that there are h isolated tours and the longest tour includes $|\mathcal{E}_h|$ edges, the time complexity of step 6 is bounded by $\mathcal{O}(|\mathcal{E}_h| \times \alpha)$.

3.2.3 Variable neighborhood descent

Variable neighborhood descent (VND) [MH97] is a general local search approach which has been applied successfully to a number of routing and TSP-like problems [ISW09; Soy15; Tod+17; WCL17]. VND explores local optima with several ordered neighborhoods N_θ ($\theta = 1, 2, \cdots, \theta_{max}$). VND starts its descent search from the first neighborhood and switches to the next neighborhood once a local optimum is reached. When neighborhood N_θ ($\theta > 1$) is examined, VND switches to the first neighborhood N_1 immediately if a better solution is found; otherwise when neighborhood N_θ ($\theta > 1$) is exhausted, VND

moves to the next neighborhood $N_{\theta+1}$. Once the last neighborhood $N_{\theta_{max}}$ is exhausted and no better solution can be found, VND stops and returns the last local optimum. In this work, we use VND to exploit six neighborhoods, where two neighborhoods (2-opt* and κ-opt) are employed to solve the minmax mTSP and the minmax multidepot mTSP for the first time. To speed up neighborhood examination, two new data structures are introduced to accelerate the search process of VND.

Neighborhoods

The six neighborhoods adopted in this work include five inter-tour neighborhoods and one intra-tour neighborhood. Let $r(\pi)$ denote the tour containing vertex π in the incumbent solution. Let vertex δ be a neighbor of vertex π, and vertices x and y the successor of π in $r(\pi)$ and δ in $r(\delta)$, and (π_a, π_b) a substring from π_a to π_b. To avoid the examination of non-promising candidate solutions, we use the α-nearness technique [HH22; Hel00] and consider, for a vertex π, only α neighbor vertices. The six neighborhoods are given by the following move operators M1-M6.

M1: If $r(\pi) \neq r(\delta)$ and $r(\pi)$ is the longest tour r_l, then remove π and place it after δ.

M2: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then, swap π and δ.

M3: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then replace (π, x) and (δ, y) by (π, y) and (δ, x).

M4: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then swap two sequencing substrings (π_a, π_b) and (δ_a, δ_b).

M5: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then swap a sequencing substring (π_a, π_b) and a reversing substring (δ_b, δ_a).

M6: This is an intra-tour optimization operator to improve a standard TSP tour. Each tour is refined by the κ-opt heuristic [LK73], which was previously used in several best heuristics for related routing problems [AGS19; AS19a; LBW21]. In this work, the upper limit of κ is set to four.

M1 corresponds to *insertion* or *relocation*, while M2 is called *swap*. M3 is the 2-opt* inter-tour move [PR95]. M4 and M5 correspond to the cross-exchange operator, where two substrings from two tours are exchanged [HH22; Tai+97]. The cross-exchange operator generalizes M1 and M2, and has been successfully used to solve the minmax mTSP [HH22]. In this work, we limit the maximum length of each substring in M4 and M5 to β (a parameter).

It is worth mentioning that M6 is used for the first time in this work and M3 was independently used in [Zhe+22b], while the other moves were previously applied to the minmax mTSP (e.g., [HH22; Kar+21; Soy15; WGW15; WCL17]). For the minmax multi-depot mTSP, it is to be noted that M3 cannot be used because each salesman must start and end at the same depot. Therefore, when solving the minmax multidepot mTSP, M3 is disabled from the VND procedure. Furthermore, this is the first time that M4-M6 are adopted to solve the minmax multidepot mTSP.

Auxiliary data structures

In order to enhance the computational efficiency of our VND procedure, we introduce two auxiliary arrays to store useful information regarding each city.

$A1$: A one-dimensional array of length n. It stores the variation of distance of the current tour when a vertex is removed from the tour. For example, $A1[\pi]$=-100 means that if vertex π is removed from tour r_a, the length of tour r_a is shortened by 100.

$A2$: A two-dimensional array of size $n \times n$. It stores the variation of distance of the tour when vertex π is inserted after vertex δ. For example, $A2[\pi][\delta]$=100 indicates that if π is placed after δ in tour r_a, the length of tour r_a is increased by 100.

In general, a neighboring solution can be obtained from the incumbent solution by exchanging several edges. Therefore, most edges in the incumbent solution are common with its neighboring solutions. This insight has been used to design static move descriptors for several vehicle routing problems [AV21; Bee+18; ZK10]. For the minmax mTSP, these two auxiliary arrays ($A1$ and $A2$) enable the VND procedure to avoid unnecessary redundant calculations. As shown in Fig. 3.3, city δ_b is removed from tour r_a and placed after δ_a in tour r_b. Therefore, we can easily compute the length of r'_a and r'_b as follows: $f(r'_a) = f(r_a) + A1[\delta_b]$ and $f(r'_b) = f(r_b) + A2[\delta_b][\delta_a]$. After placing δ_b after δ_a in tour r_b, only five values in $A1$ and $3 \times n$ values in $A2$ need to be updated, respectively. In general, the time complexity of updating $A1$ and $A2$ is $\mathcal{O}(n)$.

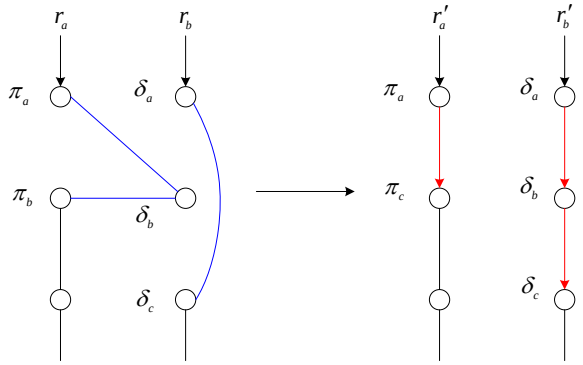


Figure 3.3 – Illustration of M1 move

In the VND procedure, these two auxiliary arrays are used to speed up the calculations of M1 and M2. Furthermore, the ejection chain operator, introduced in Section 3.2.4, also benefits from these data structures to accelerate the neighborhood examination.

3.2.4 Post-optimization

In addition to the above mEAX crossover and the VND procedure, the proposed MA algorithm includes an original post-optimization phase to further improve the quality of each global best offspring solution. The main purpose of the post-optimization is to perform an intensified search around each elite offspring solution to find possible still

better solutions. This post-optimization phase is ensured jointly by an ejection chain operator (EC) and the conventional EAX heuristic for the TSP (denoted by EAX-TSP hereafter) [NK97; NK13].

As shown in Algorithm 6, the post-optimization applies first the EC operator to improve the solution by displacing cities among different tours. A binary array T is employed to record the tours that are modified during the EC phase, such that $T[i] = 1$ $(i = 1, ..., m)$ if the ith tour is changed by EC. Then for each modified tour, the EAX-TSP heuristic is applied to shorten its distance. When neither EC nor EAX-TSP can improve the incumbent solution φ, the post-optimization stops and returns the best solution.

Algorithm 6 Pseudo code of the post-optimization procedure

1: **Input**: A solution φ;
2: **Output**: The best solution φ found;
3: $fit \leftarrow$ M;/*M is a big number*/
4: **while** $fit > f(\varphi)$ **do**
5: $\quad fit \leftarrow f(\varphi)$;
6: $\quad < \varphi, T > \leftarrow$ EC(φ);/*Ejection chain*/
7: $\quad \varphi \leftarrow$ EAX-TSP(φ, T);
8: **end while**
9: **return** φ;

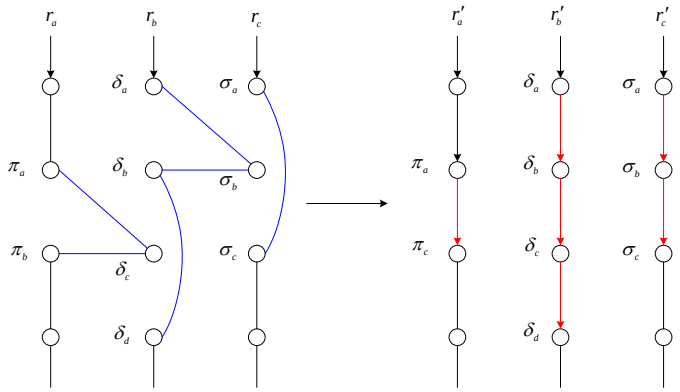


Figure 3.4 – Illustration of the ejection chain with two relocations

The ejection chain approach has been used to perform inter-tour optimization for the CVRP [AV21; AS19a]. We adopt the same approach for the first time to handle the minmax mTSP. Using Fig. 3.4 where the incumbent solution is composed of three tours, we illustrate the EC process as follows. EC starts by greedily relocating a city δ_c from the longest tour r_a into another tour r_b. This relocation operation is followed by the relocation of another city σ_b from the extended tour r_b into another tour r_c, where r_a and r_c may be same. This process continues until a maximum number of relocation moves is reached.

The EC approach is based on the following observation. Single relocation moves between two tours may increase the length of the longest tour. For example, relocating a

city from the longest tour r_a into r_b shortens r_a, but may increase tour r_b such that r_b becomes the longest tour with a distance longer than r_a. However, if we perform immediately another move to relocate a city from tour r_b into tour r_c, then it is possible that the longest tour of the solution is definitively shorten.

Function (3.1) illustrates the calculation of the move gain of an EC move based on the two auxiliary arrays introduced in Section 3.2.3, where b, c, s, t and q are indexes of r_b, r_c, the second, third and fourth longest tours, respectively.

$$\begin{aligned}
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_s)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \varnothing \\
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_t)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \{s\} \\
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_q)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \{s,t\} \\
&f(r_a^{'}) = f(r_a) + A1[\delta_c] \\
&f(r_b^{'}) = f(r_b) + A2[\delta_c][\delta_b] + A1[\sigma_b] \\
&f(r_c^{'}) = f(r_c) + A2[\sigma_b][\sigma_a]
\end{aligned} \tag{3.1}$$

Based on the M1 move introduced in Section 3.2.3, if the number of relocation is 1, the time complexity is $\mathcal{O}(n \times \alpha)$. When we continue the EC move by performing the second relocation, the time complexity becomes $\mathcal{O}((n \times \alpha)^2)$. To keep the time complexity at an acceptable level, we limit the number of relocations to 2 in this work.

One notes two differences between the EC move applied to the CVRP [AV21; AS19a] and the EC move applied in this study. First, the EC operator in our case does not need to consider the capacity constraint. Second and more importantly, even if the move gain of an EC move can be obtained in $\mathcal{O}(1)$ time in both cases, the practical computation in our case is more complicated. Indeed, for the CVRP, the move gain is simply obtained by adding up the values of $A1$ and $A2$, which themselves can be computed efficiently with the static move descriptor technique [AV21]. In our case, the static move descriptor is no more available and furthermore as shown in Eq. (3.1), the EC move gain evaluation needs to consider the second, third and fourth longest tours.

After the EC phase, the EAX-TSP heuristic [1] is triggered to optimize each individual tour that has been modified by the EC procedure. Each EAX-TSP optimization stops when the difference between the fitness of the best solution and the average fitness of individuals in the population is less than 0.01. The reason to choose the EAX-TSP heuristic is that it can effectively optimize each tour to being optimal or near-optimal in a very short time.

3.2.5 Population updating

The population updating mechanism is known to be a key component of successful memetic algorithms [Hao12]. The proposed algorithm adopts the variable population scheme presented in [Vid22; Vid+14].

The population $\mathcal{P}$ contains between μ and $\mu + \lambda$ individuals, where parameter μ is the minimum size and parameter λ is the generation size. Unlike [Vid22], clone solutions are

1. The code of EAX-TSP is available at: https://github.com/sugia/GA-for-TSP

not permitted to join the population. In each generation of MA, offspring solutions φ_O^i are progressively added to the population (Line 14, Algorithm 4). Once the population reaches $\mu + \lambda$ individuals, the survivors selection is used to eliminate λ individuals based on their contributions to the diversity of the population. The biased fitness of each individual is calculated with respect to its fitness and diversity rank in the population.

Furthermore, if the global best solution is not improved during η consecutive iterations, the algorithm is considered to be stagnating in deep local optima. In this case, diversity is introduced into the population as follows. The survivors selection phase is triggered to reduce the number of individuals in $\mathcal{P}$ to μ individuals. Then, $\mu/2$ individuals of the population are randomly and uniformly selected and replaced by new solutions generated by the initial population procedure of Section 3.2.1

3.3 Computational results and comparisons

This section is dedicated to an extensive evaluation of our MA algorithm and comparisons with state-of-the-art algorithms. Three sets of benchmark instances are used in our experiments (see Section 1.3.3): Sets I and II for the minmax mTSP and Set III for the minmax multidepot mTSP.

3.3.1 Experimental protocol and reference algorithms

Parameter setting. The MA algorithm has six parameters: population size μ, generation size λ, number of the best offspring solutions γ, neighborhood reduction parameter α, substring size β, maximum consecutive iterations (η) without an improvement. To calibrate these parameters, we employed the automatic parameters tuning package Irace [Lóp+16]. The tuning was performed on 8 instances with 150-1655 cities for the minmax mTSP and 10 instances with 100-500 cities for the minmax multidepot mTSP. The tunning budget was set to be 2000 runs. Table 3.1 shows, for each parameter, the interval of values tested by Irace, and the best value returned by the method. For the experiments presented hereafter, we used consistently these parameter values, which can be considered to be the default setting of the MA algorithm.

Table 3.1 – Parameters tuning results

Parameters	Section	Description	Considered values	Final values	
				mTSP	multidepot mTSP
μ	3.2.1	population size	{10,15,20,25,30}	30	30
λ	3.2.5	generation size in $\mathcal{P}$	{0,5,15,20,25,30}	20	15
γ	3.2.2	number of the best offspring	{1,2,3,4,5,6,7}	1	5
α	3.2.3	neighborhood reduction	{10,15,20,25,30}	15	10
β	3.2.3	substring size	{1,2,3,4,5,6,7}	4	7
η	3.2.5	maximum iterations without improvement	{2000,4000,6000,8000,10000,12000}	4000	2000

Reference algorithms. For the minmax mTSP, five algorithms (IWO [PS15], MASVND [WCL17], ES [Kar+21], HSNR [HHW21] and ITSHA [Zhe+22b]) represent the state-of-the-art for solving the problem. In [HHW21], the authors thoroughly assessed HSNR, IWO

and MASVND on the same computing platform as used in this work. The executable code of ES [Kar+21] and the source code of ITSHA [Zhe+22b] were kindly provided by their authors. For the minmax multidepot mTSP, the MD and VNS algorithms from [WGW15] are the leading algorithms in the literature (their codes are unavailable). Thus, the results of these algorithms (obtained on a computer with an Intel Pentium CPU of a 2.2GHz processor) are used as reference values to evaluate the performance of the MA algorithm. According to [WGW15], both MD and VNS terminate after five consecutive iterations without an improvement.

Experimental setting and stopping condition. The MA algorithm was written in C++ and compiled using the g++ complier with the -O3 option [2]. All experiments were conducted, like [HH22], on a computer with a Xeon E5-2670 processor of 2.5GHz CPU and 8GB RAM running Linux.

To make the comparison as fair as possible, for the minmax mTSP, we ran 20 times our MA algorithm and the executable code of the reference algorithm ES on our machine to solve each instance under the cutoff limit of $(n/100)\times 4$ minutes per run (this is the same stopping condition used in [HH22] to assess IWO, MASVND and HSNR). For the other reference algorithms (IWO, MASVND, HSNR), we cite the results reported in [HH22], which were obtained on the same computer used in this work. For the minmax multidepot mTSP, MA terminates when it reaches a maximum of 30,000 iterations.

3.3.2 Computational results and comparison

To compare MA and the reference algorithms, we report a summary of the results in Table 3.2 and the detailed results in the Appendix. The 'BKS' values show the best-known results compiled from the literature. To check the statistically significant difference between MA and each reference algorithm, the Wilcoxon signed-rank test is applied. With a confidence level of 0.05, a *p-value* lower than 0.05 indicates a significant difference.

Results on the minmax mTSP

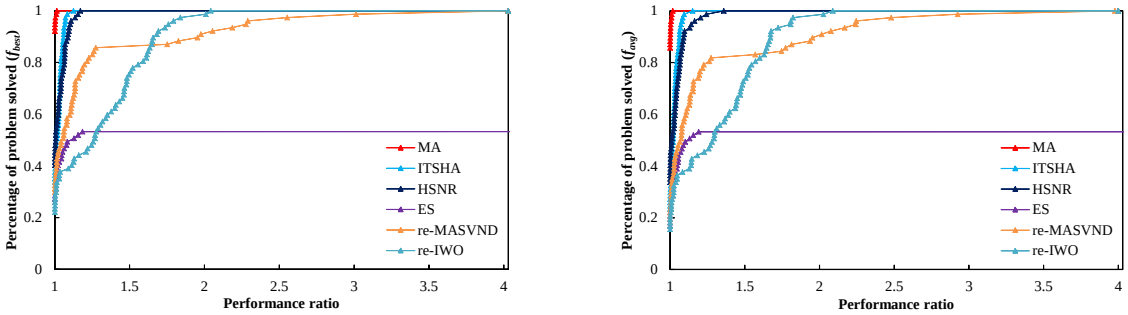
The comparative results on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP are shown in Tables 6.1 and 6.2 with the summary information in Table 3.2, where re-IWO and re-MASVND are the re-implemented IWO [PS15] and MASVND [WCL17] algorithms in [HH22]. According to these tables, the MA algorithm outperforms the five reference algorithms by achieving the best result for the vast majority of the instances. MA improves the best-known solutions of 44 instances, and matches the best-known solutions of 27 other instances. Furthermore, in terms of the average result, MA also outperforms the reference algorithms. Specifically, for $n \leq 100$, MA and the reference algorithms perform similarly in terms of f_{best}. For $n \geq 150$, MA outperforms the other algorithms (improvement gap up to 8.72%). As the number of cities increases, the difference becomes more significant, especially for the instances with few tours (e.g., $m = 3, 5$). The small *p-values* from the

2. https://github.com/pengfeihe-angers/minmax-mTSP.git

Table 3.2 – Summary of comparative results between MA and reference algorithms on the three sets of 120 instances. Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP and Set $\mathbb{M}$ for the minmax multidepot mTSP.

Instances	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Set $\mathbb{S}$ (41)	MA vs. BKS	16	23	2	1.37E-03	-	-	-	-
	MA vs. ITSHA [Zhe+22b]	19	21	1	2.93E-04	23	13	5	2.25E-04
	MA vs. HSNR [HHW21]	18	22	1	8.37E-04	22	13	6	7.51E-04
	MA vs. ES [Kar+21]	20	20	1	9.22E-05	28	9	4	7.61E-06
	MA vs. re-MASVND	20	20	1	1.23E-04	27	13	1	6.53E-06
	MA vs. re-IWO	24	17	0	1.82E-05	29	12	0	3.52E-06
Set $\mathbb{L}$ (36)	MA vs. BKS	28	3	5	2.50E-06	-	-	-	-
	MA vs. ITSHA [Zhe+22b]	32	3	1	5.91E-07	36	0	0	1.68E-07
	MA vs. HSNR [HHW21]	28	3	5	2.50E-06	29	2	5	3.18E-06
	MA vs. re-MASVND	33	3	0	5.39E-07	35	1	0	2.48E-07
	MA vs. re-IWO	36	0	0	1.68E-07	36	0	0	1.68E-07
Set $\mathbb{M}$ (43)	MA vs BKS	39	1	3	3.15E-08	-	-	-	-
	MA vs. MD [WGW15]	40	1	2	2.28E-08	-	-	-	-
	MA vs. VNS [WGW15]	41	0	2	2.04E-08	-	-	-	-

Wilcoxon signed-rank test confirm the statistically significant difference between MA and the reference algorithms for the best and average values.



(a) Best **(b)** Avg.

Figure 3.5 – The minmax mTSP: performance profiles of MA and the five reference algorithms on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

In Fig. 3.5, the average gap of MA and the five reference algorithms are analyzed through their performance profiles. Intuitively, MA dominates the reference algorithms in terms of both the best and average results. Indeed, MA has a much higher $\mathcal{Q}_s(1)$, meaning that it finds better or equal results for nearly all instances. Furthermore, MA reaches 1 firstly, which indicates MA has a higher robustness.

Results on the minmax multidepot mTSP

Tables 3.2 and 6.3 show the results of MA as well as the two reference algorithms (MD [WGW15] and VNS [WGW15]) on the 43 instances of Set $\mathbb{M}$. According to the results, MA dominates the reference algorithms by providing 39 new best-known solutions. Only

for three instances, MA obtains slightly worse results. The small *p-values* ($\ll 0.05$) also confirm the statistically significant differences between MA and the compared algorithms. The performance profiles in Fig. 3.6 illustrate that MA has a much higher $\mathcal{Q}_s(1)$ and $\mathcal{Q}_s(\tau)$ reaches 1 first. Therefore, MA competes very favorably with the best existing algorithms for solving the minmax multidepot mTSP.

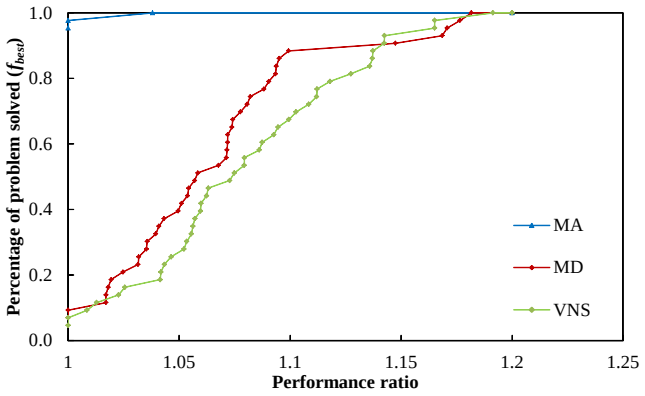


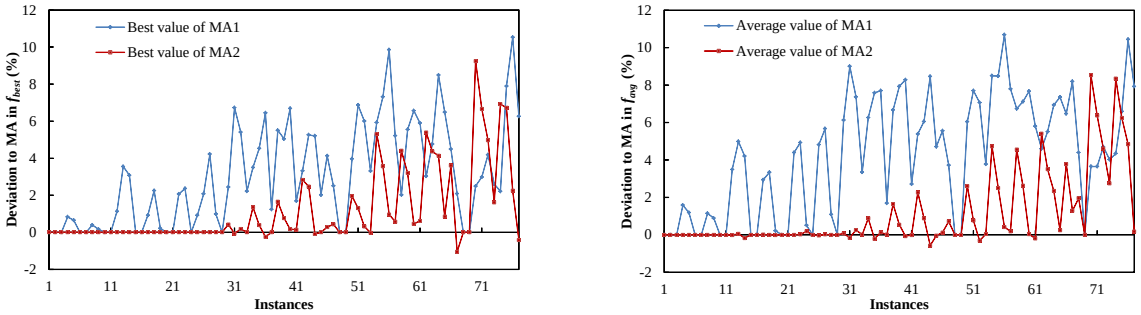
Figure 3.6 – The minmax multidepot mTSP: performance profiles of the MA and two reference algorithms on 43 instances of set $\mathbb{M}$.

Given that MA, MD and VNS were run on different computers and reported results of different qualities, it is not straightforward to make a fair comparison of their computation time. One observes that for the 18 instances where the time information is available for the compared algorithms, MA is able to reach the best-known results with a time of the same order of magnitude compared to MD and VNS, and then continue to improve these results during the rest of its execution.

According to the results of Sections 3.3.2 and 3.3.2, we conclude that the MA algorithm is highly effective for solving the minmax mTSP and the minmax multidepot mTSP compared to the best performing algorithms.

3.4 Additional experiments

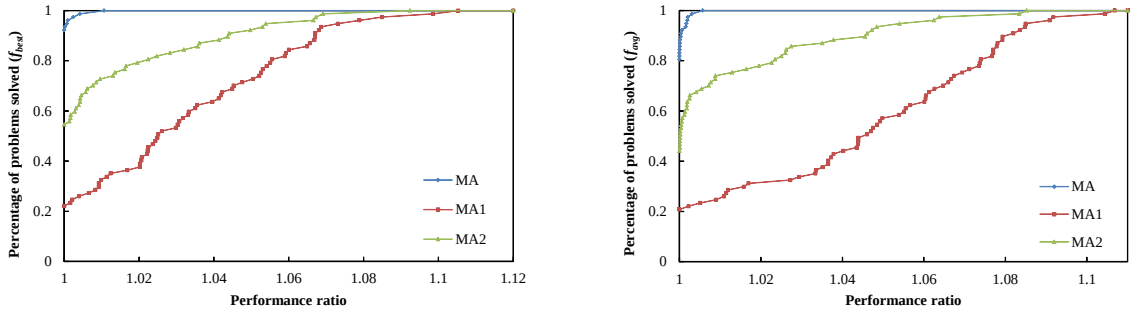
The computational results and comparisons with the existing algorithms on three sets of instances illustrated the high effectiveness and efficiency of the MA algorithm. In this section, we assess the contributions of two key components: the mEAX crossover and the post-optimization and two new neighborhood operators. Experiments are performed to compare MA and its variants where the assessed components are disabled. Furthermore, we investigate the long-term convergence behavior of the MA algorithm under a relaxed timing condition. The experiments reported in this section are based on the minmax mTSP.



(a) Best **(b)** Avg.

Figure 3.7 – Comparative results of MA with the variants MA1 (without mEAX) and MA2 (without the post-optimization) on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$

3.4.1 Benefits of the mEAX crossover and the post-optimization procedure



(a) Best **(b)** Avg.

Figure 3.8 – Performance profiles of MA and its variants MA1 and MA2 on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

To study the benefits of the mEAX operator and the post-optimization procedure, we created two MA variants MA1 and MA2 as follows. For MA1, we removed the mEAX operator (i.e., lines 6 and 7) in Algorithm 4 and replaced γ by μ in line 8. To make sure that MA1 consumes the given time budget effectively like MA, we repetitively re-start the algorithm until the time limit is reached. In other words, MA1 uses the VND procedure and the post-optimization to improve the solutions of the population within the given time limit. For the variant MA2, we just removed in Algorithm 4 the post-optimization (i.e., lines 9-12).

We ran MA1 and MA2 under the same condition of Section 3.3.1 to solve the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$. The results are summarized in Figs. 3.7 and 3.8.

Fig. 3.7 shows the deviations of the two variants MA1 and MA2 compared to MA (the reference line) in terms of the best results (Fig. 3.7(a)) and the average results (Fig. 3.7(b)). From these figures, we can make the following observations.

First, the results of MA1 indicate that removing mEAX deteriorates considerably the performance of the MA algorithm on a large majority of the tested instances in terms of the best and average results. The deterioration is more significant on large instances than on small instances. These results confirm the critical role of the proposed mEAX crossover.

Second, the results of MA2 indicate that the post-optimization doesn't really impact the performance of the MA algorithm on the first 29 small instances ($n \leq 318$). However, disabling this component deteriorates much MA's performance on many larger instances with $n > 318$. These results demonstrate the positive contributions of the post-optimization for solving large (and hard) instances.

Third, though both mEAX and post-optimization contribute to the high performance of the MA algorithm, the mEAX crossover plays a more general and more significant role compared to the post-optimization component.

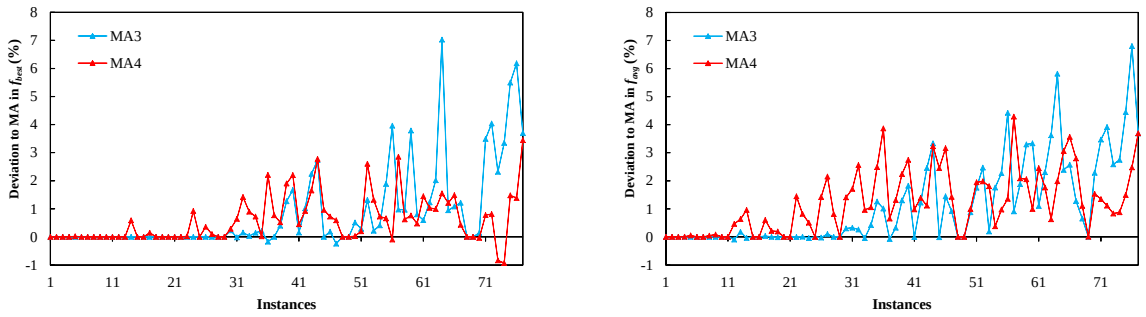
To further study the MA1 and MA2 variants, Fig. 3.8 shows the performance profiles of MA, MA1 and MA2 based on their best results (Fig. 3.8(a)) and their average results (Fig. 3.8(b)). We observe that MA dominates its two variants in terms of the best and average values. MA has a much higher $\mathcal{Q}_s(1)$ compared with MA1 and MA2. Indeed, MA reaches $\mathcal{Q}_s(\tau) = 1$ firstly, much earlier than the two variants, which indicates a higher robustness of the MA algorithm. In summary, these experiments confirm that both the mEAX crossover and the post-optimization contribute positively to the performance of MA, while the post-optimization component is especially useful for solving large instances.

3.4.2 Benefits of the new neighborhood operators

Six neighborhood operators are applied in the local search to ameliorate offspring solutions. We assess the contributions of the two new neighborhood operators: M3 independently used in [Zhe+22b] and M6 introduced in this work. For this purpose, two MA variants, MA3 (without M3) and MA4 (without M6), are compared, along with the standard MA associated with all neighborhood operators. To ensure a fair comparison, we ran MA3 and MA4 under the same condition of Section 3.3.1 to conduct the experiments. The results are summarized in Table 3.3 and illustrated in Fig. 3.9.

Table 3.3 – Summary of comparative results between MA and two variants.

Pair algorithms	f_{best}				f_{avg}			
	#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
MA vs. MA3	39	35	3	3.90E-08	47	20	10	4.42E-09
MA vs. MA4	47	26	4	9.15E-08	63	14	0	5.17E-12



(a) Best **(b)** Avg.

Figure 3.9 – Comparative results of MA with two variants MA3 (without operator M3) and MA4 (without operator M6) on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

According to Table 3.3, the two operators are critical to ensure the performance of the MA algorithm (confirmed by the small *p-value* $\ll 0.05$). Indeed, disabling them significantly worsens the results in terms of both the best and average values. Moreover, as shown in Fig. 3.9, disabling the M3 operator deteriorates MA's performance more than disabling the M6 operator on many large instances with $n > 2152$. These results demonstrate the positive contributions of the M3 operator for solving large instances. Finally, both neighborhood operators have marginal contributions when solving small and medium-sized instances ($n \leq 532$) in terms of the best results.

3.4.3 Convergence analysis of the MA algorithm

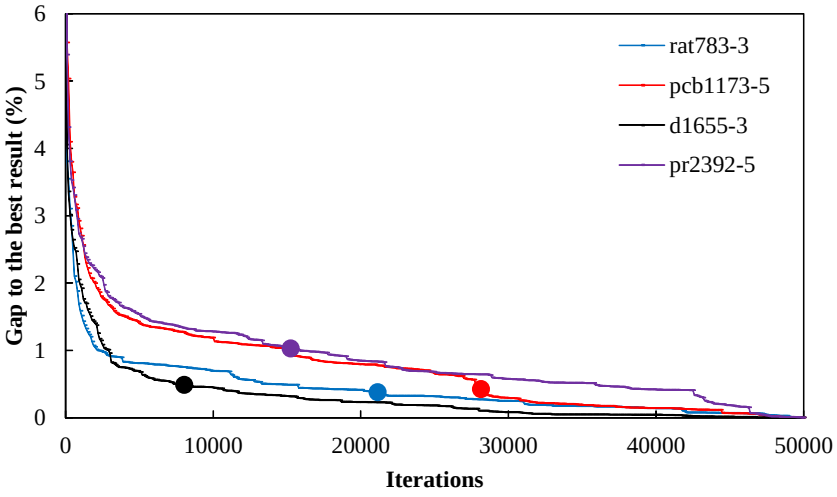


Figure 3.10 – Running profiles of the MA on four representative instances

In Section 3.3.2, the stopping condition for solving the minmax mTSP was set to the

maximum time of $(n/100) \times 4$ minutes in line with the literature. This section aims to verify the convergence behavior of the MA algorithm in the long run by using a relaxed stopping condition of 50,000 iterations. Four representative instances (rat783-3, pcb1173-5, d1655-3, pr2392-5) with different sizes (n from 783 to 2392, m from 3 to 5) were selected and each instance was solved 20 times while the best objective values are recorded during the search. Fig. 3.10 shows the evolution of the gap between the current value and the best value along the iterations. The four colored dots indicate the average objective values obtained at the end of the standard cutoff time of $(n/100)\times4$ minutes for the four instances. For these instances, 50,000 iterations lead to 4061.67, 4058.71, 16858.6, 13983.4 seconds, respectively.

From Fig. 3.10, one observes that with a higher time budget, MA is able to further improve its results reached at the end of the standard cutoff time $(n/100)\times4$ minutes). Specifically, the best result can be even improved by 1.19% while the average result can be improved by 1.03%. This experiment demonstrates that the MA algorithm has a highly desirable long-term search behavior and can effectively take advantage of a prolonged cutoff time limit to discover still better solutions.

3.5 Chapter conclusion

In this chapter, we introduced a unified memetic algorithm for solving both the minmax mTSP and the minmax multidepot mTSP. The proposed algorithm integrates a dedicated edge assembly crossover operator (mEAX), an efficient variable neighborhood descent and an aggressive post-optimization procedure. By properly inheriting edges from high-quality parent solutions, mEAX contributes to propagate favorable characteristics from elite parent solutions to offspring. The variable neighborhood descent is able to locate local optimal solutions effectively. The post-optimization procedure takes full advantage of the ejection chain method and a leading TSP heuristic to further improve the quality of new elite solutions.

The performance of the algorithm was evaluated on two sets of 77 minmax mTSP instances and one set of 43 minmax multidepot mTSP instances. The computational results indicated that the algorithm reaches a high performance compared to the reference algorithms for both problems. Specifically, it reports 44 and 39 new upper bounds for the minmax mTSP and the minmax multidepot mTSP, respectively. We performed additional experiments to assess the contributions of the two key algorithmic components (i.e., mEAX and post-optimization). We also conducted a long term convergence analysis of the algorithm to illustrate its capacity of finding still better solutions if more time is allowed.

In the next chapter, we study the traveling salesman problems with profits and propose a hybrid genetic algorithm for solving two well-known problems.

Chapter 4

A HYBRID GENETIC ALGORITHM FOR UNDIRECTED TRAVELING SALESMAN PROBLEMS WITH PROFITS

In this chapter, we introduce a hybrid genetic algorithm that addresses the orienteering problem (OP) and the prize-collecting traveling salesman problem (PCTSP) under a unified framework. The algorithm combines an extended edge assembling crossover operator to produce promising offspring solutions and an effective local search to ameliorate each offspring solution. The algorithm is further enforced by a diversification-oriented mutation and a population-diversity management. Extensive experiments show that the method competes favorably with the best existing methods both in terms of solution quality and computational efficiency. Additional experiments help to get insights into the roles of the key ingredients of the proposed method. The content of this chapter is based on an article submitted to *Networks*.

4.1 Introduction

Traveling salesman problems with profits are useful models for a broad range of applications [BM85; FDG05; FT88; GLV16; RB91; VSV11]. As it is shown in the comprehensive review of [FDG05], a number of studies have contributed to improve the state-of-the-art of solving these difficult problems. On the one hand, several exact algorithms were proposed in [Bal89; BGP09; FGT98; GLS98b; LM90; LR94] to optimally solve small and medium instances with up to 532 vertices. Remarkably, the recent revisited branch and cut algorithm presented by Kobeaga et al. [KML20] was able to find optimal solutions for OP instances with up to 2152 vertices. On the other hand, several heuristic algorithms were developed for TSPs with profits to deal with instances whose optimal solutions cannot be determined by exact algorithms. In Section 1.4.2, we provide a review of the most representative heuristic algorithms. Meanwhile, one notices that until now, these problems have been studied separately with specific algorithms designed for each problem without a general and unified approach. Moreover, compared to research on exact algorithms, effective heuristic algorithms are still rare and most existing heuristic algorithms don't compete well with the best exact algorithms on a number of benchmark instances.

This chapter aims to advance the state-of-the-art of solving for TSPs with profits with effective heuristic algorithms. For this purpose, we introduce a unified approach for the OP and the PCTSP under the hybrid genetic search framework. Hybrid genetic algorithms, also called memetic algorithms, take advantage of population-based genetic framework and neighborhood-based local search framework [Hao12]. On the one hand, thanks to the use of a population of solutions, a genetic algorithm offers, via a crossover operator, the possibility of creating new solutions by recombination of existing solutions. On the other hand, by exploring a neighborhood, a local search algorithm offers an effective means to locate high-quality solutions around a seeding solution. By combining these two complementary methods, a hybrid genetic algorithm is expected to reach a performance that cannot be attained by each individual approach applied separately. Indeed, several highly effective hybrid genetic algorithms have been proposed to solve various routing problems [NB09; NBD10; Pot09; Pri04; Vid+12; Vid+13; Vid+14].

For the OP and the PCTSP, we devise a dedicated technique to adapt the popular edge assembly crossover initially designed for the travel salesman problem [NK97; NK13] and also applied to routing problems [NB09; NBD10]. The proposed approach relies on an extended edge assembly crossover operator and benefits from the synergy with effective local search and dedicated diversification strategies such as mutation and population-diversity management. Our experiments on well-known benchmark instances in the literature show that the proposed algorithm competes very favorably with the best performing methods. In particular, the algorithm is able to improve many current best bounds for both the OP and the PCTSP.

The rest of this chapter is organized as follows. Section 4.2 presents the proposed algorithm. Section 4.3 shows computational results and comparisons. Section 4.4 analyzes the main ingredients of the algorithm. Section 4.5 draws conclusions.

4.2 Hybrid genetic algorithm for TSPs with profits

This section presents the hybrid genetic algorithm (HGA) designed for the two studied TSPs with profits, i.e., the orienteering problem and the prize-collecting TSP. This is a unified algorithm in the sense that, with slight adjustments, the same algorithm is used to solve both problems effectively.

The general HGA algorithm is composed of the following five steps.

Step 1 (Generation of an initial population): This step fills the population $\mathcal{P}$ with a number of distinct solutions with the initialization procedure presented in Section 4.2.1. This initial population is then evolved generation-by-generation through Steps 2–4.

Step 2 (Parent selection and crossover application to generate offspring solutions): From the current population, two solutions are selected as parents using the tournament selection of size of 2. The two parent solutions are then recombined by the extended edge assembly crossover presented in Section 4.2.2 to generate β (β is a parameter) offspring solutions $\{\varphi_O^1, \varphi_O^2, \cdots, \varphi_O^\beta\}$.

Step 3 (Local search to improve each offspring solution): For each offspring solution φ_O^i in $\{\varphi_O^1, \varphi_O^2, \cdots, \varphi_O^\beta\}$, the local search presented in Section 4.2.3 is applied to raise the quality of the offspring.

Step 4 (Mutation and population update): Each offspring solution φ_O^i improved by the local search is modified by the mutation presented in Section 4.2.4 to introduce diversity. The modified offspring is then used to update the population as described in Section 4.2.4.

Step 5 (Stopping) The algorithm repeats Steps 2–4 until a stopping condition is satisfied. Typical conditions are a maximum number of generations (one generation includes Steps 2–4), a maximum cutoff time and a maximum number of local search invocations. At the end of the algorithm, the best solution φ^* ever found is returned.

Throughout the course of the algorithm, the best solution φ^* found, which is initialized by the best solution in the initial population, is updated each time a solution better than the current φ^* is discovered.

The rest of this section is dedicated to detailed presentation of the methods for population initialization, crossover, local search and mutation as well as population management.

4.2.1 Population initialization

The initial population $\mathcal{P}$ is generated in two phases by a method inspired by the technique presented in [Vid22]. Phase 1 generates a pool of $4 \times \lambda$ solutions where each solution is created greedily (see below) and then improved by the local search of Section

4.2.3. Phase 2 uses the surviving strategy of Section 4.2.4 to retain λ solutions in $\mathcal{P}$ with respect to solution quality and their contribution to the diversity of the population.

Since the OP and the PCTSP pursue different optimization objectives, the first phase uses two greedy strategies to create each initial solution. For the OP, the greedy construction works as follows. First, a solution (route) is initialized by the depot v_0 and then extended by adding a random vertex $v_i \in \mathcal{N}$. Second, for the newly added vertex v_i, an unrouted vertex v_j from the δ-nearest neighborhood (Section 4.2.3) is selected and inserted after vertex v_i such that the insertion leads to the minimum increase of the travel costs. This process stops when all vertices are inserted to the solution or the current travel costs exceed $1.5 \times c_{max}$.

For the PCTSP, the greedy construction works similarly, but the selection of the next vertex to be added aims to maximize the collected profit. The construction stops when the collected profit reaches $1.5 \times p_{min}$.

Note that initial solutions generated this way are necessarily infeasible. Given that the feasibility of an initial solution can be easily established by simply removing some vertices, using an initial population of infeasible solutions is not harmful; instead, it's beneficial in terms of search diversification.

4.2.2 Extended edge assembly crossover

The HGA algorithm relies on an extended edge assembly crossover, which is an adaptation of the edge assembly crossover (EAX) designed for the TSP [NK97; NK13] to TSPs with profits. Critically, there is a difficulty of directly applying EAX to TSPs with profits since EAX assumes that all vertices are visited exactly once in a solution of the TSP.

Indeed, given a TSP instance defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a candidate TSP solution φ corresponds to a partial graph $\mathcal{G}_\varphi = (\mathcal{V}, \mathcal{E}_\varphi)$ with $\mathcal{E}_\varphi$ being the set of edges traversed by φ. Given a solution of the TSP, each vertex in $\mathcal{V}$ is visited exactly once and thus has the same degree of two in $\mathcal{G}_\varphi$. Given two parent TSP solutions and their associated partial graphs, EAX uses this property to reassemble the edges from the parents to produce offspring solutions.

However, the situation is different for TSPs with profits. Given two parent solutions, some vertices may be visited in one parent, but not visited in the other parent. Consequently, a vertex may have two distinct degrees in the partial graphs of the parent solutions. This particularity makes it impossible to apply the EAX crossover to TSPs with profit. For the OP and the PCTSP, we design the extended edge assembly crossover (E^2AX), whose key idea is to add dummy edges (self-loops) to ensure that each vertex has the same degree in the graphs of the parent solutions.

Given an instance of the OP or the PCTSP on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let φ be a solution visiting $|\varphi|$ vertices ($|\varphi| \leq n$) and let $\mathcal{G}_\varphi = (\mathcal{V}, \mathcal{E}_\varphi)$ be the corresponding partial graph where $\mathcal{E}_\varphi \subset \mathcal{E}$ is the set of edges traversed by φ. There are two cases for each vertex in $\mathcal{G}_\varphi$ such that: 1) the vertex is visited by φ and the degree is 2 in $\mathcal{G}_\varphi$, 2) the vertex is not visited by φ and the degree is 0. In the example of Fig. 4.1, red vertices are not visited

by φ_A and the degree is 0 in $\mathcal{G}_\mathcal{A}$, while the visited vertices in $\mathcal{G}_\mathcal{A}$ has a degree of 2.

Let φ_A and φ_B be two candidate solutions for the OP or the PCTSP, let $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ be the corresponding partial graphs. We define the *degree difference* of vertex v in $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$ by $\Delta_v = |deg_\mathcal{A}(v) - deg_\mathcal{B}(v)|$ where $deg_\varphi(v)$ denotes the degree of vertex v in the graph $\mathcal{G}_\varphi$. In the example of Fig. 4.1, the degree difference Δ_v of a vertex v equals 0 if v is visited by both solutions or by none of them; otherwise $\Delta_v = 2$. For each vertex v with $\Delta_v = 2$, we can add a dummy loop (v, v) in $\mathcal{G}_\mathcal{A}$ or $\mathcal{G}_\mathcal{B}$ to make the degree difference become 0 (see Fig. 4.1(left-middle)).

Let $\mathcal{G}'_\mathcal{A} = (\mathcal{V}, \mathcal{E}'_\mathcal{A})$ and $\mathcal{G}'_\mathcal{B} = (\mathcal{V}, \mathcal{E}'_\mathcal{B})$ be the graphs extended with dummy loops such that $\Delta_v = 0$ for all vertices. Clearly, these extended graphs $\mathcal{G}'_\mathcal{A}$ and $\mathcal{G}'_\mathcal{B}$ satisfy the basic property required by the EAX crossover, i.e., each vertex has the same degree in these graphs. As a result, we can now benefit the edge assembly idea of the EAX operator to create offspring solutions for the OP and the PCTSP.

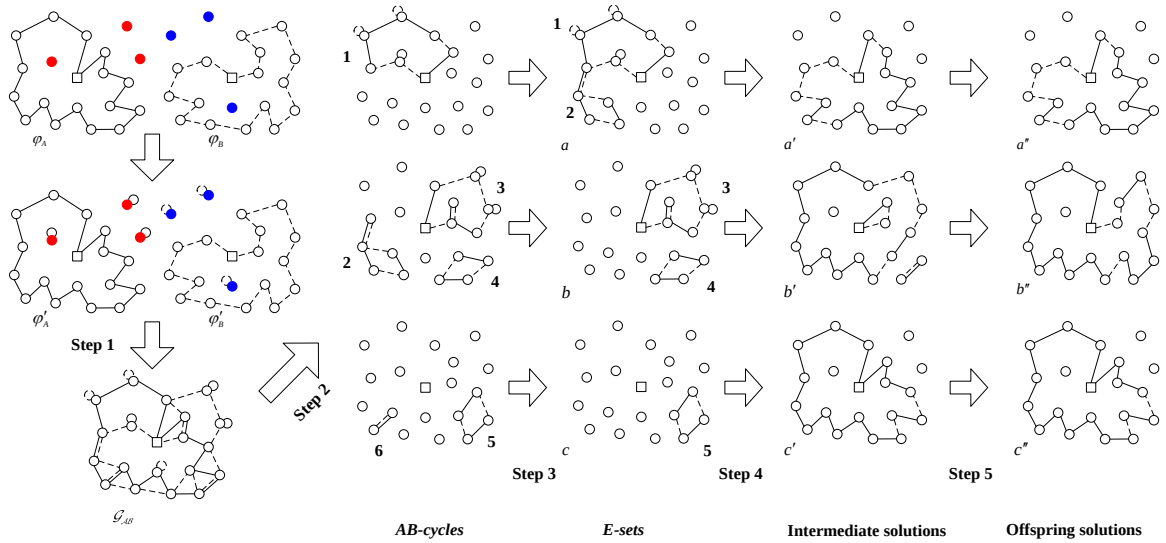


Figure 4.1 – Illustration of the E^{2}AX crossover

Given two parent solutions φ_A and φ_B, the proposed E^{2}AX crossover for the OP and the PCTSP performs the following steps to generate β offspring solutions.

1. *Generation of multigraph $\mathcal{G}_{\mathcal{AB}}$ with dummy loops.* Build partial graphs $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ for φ_A and φ_B. For each vertex v such that $\Delta_v \neq 0$ in $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$, add $\frac{|deg_\mathcal{A}(v) - deg_\mathcal{B}(v)|}{2}$ dummy self-loops in φ_A or φ_B to make $\Delta_v = 0$. Build multigraph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, \mathcal{E}'_\mathcal{A} \cup \mathcal{E}'_\mathcal{B})$ where $\mathcal{E}'_\mathcal{A}$ and $\mathcal{E}'_\mathcal{B}$ are the edge sets extended with dummy loops.
2. *Generation of AB-cycles from $\mathcal{G}_{\mathcal{AB}}$.* An *AB-cycle* is a closed-path whose edges are alternatively taken from the parents. From the multigraph $\mathcal{G}_{\mathcal{AB}}$, build a set of *AB-cycles* as follows. Initialize an *AB-cycle* by a random vertex with one adjacent edge in $\mathcal{G}_{\mathcal{AB}}$. Then add edges belonging to $\mathcal{E}'_\mathcal{A}$ and $\mathcal{E}'_\mathcal{B}$ alternatively until a cycle is obtained,

which is an *AB-cycle*. Remove the edges of the *AB-cycle* from $\mathcal{G}_{\mathcal{AB}}$. Repeat the process to build the next *AB-cycle* until all the edges in $\mathcal{G}_{\mathcal{AB}}$ are considered.

3. *Generation of E-sets.* An *E-set* is an union of AB-cycles. Divide the set of *AB-cycles* randomly and uniformly into β subsets ($\beta = 3$ in this work), each subset of *AB-cycles* defining an *E-set*.
4. *Generation of intermediate solutions.* First remove all dummy loops in the *E-sets*. Then for each *E-set*, produce an intermediate solution from φ_A by removing the edges of $\mathcal{E}_{\mathcal{A}}$ and adding the edges of $\mathcal{E}_{\mathcal{B}}$.
5. *Elimination of isolated subtours.* For each intermediate solution containing subtours, merge the subtours with the main tour by the method presented in [NK13].

Fig. 4.1 provides an illustrative example of the recombination process with the E^{2}AX crossover applied to two parent solutions φ_A and φ_B. Note that the second intermediate solution contains two small subtours that are merged with the main tour to form a single tour.

We now provide an analysis of the time complexity of the E^{2}AX crossover. Steps (1)–(4) have to assemble $|\mathcal{E}'_{\mathcal{A}}| + |\mathcal{E}'_{\mathcal{B}}|$ edges to produce β offspring solutions, implying a time complexity of $\mathcal{O}(|\mathcal{E}'_{\mathcal{A}}| + |\mathcal{E}'_{\mathcal{B}}|)$. Given that a solution is necessarily an elementary tour and $n \geq |\mathcal{E}'_{\mathcal{A}}| \geq |\mathcal{E}'_{\mathcal{B}}|$ holds. Thus the time of steps (1)–(4) is bounded by $\mathcal{O}(n)$. For the last step, suppose that there are m subtours including at most e edges, the time complexity of this step is $\mathcal{O}(e \times \delta)$ [NK13], where δ is the number of closest vertices and introduced in Section 4.2.3.

4.2.3 Offspring improvement

The HGA algorithm employs a neighborhood-based local search to improve the offspring solutions generated by the E^{2}AX crossover. As discussed in [FDG05], four neighborhood operators are usually used to transform a route for TSPs with profits: 1) adding an unrouted vertex, 2) removing a vertex from the route, 3) resequencing the route, and 4) replacing a routed vertex with an unrouted vertex. The HGA algorithm adopts the first three operators because our experiments show that the fourth operator is of little interest. Also, to resequence a route, any TSP heuristic can be used. In our case, we find the 2-opt heuristic [Cro58] quite suitable. We now explain the add and remove operators.

Add operator

This operator is applied to add unrouted vertices into the route. For the OP, a heuristic commonly used in the literature [Cam+14; SG10] is adopted to perform vertex insertions. For each unrouted vertex v_i, its move gain $\Delta = \frac{p_i}{c_{i_p i} + c_{i i_n} - c_{i_p i_n}}$ is calculated, where v_{i_p} and v_{i_n} are the vertices before and behind v_i, respectively. Then the most favorable vertex with the highest move gain is selected and added in the route. The add operator is repetitively applied until the limit c_{max} of travel costs is attained. For the PCTSP, the

vertex associated with minimum increase of the travel costs is selected and added in the route. The add operator is triggered to insert unrouted vertices if the collected profit is below the minimum profit threshold p_{min}.

The worst time complexity of the add operator is $\mathcal{O}(|\varphi| \times (n - |\varphi|))$, where $|\varphi|$ is the number of visited vertices in the solution. This complexity can be reduced to $\mathcal{O}(\delta \times (n - |\varphi|))$ by considering only the δ-nearest vertices (δ is a parameter called granularity threshold) and using streamlining techniques of [KML18].

Remove operator

This operator is applied to remove visited vertices. For the OP, given a routed vertex v_i, the move gain of removing v_i is given by $\Delta = \frac{p_i}{c_{i_p i} + c_{i i_n} - c_{i_p i_n}}$, where v_{i_p} and v_{i_n} are the vertices before and behind v_i, respectively. If the solution is infeasible, i.e., the travel costs are greater than c_{max}, the vertex with respect to the minimum Δ is removed. The remove process stops once the solution becomes feasible. For the PCTSP, if the solution is feasible, i.e., the collected profit is greater than the required minimum profit p_{min}, vertices v_i can be removed from the route such that they have the maximum move gain $\Delta = c_{i_p i} + c_{i i_n} - c_{i_p i_n}$, where v_{i_p} and v_{i_n} are the vertices before and behind v_i, respectively. The process terminates when the collected profit reaches the required minimum profit p_{min}. The time complexity of the remove operator is bounded by $\mathcal{O}(|\varphi|)$.

Application of the move operators

Given the add and remove operators as well as the 2-opt operator, it is important to decide in which order they are applied. Given that the OP and the PCTSP pursue different objectives with different constraints, the HGA algorithm applies a specific order for each problem. For both problems, the 2-opt heuristic is first applied to reduce the travel costs. Then, for the OP, the remove operator is used to restore the feasibility of the solution in terms of the travel costs, followed by the add operator to increase the profit. For the PCTSP, the add operator is used to satisfy the minimum profit constraint, followed by the remove operator to reduce the travel costs as much as possible. Once the solution cannot be improved by any operator, the local search phase terminates and returns the best solution reached.

4.2.4 Diversity preservation

Diversity is a key issue of any population-based algorithm. The HGA algorithm employs two different and complementary strategies, i.e., a specific mutation and a dedicated population management, to effectively preserve population diversity.

Mutation

An offspring solution created by the E^{2}AX crossover inherits exclusively the edges of its parents. In other words, the E^{2}AX cannot introduce vertices that are not visited by both parents in the offspring solutions. Furthermore, the local search is rarely able to introduce unrouted vertices into the solution given that adding new vertices often increase the travel costs, which is undesirable. Consequently, the offspring solution may resemble much the parents even after local optimization. To maintain sufficient diversity and avoid premature convergence, the HGA algorithm applies, with a probability τ, a mutation to modify each offspring solution by adding new vertices. Basically, the mutation removes some vertices from the solution and then greedily inserts unrouted vertices into the solution while respecting to the corresponding constraints (i.e., maximum travel costs c_{max} for the OP and minimum collected profit p_{min} for the PCTSP).

Given a solution φ, let $\mathcal{N}_\varphi$ and $\overline{\mathcal{N}}_\varphi$ be a set of routed and unrouted vertices in φ, respectively. The mutation consists of two steps. First, l vertices (l is a parameter called mutation length) are selected and removed one by one. Specifically, a vertex v_i is selected for removal if its removal leads to the minimum move gain $\Delta = \frac{p_i}{c_{i_p i}+c_{i i_n}-c_{i_p i_n}}$, where v_{i_p} and v_{i_n} are the vertices before and behind v_i, respectively. Each removed vertex is forbidden to be reinserted again into the route during the mutation. Second, a vertex v_j is selected from $\overline{\mathcal{N}}_\varphi \backslash \mathcal{T}$ such that its insertion leads to the maximum increase of $\Delta = \frac{p_i}{c_{i_p i}+c_{i i_n}-c_{i_p i_n}}$ and inserted in the solution φ. For the OP, the insertion process stops when l unrouted vertices are inserted or if the insertion makes the solution infeasible (i.e., the travel costs exceed c_{min}). For the PCTSP, the insertion terminates when l vertices are inserted or the insertion makes the solution feasible (i.e., the collected profit reaches p_{min}). In Section 4.3.2, we experimentally show the importance of the mutation.

Population management

To maintain a suitable diversity of the population $\mathcal{P}$, the HGA algorithm adopts a variable population scheme similar to that used in [Vid22]. From an initial population of λ solutions, the population is extended by offspring solutions until its size reaches a upper limit $\mu + \lambda$ where μ is the generation size. When this happens, the surviving selection is triggered to remove μ solutions with respect to the fitness and their contributions to the diversify of the population. Similar to Vidal [Vid22], the distance between two solutions is defined as the number of distinct edges. Let $|\mathcal{P}|$ donate the number of solutions in $\mathcal{P}$. Given a solution φ, the distance between φ and other $|\mathcal{P}| - 1$ solutions is computed and sorted from the smallest to the largest. Then, the sum of the first *nbClost* values (*nbClost* is a parameter) is used as the diversity contribution of φ to $\mathcal{P}$, donated by div_φ. Each solution $\varphi \in \mathcal{P}$ is thus associated with a div_φ value. All these values are sorted from the smallest to the largest, and each solution is associated with a rank rd_φ with respect to div_φ. Furthermore, we rank the solutions of $\mathcal{P}$ according to their objective values from the worst to the best, leading to another rank ro_φ for each solution φ. Finally, the biased fitness of solution φ is defined as $f(\varphi)_{biased} = ro_\varphi + (1 - \frac{nbElite}{|\mathcal{P}|}) \times rd_\varphi$ where *nbElite* is

a parameter. The solution associated with the smallest biased fitness is removed from $\mathcal{P}$ and the biased fitness for each remaining solution in $\mathcal{P}$ is updated. The solution removal process is repeated until there are λ solutions in $\mathcal{P}$. Following [Vid22], we set $nbClost = 5$ and $nbElite = 4$.

If the best solution found so far $\varphi*$ cannot be improved during γ consecutive iterations (γ is a parameter called population rebuilding threshold and one iteration is the generation of one offspring solution followed by the local search), the algorithm restarts by generating a totally new population.

4.3 Computational results and comparisons

In this section, we evaluate the performance of the proposed algorithm on the OP and the PCTSP. We present the benchmark instances (see Section 1.4.3), experimental protocol, reference algorithms, and comparisons with the state-of-the-art methods.

4.3.1 Experimental protocol and reference algorithms

Parameter setting. HGA has six main parameters: minimum population size λ and generation size μ, granularity threshold δ used in local search, mutation probability τ, mutation length l and population rebuilding threshold γ. In order to identify suitable values for the parameters, the automatic parameter tuning package Irace [Lóp+16] is used. The candidate and final values are shown in Table 4.1 . These parameter values can be considered to form the default setting and are used consistently in our experiments.

Table 4.1 – Parameter tuning results.

Parameter	Section	Description	Considered values	Final values	
				OP	PCTSP
λ	4.2.1 and 4.2.4	minimal size of population	$\{50, 100, 150, 200, 250\}$	100	100
μ	4.2.1 and 4.2.4	generation size	$\{25, 50, 75, 100, 125\}$	50	100
δ	4.2.3	granularity threshold	$\{5, 8, 10, 12, 15, 20\}$	10	12
τ	4.2.4	mutation probability	$\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$	0.15	0.1
l	4.2.4	mutation length	$\{0.05, 0.1, 0.15, 0.2, 0.25\}$	0.25	0.25
γ	4.2.4	population rebuilding threshold	$\{5000, 10000, 20000, 30000, 50000, 80000\}$	30000	30000

Reference algorithms. According to the review of Section 1.4.2, we identify the following best heuristic and exact algorithms for the OP and use them for our comparative study.

— BKS. This indicates the best known solutions (best lower bounds) that are compiled from all reference heuristic and exact approaches [KML18; KML20; San19].

— RB&C [KML20]. This exact algorithm [KML20] was applied to solve the first three sets of instances and was able to obtain optimal solutions for many instances under a time limit of 18000s.

— ALNS [San19]. It was implemented in C++ and executed on an Intel Xeon E5 processor, running at 2.2 GHz under a time limit of 18000s or after 250000 iterations without improvement. The algorithm was executed 10 times on each instance. It was tested on the four sets of instances.

— EA4OP [KML18]. The hybrid algorithm was implemented in C and executed on an Intel Xeon E5-2609 v3 1.90 GHz processor with 4 GB RAM. The algorithm terminates either when the first quartile of the population's fitness is the same as the best fitness or when the maximum running time exceeds 18000s. The algorithm was executed 10 times on each instance. The EA4OP algorithm reported its results on the four sets of instances.
— B&C [KML18]. This is the branch and cut algorithm presented in [FGT98] and rerun in [KML18]. It stops when the maximum running time (18000s) is met or when the optimal solution is found. This algorithm reported the results on the fourth set only.

For the PCTSP, only the branch & cut algorithm (B&C) [BGP09] reported results on medium-sized instances with up to 532 vertices. To have a reference algorithm for the large-sized instances with up to 7397 vertices, we created a HGA variant (called HGA-Giant) where we replaced the E^2AX crossover by a giant tour crossover described in Appendix 6.2.

Experimental setting and stopping criterion. The HGA algorithm was implemented in C++ and compiled using the g++ compiler with the -O3 option [1]. All experiments were run on an Intel Xeon ES-2630 processor of 2.66 GHz and 6 GB RAM running Linux with a single thread. The algorithm was executed 20 times on each instance with distinct random seeds. Following the literature, the HGA algorithm terminates when it reaches a time limit of 18000s or a maximum of 500,000 iterations (one iteration means the generation of one offspring solution followed by one local search run).

4.3.2 Computational results

To compare HGA and the reference algorithms, two summarizing tables are presented for the OP and the PCTSP, respectively.

Comparative results on the OP

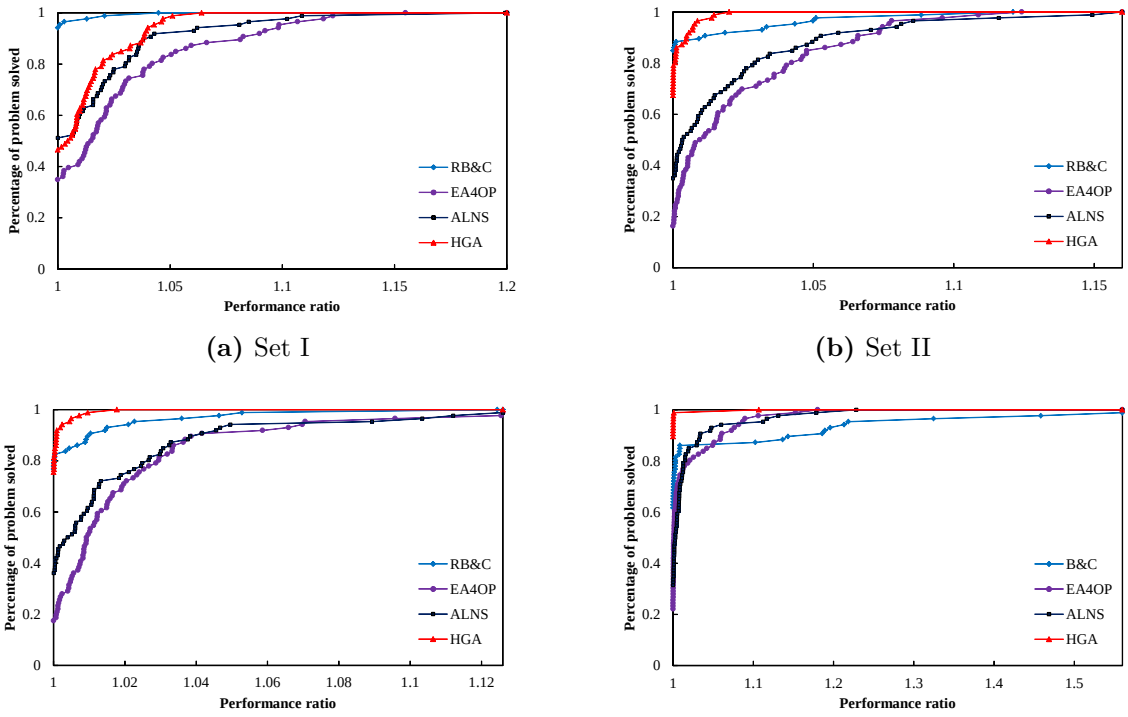
Since the two reference heuristic algorithms ALNS [San19] and EA4OP [KML18] did not report their average values, we focus on the best objective values of the compared algorithms in Table 4.2. The detailed results on the four sets of 344 instances are provided in Tables 6.4-6.11.

Compared to the BKS values that represent the best values ever reported by all the algorithms, HGA updates 67 BKS values (new lower bounds) out of 344 instances (19.5%) and matches 172 other BKS values (50%). Given that the BKS values are the best results compiled from all existing approaches, the HGA algorithm can be considered to reach a remarkable performance.

HGA significantly outperforms the two best heuristic algorithms ALNS and EA4OP ($p\text{-}value \ll 0.05$), except ALNS on the first set. Furthermore, the two best exact algorithm

1. The code of the HGA algorithm will be available at: https://github.com/pengfeihe-angers/tsps-with-profits.git

RB&C and B&C can obtain many optimal solutions for medium-sized instances within a reasonable running time, but their results and running time become unacceptable with the increase of instance sizes. As shown in Tables 6.7, 6.9 and 6.11, HGA is able to provide significant improvements for large-sized instances, especially for instances with at least 2000 vertices.



(a) Set I

(b) Set II

(c) Set III

(d) Set IV

Figure 4.2 – The OP: performance profiles of the compared algorithms on the 86 instances of each set

The performance profiles illustrate the performance differences of the compared algorithms. As shown in Fig. 4.2, the algorithms have different behaviors on the four sets of instances. For the first set, it is clear that RB&C outperforms all approaches. HGA has a lower $\mathcal{Q}_s(1)$ but reaches 1 more quickly than the two heuristic algorithms. However, for the other three sets, HGA dominates the reference algorithms since it reaches 1 firstly, which indicates a high robustness. The performance profiles confirm that the HGA algorithm dominates the state-of-the-art algorithms for the OP, except the exact algorithm RB&C on the first set.

Tables 6.4-6.11 show the detailed results on all 344 OP instances. Although EA4OP reports very short running time, its results are much worse than those of ALNS and HGA. Compared with ALNS, our HGA algorithm can find better results with less running time. It is noticeable that exact algorithms spend very short time for medium-sized instances to obtain the optimal solutions, but the gap becomes unacceptable for large-sized instances.

Table 4.2 – The OP: summary of results between HGA and reference algorithms on the four sets of 344 instances in terms of the best objective values.

Instances	Pair algorithms	Medium-sized (45)				Large-sized (41)			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Set I	HGA vs. BKS	0	35	10	2.00E-03	3	1	37	5.51E-06
	HGA vs. RB&C [KML20]	0	35	10	2.00E-03	5	1	35	4.62E-05
	HGA vs. EA4OP [KML18]	**12**	29	4	1.80E-02	**32**	2	7	6.70E-06
	HGA vs. ALNS [San19]	3	35	7	4.59E-01	**20**	4	17	7.61E-02
Set II	HGA vs. BKS	0	43	2	5.00E-01	13	2	26	5.53E-01
	HGA vs. RB&C [KML20]	0	43	2	5.00E-01	13	2	26	7.64E-01
	HGA vs. EA4OP [KML18]	**31**	14	0	1.17E-06	**41**	0	0	2.42E-08
	HGA vs. ALNS [San19]	**16**	29	0	4.35E-04	**40**	0	1	2.61E-08
Set III	HGA vs. BKS	0	43	2	5.00E-01	19	3	19	7.10E-02
	HGA vs. RB&C [KML20]	0	43	2	5.00E-01	19	3	19	6.24E-02
	HGA vs. EA4OP [KML18]	**28**	15	2	1.64E-05	**39**	0	2	5.26E-08
	HGA vs. ALNS [San19]	**14**	29	2	1.13E-02	**38**	0	3	6.14E-08
Set IV	HGA vs. BKS	2	41	2	8.75E-01	**30**	4	7	1.54E-05
	HGA vs. B&C [KML18]	2	41	2	8.75E-01	**30**	4	7	4.15E-06
	HGA vs. EA4OP [KML18]	**27**	17	1	6.57E-05	**39**	0	2	7.81E-08
	HGA vs. ALNS [San19]	**20**	24	1	1.01E-03	**39**	2	0	5.25E-08
Summary	HGA vs. BKS	2	162	16	-	65	10	89	-

Thus, even if HGA can find high-quality solutions in a short time for small and medium-sized instances, its main interest remains its capacity of solving large-sized OP instances.

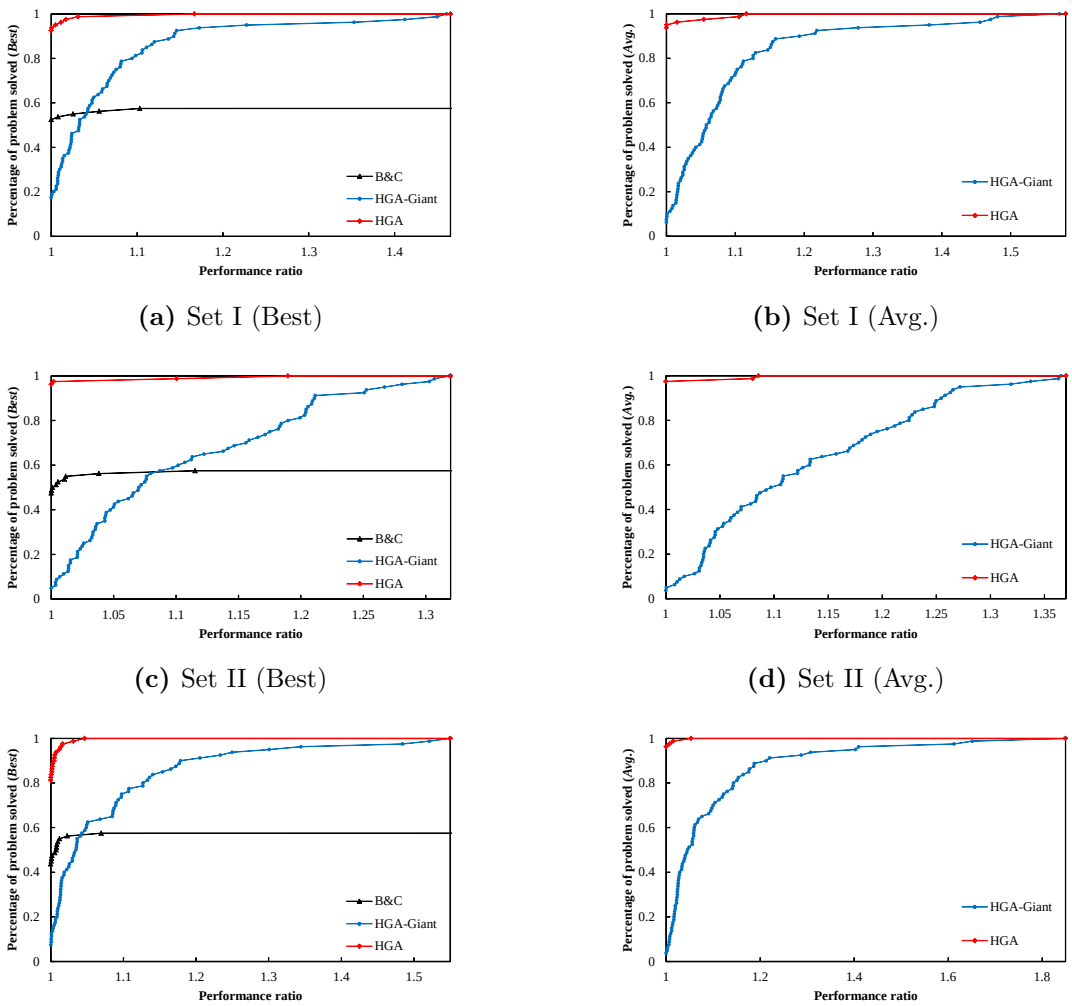
Comparative results on the PCTSP

Table 4.3 – The PCTSP: summary of results between HGA and reference algorithms on the three sets of 240 instances.

Instances	Pair algorithms	Medium-sized (46)							
		Best				*Avg.*			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Set I	HGA vs. B&C [BGP09]	4	37	5	8.20E-01	-	-	-	-
	HGA vs. HGA-Giant	**32**	12	2	6.63E-06	**42**	0	4	5.35E-06
Set II	HGA vs. B&C [BGP09]	**7**	36	3	4.92E-01	-	-	-	-
	HGA vs. HGA-Giant	**41**	3	2	1.21E-06	**43**	1	2	3.68E-07
Set III	HGA vs. B&C [BGP09]	11	21	14	3.06E-01	-	-	-	-
	HGA vs. HGA-Giant	**40**	5	1	2.92E-08	45	0	1	4.90E-09
	Summary	**20**	96	22	-	**130**	1	7	-
		Large-sized (34)							
		Best				*Avg.*			
Set I	HGA vs. HGA-Giant	**33**	0	1	4.78E-07	**33**	0	1	4.00E-07
Set II	HGA vs. HGA-Giant	**34**	0	0	3.65E-07	**34**	0	0	3.65E-07
Set III	HGA vs. HGA-Giant	**33**	0	1	1.62E-06	**32**	0	2	1.77E-06
	Summary	**100**	0	2	-	**99**	0	3	-

As shown in Table 4.3 and Fig. 4.3, the HGA algorithm significantly outperforms HGA-Giant since all *p-values* are less than 0.05. For the medium-sized instances, the exact algorithm (B&C) performs well by obtaining many optimal solutions within a reasonable running time. On the other hand, our algorithm finds 120 new upper bounds out of 240 instances (50%), matches the best solutions for 96 instances (40%) and only misses 24 best known values (10%). The results between HGA and HGA-Giant are significantly different on both medium-sized and large-sized instances, which indicates that E^2AX is more powerful than the giant tour crossover when solving the PCTSP. Furthermore, the

performance profiles confirm the superiority of HGA on both the best and average values compared to the reference algorithms.



(a) Set I (Best)

(b) Set I (Avg.)

(c) Set II (Best)

(d) Set II (Avg.)

(e) Set III (Best)

(f) Set III (Avg.)

Figure 4.3 – The PCTSP: performance profiles of the compared algorithms on the 80 instances of each set

From the detail results on the PCTSP shown in Tables 6.12-6.17, we make the following observations. First, under the same stopping condition, HGA requires only half of the time needed by HGA-Giant to find solutions of equal or better quality on the medium-sized instances. More importantly, HGA reaches better solutions than HGA-Giant by spending shorter running time for the large-sized instances. Second, although B&C can solve medium-sized instances optimally, the running time increases significantly with the increase of the instance size. For example, for Sets II and III, B&C fails to obtain the optimal solution of several instances with more than 400 vertices. Meanwhile, HGA can

Table 4.4 – Summary of results of HGA compared to the results of HGA-Giant (using the giant tour crossover) and HGA1 (without any crossover) on Sets II and III of the OP.

Instances	Pair algorithms	*Best*				*Avg.*			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Set II (86)	HGA vs. HGA-Giant	**51**	34	1	4.25E-09	**65**	18	3	3.65E-11
	HGA vs. HGA1	**81**	5	0	5.36E-15	**83**	3	0	2.50E-15
Set III (86)	HGA vs. HGA-Giant	**57**	27	2	1.55E-10	**68**	15	3	4.84E-12
	HGA vs. HGA1	**84**	2	0	1.71E-15	**85**	1	0	1.17E-15

find high-quality solutions for large-sized instances within a short running time.

4.4 Additional experiments

In this section, we conduct additional experiments to study the benefits of two key components of the proposed algorithm. The experiments are based on the instances of Sets II and III of the OP.

4.4.1 Significance of the crossover

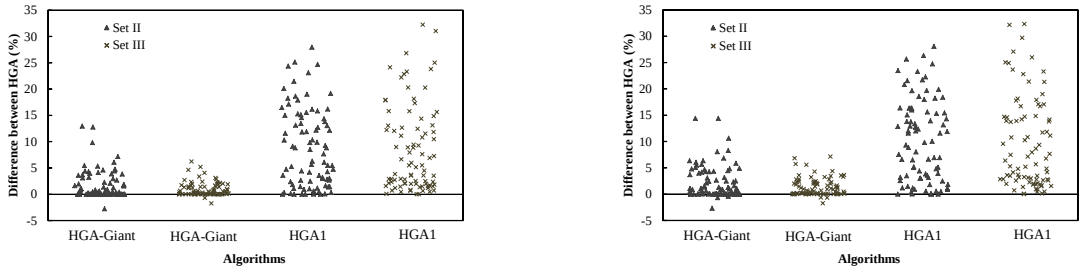
To assess the significance of the E^{2}AX crossover within the HGA algorithm, we create a HGA variant (HGA-Giant) where E^{2}AX is replaced by the giant tour crossover [BDM10] (see Appendix 6.2) and another HGA variant (HGA1) where the E^{2}AX crossover is disabled in HGA. We run these algorithms under the same stopping condition as before and report the comparative results in Table 4.4 and Fig. 4.4.

From these results, one observes that the E^{2}AX crossover plays a highly positive role in the good performance of HGA. Indeed, HGA dominates HGA-Giant by obtaining 108 better results and 61 equal results out of the 172 tested instances. HGA1 (without crossover) has the worst performance even compared to HGA and HGA-Giant, indicating that crossovers such as E^{2}AX and giant tour are highly useful for the performance of the hybrid algorithm.

To sum, we conclude that E^{2}AX positively contributes to the performance of HGA, and it also outperforms the giant tour crossover.

4.4.2 Benefits of the mutation

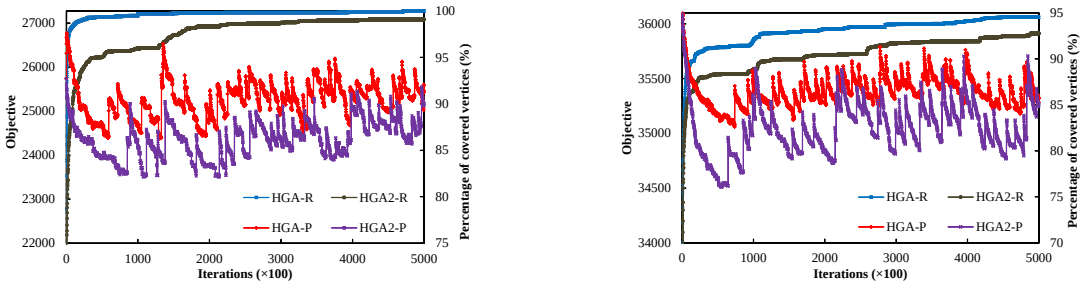
In the HGA algorithm, the mutation operator is used as a means to preserve diversity of the population. To assess its usefulness, a HGA variant (HGA2) is created by disabling the mutation operator. We compare HGA and HGA2 in terms of population diversity by using the following diversity measure. Let $|\mathcal{P}|$ be the number of solutions in the population $\mathcal{P}$. Let $\mathcal{N}_\varphi$ be the set of vertices visited by solution φ in $\mathcal{P}$. Let $\mathcal{H}$ be the set of vertices visited by all the solutions in $\mathcal{P}$ and $\mathcal{H} = \cup_{i=1}^{|\mathcal{P}|}\mathcal{N}_{\varphi_i}$. Let ξ be the proportion of vertices covered by $\mathcal{P}$ and $\xi = \frac{|\mathcal{H}|}{n}$, $0 < \xi \leq 1$. We use the value of ξ to measure the diversity of



(a) Best **(b)** Avg.

Figure 4.4 – The differences between HGA and two variants for solving the instances of Sets II and III of the OP.

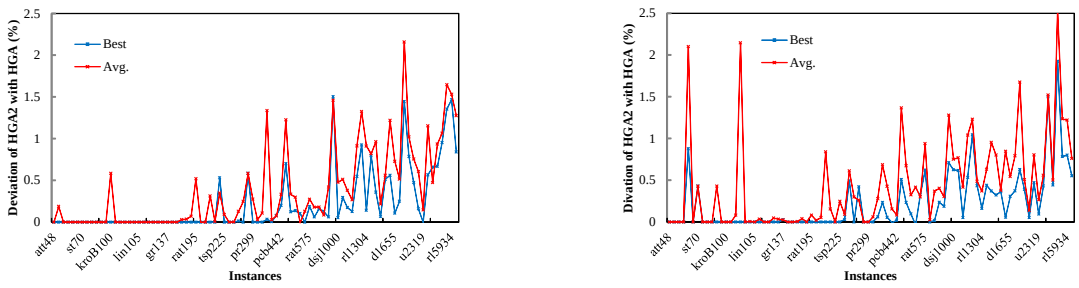
the population. If $\xi \to 1$, it means $\mathcal{P}$ covers many vertices, offering good possibilities for the algorithm to explore larger search spaces, and vice versa. We present the convergence charts of HGA and HGA2 together with the evolution of the population diversity, based on two instances (rat783-gen3 and u1060-gen2). The results are shown in Fig. 4.5, where HGA-R and HGA2-R indicate the best results found while HGA-P and HGA2-P are the current diversity values ξ of the population. One notes that HGA has a better convergence and dominates its counterpart in both instances. It's observed that HGA always keeps a higher value ξ along its evolution compared to HGA2, which indicates the contributions of the mutation to the diversity and the performance of the HGA algorithm.



(a) rat783-gen3 **(b)** u1060-gen2

Figure 4.5 – Convergence charts of HGA and HGA2 for solving two representative instances.

Finally, Fig. 4.6 shows the comparative results of HGA and HGA2 in terms of both the best and average objective values on the 86 instances of Set II and 86 instances of Set III (the names of 15 instances are shown). The results are presented as the deviation in percentage of the results of HGA2 compared to the results of HGA. For the medium-sized instances, HGA and HGA2 obtains similar results. However, for instances with more than 200 vertices, HGA2 performs worse than HGA and the difference becomes more significant



(a) Set II **(b)** Set III

Figure 4.6 – Results of HGA2 (without mutation) in terms of deviation in percentage compared to the results of HGA (with mutation).

as the size of instances increases. These results confirm that the mutation operator plays a crucial role in the HGA algorithm, especially for large-sized instances.

4.5 Chapter conclusion

This chapter proposed a new hybrid genetic algorithm to efficiently address two traveling salesman problems with profits. We introduced several methodological contributions including an extended edge assembly crossover for producing promising solutions, an effective local search for solution refinement and specific strategies for diversity preservation of the population.

Extensive experiments were conducted on the orienteering problem and the prize-collecting traveling salesman problem. For the OP, four sets of 344 commonly used instances were tested and 67 new lower bounds were discovered. The algorithm also matches the best known results for 172 other instances. For the PCTSP, results on three sets of 240 instances showed a high performance on large-sized instances including 120 new best results never reported in the literature. Additional experiments were conducted to get insights into the benefits of the proposed crossover and the mutation. The new bounds reported in this work can be useful for future research on these problems. Moreover, the code of our algorithm that we make available can be used by researchers and practitioners.

In the next chapter, we present a general edge assembly crossover operator for solving the split delivery vehicle routing problem.

Chapter 5

General edge assembly crossover driven memetic search for the split delivery vehicle routing problem

---

In this chapter, we present an effective memetic algorithm for solving the split delivery vehicle routing problem with a fleet of limited or unlimited vehicles. The algorithm features a general edge assembly crossover to generate promising offspring solutions from the perspective of assembling suitable edges and an effective local search to improve each offspring solution. The algorithm is further reinforced by a feasibility-restoring procedure, a diversification-oriented mutation and a quality-and-distance pool updating technique. Extensive experiments on 324 benchmark instances indicate that our algorithm is able to update 143 best upper bounds in the literature and match the best results for 156 other instances. Additional experiments are presented to obtain insights into the roles of the key search ingredients of the algorithm. The method was ranked second at the 12th DIMACS Implementation Challenge on Vehicle Routing - SDVRP Track. The content of this chapter is based on an article submitted to *Transportation Science*.

5.1 Introduction

Like the conventional VRP, the SDVRP has many applications such as determining routes and schedules for newspaper delivery [SLK02] and waste collection [AS04]. Meanwhile, the SDVRP has been much less investigated compared to the VRP and its variants such as the capacitated VRP, the VRP with time windows and the VRP with profits. Still, since the introduction of the SDVRP, a number of algorithms using exact and heuristic approaches have been proposed. Representative exact algorithms are based on various formulations [BMM00; OKY18] and the branch-and-cut framework [ABS14; MS22]. These exact approaches are able to provide the optimal solutions for some small or medium-sized instances with up to some 100 customers. For larger instances, heuristics and metaheuristics are preferred to find suboptimal solutions with a reasonable time, as reviewed in Section 1.5.2.

This chapter aims to advance the state-of-the-art for solving large SDVRP instances effectively and efficiently. The contributions of this paper are summarized as follows.

1. The proposed memetic algorithm (SplitMA) [1] combines several complementary search components including a general edge assembly crossover (gEAX) to generate promising offspring solutions and a local search associated with a maximum splits strategy to improve offspring solutions. The gEAX crossover transmits common edges from parent solutions to offspring solutions while reassembling non-common edges of parent solutions. The local search exploits both VRP neighborhood operators and SDVRP neighborhood operators reinforced by the maximum splits strategy, which ensures that a customer will not be served by too many vehicles. The algorithm additionally integrates dedicated repairing techniques to ensure the feasibility of offspring solutions, a mutation to diversify each new solution, and an advanced updating strategy to maintain a healthy population.
2. We illustrate the competitiveness of the algorithm on four sets of 324 instances of the SDVRP-LF and SDVRP-UF problems compared to the state-of-the-art algorithms. In particular, we report 143 new best upper bounds that can be useful for future studies. We investigate the underlying algorithmic components to shed light on their contributions to the performance of the algorithm. Specifically, we provide insights about why the gEAX crossover works well on the SDVRP and present for the first time experimental evidences that high-quality solutions are close to each other and are also close to optimal solutions.
3. This work shows the interest of the general idea of the edge assembly crossover. The gEAX crossover, which generalizes the popular EAX crossover for the TSP nagata1997edge,nagata2013powerful, provides a powerful solution recombination mechanism that can be advantageously applied not only to the SDVRP, but also to other routing problems where the associated graphs of candidate solutions do not necessarily have the same degree for their vertices.

1. The SplitMA algorithm was ranked second at the 12th DIMACS Implementation Challenge on Vehicle Routing - SDVRP Track (http://dimacs.rutgers.edu/programs/challenge/vrp/).

The remainder of this chapter is organized as follows. Section 5.2 presents the details of the proposed algorithm. Section 5.3 shows computational results and comparisons. Section 5.4 investigates key ingredients of the proposed algorithm. Section 5.5 draws conclusions.

5.2 General edge assembly crossover driven memetic algorithm

Population-based evolutionary algorithms have been successfully applied to the traveling salesman problem [NK97; NK13] and several vehicle routing problems [NB09; NBD10; Pot09; Pri04; Vid+12; Vid+13; Vid+14]. The proposed SplitMA algorithm for the SDVRP is a population-based hybrid algorithm that uses a dedicated edge assembly crossover to generate new solutions and an effective local optimization to improve the offspring solutions. SplitMA also applies a mutation to diversify each offspring solution and an advanced pool updating strategy to manage the population.

Algorithm 7 The memetic algorithm for the SDVRP

Input: Instance I;
Output: The best solution φ^* found so far;

begin
  $\mathcal{P} \leftarrow PopulationInitial(I)$; /* Initializing the population $\mathcal{P}$, Section 5.2.1 */
  $\varphi^* \leftarrow \arg\min\{f(\varphi_i)|i = 1, 2, \cdots, |\mathcal{P}|\}$; /* φ^* Record the best solution found so far */
  while *Stopping condition is not met* **do**
    $\{\varphi_A, \varphi_B\} \leftarrow ParentSelection(\mathcal{P})$; /* Selecting two parental solutions randomly */
    $\{\varphi_O^1, \varphi_O^2, \cdots, \varphi_O^\beta\} \leftarrow gEAX(\varphi_A, \varphi_B)$; /* Generating offspring solutions, Section 5.2.2 */
    for $i = 1$ *to* β **do**
      $\varphi_O^i \leftarrow RestoringFeasibility(\varphi_O^i)$; /* Restoring feasibility, Section 5.2.3 */
      $\varphi_O^i \leftarrow Mutation(\varphi_O^i)$; /* Generating mutation, Section 5.2.4 */
      $\varphi_O^i \leftarrow LocalSearch(\varphi_O^i)$; /* Improving the offspring solution, Section 5.2.5 */
      if *SDVRP-LF* **then**
        $\varphi_O^i \leftarrow EmptyRoute(\varphi_O^i)$; /* Reducing routes to K_{min}, Section 5.2.5 */
      end
      if $f(\varphi_O^i) < f(\varphi^*)$ **then**
        $\varphi^* \leftarrow \varphi_O^i$;
      end
      $\mathcal{P} \leftarrow PoolUpdating(\mathcal{P}, \varphi_O^i)$; /* Managing the population, Section 5.2.6 */
    end
  end
  return φ^*;
end

The general scheme of SplitMA is outlined in Algorithm 7. SplitMA starts from an

initial population $\mathcal{P}$ constructed by the population initialization procedure (Line 2 of Algorithm 7). Then the algorithm evolves the population through a number of generations by applying the crossover operator, the local optimization procedure and the population updating procedure (Lines 4-19). Of particular interest is the general edge assembly crossover operator (gEAX) (Line 6) that creates at each generation β offspring solutions by assembling the edges of two parent solutions. After restoring the feasibility of each offspring solution in terms of customer demand and vehicle capacity (Line 8), the solution is diversified by the mutation operator (Line 9) and then submitted to local optimization for quality improvement (Line 10). Finally, each improved solution is used to update the population by the pool updating strategy (Line 17). For the SDVRP-LF where the fleet size is set to K_{min}, the number of the used vehicles is reduced to this fleet size by emptying some routes if needed (Lines 11-13). During the search, the best solution found so far φ^* is updated each time a solution than it is discovered (Lines 14-16). The algorithm stops and returns the best solution φ^* when a predefined stopping condition is met (e.g., a maximum cutoff time or maximum number of generations).

5.2.1 Population initialization

SplitMA starts its evolution from an initial population $\mathcal{P}$, whose size varies between p_{min} and p_{max} ($p_{max} > p_{min}$) during the search process. Similar to [Vid22], $4 \times p_{min}$ solutions are first constructed and subsequently improved by the local search (Section 5.2.5), and then inserted into $\mathcal{P}$ one by one. Once $|\mathcal{P}| = p_{max}$, the surviving strategy (Section 5.2.6) is triggered to shrink the population $\mathcal{P}$ to p_{min} solutions.

The construction process of each solution works as follows. First, $K_{min} = \lceil (\sum_{i=1}^{n} d_i/Q) \rceil$ routes are created where each route is initialized by the depot and a random customer. Then, for each newly routed customer i, a random unrouted customer j from the δ-nearest neighborhood (see Section 5.2.5) is selected and inserted into the route after the customer i without split. This insertion process stops when no customer can be inserted into the solution without violating the capacity constraint. Finally, if there are unrouted customers, these customers are dividedly inserted into routes in a greedy way such that the insertions lead to the minimum increase of the objective value (i.e., the total traveling distance). Once all customers are routed, a complete solution is obtained.

5.2.2 The general edge assembly crossover operator

Crossover is a key component of memetic algorithms and constitutes one leading force to explore the search space [Hao12]. In this section, we introduce the gEAX crossover for the SDVRP that generalizes the edge assembly crossover (EAX) designed for the VRP [NB09], which itself comes from the popular EAX crossover initially designed for the TSP.

The main difficulty of applying EAX to the SDVRP lies in the fact that EAX assumes that each customer is served by exactly one vehicle. Indeed, for a given TSP and VRP instance defined on a graph $\mathcal{G}$, a candidate solution can be identified by a partial graph of

graph $\mathcal{G}$. Given two parent solutions, each customer vertex necessarily has the same degree of two and EAX uses this property to assemble edges from the parents. However, for the SDVRP, given that each customer can be served by several vehicles, a solution corresponds to a multigraph where parallel edges may exist between two vertices (see Definition 5.2.2). Indeed, given the assumption that triangle inequity holds, each edge between customers is traversed at most once in the optimal solution. However, each edge between the depot and a customer may still be traversed several times. Without loss of generality, we use the term 'vertex' to denote both 'depot' and 'customer' in this paper. As a result, the same customer vertex may have different degrees in the multigraphs of the parent solutions, making the EAX crossover inoperative. On the other hand, the idea of assembling specific (promising) edges from the routes of high-quality solutions is highly appealing from the perspective of solution recombination. The general edge assembly crossover gEAX that we introduce in this work benefits from the basic idea of assembling suitable edges and gets around the aforementioned difficulty related to the EAX crossover.

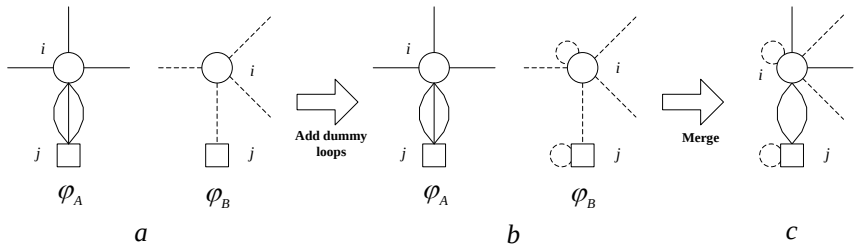


Figure 5.1 – Illustration of adding dummy edges. (a) A portion of the multigraphs $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$ associated to solutions φ_A and φ_B. (b) multigraph $\mathcal{G}_\mathcal{A}$ and extended multigraph $\mathcal{G}_\mathcal{B}$ with two dummy loops. (c) Joint multigraph of $\mathcal{G}_\mathcal{A}$ and extended $\mathcal{G}_\mathcal{B}$.

The key idea of the gEAX crossover is to ensure that each vertex has the same degree in the multigraphs of the parent solutions by introducing dummy edges, rendering it possible to apply the edge assembling operations. To describe the gEAX crossover, we first introduce the following notations.

For a SDVRP instance on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let φ be a solution composed of K routes. Following the notation used in Section 1.5.1, let x_{ij}^k be a Boolean variable such that $x_{ij}^k = 1$ if route (or vehicle) k goes from vertex i to vertex j and $x_{ij}^k = 0$ otherwise. Then $x_{ij}(\varphi) = \sum_{k=1}^{K} x_{ij}^k$ is the number of times edge (i, j) is traversed in the solution φ and $x_{ij}(\varphi) \geq 1$ holds for each edge (i, j). For example, in Fig. 5.1(a) (the square is the depot j and the circle represents customer i), three vehicles (say k_1, k_2 and k_3) of solution φ_A (solid lines) go through the edge (i, j). These three distinct traversals on (i, j) are identified as $x_{ij}^{k_1} = 1$, $x_{ij}^{k_2} = 1$ and $x_{ij}^{k_3} = 1$. Thus $x_{ij}(\varphi_A) = 3$. For solution φ_B (dot lines), there is only one route k passing through the edge (i, j), thus $x_{ij}^k = 1$ and $x_{ij}(\varphi_B) = 1$.

For a solution φ of the SDVRP instance on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define its corresponding multigraph $\mathcal{G}_\varphi = (\mathcal{V}, \mathcal{E}_\varphi)$ with the multiset of parallel edges $\mathcal{E}_\varphi$ such that for an edge (i, j) of $\mathcal{E}$, there are $x_{ij}(\varphi)$ parallel edges in $\mathcal{E}_\varphi$.

Fig. 5.1(a) shows a portion of the multigraphs associated to solutions φ_A and φ_B. For solution φ_A, there are three parallel edges between the depot j and the customer i, because three vehicles traverse edge (i, j).

Given two solutions φ_A and φ_B, let $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ be the corresponding multigraphs. The *degree difference* of vertex i in $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$ is $\Delta_i = |deg_\mathcal{A}(i) - deg_\mathcal{B}(i)|$ where $deg_\varphi(i)$ denotes the degree of vertex i in solution φ. For a vertex i, if $\Delta_i \neq 0$, $\mathcal{G}_\mathcal{A}$ or $\mathcal{G}_\mathcal{B}$ is extended by adding one or more dummy loops (i, i) to the vertex to render $\Delta_i = 0$.

In the example of Fig. 5.1(a), $\Delta_i = |deg_\mathcal{A}(i) - deg_\mathcal{B}(i)| = 6 - 4 = 2$ and $\Delta_j = |deg_\mathcal{A}(j) - deg_\mathcal{B}(j)| = 3 - 1 = 2$. Thus, $\mathcal{G}_\mathcal{B}$ is extended by dummy loops (i, i) and (j, j) as shown in see Fig. 5.1(b). In what follows, an edge $e \in \mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}$ is called a common edge of φ_A and φ_B if $e \in \mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}$; otherwise, e is a non-common edge.

Given two solutions φ_A and φ_B, let $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ be their extended multigraphs such that $\Delta_i = 0$ holds for each vertex i, we define the joint multigraph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, \{\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}\} \backslash \{\mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}\})$ by the symmetric difference of $\mathcal{E}_\mathcal{A}$ and $\mathcal{E}_\mathcal{B}$.

Fig. 5.1(c) shows the joint multigraph $\mathcal{G}_{\mathcal{AB}}$ associated to two solutions φ_A and φ_B.

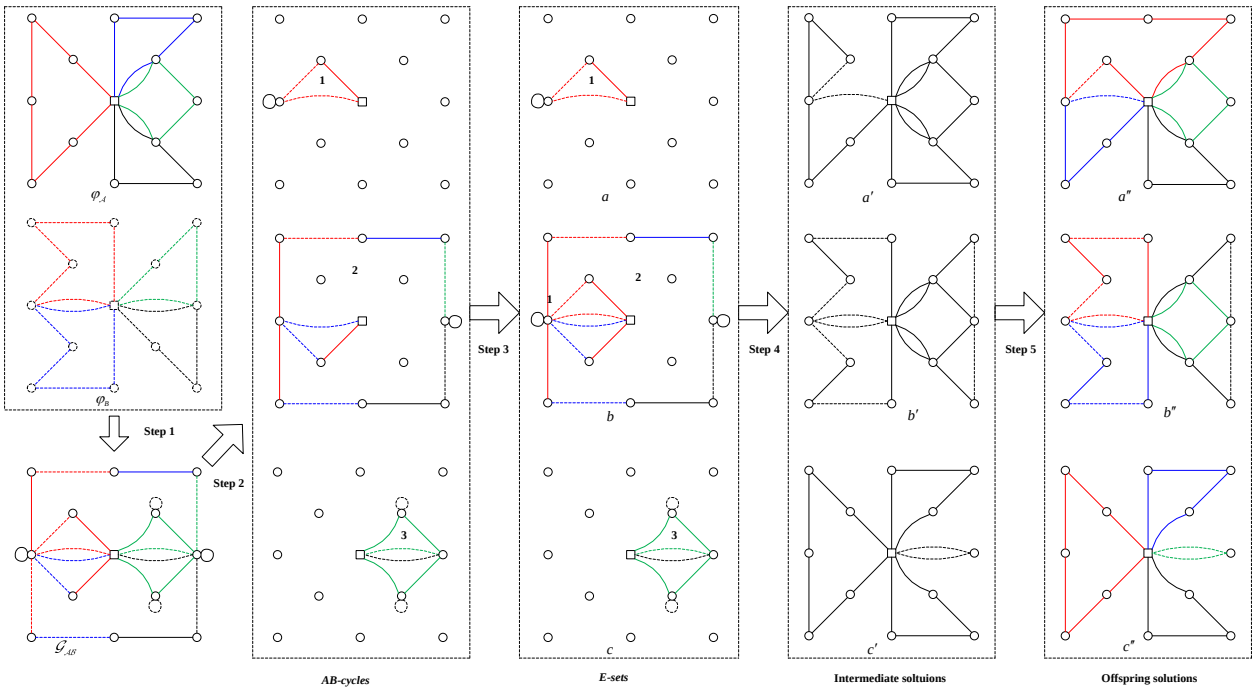


Figure 5.2 – Illustration of the gEAX crossover.

Given two solutions φ_A, φ_B as well as their corresponding multigraphs $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$, the proposed gEAX crossover generates several offspring solutions in five steps (see Fig. 5.2 for an illustrative example).

1. **Addition of dummy loops and generation of graph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, \mathcal{E}_{\mathcal{AB}})$.** At the beginning, dummy loops are added to make the *degree difference* become 0 for all vertices in the multigraphs $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$. Specifically, for each vertex i, the number

of added dummy loops (i, i) is $\frac{|deg_{\mathcal{A}}(i)-deg_{\mathcal{B}}(i)|}{2}$. If $deg_{\mathcal{A}}(i) > deg_{\mathcal{B}}(i)$, dummy loops are added into $\mathcal{E}_{\mathcal{B}}$, and vice versa, as illustrated in Fig. 5.1(b). Once the *degree difference* becomes 0 for all vertices in the multigraphs $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$, we create the joint multigraph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, \mathcal{E}_{\mathcal{AB}})$ with $\mathcal{E}_{\mathcal{AB}} = \{\mathcal{E}_{\mathcal{A}} \cup \mathcal{E}_{\mathcal{B}}\} \backslash \{\mathcal{E}_{\mathcal{A}} \cap \mathcal{E}_{\mathcal{B}}\})$. In the example of Fig. 5.2, four dummy loops are added.

2. **Generation of *AB-cycles***. From the joint multigraph $\mathcal{G}_{\mathcal{AB}}$, a number of *AB-cycles* are generated where each new *AB-cycle* is constructed as follows. A random vertex is selected to initialize an empty *AB-cycle*; then edges from $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$ are traced alternatively to extend the ongoing *AB-cycle*, and each traced edge is removed from $\mathcal{G}_{\mathcal{AB}}$; the *AB-cycle* is constructed successfully when the traced edges lead to a cycle. After the construction of the current *AB-cycle*, if $\mathcal{G}_{\mathcal{AB}}$ is not empty, the process continues to build the next *AB-cycle*. The process stops and returns all *AB-cycles* once $\mathcal{G}_{\mathcal{AB}}$ becomes empty. As shown in Fig. 5.2, three *AB-cycles* are generated from $\mathcal{G}_{\mathcal{AB}}$. One notices that each *AB-cycle* contains at least four edges. Let C denote the set of m *AB-cycles* obtained from this step.
3. **Generation of *E-sets***. From the set of m *AB-cycles* $C = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$, a set of *E-sets* is created, where an *E-set* is an union of *AB-cycles*. Each new *E-set* $\mathcal{E}_i$ is initialized by an *AB-cycle* $\mathcal{C}'$ in C and $\mathcal{C}'$ is removed from C. Then, each remaining *AB-cycle* $\mathcal{C}''$ of C are checked. If $\mathcal{C}''$ shares at least one vertex with $\mathcal{E}_i$, $\mathcal{C}''$ is added to $\mathcal{E}_i$ and removed from C. A complete *E-set* ($\mathcal{E}_i$) is achieved when no *AB-cycles* can be added into $\mathcal{E}_i$. This process stops when no *AB-cycle* is left (i.e., C becomes empty). In the example of Fig. 5.2, the three *AB-cycles* should be combined to form one single *E-set* since the depot is shared. However, for illustrative purpose of steps 4 and 5 below, we suppose three *E-sets* as shown in Fig. 5.2. Let E denote the set of *E-sets* obtained from this step.
4. **Generation of intermediate solutions**. For each *E-set* $\mathcal{E}_i$ of E, an intermediate solution is generated by using a random parent (say φ_A) as the basic solution. The dummy loops in the *E-sets* $\mathcal{E}_i$ are first removed. Then, the intermediate solution φ'_i is constructed based on φ_A by removing from it the edges of $\mathcal{E}_{\mathcal{A}}$ shared with $\mathcal{E}_i$ and adding the edges of $\mathcal{E}_{\mathcal{B}}$ shared with $\mathcal{E}_i$, that is, $\varphi'_i \leftarrow (\mathcal{E}_{\mathcal{A}} \setminus (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{A}})) \cup (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{B}})$. Such a strategy guarantees that all common edges in φ_A and φ_B are necessarily inherited by intermediate solutions. Moreover, all edges in intermediate solutions come from parent solutions. Fig. 5.2($a' - c'$) illustrate the three intermediate solutions from this step.
5. **Elimination of isolated subtours**. An intermediate solution may include one or more isolated subtours, such as the triangle subtour in the upper left corner of Fig. 5.2(a'). The 2-opt* heuristic [PR95] is then adopted to eliminate these subtours. For each randomly selected subtour, an edge is removed from the subtour and an edge is removed from another route. Then two new edges are introduced to connect two routes. This process is exactly the same as the M8 and M9 introduced in Section 5.2.5. Fig. 5.2(a'') illustrates the offspring solution after subtour elimination in the solution of Fig. 5.2(a').

The complexity of gEAX can be summarized as follows. Suppose without loss of generality that $|\mathcal{E}_\mathcal{A}| \geq |\mathcal{E}_\mathcal{B}|$. In the first four steps, there are $|\mathcal{E}_\mathcal{A}|+|\mathcal{E}_\mathcal{B}|$ edges involved, leading to a time complexity of $|\mathcal{E}_\mathcal{A}|$. For the fifth step, the time complexity of 2-opt* is $\mathcal{O}(n \times \delta)$, where δ is a parameter (Introduced in Section 5.2.5). Thus, the time complexity of gEAX is $\mathcal{O}(n \times \delta)$. Moreover, $|\mathcal{E}_\mathcal{A}|$ edges are invoked and thus the space complexity is $\mathcal{O}(|\mathcal{E}_\mathcal{A}|)$.

The gEAX crossover follows the idea of the EAX crossover initially designed for the VRP [NB09] and inherits its advantages, while relaxing the customer demand and capacity constraints. A pair of solutions can generate a variety of offspring solutions with relatively short edges from the parent solutions. More importantly, gEAX overcomes the limitation of EAX that parent solutions (precisely their multigraphs) need to possess the same degree for each vertex. As we show in Sections 5.3 and 5.4.1, gEAX significantly contributes to the performance of the proposed algorithm. In Section 5.4.2, we provide experimental evidences to understand why gEAX is a meaningful crossover for the SDVRP. Finally, the idea behind gEAX also provides a basis for designing meaningful edge assembly crossovers for other rich routing problems such as team orienteering, location routing as well as arc routing.

5.2.3 Restoring the feasibility of offspring solutions

The customer demand and vehicle capacity are ignored during the gEAX crossover process. As such, an offspring solution may be infeasible in terms of these constraints. This section describes how the feasibility of an offspring solution is restored.

Restoring customers' demand

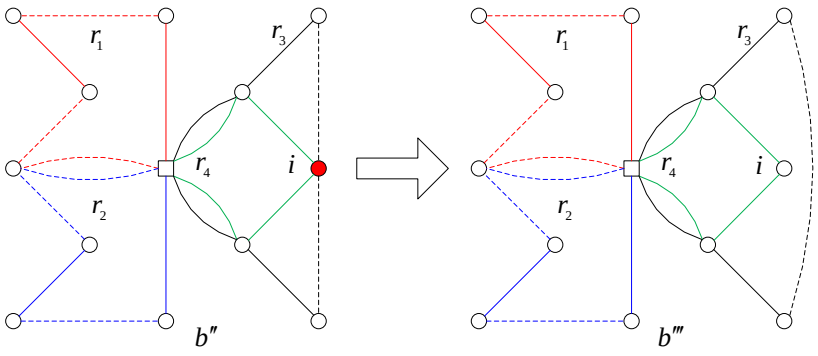


Figure 5.3 – Illustration of balancing demands

When the routes from the parent solutions are recombined by gEAX, the total amount of served demand of a customer in an offspring solution can be different from the customer's demand. Suppose that $d_i(r_k)$ is the served demand of customer i by route r_k. For example, for the offspring b'' of Fig. 5.3, customer i (denoted by the red dot) is visited by two routes r_3 and r_4 with the total amount of served demand $d_i(r_3) + d_i(r_4)$. However,

since route r_4 in solution b'' entirely comes from φ_A that serves the full demand d_i already, we have $d_i(r_3) + d_i(r_4) > d_i$. Thus, for each customer i, we need to adjust the demand distribution among the routes visiting the customer and make sure that $\sum_{k=1}^{K} d_i(r_k) = d_i$.

We distinguish two cases (i) $\sum_{k=1}^{K} d_i(r_k) > d_i$, and (ii) $\sum_{k=1}^{K} d_i(r_k) < d_i$. Let d_{r_k} be the total load of route r_k. For the first case, the capacity excess $d_{r_k} - Q$ (Q is the vehicle capacity) of each route r_k visiting customer i is calculated, and the resulting values are sorted from the largest to the smallest. Then, the route r_k with the largest capacity excess is identified. If $\sum_{k=1}^{K} d_i(r_k) - d_i > d_i(r_k)$, the customer i is removed from route r_k. Otherwise the amount of demand $d_i(r_k) - (\sum_{k=1}^{K} d_i(r_k) - d_i)$ is removed from route r_k, and the demand of customer i is restored, that is $\sum_{k=1}^{K} d_i(r_k) = d_i$. This process is looped until the demand of all customers is restored. For the second case, the process is similar and operates with the residual capacity of $Q - d_{r_k}$.

Restoring the capacity constraints

In addition to the customer demand, the offspring solutions generated by the gEAX crossover may violate the capacity constraint as well. To restore the capacity feasibility of an offspring solution, we apply two well-known inter-route move operators (i.e., insert* and 2-opt*).

Specifically, let φ be an infeasible offspring solution and $f_c(\varphi)$ be its fitness as defined by $f_c(\varphi) = f(\varphi) + p_c \times f_p(\varphi)$, where $f(\varphi)$ is the traveling cost, $f_p(\varphi)$ is the total over-capacity in solution φ, and p_c is a penalty parameter initialized to be the ratio between the longest edge and the largest demand. The repair process operates on an overcapacitated route r and uses insert* [ASH06] and 2-opt* (2-opt* corresponds to M8 and M9 of Section 5.2.5) to repair the route. During this process, a tabu list is used to prevent a performed move from being reversed. After each repair operation involving two routes, the set of infeasible routes $\mathcal{R}_{inf}$ is updated. The penalty parameter p_c is multiplied by 10 if no feasible move can be found while there are still infeasible routes ($\mathcal{R}_{inf} \neq \emptyset$). The procedure continues until all routes becomes feasible ($\mathcal{R}_{inf} = \emptyset$), and returns the repaired solution φ.

5.2.4 Mutation

Given that an offspring solution inherits exclusively the edges of its parents, it may resemble much the parents even after the feasibility restoring operations. To introduce some diversity into an offspring solution, we modify the solution with a probability p_m with the removal operator presented in [Sha98]. Basically, this operator deletes some customers from their routes and then greedily reinserts these customers into the solution while respecting the capacity constraint.

Specifically, the mutation removes a number of customers that are similar with respect to a predefined characteristic (e.g., location or demand). In this work, we use the distance between customers to define the similarity. The mutation works in two steps as follows. Firstly, a random customer i in route r_k with its served demand $d_i(r_k)$ is selected to

initialize set $\mathcal{C}$. Then, the similarity between customer i and other customers ($\mathcal{N} \setminus \mathcal{C}$) is calculated and sorted in ascending order, where the first customer has the maximum similarity. A customer with its served demand in the route is selected with the roulette-wheel selection and saved in set $\mathcal{C}$ subsequently. For each selected customer i, if it is visited by more than one route, a random route is retained. The first step terminates when l customers are considered ($|\mathcal{C}| = l$) (l is a parameter called the mutation length) . More details about this step can be found in [RP06]. The second step reinserts greedily the removed customers of set $\mathcal{C}$. For each customer $i \in \mathcal{C}$, a customer $j \in \mathcal{N} \setminus \mathcal{C}$ from its δ-nearest neighborhood is selected, and the customer i is inserted after the customer j with respect to the capacity constraint and the minimum traveling distance. This procedure terminates when all customers in $\mathcal{C}$ are inserted into the solution. The worst-case time complexity of the mutation is $\mathcal{O}(l \times \delta)$.

5.2.5 Local search

Local search is among the core components of the state-of-the-art heuristic algorithms for several related VRPs. Enriched neighborhood operators, exploration strategies, and speed-up techniques have been developed to allow the local search to attain high-quality solutions within a limited time. The local search procedure of SplitMA for the SDVRP adopts nine popular VRP neighborhood operators used in [Vid22], including eight inter-route and one intra-route structures. To reinforce its search capacity, our local search additionally employs four tailored SDVRP neighborhood operators proposed in [BPR07] and [DT89; DT90]. These 13 operators are explored under the framework of variable neighborhood descent according to the order in which they are presented in the forthcoming subsections.

Before introducing the neighborhood operators, we first present three application rules. The first rule is that once an improvement occurs with an inter-route structure, the procedure checks whether a vehicle visits some customers twice. If so, the duplicated visits with the largest distance reduction are removed. The second rule defines the neighborhood of each customer as the δ-nearest vertices, where δ ($\delta < |\mathcal{N}|$) is the granularity threshold restricting the search to nearby vertices. This rule aims to avoid the examination of non-promising neighboring solutions and speeds up the local search. The last rule is that the first improvement strategy is adopted to explore each neighborhood.

To present the different neighborhood operators, we adopt the following notations. $r(u)$ and $r(v)$ denote the routes which visit vertices u and v, respectively. Let v be a neighbor of u, and x and y the successors of u in $r(u)$ and v int $r(v)$, respectively. (u, x) is the substring from vertex u to x, and (v, y) is the substring from vertex v to y.

VRP neighborhood operators

We first summarize the nine commonly used VRP neighborhood operators, named as M1–M9. Detailed presentations of these operators are provided in [Vid22]. Basically, M1–M3 are based on the insertion operation and M4-M6 use the interchange (or swap)

operation. M7 is the classical 2-opt for intra-route move, while M8 and M9 apply 2-opt* [PR95] for inter-route optimization.

— M1: If u is a customer visit, remove u from route $r(u)$ and place u after v;
— M2: If u and x are customer visits, remove them from route $r(u)$ and place (u, x) after v;
— M3: If u and x are customer visits, remove them from route $r(u)$ and place (x, u) after v;
— M4: Interchange u and v if they are customer visits;
— M5: Interchange (u, x) and v if they are customer visits;
— M6: Interchange (u, x) and (v, y) if they are customer visits;
— M7: This is 2-opt. If $r(u) = r(v)$, replace (u, x) and (v, y) by (u, v) and (x, y);
— M8: This is 2-opt*. If $r(u) \neq r(v)$, replace (u, x) and (v, y) by (u, v) and (x, y);
— M9: This is 2-opt*. If $r(u) \neq r(v)$, replace (u, x) and (v, y) by (u, y) and (v, x).

SDVRP inter-route neighborhood operators

We describe now the four inter-route neighborhood operators M10–M13 specifically designed for the SDVRP [BPR07; DT89].

— M10: This operator extends M4 by modifying the amounts to be delivered to customers with respect to the capacity constraint. Suppose that customers u and v (customer v is a neighbor of customer u) are visited on two distinct routes, that is $r(u) \neq r(v)$. There are two cases: (i) if $d_u(r(u)) > d_v(r(v))$, then customer v with demand $d_v(r(v))$ is inserted before or after customer u in route $r(u)$, and a copy of u with $d_v(r(v))$ is inserted into route $r(v)$ at the position of customer v; (ii) if $d_u(r(u)) < d_v(r(v))$, customer u with $d_u(r(u))$ is inserted before or after customer v, while a copy of v with $d_u(r(u))$ is removed from route $r(v)$ and repositioned at the position of customer u in route $r(u)$. Please refer to [BPR07; SSO15] for a detailed description and illustration.
— M11: It extends M5 by adjusting the amounts to be delivered to customers while satisfying the capacity constraint. Suppose that customers u and v come from two different routes. Two cases are considered: (i) if $d_u(r(u)) + d_x(r(u)) > d_v(r(v))$ and $d_u(r(u)) < d_v(r(v))$, then customer u with $d_u(r(u))$ and a copy of x with $d_v(r(v)) - d_u(r(u))$ are interchanged with customer v with $d_v(r(v))$; (ii) if $d_u(r(u)) + d_x(r(u)) < d_v(r(v))$, customers u, x are inserted before or after v in route $r(v)$, and a copy of customer v with $d_u(r(u)) + d_x(r(u))$ is removed from $r(v)$ and replaced at the position of u in route $r(u)$. One notices that if $d_u(r(u)) + d_x(r(u)) = d_v(r(v))$, M11 becomes M5. A detailed description of M11 can be found in [BPR07; SSO15].
— M12 (RouteAddition): This operator was introduced by [DT89]. Firstly, suppose that a customer u is served by two routes $r(u)$ and $r'(u)$, and the customer u is removed from the routes and inserted in a new empty route. Then, four subtours of routes $r(u)$ and $r'(u)$ split by customer u are considered. The best component of combining these four route segments together with customer u is constructed to minimize the traveling cost, and three new routes are generated. Following

[DT89], we only consider the customer u involved in two or three routes to limit the computational complexity of exploring this neighborhood. For example, if customer u is visited by two routes, there are 9 components; however, if customer u is visited by three routes, there are 19 components.

— M13(k-Split): This operator was also introduced by [DT89]. It splits a customer and inserts the split demands into different routes with respect to the minimum move gain and capacity constraint. A greedy heuristic is adopted to find the best move quickly. For a detailed description, please refer to [SSO15].

Route elimination

For the SDVRP-LF, feasible solutions are limited to K_{min} vehicles. However, this constraint is relaxed during the mutation and local search with different neighborhood operators. In order to obtain feasible solutions after the local search, the k-Split neighborhood operator is employed to eliminate the least loaded route one by one until the number of routes equals K_{min}. For route elimination, we adopt the *EmptyRoutes* procedure presented in [SSO15].

Maximum splits per customer

Intuitively, to minimize the objective function, it is not desirable to split too much a customer's demand. As a result, in SplitMA, for each customer i, a maximum number of splits s_i is determined by $s_i = max\{s_{min}, \lceil \theta \times \frac{d_i}{Q} \rceil\}$, where θ is a control parameter and s_{min} sets the minimum of s_i, which prevents the maximum splits per customer from becoming too small. In SplitMA, we experimentally set $\theta = 50$ and $s_{min} = 5$, and apply the maximum splits strategy in neighborhood operators M10, M11 and M13. The benefits of this strategy are investigated in Section 5.4.4.

5.2.6 Population management

Population management is known as an important ingredient of successful memetic algorithms. SplitMA adopts a variable population scheme inspired by that used in [Vid+12].

The number of individuals in $\mathcal{P}$ varies between p_{min} and p_{max} ($p_{min} < p_{max}$) during the evolution process. Unlike the population management strategy used in [Vid+12], clone individuals are not allowed. Along with the evolution, the size of $\mathcal{P}$ increases since offspring individuals are progressively added to the population. Once $|\mathcal{P}| > p_{max}$, the surviving selection is triggered to remove p_{max} - p_{min} individuals by considering their contributions to the diversify of the population and traveling cost. Similar to [BPR07], the normalized Hamming distance h_{AB} between φ_A and φ_B is defined as the ratio between the number of non-common edges and the number of total edges in φ_A and φ_B, $h_{AB} = \frac{|\{\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}\} \setminus \{\mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}\}|}{|\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}|}$. Then, the biased fitness of each solution is calculated with respect to its initial fitness and diversity rank in $\mathcal{P}$.

If the best solution found so far φ^* cannot be improved during γ consecutive iterations, the algorithm restarts by generating a totally new population.

5.3 Computational results and comparisons

In this section, we report extensive experiments to evaluate the performance of SplitMA on popular benchmark instances (see Section 1.5.3]) in comparison with the state-of-the-art SDVRP algorithms in the literature.

5.3.1 Experimental protocol and reference algorithms

Parameter setting. The SplitMA algorithm involves six main parameters: the minimal population size p_{min}, the maximal population size p_{max}, the mutation probability p_m, the mutation length l, the granularity threshold δ and the maximum iterations without improvement γ. To tune these parameter, we applied the automatic parameter tuning package Irace [Lóp+16], leading to the setting shown in Table 5.1. This setting can be considered as the default setting of the SplitMA algorithm and is consistently used for our experiments.

Table 5.1 – Parameter tuning results.

Parameter	Section	Description	Considered values	Final value
p_{min}	5.2.1 and 5.2.6	minimal size of population	$\{10, 15, 20, 25, 30\}$	30
p_{max}	5.2.1 and 5.2.6	maximal size of population	$\{45, 50, 55, 60, 65, 70, 75\}$	60
p_m	5.2.4	mutation probability	$\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$	0.2
l	5.2.4	length of mutation	$\{0.05, 0.1, 0.15, 0.2, 0.25\}$	0.05
δ	5.2.5	granularity threshold	$\{10, 15, 20, 25, 30\}$	20
γ	5.2.6	maximum iterations without improvement	$\{5000, 10000, 15000, 20000, 25000\}$	10000

Reference algorithms. Following the review of Section 1.5.2, we adopt the following references for the comparative study.

— BKS. This indicates the best known solutions (best upper bounds) that are compiled from all reference heuristic and exact approaches [ABS14; MS22; OKY18].

— SplitILS. This multistart iterated local search algorithm was proposed by [SSO15] for solving the SDVRP-LF and SDVRP-UF. It remains one of the current best SDVRP algorithms. The algorithm was implemented in the C++ language and executed on an Intel Core i7 2.93 GHz with 8.0 GB of RAM memory running Linux. Each instance was executed 20 times with distinct seeds under the single thread. The stopping condition is the maximum iterations given by $\min\{K_{min} \times n, 5000\} \times 10$.

— iVNDiv. The algorithm was proposed by [AH10] for solving the SDVRP-LF only. The algorithm was implemented in the C# language and executed on a Pentium 4, 2.8 GHz with 512 MB of RAM. The stopping condition is a maximum number of iterations.

— RGTS. This random granular tabu search algorithm was proposed by [BGN14] for solving the SDVRP-LF and SDVRP-UF. It was written in C++ and executed on a personal computer with 2.10 GHz and 4 GB RAM. The algorithm stops when the given number of non-improving moves is met.
— SS. This scatter search algorithm was proposed by [CCM08] for solving the SDVRP-LF only. It was encoded by C and executed on a Pentium IV, 2.4 GHz, 1 GB RAM. The algorithm stops when the reference set remains unchanged after combining all the solutions or the maximum number of iterations is reached.
— HGA. The hybrid genetic algorithm was presented by [WC12] and tested on some instances of Set I and Set IV. It was implemented in FORTRAN 95 and executed on an Intel Xeon 2.94 GHz with 8 GB RAM.
— TSVBA. The tabu search with vocabulary building approach was proposed by [AH10] for solving the SDVRP-UF. It was implemented in C# and run on a Pentium 4, 2.8 GHz, 512 MB of RAM. The algorithm stops when a predefined number of iterations without improving is reached.
— FBTS. The forest-based tabu search was proposed by [Zha+15]. For solving the SDVRP-UF. It was written in C++ and executed on an Intel i5-2410 2.3 GHz, 4 GB RAM. The algorithm terminates when the number of non-improvement steps is met.
— MAPM. The memetic algorithm with population management was proposed by [BPR07]. For solving the SDVRP-UF. The algorithm was implemented in Delphi and executed on a 3.0 GHz personal computer. The algorithm stops when a maximum number of iterations is reached.
— ABHC. The attribute based hill climber heuristic was proposed by [DLV10] for solving the SDVRP-UF. It was executed on a 3 GHz personal computer with 2 GB RAM.

Among these references, the BKS values can be considered as the most reliable because they are the best results ever reached by an existing SDVRP algorithm in the literature. On the other hand, the results of the cited algorithms enable an assessment of the proposed algorithm compared to the current state-of-the-art methods. We contacted the authors of the reference algorithms, and obtained the source codes of RGTS [BGN14] and FBTS [Zha+15]. Unfortunately, for RGTS when we ran it with large scale instances such as p03-100D4, the program terminated with unknown errors. For FBTS, when we compiled the C++ code with g++ on our computer, there were several errors. Furthermore, two studies [Che+17; Shi+18] are excluded for our comparative experiments because they report inconsistent results. For several instances, their results are even better than the proven optimal values reported in [ABS14; MS22].

Experimental setting and stopping criterion. The SplitMA algorithm was implemented in C++ and compiled using the g++ compiler with the -O3 option [2]. Experiments were executed on a computer with a Xeon E5-2670 processor of 2.5 GHz and 2 GB RAM

2. Upon the publication of the paper, the code of our algorithm will be made available at https://github.com/pengfeihe-angers/SplitMA

running Linux with a single thread. The algorithm was executed 20 times for each instance with distinct random seeds. In order to provide a good compromise between computing time and solution quality, the SplitMA algorithm terminates when it reaches a maximum of 40,000 iterations. Since each application of the gEAX crossover produces β offspring solutions, each iteration means an offspring solution is constructed and improved by the local search subsequently. On our computer, one run of SplitMA under this stopping condition corresponds to a maximum of 0.04 to 4470.13 seconds (only one instance requires this longest time) according to the instance size, which is quite reasonable compared to the time reported by most reference algorithms in the literature.

5.3.2 Computational results and comparisons

In the tables presented hereafter, column *Instance* indicates the name of instances; *#Instances* is the number of instances; *LB* is the lower bound extracted from state-of-the-art exact algorithms [ABS14; BMM00; MS22; OKY18]; *Best* and *Avg.* are the best and average results obtained by the corresponding algorithm in the column header, respectively; *Gap* is calculated as $Gap = 100 \times (f_{best} - BKS)/BKS$, where f_{best} is the best objective value of SplitMA. Since the SDVRP is a minimization problem, a negative *Gap* (in bold) indicates an improved upper bound. *Time* is the average time in seconds of 20 executions. *TMB* is the average time needed by the algorithm to hit its best solution. Furthermore, the dark gray color indicates that the corresponding algorithm obtains the best result among all compared algorithms on the corresponding instance; the medium gray color displays the second best results, and so on.

We also provide the summarizing information as follows. *Average* is the average value over the instances of a benchmark set. *#Best* is the number of instances for a set where an algorithm gets the best objective value. Finally, to access the statistically significant difference between SplitMA and each reference algorithm, the *p-value* is shown in each table and it is the result of the Wilcoxon signed-rank test with a confidence level of 0.05. If the *p-value* is less than 0.05, the null hypothesis is rejected.

In the following subsections, we present the results obtained by SplitMA on all the benchmark instances and compare them with the reference algorithms.

Comparative results on the SDVRP-LF

Table 5.2 summarizes the results of the SplitMA algorithm for the SDVRP-LF (upper part) compared to the reference algorithms in terms of the best objective values while Tables 6.18 - 6.22 show the detailed results on the 162 instances. From these tables, the following observations can be made. First, as shown in Table 5.2, SplitMA finds 70 new upper bounds out of the 162 instances (43%), matches the BKS values for 75 other instances (46%) and only misses 17 BKS values (10%). This performance can be considered as remarkable given that the BKS values are the best results compiled from all existing algorithms. Furthermore, compared to the most effective heuristic SplitILS, SplitMA obtains 76 and 97 better results in terms of the best and average values, respectively,

Table 5.2 – Summary of comparative results between SplitMA and reference algorithms in terms of the best objective values.

Pair algorithms	#Instances	*Best*				*Avg.*			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
SDVRP-LF	162	-	-	-	-	-	-	-	-
SplitMA vs. BKS	162	70	75	17	4.28E-09	-	-	-	
SplitMA vs. SplitILS	162	76	74	12	1.11E-12	97	29	36	7.42E-09
SplitMA vs. iVNDiv	99	92	7	0	3.15E-17	-	-	-	-
SplitMA vs. RGTS	88	78	9	1	2.15E-14	79	8	1	2.76E-14
SplitMA vs. SS	49	44	5	0	1.74E-09	-	-	-	-
SplitMA vs. HGA	21	12	8	1	3.09E-03	-	-	-	-
SDVRP-UF	162	-	-	-	-	-	-	-	-
SplitMA vs BKS	162	73	81	8	2.08E-12	-	-	-	-
SplitMA vs. SplitILS	162	82	76	4	4.35E-16	112	33	17	6.24E-18
SplitMA vs. TSVBA	120	105	13	2	8.69E-20	-	-	-	-
SplitMA vs. FBTS	67	67	0	0	1.12E-12	-	-	-	-
SplitMA vs. MAPM	74	62	12	0	1.72E-12	-	-	-	-
SplitMA vs. ABHC	36	34	2	0	1.83E-07	-	-	-	-

while the reverse is true for 12 and 36 cases. For the remaining reference algorithms, the dominance of SplitMA is even more evident by achieving the best results for the vast majority of the instances. According to the Wilcoxon signed-rank test, the small *p-values* ($\ll 0.05$) between SplitMA and the competitors indicate that the performance differences are statistically significant.

From the detailed results shown in Tables 6.18 - 6.22, we have several observations. First, for each benchmark set, SplitMA competes favorably with the corresponding reference algorithms in terms of the best and average results. Second, in terms of running time, SplitMA spends a little more time to obtain slightly better results compared to SplitILS for Set I with both rounded and unrounded costs. For the three remaining Sets, SplitMA finds better results than SplitILS with less computation time. Some algorithms, such as RGTS, show very short times, but their results are much worse than SplitMA (and SplitILS). It's worth saying that given the reference algorithms were programmed in different languages and performed on different computers under different stopping conditions, the comments on running times are provided for indicative purposes only.

Comparative results on the SDVRP-UF

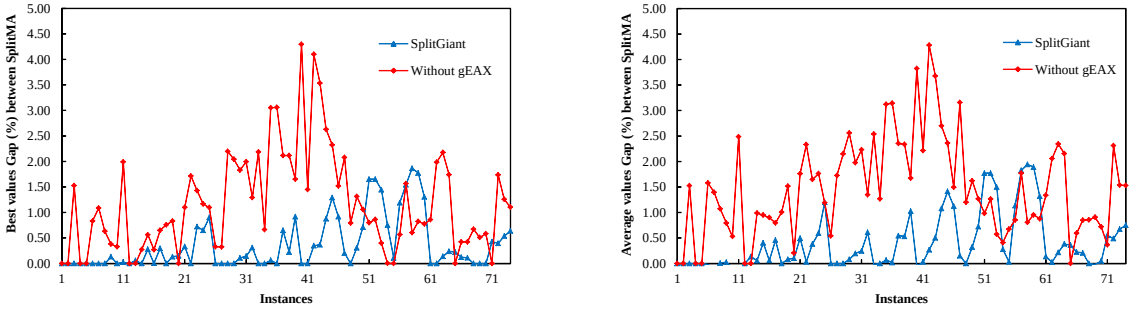
Table 5.2 summarizes the results of the SplitMA algorithm for the SDVRP-UF (lower part) compared to the reference algorithms in terms of the best objective values while Tables 6.23 - 6.27 show the detailed results on the 162 instances. One notices that our algorithm updates 73 BKS values (new upper bounds) and matches 81 other BKS values. Compared to the best reference algorithm SplitILS, our algorithm reports 82 better, 76 equal and 8 worse results, respectively. For the average results, SplitMA obtains 112 better results compared to SplitILS. SplitMA performs much better than the other reference algorithms (weaker than SplitILS) by obtaining the best results for the vast majority of the instances. The small *p-values* ($\ll 0.05$) from the Wilcoxon signed-rank test indicate that

the performance differences between SplitMA and the reference algorithms are statistically significant.

5.4 Analysis

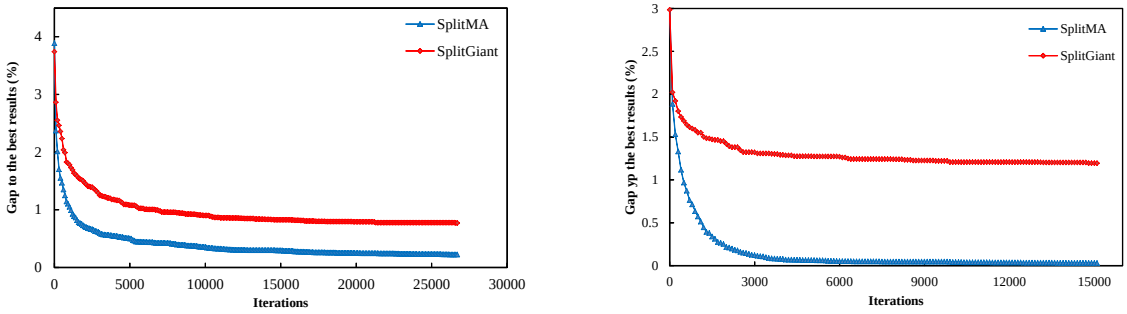
In this section, we conduct additional experiments to assess the contributions of two key components of the SplitMA algorithm, that is gEAX and local search. For this, we focus on the SDVRP-UF and the 74 instances of Sets I and II.

5.4.1 Significance of the gEAX crossover



(a) Best **(b)** Avg.

Figure 5.4 – Performance gaps of SplitGiant (with the giant tour crossover) and SplitMA1 (with the gEAX crossover disabled) compared to SplitMA on the 74 instances of Sets I and II (a positive gap indicates a deteriorating result).



(a) S101D3 **(b)** S101D5

Figure 5.5 – Convergence charts of SplitMA and SplitGiant for solving two representative instances.

Table 5.3 – Summary of comparative results of SplitMA compared to the results of SplitGiant (using the giant tour crossover) and SplitMA1 (without any crossover).

Pair algorithms	*Best*				*Avg.*			
	#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
SplitMA vs. SplitGiant	46	28	0	3.52E-09	54	13	7	3.52E-12
SplitMA vs. SplitMA1	64	10	0	4.63E-10	68	6	0	7.64E-13

To assess the interest of the gEAX crossover, we create two variants of SplitMA as follows. The first variant (SplitGiant) replaces in SplitMA the gEAX crossover by the popular giant tour crossover, which has been very successful for solving routing problems [Pot09; Vid+14] as well as the SDVRP [BPR07]. To implement this variant, we faithfully follow the description of [BPR07] and adopt the source code of the split procedure from [Vid22]. The second variant (SplitMA1) just disables the gEAX crossover of SplitMA. To ensure a fair comparison, we use the average running time of SplitMA shown in Tables 6.23 and 6.25 as the stopping condition of these two variants to solve each instance. Like SplitMA, each variant is run 20 times independently on each instance. The summarized results are shown in Table 5.3 while the detailed results are illustrated in Fig. 5.4 where the results of SplitMA are used as the basis and the results of SplitGiant and SplitMA1 are presented related to this basis.

From Table 5.3 and Fig. 5.4, one observes that SplitMA outperforms SplitGiant (using the giant tour crossover) in terms of both the best and average values, by reaching 46 better results and 28 equal results out of the 74 instances. Furthermore, when the gEAX crossover is removed from SplitMA, the results become much worse since SplitMA1 can only matches 10 and 6 best solutions in terms of the best and average results.

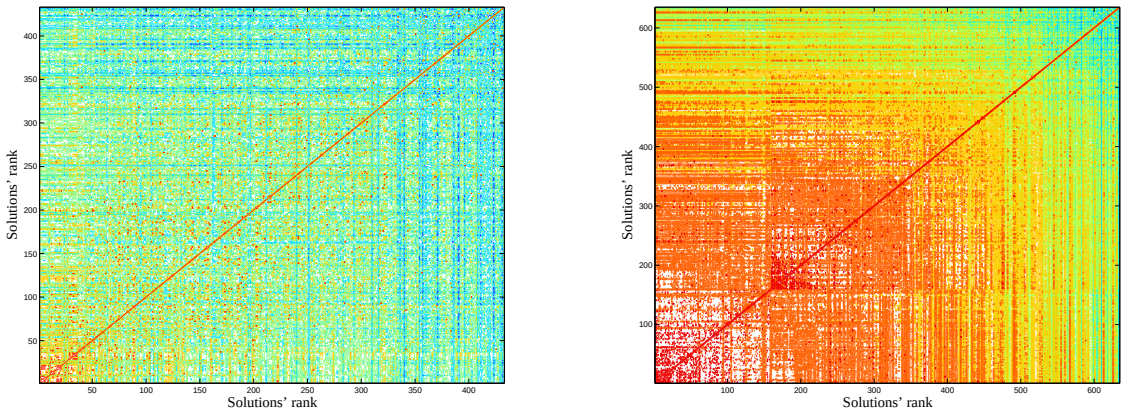
To further compare SplitMA and SplitGiant, we investigate their convergence behaviors. Specifically, we obtain the running profiles of these algorithms on two representative instances (S101D3 and S101D5). Each algorithm is run 20 times with the same time budget and the best results were recorded during the process. The results of this experiment are shown in Fig. 5.5. One observes that SplitMA converges not only faster than SplitGiant, but also converges better.

We conclude that gEAX is not only a critical search operator contributing greatly to the performance of SplitMA, but also a more suitable crossover compared to the giant tour crossover.

5.4.2 Rationale behind the crossover

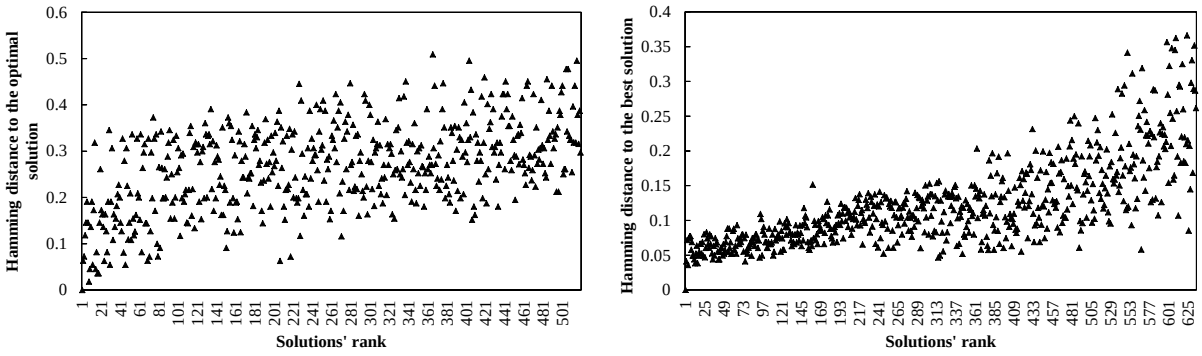
To shed insights on why the gEAX crossover is a suitable operator for the SDVRP, we investigate the relationship between high-quality local optimal solutions in terms of the Hamming distance. Indeed, relevant studies on the TSP [Müh90; NK13] and VRP [AS19b; NB09] have found that high-quality solutions share many common edges, which form the backbone of optimal solutions. EAX thus benefits from this property to construct

promising offspring solutions by inheriting the backbone information while introducing a certain degree of diversity [NK13]. In this section, we show experimentally that the same property remains valide for the SDVRP.



(a) eil51 **(b)** S101D5

Figure 5.6 – Hamming distance between each pair of local optimal solutions



(a) eil51 **(b)** S101D5

Figure 5.7 – Hamming distance between solutions and the best/optimal solution

For our experiment, we select two representative instances: eil51 whose optimal value is known and S101D5 whose best result is shown in Table 6.23. We run SplitMA on these two instances and record a large number of high-quality solutions whose objective value is within 5% of the best/optimal value. As such, 501 solutions for eil51 and 625 solutions for S101D5 are collected. Then, we calculate the normalized Hamming distance (see the definition in Section 5.2.6) between each pair of the solutions. Informally, this distance indicates the percentage of the non-common edges between two solutions over the total edges of the two solutions. A value close to 0 means that the two solutions are

very similar and vice versa. The results are showed in the two dimensional heat map of Fig. 5.6. The abscissa and ordinate represent the rank of solutions from smallest (best) to largest (worst) with respect to the objective value. Each colored pixel corresponds to the normalized Hamming distance between two solutions. Hot colors show small Hamming distances, corresponding to pairs of similar (or close) solutions, while cold colors indicate large Hamming distances, thus pairs of distant solutions.

As one observes in Fig. 5.6, hot colors are around the bottom left corner of both figures, while cold colors are around the upper right corner. This indicates that plus the solutions are good, more they share common edges and vice versa. Furthermore, Fig. 5.7 illustrates the Hamming distance between high-quality solutions and the best/optimal solution. Once again, one notes that high-quality solutions are closer to the best/optimal solution compared to less good solutions. This is particularly true for S101D5, for which high-quality solutions are very close to the best known solution (with more than 90% common edges).

These findings explain why the gEAX crossover performs well for the SDVRP. Indeed, gEAX transmits the common edges from parents (high-quality solutions) to offspring and conserves the backbone information of high-quality solutions while reassembling non-common edges. It is worth noting that these findings are fully consistent with the cases of the TSP and VRP, which motivated the design of the EAX crossover.

5.4.3 Benefits of the local search and mutation

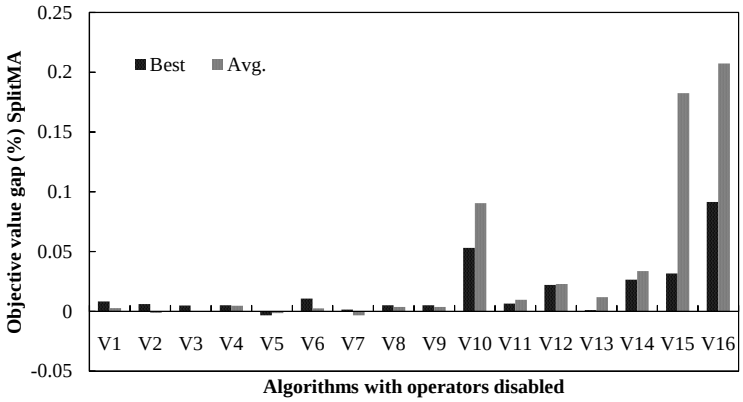


Figure 5.8 – Illustration of the effects of the neighborhood operators and the mutation operator in terms of the gap with respect to the results of the SplitMA algorithm with all neighborhoods and the mutation operator.

SplitMA uses thirteen neighborhood operators in its local search procedure and one mutation operator. It is interesting to know how each of these operators contributes to the performance of the algorithm. For this purpose, we create fourteen SplitMA variants (named V1 to V14) by disabling each of these operators. For example, variant V1 is

Table 5.4 – Effect of each neighborhood and the mutation operator.

Pair algorithms	*Best*				*Avg.*			
	#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
SplitMA vs. V1	18	42	14	5.88E-01	31	20	23	9.01E-01
SplitMA vs. V2	13	48	13	6.94E-01	25	21	28	8.49E-01
SplitMA vs. V3	14	44	16	9.92E-01	29	22	23	9.67E-01
SplitMA vs. V4	14	45	15	9.66E-01	32	22	20	3.16E-01
SplitMA vs. V5	13	46	15	4.73E-01	27	22	25	9.75E-01
SplitMA vs. V6	18	44	12	1.53E-01	32	22	20	8.04E-02
SplitMA vs. V7	13	45	16	3.36E-01	26	22	26	2.70E-01
SplitMA vs. V8	14	45	15	9.31E-01	32	21	21	4.08E-01
SplitMA vs. V9	14	45	15	9.31E-01	32	21	21	4.08E-01
SplitMA vs. V10	32	39	3	1.05E-05	49	21	4	1.70E-09
SplitMA vs. V11	16	44	14	5.30E-01	32	21	21	3.40E-02
SplitMA vs. V12	15	42	17	4.54E-01	29	20	25	4.36E-01
SplitMA vs. V13	12	45	17	2.39E-01	39	16	19	7.13E-03
SplitMA vs. V14	26	40	8	3.76E-03	44	21	9	2.70E-07
SplitMA vs. V15	24	41	9	2.17E-02	55	14	5	1.02E-10
SplitMA vs. V16	35	36	3	3.71E-07	58	14	2	2.56E-11

the SplitMA algorithm with the M1 neighborhood being removed from the local search procedure and V14 is SplitMA without the mutation operator. To assess the contributions of the nine VRP neighborhoods (M1-M9) and the four SDVRP neighborhoods (M10-M13), we create two additional SplitMA variants V15 and V16 where M1-M9 and M10-M13 are disabled, respectively. For each of these variants, we compare its best and average results with those obtained by SplitMA. The gaps between these variants and SplitMA are shown in Fig. 5.8, and a positive gap implies a deteriorating performance with respect to the original SplitMA algorithm.

From the results of Table 5.4 and Fig. 5.8, the contribution of each operator can be summarized as follows. First, all operators influence the overall process with variable impacts. Specifically, M10 can be considered as the most critical neighborhood operator since SplitMA deteriorates significantly its performance if M10 is disabled. Meanwhile, the roles of M2 and M9 are rather marginal. Second, the four tailored SDVRP neighborhood operators (M10–M13) are important for the local search procedure. Third, the mutation operator cannot be ignored since it considerably influences the performance of SplitMA for the best and average results. Finally, both V15 (without the VRP neighborhoods) and V16 (without the SDVRP neighborhoods) perform very badly, confirming that both types of neighborhoods are indispensable for the local search. Meanwhile, we observe that the SDVRP neighborhoods are more critical than the VRP neighborhoods. In summary, all the neighborhood operators and mutation contribute to the performance of the SplitMA algorithm, even if their contributions vary significantly.

5.4.4 Benefits of the maximum splits per customer

We now study how the maximum splits strategy contributes to the performance of SplitMA. For this purpose, we create 10 SplitMA variants with different values of θ,

Table 5.5 – Effect of the maximum splits per customer.

Pair algorithms	*Best*				*Avg.*			
	#Wins	#Ties	#Losses	*p-value*	#Wins	#Ties	#Losses	*p-value*
SplitMA vs. MaxS20	11	46	17	4.25E-01	19	29	26	3.07E-01
SplitMA vs. MaxS30	14	47	13	8.29E-01	20	31	23	4.69E-01
SplitMA vs. MaxS40	12	53	9	2.97E-01	22	32	20	5.28E-01
SplitMA vs. MaxS60	13	56	5	8.54E-02	22	32	20	9.70E-01
SplitMA vs. MaxS70	13	57	4	5.52E-02	25	31	18	8.75E-01
SplitMA vs. MaxS80	12	56	6	1.33E-01	26	30	18	6.12E-01
SplitMA vs. MaxS90	12	46	16	5.24E-01	19	28	27	2.92E-01
SplitMA vs. MaxS100	13	57	4	5.52E-02	27	29	18	2.29E-01
SplitMA vs. MaxS150	13	57	4	4.94E-02	33	26	15	1.05E-01
SplitMA vs. MaxS200	14	55	5	3.29E-02	34	25	15	3.81E-02

which controls the number of maximum splits per customer (the larger θ, the higher the allowed maximum splits). For example, variant MaxS30 uses $\theta = 30$. For each of these variants, we compare its best and average results with those obtained by SplitMA ($\theta = 50$). This experiment follows the same experimental protocal as before and the results are summarized in Table 5.5.

From Table 5.5, we find that SplitMA performs significantly better than MaxS150 and MaxS200 in terms of the best results. Indeed, the value of θ used in variant MaxS200 is four times larger than SplitMA ($\theta = 50$). Furthermore, if the maximum splits strategy is removed from SplitMA, the results we obtain are nearly the same as with the variant MaxS200. Thus, the maximum splits strategy positively contributes to the performance of SplitMA. On the other hand, SplitMA is marginally better than the other variants except two cases for these 74 SDVRP-UF instances, which indicates that SplitMA performs similarly well when the maximum splits per customer are limited to an reasonable range.

5.5 Chapter conclusion

The split delivery vehicle routing problem is a useful model for a broad range of applications in various domains. This work introduced a new memetic algorithm SplitMA that features a general edge assembly crossover for creating promising offspring solutions and an effective local search for solution refinement. It also employs dedicated repairing techniques to ensure the feasibility of offspring solutions, a mutation to diversify new offspring solutions, and an advanced quality-and-distance strategy for maintaining a healthy population.

Extensive experiments on four sets of 324 commonly used instances demonstrate that our algorithm significantly outperforms all existing SDVRP algorithms available in the literature. The algorithms discovers 143 new upper bounds (70 for the SDVRP with a fleet of limited vehicles and 73 cases for the SDVRP with a fleet of unlimited vehicles) and matches the best known results for the majority of the remaining instances. Additional experiments are shown to understand the contributions of main algorithmic components

including the gEAX crossover, local search neighborhoods and mutation.

Part III

Conclusions

CONCLUSIONS

This thesis investigates effective hybrid genetic algorithms for solving four routing problems: colored traveling salesmen problem, minmax multiple traveling salesmen problem, traveling salesman problems with profits and split delivery vehicle routing problem. As the literature review shown in **Chapter 1**, given that these problems are $\mathcal{NP}$-hard and imply numerous real-life applications, many methods have been presented for solving these problems. In this thesis, we present a hybrid genetic algorithmic framework to solve these problems efficiently and robustly. Extensive experiments on commonly used benchmark instances indicate that our algorithms outperform state-of-the-art algorithms.

In **Chapter 2**, a grouping memetic algorithm is presented for solving the CTSP. The algorithm integrates two complementary components: a specific backbone-based crossover to produce promising offspring solutions and a powerful local optimal exploration for offspring improvement. The crossover operator emphasizes diversification when inheriting information from parent solutions to offspring individuals while the local optima exploration is devoted to intensified search by finding local optimal solutions as good as possible. Extensive experimental results on three sets of 65 benchmark instances indicate that our algorithm is very competitive compared with existing leading algorithms. In particular, it is able to update 38 new upper bounds and match 24 best-known results. We also investigate the interest of CPLEX for solving the CTSP and reported 10 proven optimal solutions for the first time.

In **Chapter 3**, even that the backbone information is easy to be applied to the CTSP, it becomes difficult when solving the minmax mTSP and other routing problems. Thus, we need more powerful crossover operators for these problems. For this reason, we design a dedicated edge assembly crossover operator (mEAX) for the minmax mTSP with single and multiple depots. The proposed algorithm integrates an efficient variable neighborhood descent to do intensive search and an aggressive post-optimization procedure to further advance solutions' quality. By properly inheriting edges from high-quality parent solutions, mEAX contributes to propagate favorable characteristics from elite parent solutions to offspring. Extensive experimental results on commonly used instances indicate that our algorithm performs very well for both problems by reporting 44/77 and 39/43 new upper bounds for the minmax mTSP and the minmax multidepot mTSP, respectively. We performe additional experiments to assess the contributions of the two key algorithmic components (i.e., mEAX and post-optimization). We also conducte a long term convergence analysis of the algorithm to illustrate its capacity of finding still better solutions if more time is allowed.

In **Chapter 4**, we present an extended edge assembly crossover (E^2AX) for traveling salesman problems with profits and its associated hybrid genetic algorithm. In addition to the E^2AX, the proposed hybrid genetic algorithm integrates an effective local search

to ameliorate each offspring solution and diversification-oriented mutation as well as a population-diversity management. Extensive experiments are conducted on the orienteering problem (OP) and the prize-collecting traveling salesman problem (PCTSP). For the OP, four sets of 344 commonly used instances are tested and 67 new lower bounds are discovered. The algorithm also matches the best known results for 172 other instances. For the PCTSP, results on three sets of 240 instances show a high performance on large-sized instances including 120 new best results never reported in the literature. Additional experiments are conducted to get insights into the benefits of the proposed crossover and the mutation.

In **Chapter 5**, since **Chapter 4** proposes an extended edge assembly crossover that the limited extension cannot be applied to more complex problems. such as the split delivery vehicle routing problem, the more general edge assembly crossover should be presented for rich routing problems. This chapter presents an effective memetic algorithm for solving the problem with a fleet of limited or unlimited vehicles. The algorithm features a general edge assembly crossover to generate promising offspring solutions from the perspective of assembling suitable edges and an effective local search to improve each offspring solution. The algorithm is further reinforced by a feasibility-restoring procedure, a diversification-oriented mutation and a quality-and-distance pool updating technique. Extensive experiments on 324 benchmark instances indicate that our algorithm is able to update 143 best upper bounds in the literature and match the best results for 156 other instances. Additional experiments are presented to obtain insights into the roles of the key search ingredients of the algorithm.

The primary interest of this thesis is to investigate hybrid genetic algorithms to tackle a class of routing problems. We focus on a versatile and scalable edge crossover crossover and generalize it to rich routing problems. Results on four problems show positive contributions of the crossover as well as local search procedures.

Perspectives

In this thesis, a hybrid genetic algorithmic framework is presented for four routing problems. At the time of concluding thesis, the following perspectives would be interested in the development of efficient solution methods.

The CTSP is strongly related to the mTSP and TSP, for which powerful algorithms exist. Ideas of these algorithms could be useful for solving the CTSP. To advance heuristic algorithms for the mTSP, the dedicated local search techniques could be further investigated to speed up neighborhood examinations without sacrificing quality. Furthermore, efficient exact algorithms are still missing for the minmax mTSP and minmax multidepot mTSP. Research on this topic is thus valuable.

The hybrid genetic algorithm can be further improved by investigating powerful streamline techniques to increase the computational efficiency and to deal with still large problem instances, in terms of the traveling salesman problems with profits. Several directions can be envisaged to address the split delivery vehicle routing problem. First, the local search is

the most time-consuming component, thus it is of interest to develop speed-up techniques, such as static move descriptors designed for the CVRP. Second, our algorithm basically explores feasible solutions. It is interesting to carry out mixed search approaches allowing the examination of both feasible and infeasible solutions.

Another interesting research direction would be to investigate the general idea of assembling promising edges from elite parents. Indeed, the unified hybrid genetic algorithmic framework including the gEAX for rich routing problems can be further studied. For example, it is possible to design hybrid genetic algorithms for several routing problems, such as location routing problems and two-echelon vehicle routing problems. Many other computational challenges arise in these cases.

List of Figures

List of Tables

Part IV

Appendix

Chapter 6

Appendix

6.1 Computational results on the minmax mTSP and minmax multidepot mTSP

This appendix presents detailed computational results of the proposed MA algorithm together with the results of reference algorithms: re-IWO, re-MASVND, ES [Kar+21], HSNR [HHW21] and ITSHA [Zhe+22b]. In the tables presented hereafter, column 'Instances' indicates the name of the benchmark instance; column 'BKS' shows the best-known solution summarized from the literature. For the minmax mTSP, the starred BKS values are optimal values. 'f_{best}' and 'f_{avg}' are the best and average solution found by the algorithm in the column header, respectively. 'Gap' is calculated as $Gap = 100\times(f_{best}\text{-}f_{bk})/f_{bk}$ where f_{best} and f_{bk} are the best objective value of MA and the best objective value from all reference algorithms (including BKS), respectively. Since both problems have a minimization objective, a negative Gap indicates an improvement over the BKS value (i.e., a new upper bound). Furthermore, the dark gray color indicates that the algorithm obtains the best result among the compared algorithms on the corresponding instance; the medium gray color displays the second best result, and so on. We provide additionally information for each algorithm in terms of the best and average value. 'Average' is the average value over the instances of a benchmark set.

As shown in Table 6.3, the time information in Table 6.3 is provided only for indicative purposes (The '-' symbol indicates the time information is unavailable for MD and VNS or non-applicable for MA). The time (in seconds) for MD and VNS corresponds to the average time of one run under the stopping conditions indicated in Section 3.3.1. For the MA algorithm, 'TTB' indicates the average time in seconds needed for MA to hit the BKS values, while 'AT' is the average time of one run.

6.2 The giant tour crossover for the prize-collecting traveling salesman problem

Crossover operators based on the giant tour have been used to solve various routing problems [Vid+14], which rely on efficient split algorithms designed for specific constraints, such as capacity or time windows. Indeed, the giant tour can also be applied to TSPs with profits with respect to the corresponding constraints. In this section, we introduce a giant tour crossover and an optimal split algorithm.

We take the PCTSP as an example. Given a solution φ, let $\mathcal{N}_\varphi$ and $\overline{\mathcal{N}}_\varphi$ be a set of routed and unrouted vertices in φ, respectively. Let φ_A and φ_B be two parent solutions. First, all routed vertices ($v_i \in \mathcal{N}_{\varphi_A}$) in solution φ_A are arranged into an array A. Second, all unrouted vertices ($v_i \in \overline{\mathcal{N}}_{\varphi_A}$) are arranged into A after routed vertices in the sequential order. An array B is produced using solution φ_B in the same way. Second, given two giant tours A and B, an ordered crossover [OSH87] is used on a simple permutation-based representation. Then a new giant tour S is produced. Finally, a linear-time split algorithm with respect to the collecting prize optimally splits each giant tour by inserting a trip delimiter. Specifically, for each vertex in S, if the delimiter is inserted after it, there are two tours and we need $\mathcal{O}(1)$ time to compute profits and travel costs. Since there are n vertices in S, we can optimally splits S in $\mathcal{O}(n)$ time. After the split, a feasible offspring is returned.

Table 6.1 – The minmax mTSP: comparative results of MA with five reference algorithms on the 41 instances of Set $\mathbb{S}$.

Instances	BKS	f_{best}						Gap(%)	f_{avg}					
		re-IWO	re-MASVND	ES [Kar+21]	HSNR [HHW21]	ITSHA [Zhe+22b]	MA		re-IWO	re-MASVND	ES [Kar+21]	HSNR [HHW21]	ITSHA [Zhe+22b]	MA
mtsp51-3	159.57	159.57	159.57	159.57	159.57	159.57	159.57	0.00	159.57	159.57	159.57	159.85	159.57	159.57
mtsp51-5	118.13	118.13	118.13	118.13	118.13	118.13	118.13	0.00	118.13	118.13	118.13	118.13	118.13	118.13
mtsp51-10	112.07*	112.07	112.07	112.07	112.07	112.07	112.07	0.00	112.07	112.07	112.07	112.07	112.07	112.07
mtsp100-3	8509.16	8509.16	8509.16	8509.16	8509.16	8509.16	8509.16	0.00	8510.86	8602.30	8649.75	8513.75	8509.16	8509.16
mtsp100-5	6765.73	6780.37	6767.02	6767.02	6765.73	6767.82	6765.73	0.00	6833.45	6801.75	6832.74	6770.67	6772.95	6766.78
mtsp100-10	6358.49*	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49
mtsp100-20	6358.49*	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49
rand100-3	3031.95	3031.95	3031.95	3031.95	3031.95	3031.95	3031.95	0.00	3031.95	3047.71	3084.49	3032.67	3033.65	3031.95
rand100-5	2409.63	2409.90	2409.63	2409.63	2411.68	2412.35	2409.63	0.00	2429.50	2428.35	2422.41	2415.00	2414.65	2409.64
rand100-10	2299.16*	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16
rand100-20	2299.16*	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16
mtsp150-3	13075.80	13078.40	13234.10	13303.80	13075.80	13088.74	13038.30	**-0.29**	13259.97	13411.26	13526.70	13169.37	13210.69	13079.74
mtsp150-5	8466.00	8477.96	8493.62	8563.08	8477.96	8492.97	8417.02	**-0.58**	8650.11	8686.61	8757.22	8538.83	8572.77	8453.15
mtsp150-10	5557.00	5751.41	5666.45	5625.32	5590.64	5593.56	5557.41	0.00	5851.79	5763.28	5718.45	5604.92	5608.50	5588.86
mtsp150-20	5246.49*	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	0.00	5247.62	5246.49	5247.21	5246.49	5246.49	5246.49
mtsp150-30	5246.49*	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	0.00	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49
gtsp150-3	2407.34	2407.34	2433.80	2423.17	2407.34	2407.34	2401.63	**-0.24**	2421.40	2468.12	2491.00	2435.49	2416.87	2401.86
gtsp150-5	1741.61	1744.57	1744.26	1751.85	1741.71	1741.13	1741.13	0.00	1777.90	1779.32	1797.71	1743.48	1752.06	1741.13
gtsp150-10	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1557.22	1559.10	1554.64	1554.64	1554.64	1554.76
gtsp150-20	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64
gtsp150-30	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64
kroA200-3	10748.10	10801.30	10833.60	10883.30	10748.10	10700.57	10691.00	**-0.09**	10965.59	11136.70	11174.70	10987.69	10819.85	10691.41
kroA200-5	7415.54	7497.21	7484.17	7536.91	7418.87	7449.22	7412.12	**-0.05**	7697.45	7634.61	7770.43	7494.44	7513.67	7414.21
kroA200-10	6223.22*	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00	6255.37	6266.44	6240.52	6223.22	6223.22	6249.10
kroA200-20	6223.22*	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22
lin318-3	15902.50	16133.40	16551.60	16349.60	15902.50	15930.04	15663.50	**-1.50**	17006.92	16886.01	16797.80	16207.05	16088.56	15699.92
lin318-5	11295.20	12291.60	11741.60	11619.60	11295.20	11430.65	11276.80	**-0.16**	12882.38	12023.74	11907.90	11596.35	11601.67	11291.59
lin318-10	9731.17*	9861.64	9731.17	9731.18	9731.17	9731.17	9731.17	0.00	9963.31	9797.38	9736.17	9731.17	9731.17	9731.17
lin318-20	9731.17*	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	0.00	9731.18	9731.17	9731.18	9731.17	9731.17	9731.17
att532-3	10231.00	11258.00	10566.00	10585.00	10231.00	10158.00	9966.00	**-1.89**	11525.50	10853.05	10953.90	10565.30	10344.50	10064.00
att532-5	7067.00	8518.00	7279.00	7344.00	7067.00	7067.00	6986.00	**-1.15**	8895.80	7429.50	7463.50	7334.00	7156.80	7070.95
att532-10	5709.00	6427.00	5745.00	5761.00	5709.00	5731.00	5770.00	1.07	6552.90	5809.00	5806.75	5738.90	5787.50	5796.75
att532-20	5580.00*	5745.00	5580.00	5580.00	5580.00	5583.00	5580.00	0.00	5836.90	5582.90	5580.05	5580.00	5601.75	5589.35
rat783-3	3158.34	3688.79	3295.90	3444.20	3187.90	3131.99	3052.41	**-2.54**	3786.16	3364.20	3485.74	3237.29	3180.79	3083.52
rat783-5	2006.46	2627.74	2120.74	2125.53	2006.46	2018.44	1961.12	**-2.26**	2781.71	2145.38	2189.92	2044.32	2058.65	1989.68
rat783-10	1334.76	1692.31	1396.92	1373.46	1334.76	1357.65	1313.01	**-1.63**	1718.75	1424.76	1396.78	1345.88	1381.69	1325.54
rat783-20	1231.69*	1371.32	1237.97	1231.69	1231.69	1231.69	1231.69	0.00	1390.09	1244.26	1231.69	1231.69	1231.84	1235.37
pcb1173-3	20292.61	25557.90	22255.20	23193.10	20813.80	20288.75	19569.50	**-3.55**	26439.83	22941.19	23640.00	21144.92	20473.45	19858.77
pcb1173-5	12952.97	18703.50	14088.40	14333.00	13032.30	12816.55	12406.60	**-3.20**	19226.82	14305.57	14601.30	13216.99	13045.14	12639.49
pcb1173-10	7758.26	11170.00	8452.28	8222.40	7758.26	7801.18	7623.59	**-1.74**	11388.41	8637.95	8352.07	7897.20	7910.09	7745.00
pcb1173-20	6528.86*	8132.08	6549.14	6549.14	6528.86	6528.86	6528.86	0.00	8356.53	6623.91	6577.59	6528.86	6534.75	6548.87
Average	5998.71	6553.84	6152.15	6177.75	6015.33	6000.98	5943.29	-	6689.21	6241.85	6268.40	6076.73	6043.73	5971.30

Table 6.2 – The minmax mTSP: comparative results of MA with four reference algorithms on the 36 instances of Set 𝕃.

Instances	BKS	f_{best}					Gap(%)	f_{avg}				
		re-IWO	re-MASVND	HSNR [HHW21]	ITSHA [Zhe+22b]	MA		re-IWO	re-MASVND	HSNR [HHW21]	ITSHA [Zhe+22b]	MA
nrw1379-3	20495.90	24401.20	22236.40	20495.90	19871.21	19222.10	**-3.27**	25204.30	23349.12	20765.70	20085.29	19472.39
nrw1379-5	12416.50	17636.70	13368.80	12416.50	12218.09	11913.40	**-2.49**	18019.88	13847.14	12652.56	12493.99	12203.76
nrw1379-10	7114.71	10145.40	7583.59	7114.71	7008.61	6846.27	**-2.32**	10404.29	7748.69	7212.24	7136.56	6987.31
nrw1379-20	5370.82*	7082.11	5495.31	5370.82	5370.82	5381.59	0.20	7310.74	5571.01	5371.08	5402.42	5388.99
fl1400-3	9192.38	9860.63	9562.25	9192.38	8310.34	7854.97	**-5.48**	10140.92	10094.49	9621.59	8482.77	7989.25
fl1400-5	6268.25	8422.09	6803.42	6268.25	6360.45	6116.28	**-2.42**	8590.95	7134.69	6783.62	6572.53	6185.40
fl1400-10	5763.26*	7359.74	5763.26	5763.26	5763.26	5763.26	0.00	7485.43	5763.74	5763.26	5785.69	5763.26
fl1400-20	5763.26*	6687.79	5763.26	5763.26	5763.26	5763.26	0.00	6819.01	5763.26	5763.26	5763.26	5763.26
d1655-3	25229.30	32293.30	30143.30	25229.30	25403.26	23921.90	**-5.18**	33051.92	42813.48	25635.98	25804.06	24149.72
d1655-5	17181.20	24146.80	18719.10	17181.20	17502.88	16512.20	**-3.89**	24854.66	19376.15	17454.32	17824.61	16754.81
d1655-10	11660.00	15868.50	12454.00	11660.00	11814.34	11320.10	**-2.92**	16569.21	12623.92	11816.04	11971.60	11528.33
d1655-20	9598.94	12165.20	9893.04	9598.94	9910.12	9627.28	0.30	12605.33	10120.03	9607.73	10172.24	9669.97
u2152-3	24207.40	32354.70	43724.70	24207.40	23295.73	22127.80	**-5.01**	33246.73	44187.24	24747.01	23746.41	22629.25
u2152-5	15055.10	23356.00	17653.10	15055.10	14778.56	14094.00	**-4.63**	23534.98	18404.22	15394.85	15236.58	14442.21
u2152-10	8624.61	13454.40	9458.60	8624.61	8763.71	8332.12	**-3.39**	13985.98	9600.79	8780.91	9018.30	8499.64
u2152-20	6171.89	9223.98	6550.73	6171.89	6605.48	6253.35	1.32	9532.87	6727.13	6225.82	6676.91	6339.15
pr2392-3	141627.00	186013.00	254034.00	141627.00	135763.02	130015.00	**-4.23**	190584.70	256052.65	143703.00	137589.12	132228.70
pr2392-5	88083.20	133780.00	104977.00	88083.20	87465.60	82408.50	**-5.78**	135073.30	132626.10	89582.83	88179.42	84448.64
pr2392-10	51085.30	80135.10	55337.60	51085.30	50514.84	49033.60	**-2.93**	83131.04	56650.63	52100.80	50929.73	49985.84
pr2392-20	35325.30	56941.00	38175.60	35325.30	35999.41	35455.50	0.37	58490.18	39420.33	35709.02	36546.00	36107.77
pcb3038-3	51049.90	66159.20	85795.40	51049.90	48351.41	46994.60	**-2.81**	68931.99	86481.39	51582.38	49081.79	47686.85
pcb3038-5	31140.20	46465.70	66560.40	31140.20	30089.85	29223.00	**-2.88**	46938.10	67071.90	31495.59	30603.78	29864.61
pcb3038-10	16949.90	26954.20	19198.20	16949.90	16878.69	16031.70	**-5.02**	27659.07	19620.41	17450.44	16645.10	16509.95
pcb3038-20	10835.00	17772.50	12012.20	10835.00	10827.78	10769.60	**-0.54**	18323.59	12643.54	11004.40	11196.47	10961.26
fl3795-3	11971.00	16611.70	22444.50	11971.00	12290.18	10927.40	**-8.72**	17207.03	22801.50	12815.54	13022.12	11321.19
fl3795-5	7923.71	13391.00	19698.50	7923.71	8151.43	7715.36	**-2.63**	13809.93	19877.81	8610.84	8657.18	8014.97
fl3795-10	5763.26*	10132.30	6715.07	5763.26	5824.14	5764.85	0.03	10500.27	7120.46	5823.89	5990.03	5810.07
fl3795-20	5763.26*	8519.26	5763.26	5763.26	5763.26	5763.26	0.00	8679.69	5763.75	5763.26	5766.51	5763.74
fnl4461-3	66903.70	90062.10	108622.00	66903.70	64140.70	62016.70	**-3.31**	91143.24	109798.50	67971.34	65268.41	62894.65
fnl4461-5	40721.20	59532.70	83650.40	40721.20	39839.40	38265.50	**-3.95**	60170.68	84430.87	41777.11	39839.40	38935.22
fnl4461-10	22041.50	34068.20	25385.20	22041.50	22145.65	20671.60	**-6.22**	34741.06	43581.63	22891.45	22954.31	20996.25
fnl4461-20	12630.10	22142.80	14611.30	12630.10	12789.54	12347.80	**-2.24**	22852.80	15262.97	13046.38	12987.65	12542.14
rl5915-3	213864.00	328020.00	443748.00	213864.00	210056.49	193879.00	**-7.70**	332327.15	445979.00	226819.75	215466.06	198796.00
rl5915-5	133457.00	221495.00	362776.00	133457.00	125537.65	120418.00	**-4.08**	225566.65	364717.65	145173.07	132524.74	124718.05
rl5915-10	76585.20	133266.00	267295.00	76585.20	70853.30	66329.40	**-6.38**	137737.55	270354.45	84459.02	71353.13	67961.45
rl5915-20	48958.50	88081.70	51115.20	48958.50	44716.69	43121.00	**-3.57**	92716.25	53066.77	60306.22	46080.96	44373.90
Average	35077.55	52611.17	63141.32	35077.55	34076.09	32450.03	-	53831.71	65456.87	36713.40	34801.53	33158.00

Table 6.3 – The minmax multidepot mTSP: comparative results of MA with two reference algorithms on the 43 instances of Set $\mathbb{M}$.

Instances	BKS	MD [WGW15]		VNS [WGW15]		MA				
		f_{best}	Time (s)	f_{best}	Time (s)	f_{best}	f_{avg}	Gap(%)	TTB (s)	AT (s)
MM1	170.109	170.909	1	170.909	1	170.908	170.908	0.470	-	2
MM2	130.8	130.800	11	131.497	6	124.067	125.8095	**-5.148**	18	277
MM3	238.973	238.973	18	240.397	13	230.821	231.9083	**-3.411**	28	371
MM4	479.676	479.676	18	481.595	340	438.039	442.2707	**-8.680**	52	1446
MM5	315.889	315.889	33	333.376	18	299.751	299.9226	**-5.109**	13	1546
MM6	82.187	82.226	44	82.226	16	85.356	86.94082	3.856	-	306
MM7	189.016	189.016	2	189.016	5	189.017	189.017	0.001	-	9
MM8	217.383	217.383	30	231.493	27	203.585	204.2455	**-6.347**	12	460
MM9	152.504	152.504	112	156.972	57	142.355	143.4553	**-6.655**	55	840
MM10	182.926	197.390	4	182.926	9	181.382	181.382	**-0.844**	1	33
MM11	102.346	102.346	3	103.663	8	102.346	102.346	0.000	12	99
MM12	78.903	78.903	3	80.828	5	72.921	73.23033	**-7.581**	1	83
MM13	120.688	121.872	5	120.688	11	117.681	117.7794	**-2.492**	13	215
MM14	134.613	134.613	8	137.219	11	125.585	126.0707	**-6.707**	11	262
MM15	96.524	99.805	5	96.524	7	90.787	91.27053	**-5.943**	11	221
MM16	101.68	101.680	23	103.696	28	96.068	98.70936	**-5.519**	204	548
MM17	248.588	248.588	235	259.255	28	236.859	238.7844	**-4.718**	98	1038
MM18	390.16	390.160	619	400.269	58	383.617	385.4731	**-1.677**	324	985
MM19	365.657	365.657	616	395.371	159	339.333	344.665	**-7.199**	6	1046
MM20	339.92	339.920	360	356.176	152	311.737	315.116	**-8.291**	10	1213
MM21	259.14	259.140	-	274.100	-	245.165	246.5903	**-5.393**	18	551
MM22	400.6	400.600	-	413.270	-	390.934	393.2435	**-2.413**	61	510
MM23	374.97	374.970	-	378.710	-	363.504	363.5538	**-3.058**	22	280
MM24	204	204.000	-	206.220	-	195.992	198.7478	**-3.925**	146	801
MM25	272.61	272.610	-	274.840	-	230.690	233.9735	**-15.377**	4	746
MM26	364.56	364.560	-	369.100	-	349.459	351.2541	**-4.142**	40	1047
MM27	290.37	290.370	-	298.460	-	285.220	286.3728	**-1.774**	75	568
MM28	354.31	354.310	-	367.720	-	348.377	351.8489	**-1.675**	556	1343
MM29	364.01	364.010	-	376.180	-	357.100	359.0724	**-1.898**	246	1001
MM30	140.34	140.340	-	149.540	-	128.349	130.7342	**-8.544**	85	995
MM31	112.52	124.320	-	112.520	-	106.189	107.7453	**-5.627**	29	415
MM32	98.45	103.150	-	98.450	-	96.260	96.35376	**-2.225**	1	50
MM33	97.56	97.560	-	100.930	-	92.820	92.8197	**-4.859**	1	469
MM34	84.64	84.640	-	85.580	-	78.796	80.94745	**-6.905**	168	703
MM35	107.86	109.300	-	107.860	-	99.928	99.92959	**-7.354**	2	282
MM36	153.27	155.990	-	153.270	-	135.947	136.9038	**-11.302**	8	1187
MM37	151.19	156.410	-	151.190	-	132.937	132.937	**-12.073**	1	744
MM38	155.46	155.460	-	166.300	-	149.556	149.764	**-3.798**	10	331
MM39	209.85	209.850	-	223.670	-	195.788	198.0532	**-6.701**	28	676
MM40	243.47	243.470	-	250.680	-	235.961	236.2019	**-3.084**	15	213
MM41	255.27	257.160	-	255.270	-	237.959	241.4352	**-6.781**	158	1312
MM42	357.17	367.440	-	357.170	-	314.451	318.7805	**-11.960**	23	1046
MM43	375.16	375.160	-	375.550	-	349.380	351.95	**-6.872**	32	811
Average	222.449	223.794	-	227.923	-	211.132	212.964	-	-	-

6.3 Computational results on the OP and PCTSP instances

In the tables presented hereafter, column *Instance* indicates the name of instances; column BKS is the best known values summarized from the literature; LB and UB are lower and upper bounds obtained by the corresponding algorithm in the column header, respectively; *Gap* associated with exact algorithms (RB&C and B&C) is calculated as $Gap = 100 \times (LB - UB)/UB$; *Best* and *Avg.* are the best and average results obtained by the corresponding algorithm in the column header, respectively; Time in each column means the running time of the corresponding algorithm; TMB is the average running time needed by the algorithms to attain its best results. In Tables 6.4-6.11, *Gap* in the last column is calculated as $Gap = 100 \times (BKS - f_{best})/f_{best}$, where f_{best} is the best objective value of the proposed HGA algorithm. In Tables 6.12-6.17, if a value is associated with a star, it means it is the optimal solution verified by exact algorithms; *Gap* in the last column is calculated as $Gap = 100 \times (f_{best} - BKS)/BKS$, where f_{best} is the best objective value of HGA and the *BKS* is the best results of the B&C and HGA-Giant algorithms. In the tables, the *Average* row is the average value over the instances of a benchmark set. Improved best results (new bounds) are indicated by negative *Gap* values highlighted in boldface.

Table 6.4 – Results for the OP on the medium-sized instances of Set I.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
att48	31	31	31	31	*	0.03	31	0.25	31	6.77	31	31.00	0.85	0.84	0.00
gr48	31	31	31	31	*	0.02	31	0.13	31	9.99	31	31.00	0.04	0.01	0.00
hk48	30	30	30	30	*	0.01	30	0.24	30	7.20	30	30.00	2.51	2.51	0.00
eil51	29	29	29	29	*	0.01	29	0.24	29	9.51	29	28.85	11.92	7.16	0.00
berlin52	37	37	37	37	*	0.02	37	0.30	37	9.42	37	37.00	0.04	0.01	0.00
brazil58	46	46	46	46	*	0.07	46	1.00	46	9.13	46	45.30	44.65	6.38	0.00
st70	43	43	43	43	*	0.05	43	0.32	43	15.99	43	43.00	0.66	0.66	0.00
eil76	47	47	47	47	*	0.04	46	0.32	47	20.51	47	46.05	59.01	1.96	0.00
pr76	49	49	49	49	*	0.06	49	0.61	49	18.64	49	48.05	63.37	0.94	0.00
gr96	64	64	64	64	*	0.08	64	1.44	64	20.31	64	64.00	15.44	15.44	0.00
rat99	52	52	52	52	*	0.47	52	0.66	52	27.75	52	51.80	33.95	20.86	0.00
kroA100	56	56	56	56	*	0.41	55	0.34	56	34.75	56	56.00	9.92	9.92	0.00
kroB100	58	58	58	58	*	0.27	57	0.63	58	43.06	58	56.45	68.74	25.52	0.00
kroC100	56	56	56	56	*	0.25	56	0.48	56	34.32	56	56.00	14.68	14.68	0.00
kroD100	59	59	59	59	*	0.09	58	0.65	59	34.61	59	59.00	5.82	5.82	0.00
kroE100	57	57	57	57	*	5.53	57	0.50	57	32.26	57	56.35	60.60	27.06	0.00
rd100	61	61	61	61	*	0.12	61	0.74	61	29.49	61	60.90	40.18	33.52	0.00
eil101	64	64	64	64	*	0.06	64	0.79	64	31.73	64	64.00	7.20	7.20	0.00
lin105	66	66	66	66	*	0.48	66	1.42	66	32.11	66	66.00	0.45	0.44	0.00
pr107	54	54	54	54	*	0.08	54	0.93	54	78.46	54	54.00	0.11	0.01	0.00
gr120	75	75	75	75	*	0.28	74	1.20	75	29.58	75	75.00	28.58	28.58	0.00
pr124	75	75	75	75	*	0.35	75	1.11	75	49.64	75	75.00	0.86	0.86	0.00
bier127	103	103	103	103	*	0.38	103	1.18	103	40.84	103	103.00	5.05	5.05	0.00
pr136	71	71	71	71	*	1.75	71	0.96	71	29.97	71	70.95	40.26	35.01	0.00
gr137	81	81	81	81	*	0.24	78	3.44	81	59.21	81	81.00	7.44	7.44	0.00
pr144	77	77	77	77	*	1.46	77	2.61	77	87.82	77	76.50	74.23	46.61	0.00
kroA150	86	86	86	86	*	33.87	86	1.17	86	82.79	86	85.05	113.12	33.65	0.00
kroB150	87	87	87	87	*	2.21	86	1.00	87	61.64	86	86.00	146.01	36.24	1.16
pr152	77	77	77	77	*	1.29	77	3.64	77	91.38	77	76.45	90.19	30.72	0.00
u159	93	93	93	93	*	1.82	92	1.11	93	99.63	93	92.15	122.50	37.65	0.00
rat195	102	102	102	102	*	3.71	99	1.78	102	195.57	101	100.45	139.73	56.95	0.99
d198	123	123	123	123	*	5.28	123	6.68	123	65.57	123	122.70	118.46	60.17	0.00
kroA200	117	117	117	117	*	2.5	117	1.74	117	114.75	116	114.05	227.36	83.39	0.86
kroB200	119	119	119	119	*	9.91	119	1.66	119	86.58	118	117.70	211.44	81.31	0.85
gr202	145	145	145	145	*	2.71	145	6.89	145	187.56	145	144.60	157.66	77.48	0.00
ts225	124	124	124	126	1.59	18000.00	124	1.28	124	279.52	124	124.00	0.22	0.06	0.00
tsp225	129	129	129	129	*	4.31	127	2.29	128	198.47	128	126.05	223.06	102.75	0.78
pr226	126	126	126	126	*	107.69	126	6.61	126	181.94	126	125.20	168.44	16.25	0.00
gr229	176	176	176	176	*	0.32	176	8.81	173	108.27	175	174.30	324.03	84.10	0.57
gil262	158	158	158	158	*	0.35	156	2.84	158	240.02	155	153.50	323.80	125.41	1.94
pr264	132	132	132	132	*	3.92	132	5.62	132	314.29	132	132.00	2.44	2.35	0.00
a280	147	147	147	147	*	40.65	143	3.00	144	239.06	145	142.95	272.42	134.60	1.38
pr299	162	162	162	162	*	48.85	160	3.12	162	410.90	160	159.60	280.80	87.62	1.25
lin318	205	205	205	205	*	5.49	202	7.15	203	294.23	205	203.55	403.82	153.07	0.00
rd400	239	239	239	239	*	36.71	234	6.59	237	422.56	236	233.50	623.83	294.78	1.27
Average	89.31	89.31	89.31	89.36	-	407.20	88.62	2.12	89.07	99.51	88.96	88.44	101.02	40.07	-

Table 6.5 – Results for the OP on the large-sized instances of Set I.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
fl417	228	231	228	231	1.3	18000.00	224	11.84	228	1056.07	227	224.20	372.44	162.00	0.44
gr431	350	350	350	350	*	29.05	349	32.84	347	533.55	347	345.35	743.86	251.44	0.86
pr439	313	313	313	313	*	414.00	310	9.92	307	1263.74	312	309.85	617.75	281.02	0.32
pcb442	251	251	251	251	*	7.21	244	6.93	249	1328.72	249	248.40	544.55	161.21	0.80
d493	320	320	320	320	*	13.37	315	19.10	317	1291.93	315	310.10	852.24	312.30	1.59
att532	363	363	363	363	*	312.50	347	23.14	359	1380.54	357	356.10	1131.88	499.40	1.68
ali535	425	426	425	426	0.23	18000.00	424	73.03	422	1846.10	419	416.40	1609.74	624.85	1.43
pa561	357	357	357	357	*	245.42	348	23.18	346	1605.42	344	340.65	1141.05	526.90	3.78
u574	354	354	354	354	*	24.00	344	17.93	347	1204.18	350	346.60	1106.06	536.17	1.14
rat575	322	322	322	322	*	42.82	309	13.76	317	3109.65	310	307.95	948.33	425.64	3.87
p654	344	396	342	396	13.64	18000.00	336	28.89	343	10866.70	343	340.70	1074.49	475.42	0.29
d657	386	386	386	386	*	92.48	377	23.24	380	3152.17	381	375.30	1334.10	731.54	1.31
gr666	503	503	503	503	*	400.56	497	109.54	486	660.30	495	491.85	1746.74	977.59	1.62
u724	439	439	439	439	*	188.61	429	27.77	434	4157.30	429	426.00	1591.42	755.99	2.33
rat783	438	438	438	438	*	514.68	422	34.59	428	2962.52	420	416.80	1247.89	637.76	4.29
dsj1000	656	656	656	656	*	3828.50	632	81.20	630	17284.30	638	631.90	2962.76	1594.68	2.82
pr1002	606	606	606	606	*	4483.81	572	45.92	581	18000.00	587	581.20	2395.05	960.38	3.24
u1060	660	660	660	660	*	16716.01	627	90.04	644	18000.00	647	641.65	2440.49	1076.03	2.01
vm1084	777	777	777	777	*	5012.60	770	56.29	765	18000.00	772	770.00	3806.86	2003.13	0.65
pcb1173	675	675	675	675	*	6819.83	633	60.65	652	18000.00	650	643.10	3283.19	1529.82	3.85
d1291	715	715	715	715	*	7916.85	646	434.87	699	18000.00	698	682.75	2494.85	1392.79	2.44
rl1304	802	802	802	802	*	6269.39	766	102.45	788	18000.00	789	782.35	3464.28	2231.75	1.65
rl1323	814	814	814	814	*	7740.17	782	89.68	785	14585.10	806	795.75	4187.98	2125.59	0.99
nrw1379	815	817	815	817	0.24	18000.00	771	106.97	790	18000.00	779	772.30	4687.57	2410.17	4.62
fl1400	1048	1084	1003	1084	7.47	18000.00	1043	518.25	1048	18000.00	1041	1036.85	5391.92	2387.03	0.67
u1432	754	764	754	764	1.31	18000.00	738	121.46	749	14573.50	739	732.90	5068.57	2481.63	2.03
fl1577	897	900	897	900	0.33	18000.00	880	286.47	748	18000.00	865	858.50	4425.22	2523.23	3.70
d1655	922	924	922	924	0.22	18000.00	846	757.70	890	18000.00	887	880.95	4708.59	2906.87	3.95
vm1748	1276	1282	1276	1282	0.47	18000.00	1246	178.50	1252	16959.80	1262	1253.65	7284.97	3624.73	1.11
u1817	983	983	983	983	*	11226.88	879	975.58	947	18000.00	939	928.25	5700.99	3433.93	4.69
rl1889	1226	1226	1226	1226	*	17010.43	1167	269.81	1156	18000.00	1208	1195.95	7920.36	4909.81	1.49
d2103	1200	1200	1200	1200	*	15855.62	1069	951.27	1171	18000.00	1198	1194.60	7188.64	4012.64	0.17
u2152	1151	1151	1151	1151	*	14703.25	1048	1350.23	1111	18000.00	1095	1082.00	8348.35	6032.38	5.11
u2319	1170	1171	1170	1171	0.09	18000.00	1167	423.26	1170	6088.42	1170	1170.00	8933.50	3082.05	0.00
pr2392	1316	1415	1316	1415	7	18000.00	1292	402.29	1294	18000.00	1333	1323.25	8904.12	5243.50	**-1.28**
pcb3038	1727	1730	1727	1730	0.17	18000.00	1572	681.94	1626	18000.00	1623	1604.35	14596.03	9637.53	6.41
fl3795	1965	2249	1965	2249	12.63	18000.00	1815	2994.90	1818	18000.00	1904	1900.85	18000.25	10663.92	3.20
fnl4461	2541	2570	2541	2570	1.13	18000.00	2350	2462.65	2342	18000.00	2443	2410.40	18000.28	14495.34	4.01
rl5915	3593	3786	3593	3786	5.1	18000.00	3358	5361.54	3328	18000.00	3668	3626.80	18000.45	15975.74	**-2.04**
rl5934	3632	3752	3632	3752	3.2	18000.00	3145	5382.25	3276	18000.00	3602	3561.37	18000.66	16239.41	0.83
pla7397	5289	5657	5289	5657	6.51	18000.00	5141	15981.78	5140	18000.00	5294	5263.80	18000.02	14906.27	**-0.09**
Average	1039.10	1068.66	1037.95	1068.66	-	10387.03	981.22	990.82	992.93	11802.68	1022.80	1014.19	5470.48	3542.43	-

Table 6.6 – Results for the OP on the medium-sized instances of Set II.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
att48	1717	1717	1717	1717	*	0.04	1717	0.32	1717	6.77	1717	1717.00	0.62	0.62	0.00
gr48	1761	1761	1761	1761	*	1.32	1749	0.2	1761	7.87	1761	1761.00	1.01	1.01	0.00
hk48	1614	1614	1614	1614	*	0.10	1614	0.16	1614	7.19	1614	1614.00	0.03	0.01	0.00
eil51	1674	1674	1674	1674	*	0.96	1668	0.18	1674	10.13	1674	1674.00	0.68	0.68	0.00
berlin52	1897	1897	1897	1897	*	3.23	1897	0.35	1897	10.74	1897	1897.00	0.11	0.10	0.00
brazil58	2220	2220	2220	2220	*	0.46	2218	1.52	2220	12.32	2220	2220.00	8.77	8.77	0.00
st70	2286	2286	2286	2286	*	1.77	2285	0.31	2286	21.65	2286	2286.00	8.18	8.18	0.00
eil76	2550	2550	2550	2550	*	0.62	2550	0.43	2550	16.06	2550	2550.00	9.47	9.47	0.00
pr76	2708	2708	2708	2708	*	1.46	2708	0.48	2708	19.48	2708	2708.00	3.15	3.14	0.00
gr96	3396	3396	3396	3396	*	9.50	3394	1.44	3394	31.98	3396	3396.00	24.04	24.04	0.00
rat99	2944	2944	2944	2944	*	3.25	2944	0.49	2944	32.08	2944	2944.00	1.82	1.81	0.00
kroA100	3212	3212	3212	3212	*	0.70	3212	0.57	3212	32.85	3212	3212.00	4.74	4.74	0.00
kroB100	3241	3241	3241	3241	*	13.28	3238	0.52	3239	48.39	3241	3218.25	567.89	197.48	0.00
kroC100	2947	2947	2947	2947	*	2.22	2931	0.60	2947	39.27	2947	2947.00	10.97	10.97	0.00
kroD100	3307	3307	3307	3307	*	3.62	3307	0.65	3307	30.52	3307	3307.00	3.59	3.59	0.00
kroE100	3090	3090	3090	3090	*	11.31	3082	0.50	3090	39.57	3090	3090.00	1.79	1.79	0.00
rd100	3359	3359	3359	3359	*	0.36	3359	0.50	3359	30.80	3359	3359.00	1.46	1.46	0.00
eil101	3655	3655	3655	3655	*	4.15	3655	0.82	3655	26.19	3655	3655.00	12.71	12.70	0.00
lin105	3544	3544	3544	3544	*	2.51	3530	1.10	3544	36.22	3544	3544.00	15.90	15.90	0.00
pr107	2667	2667	2667	2667	*	0.20	2667	1.05	2667	69.67	2667	2667.00	0.07	0.01	0.00
gr120	4371	4371	4371	4371	*	6.57	4356	1.37	4371	40.41	4371	4371.00	7.86	7.86	0.00
pr124	3917	3917	3917	3917	*	1.07	3899	1.34	3917	55.25	3917	3917.00	3.80	3.80	0.00
bier127	5383	5383	5383	5383	*	0.96	5381	1.71	5366	23.01	5383	5383.00	23.60	23.60	0.00
pr136	4309	4309	4309	4309	*	1.25	4309	1.15	4309	35.63	4309	4309.00	5.39	5.38	0.00
gr137	4286	4286	4286	4286	*	10.65	4099	3.09	4286	639.80	4286	4286.00	2.33	2.33	0.00
pr144	4003	4003	4003	4003	*	32.23	3965	3.02	3969	100.20	4003	4003.00	7.49	7.49	0.00
kroA150	4918	4918	4918	4918	*	60.43	4902	1.26	4918	80.06	4918	4918.00	135.04	135.04	0.00
kroB150	4869	4869	4869	4869	*	16.94	4869	1.19	4869	61.96	4869	4869.00	82.07	82.05	0.00
pr152	4279	4279	4279	4279	*	1.85	4245	3.47	4279	67.41	4279	4279.00	165.93	165.93	0.00
u159	4960	4960	4960	4960	*	14.96	4941	1.44	4950	109.59	4960	4960.00	43.12	43.11	0.00
rat195	5791	5791	5791	5791	*	46.09	5703	1.55	5782	263.23	5791	5791.00	158.24	158.22	0.00
d198	6670	6670	6670	6670	*	298.24	6660	7.33	6661	88.47	6670	6669.35	798.66	273.99	0.00
kroA200	6547	6547	6547	6547	*	16.18	6534	1.71	6547	116.11	6547	6547.00	38.12	38.09	0.00
kroB200	6419	6419	6419	6419	*	20.62	6278	1.97	6413	189.98	6419	6387.70	721.43	396.77	0.00
gr202	7789	7789	7789	7789	*	139.90	7789	8.77	7719	188.27	7789	7789.00	92.11	92.11	0.00
ts225	6834	6834	6834	6834	*	95.22	6819	1.47	6782	394.00	6834	6812.30	619.32	286.64	0.00
tsp225	6987	6987	6987	6987	*	54.09	6936	1.87	6980	299.73	6987	6985.60	514.68	370.74	0.00
pr226	6662	6662	6662	6662	*	2894.81	6658	7.29	6662	201.68	6662	6662.00	3.39	3.39	0.00
gr229	9177	9177	9177	9177	*	16.67	9174	13.19	9177	1379.35	9177	9176.95	317.14	272.63	0.00
gil262	8321	8321	8321	8321	*	64.63	8175	3.47	8269	487.41	8321	8316.25	836.53	383.89	0.00
pr264	6654	6654	6654	6654	*	13.33	6173	5.94	6654	314.27	6654	6654.00	148.53	148.42	0.00
a280	8428	8428	8428	8428	*	519.95	8304	2.85	8404	215.31	8427	8405.70	1113.10	486.86	0.01
pr299	9182	9182	9182	9182	*	623.34	9112	3.23	9147	393.12	9182	9180.50	841.19	356.78	0.00
lin318	10923	10923	10923	10923	*	367.53	10866	8.29	10801	370.64	10923	10921.90	935.97	376.82	0.00
rd400	13652	13652	13652	13652	*	769.66	13442	6.80	13562	1174.91	13650	13648.90	1430.88	652.59	0.01
Average	4869.33	4869.33	4869.33	4869.33	-	136.63	4829.20	2.38	4857.31	173.77	4869.27	4866.88	216.06	112.91	-

Table 6.7 – Results for the OP on the large-sized instances of Set II.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
fl417	11933	12294	11933	12387	3.67	18000.00	11787	16.73	11923	2144.94	11938	11934.10	1279.24	582.48	**-0.04**
gr431	18318	18318	18318	18318	*	2809.41	18287	51.38	18318	2740.82	18315	18313.20	1900.40	935.74	0.02
pr439	16171	16171	16171	16171	*	3765.86	16085	11.77	16128	629.44	16171	16170.05	1283.17	574.48	0.00
pcb442	14484	14484	14484	14484	*	13760.94	14273	6.83	14411	4410.74	14465	14452.45	1677.34	700.55	0.13
d493	16995	17007	16995	17007	0.07	18000.00	16729	17.15	16820	6231.42	16991	16972.20	1956.30	1016.13	0.02
att532	19635	19800	19635	19800	0.83	18000.00	19265	23.43	19465	1564.89	19658	19653.25	2193.88	765.79	**-0.12**
ali535	21954	21954	21954	21973	0.09	18000.00	21910	95.05	21761	1537.87	21953	21945.15	2257.58	1432.75	0.00
pa561	19576	19576	19576	19576	*	1961.95	18894	23.45	19092	790.31	19551	19497.32	2127.67	1438.76	0.13
u574	19351	19351	19351	19351	*	1026.82	18966	16.33	19028	5389.10	19351	19330.90	2063.79	1188.99	0.00
rat575	18251	18251	18251	18251	*	9616.70	17705	14.97	17984	2089.02	18235	18215.25	1986.76	1063.03	0.09
p654	17900	21566	17753	22248	20.2	18000.00	17821	42.82	17900	18000.00	17917	17876.60	1976.04	642.42	**-0.09**
d657	21503	21503	21503	21503	*	554.67	21162	22.90	21231	4161.44	21499	21490.30	2421.26	1280.89	0.02
gr666	26514	26569	26514	26569	0.21	18000.00	26336	136.48	25971	1024.22	26486	26455.60	2714.55	1322.13	0.11
u724	24223	24223	24223	24223	*	9829.42	23793	28.71	23878	5755.06	24198	24143.95	2832.77	1629.65	0.10
rat783	25474	25474	25474	25474	*	12246.90	24861	32.36	24987	6622.62	25353	25214.85	2873.49	1785.02	0.48
dsj1000	35835	35915	35835	35915	0.22	18000.00	34463	83.34	34641	18000.00	35524	35436.00	4857.53	3587.65	0.88
pr1002	33030	33092	33030	33092	0.19	18000.00	31746	46.19	32120	18000.00	33005	32949.80	4285.52	2802.95	0.08
u1060	36151	36291	36151	36291	0.39	18000.00	35110	77.78	35284	18000.00	36146	36027.30	4311.82	3023.31	0.01
vm1084	40777	40952	40777	40952	0.43	18000.00	40308	55.67	40240	18000.00	40774	40750.45	4942.20	2588.10	0.01
pcb1173	37035	37100	37035	37100	0.18	18000.00	35826	69.94	35946	18000.00	36874	36741.90	4912.86	2811.80	0.44
d1291	37778	37854	37778	37854	0.2	18000.00	35153	289.25	36815	18000.00	37564	37421.15	4538.07	2613.11	0.57
rl1304	42275	42359	42275	42359	0.2	18000.00	40561	97.68	40893	12853.40	42266	41963.65	5386.89	3415.69	0.02
rl1323	43377	43450	43377	43450	0.17	18000.00	41459	89.78	41210	18000.00	43375	43160.90	5669.84	3608.18	0.00
nrw1379	46676	46787	46676	46787	0.24	18000.00	45602	117.51	45576	18000.00	46529	46398.05	6638.98	4604.15	0.32
fl1400	56692	64298	54124	64298	15.82	18000.00	56258	794.15	56692	18000.00	56883	56832.60	7843.46	6002.60	**-0.34**
u1432	46946	47018	46946	47018	0.15	18000.00	44810	100.91	44982	18000.00	46617	46281.40	6328.82	4648.08	0.71
fl1577	45505	50154	45326	50154	9.63	18000.00	45505	334.28	41148	18000.00	47295	46971.35	6779.32	5151.11	**-3.78**
d1655	49319	53083	46158	53083	13.05	18000.00	47211	683.17	49319	18000.00	50239	50038.55	7663.78	5819.46	**-1.83**
vm1748	68042	68303	68042	68303	0.38	18000.00	66685	195.85	66636	18000.00	68090	67938.90	11054.04	7804.15	**-0.07**
u1817	54245	54554	54245	54554	0.57	18000.00	50366	734.39	51676	18000.00	53506	53280.05	8443.16	7270.04	1.38
rl1889	63308	64425	63308	64425	1.73	18000.00	60084	286.07	60928	18000.00	64036	63690.90	9906.39	7719.85	**-1.14**
d2103	63426	63426	63426	63426	*	16593.51	57202	682.28	61636	18000.00	62977	62682.10	9309.68	7721.66	0.71
u2152	64649	64775	64649	64775	0.19	18000.00	60211	1164.38	61052	18000.00	63718	63324.30	10367.26	8976.34	1.46
u2319	80914	81139	80914	81139	0.28	18000.00	78102	447.06	77610	18000.00	80521	80272.70	13356.77	9988.59	0.49
pr2392	72843	78237	72843	78237	6.89	18000.00	71018	440.57	71851	18000.00	75272	74926.15	14470.13	12654.40	**-3.23**
pcb3038	97902	97995	97902	97995	0.09	18000.00	91842	820.37	91457	18000.00	95980	95276.50	18000.55	16152.89	2.00
fl3795	103397	142895	98998	142895	30.72	18000.00	103397	4788.96	102642	18000.00	110988	110604.45	18000.35	15572.33	**-6.84**
fnl4461	147109	150189	147109	150189	2.05	18000.00	140424	2618.15	135515	18000.00	145968	145165.15	18000.58	17089.11	0.78
rl5915	184424	197729	184424	197729	6.73	18000.00	176678	5512.40	173500	18000.00	193626	192284.80	18000.59	17431.83	**-4.75**
rl5934	187034	196805	187034	196805	4.96	18000.00	171649	5757.80	166368	18000.00	192968	191017.65	18000.56	17164.14	**-3.08**
pla7397	281977	297246	281977	297246	5.14	18000.00	272452	18000	266038	18000.00	287214	286523.55	18000.91	17905.70	**-1.82**
Average	56413.37	59088.10	56158.39	59107.46	-	15369.91	54195.02	1093.37	53918.83	12827.93	57074.05	56820.13	7088.20	5621.61	-

Table 6.8 – Results for the OP on the medium-sized instances of Set III.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
att48	1049	1049	1049	1049	*	1.17	1049	0.26	1049	7.18	1049	1049.00	0.21	0.21	0.00
gr48	1480	1480	1480	1480	*	0.72	1480	0.13	1480	8.87	1480	1480.00	1.32	1.32	0.00
hk48	1764	1764	1764	1764	*	0.06	1764	0.22	1764	8.51	1764	1764.00	0.06	0.06	0.00
eil51	1399	1399	1399	1399	*	1.46	1398	0.22	1399	6.87	1399	1399.00	1.04	1.04	0.00
berlin52	1036	1036	1036	1036	*	4.61	1034	0.64	1036	12.84	1036	1036.00	43.83	43.83	0.00
brazil58	1702	1702	1702	1702	*	0.02	1702	0.71	1702	11.09	1702	1702.00	1.32	1.32	0.00
st70	2108	2108	2108	2108	*	0.49	2108	0.31	2108	9.65	2108	2108.00	36.52	36.51	0.00
eil76	2467	2467	2467	2467	*	2.96	2467	0.36	2467	20.48	2467	2467.00	3.09	3.09	0.00
pr76	2430	2430	2430	2430	*	1.07	2430	0.57	2430	20.43	2430	2430.00	0.82	0.82	0.00
gr96	3170	3170	3170	3170	*	5.66	3166	1.41	3166	15.22	3170	3170.00	1.16	1.16	0.00
rat99	2908	2908	2908	2908	*	3.01	2886	0.78	2908	42.03	2908	2907.80	202.43	123.29	0.00
kroA100	3211	3211	3211	3211	*	1.81	3180	0.38	3211	32.31	3155	3155.00	586.14	6.30	1.77
kroB100	2804	2804	2804	2804	*	0.35	2785	0.51	2804	35.83	2804	2804.00	2.13	2.13	0.00
kroC100	3155	3155	3155	3155	*	1.82	3155	0.44	3155	34.67	3155	3155.00	1.78	1.76	0.00
kroD100	3167	3167	3167	3167	*	0.70	3141	0.58	3167	31.08	3167	3167.00	2.88	2.88	0.00
kroE100	3049	3049	3049	3049	*	1.36	3049	0.47	3049	31.96	3049	3049.00	26.43	26.42	0.00
rd100	2926	2926	2926	2926	*	23.20	2923	0.48	2926	16.35	2926	2926.00	3.27	3.27	0.00
eil101	3345	3345	3345	3345	*	1.37	3345	0.56	3345	28.61	3345	3345.00	25.05	25.04	0.00
lin105	2986	2986	2986	2986	*	16.02	2973	2.09	2986	38.24	2986	2986.00	2.32	2.31	0.00
pr107	1877	1877	1877	1877	*	3297.37	1802	0.82	1877	65.16	1877	1874.45	61.01	26.07	0.00
gr120	3779	3779	3779	3779	*	2.65	3748	1.36	3777	37.94	3779	3779.00	36.22	36.22	0.00
pr124	3557	3557	3557	3557	*	4507.38	3455	0.88	3557	99.87	3557	3557.00	5.67	5.66	0.00
bier127	2365	2365	2365	2365	*	40.07	2361	2.62	2361	49.90	2365	2365.00	150.74	150.73	0.00
pr136	4390	4390	4390	4390	*	30.50	4390	1.13	4390	61.84	4390	4390.00	14.83	14.82	0.00
gr137	3954	3954	3954	3954	*	14.01	3954	1.88	3954	637.09	3954	3954.00	21.85	21.84	0.00
pr144	3745	3745	3745	3745	*	116.68	3700	2.41	3744	112.92	3745	3745.00	60.76	60.76	0.00
kroA150	5039	5039	5039	5039	*	46.43	5019	1.07	5037	104.23	5003	5003.00	782.62	24.47	0.72
kroB150	5314	5314	5314	5314	*	28.53	5314	1.04	5314	63.05	5314	5314.00	57.85	57.85	0.00
pr152	3905	3905	3905	3905	*	83.51	3902	3.62	3539	184.38	3905	3904.75	374.94	212.53	0.00
u159	5272	5272	5272	5272	*	8.59	5272	0.94	5272	94.27	5272	5272.00	4.17	4.17	0.00
rat195	6195	6195	6195	6195	*	33.56	6139	2.00	6188	188.56	6195	6195.00	198.01	198.01	0.00
d198	6320	6320	6320	6320	*	461.18	6290	7.14	6320	105.70	6320	6320.00	94.10	94.09	0.00
kroA200	6123	6123	6123	6123	*	92.41	6114	1.72	6118	232.20	6123	6123.00	90.58	90.58	0.00
kroB200	6266	6266	6266	6266	*	3.87	6213	1.77	6266	188.77	6266	6224.40	424.28	356.42	0.00
gr202	8616	8616	8616	8616	*	315.26	8605	10.45	8564	57.88	8616	8615.30	413.75	323.12	0.00
ts225	7575	7575	7575	7575	*	6.62	7575	1.14	7575	450.25	7575	7575.00	20.38	20.38	0.00
tsp225	7740	7740	7740	7740	*	38.61	7488	2.58	7514	188.53	7740	7713.50	925.06	496.93	0.00
pr226	6993	6993	6993	6993	*	1170.00	6908	8.01	6993	177.59	6993	6986.00	571.88	168.34	0.00
gr229	6328	6328	6328	6328	*	42.63	6297	11.65	6328	1298.80	6328	6301.90	837.83	277.42	0.00
gil262	9246	9246	9246	9246	*	83.29	9094	3.94	9210	649.54	9246	9245.70	787.73	500.85	0.00
pr264	8137	8137	8137	8137	*	186.59	8068	3.62	8137	357.80	8137	8095.95	780.66	229.32	0.00
a280	9774	9774	9774	9774	*	126.80	8684	3.22	8789	378.80	9774	9774.00	74.80	74.80	0.00
pr299	10343	10343	10343	10343	*	913.13	9959	3.95	10233	549.11	10343	10343.00	216.38	216.38	0.00
lin318	10368	10368	10368	10368	*	327.58	10273	6.33	10337	528.20	10368	10368.00	124.69	124.69	0.00
rd400	13223	13223	13223	13223	*	214.40	13088	7.74	13122	727.58	13223	13212.85	1431.43	720.01	0.00
Average	4724.44	4724.44	4724.44	4724.44	-	272.43	4661.04	2.31	4681.51	177.83	4722.40	4718.92	211.20	106.43	-

Table 6.9 – Results for the OP on the large-sized instances of Set III.

Instances	BKS		RB&C [KML20]				EA4OP [KML18]		ALNS [San19]		HGA				
	LB	UB	LB	UB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
fl417	14220	14220	14219	14387	1.17	18000.00	14186	12.45	14220	1131.05	14214	14201.95	952.43	401.78	0.04
gr431	10911	10911	10911	10911	*	7814.17	10817	54.5	10907	2411.45	10911	10895.40	1586.08	863.30	0.00
pr439	15176	15296	15176	15331	1.01	18000.00	15097	10.96	15080	1328.74	15162	15161.35	1669.39	613.07	0.09
pcb442	14819	14819	14819	14819	*	11574.76	14522	6.58	14695	1192.19	14817	14800.60	1623.75	739.64	0.01
d493	25167	25188	25167	25195	0.11	18000.00	24981	19.18	24849	3829.32	25150	24814.15	1912.07	1221.13	0.07
att532	15498	15498	15498	15498	*	318.44	15342	22.75	15335	4533.36	15422	15402.35	2081.94	1337.68	0.49
ali535	9414	9472	9414	9472	0.61	18000.00	9328	94.09	9308	13313.50	9406	9402.40	2068.92	1061.64	0.09
pa561	14482	14482	14482	14482	*	2539.41	14034	21.35	14162	3020.53	14447	14419.70	2042.58	1212.60	0.24
u574	20064	20064	20064	20064	*	2693.59	19691	19.77	19841	1671.01	20061	20025.05	1873.35	1058.58	0.01
rat575	20109	20109	20109	20109	*	929.99	19879	18.03	19954	7175.13	20107	20068.95	1915.85	1179.53	0.01
p654	24492	24518	24492	24518	0.11	18000.00	24130	18.54	24427	7543.02	24492	24491.15	550.26	449.63	0.00
d657	24562	24562	24562	24562	*	8777.39	23772	21.89	23829	4600.87	24562	24554.30	2272.47	1491.07	0.00
gr666	17023	17048	17023	17060	0.22	18000.00	16902	143.87	16709	2734.75	17011	16994.40	2817.11	1577.29	0.07
u724	28348	28348	28348	28348	*	10332.54	27932	29.26	28033	12058.60	28339	28316.65	2634.43	1488.15	0.03
rat783	27566	27566	27566	27566	*	3812.98	26797	30.64	27306	16331.50	27442	27297.55	2552.72	1842.65	0.45
dsj1000	31434	31454	31434	31454	0.06	18000.00	30943	79.18	31040	15962.00	31423	31340.65	3780.24	2725.06	0.04
pr1002	39526	39526	39526	39526	*	13955.69	38762	47.30	38502	18000.00	39519	39458.30	4103.94	2631.37	0.02
u1060	37492	37569	37492	37569	0.2	18000.00	36570	75.88	36598	18000.00	37496	37392.40	3732.20	2808.55	**-0.01**
vm1084	37669	37669	37669	37669	*	8710.50	37508	54.21	37178	3286.89	37665	37597.50	4614.19	2735.14	0.01
pcb1173	41257	41257	41257	41257	*	15133.74	40069	66.16	40513	18000.00	40865	40731.50	4694.10	2957.94	0.96
d1291	41509	42153	41509	42153	1.53	18000.00	38132	299.87	39919	18000.00	41784	41667.40	4604.59	2792.38	**-0.66**
rl1304	41881	42075	41881	42075	0.46	18000.00	41214	81.11	41679	18000.00	41893	41848.70	5052.61	2756.05	**-0.03**
rl1323	47213	47384	47213	47384	0.36	18000.00	46641	93.53	45500	8544.44	47173	47057.05	5421.88	3513.72	0.08
nrw1379	42920	42975	42920	42975	0.13	18000.00	-	-	-	-	42838	42763.35	4714.91	2822.60	0.19
fl1400	57470	59491	54661	59491	8.12	18000.00	57226	599.81	57470	18000.00	57548	57360.85	7359.73	5660.92	**-0.14**
u1432	47778	47895	47778	47895	0.24	18000.00	46657	138.02	47242	18000.00	47742	47660.40	5448.09	3420.03	0.08
fl1577	45935	48809	45768	48809	6.23	18000.00	45692	295.62	45935	18000.00	46205	46159.65	5666.76	3601.55	**-0.58**
d1655	62048	62945	62048	62945	1.43	18000.00	58728	674.25	60956	18000.00	62598	62491.45	6934.37	4031.41	**-0.88**
vm1748	71885	72010	71885	72010	0.17	18000.00	70958	225.29	71244	18000.00	71911	71865.85	8209.77	4583.09	**-0.04**
u1817	63639	67670	63618	67670	5.99	18000.00	63639	1302.35	63016	18000.00	65061	64851.25	7319.32	5187.11	**-2.19**
rl1889	70065	71106	70065	71106	1.46	18000.00	68422	244.97	68096	18000.00	70704	70532.50	9670.58	6948.90	**-0.90**
d2103	82787	82973	82787	82973	0.22	18000.00	77333	1168.90	81081	18000.00	82789	82710.95	9355.54	5977.87	**0.00**
u2152	74007	78066	74007	78066	5.2	18000.00	73400	1619.61	72733	18000.00	75117	74822.30	8681.40	6364.49	**-1.48**
u2319	79351	81050	79343	81050	2.11	18000.00	78113	569.76	79130	18000.00	79611	79551.90	11286.34	9192.20	**-0.33**
pr2392	85409	90261	85409	90261	5.38	18000.00	84094	422.73	85084	18000.00	87200	86787.50	12106.64	9306.23	**-2.05**
pcb3038	106928	112006	106928	112006	4.53	18000.00	104667	917.39	105337	18000.00	108475	107827.45	15247.40	13511.11	**-1.43**
fl3795	97707	116792	89218	116792	23.61	18000.00	97707	3158.89	95580	18000.00	100319	99971.05	18000.24	16045.69	**-2.60**
fnl4461	146995	152562	146995	152562	3.65	18000.00	-	-	-	-	147641	146824.75	18000.37	17191.81	**-0.44**
rl5915	203695	217366	203695	217366	6.29	18000.00	199336	5593.23	201814	18000.00	211017	210311.10	18000.31	17191.88	**-3.47**
rl5934	212021	229405	212021	229405	7.58	18000.00	207385	5881.87	203667	18000.00	221855	219778.65	18000.61	17148.22	**-4.43**
pla7397	322285	334885	322285	334885	3.76	18000.00	320744	18000	312645	18000.00	325632	323903.85	18000.84	17367.57	**-1.03**
Average	60311.15	62669.63	60030.78	62675.02	-	14843.74	57470.51	1080.35	57451.64	12529.96	61064.00	60832.05	6501.59	5000.26	-

Table 6.10 – Results for the OP on the medium-sized instances of Set IV.

Instances	BKS	B&C [KML18]			EA4OP [KML18]		ALNS [San19]		HGA				
		LB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
att48	1870	1870	0.00	106.00	1870	0.52	1870	8.99	1870	1870.00	185.07	1.58	0.00
gr48	2264	2264	0.00	22.40	2264	0.40	2264	3.82	2264	2264.00	279.14	1.82	0.00
hk48	2177	2177	0.00	0.20	2177	0.15	2177	7.76	2177	2177.00	271.23	1.46	0.00
eil51	2490	2490	0.00	82.10	2490	0.24	2489	6.65	2490	2490.00	327.17	3.69	0.00
berlin52	2089	2089	0.00	115.00	2085	0.48	2089	10.75	2089	2089.00	253.33	9.03	0.00
brazil58	2070	2070	0.00	132.00	2060	1.08	2070	10.71	2070	2070.00	308.02	1.27	0.00
st70	3316	3316	0.00	127.70	3314	0.42	3316	7.82	3316	3315.50	450.39	118.57	0.00
eil76	3646	3646	0.00	45.10	3646	0.52	3640	9.38	3646	3646.00	471.84	31.66	0.00
pr76	3361	3361	0.00	1047.70	3361	0.62	3358	10.78	3361	3361.00	438.33	18.70	0.00
gr96	4851	4851	0.00	212.30	4851	0.37	4851	6.68	4851	4851.00	505.50	3.54	0.00
rat99	3502	3502	0.00	16.00	3502	0.60	3502	31.01	3502	3501.60	551.80	160.40	0.00
kroA100	4999	4999	0.00	187.10	4999	0.36	4999	6.44	4999	4999.00	590.22	66.04	0.00
kroB100	2935	2935	0.00	34.40	2935	0.61	2935	31.84	2935	2904.90	554.57	61.39	0.00
kroC100	1962	1962	0.00	261.60	1955	0.46	1962	31.46	1962	1962.00	336.67	2.33	0.00
kroD100	1212	1212	0.00	11.80	1212	0.41	1212	14.33	1212	1212.00	149.85	0.02	0.00
kroE100	4635	4635	0.00	203.40	4616	0.69	4631	13.14	4635	4635.00	681.46	22.99	0.00
rd100	3815	3815	0.00	164.60	3808	0.75	3815	22.99	3815	3815.00	647.49	21.46	0.00
eil101	4308	4308	0.00	90.80	4306	0.83	4308	39.55	4308	4308.00	609.52	4.68	0.00
lin105	2455	2455	0.00	1020.60	2453	0.81	2455	33.74	2455	2455.00	416.77	3.38	0.00
pr107	2072	2072	0.00	159.00	2072	1.95	2072	10.20	2072	2072.00	227.42	0.58	0.00
gr120	5830	5830	0.00	236.70	5830	1.25	5830	18.10	5830	5830.00	677.32	37.24	0.00
pr124	2036	2036	0.00	163.80	1937	1.18	2036	48.00	2036	2036.00	329.39	0.47	0.00
bier127	5068	5068	0.00	278.40	5067	2.28	5053	42.94	5068	5068.00	614.18	81.69	0.00
pr136	2860	2860	0.00	6303.60	2820	0.74	2860	51.86	2860	2858.80	542.13	210.35	0.00
gr137	6523	6523	0.00	203.10	6516	2.52	6523	35.45	6523	6523.00	779.57	15.72	0.00
pr144	5641	5641	0.00	357.90	5639	4.53	5641	70.02	5641	5639.30	855.54	228.20	0.00
kroA150	6858	6858	0.00	415.90	6855	1.69	6855	42.88	6858	6858.00	816.73	11.97	0.00
kroB150	7023	7023	0.00	303.00	7020	1.16	7014	23.86	7023	7023.00	890.59	4.47	0.00
pr152	5823	5823	0.00	483.60	5820	5.21	5823	43.79	5261	5261.00	823.84	20.03	10.68
u159	3147	3147	0.00	1145.20	3147	0.92	3147	161.92	3147	3147.00	499.78	5.35	0.00
rat195	9753	9753	0.00	205.40	9750	1.69	9737	27.11	9753	9752.75	928.42	233.11	0.00
d198	4661	4661	0.00	492.70	4654	4.95	4658	122.01	4661	4661.00	786.39	151.19	0.00
kroA200	9892	9892	0.00	340.30	9892	2.73	9854	47.90	9892	9889.85	1034.46	363.36	0.00
kroB200	9849	9849	0.00	253.20	9842	1.62	9846	20.18	9849	9849.00	1084.16	153.14	0.00
gr202	1071	1071	0.00	376.10	995	1.47	1055	30.88	1071	1071.00	116.43	0.31	0.00
ts225	11002	11002	0.00	3524.60	11002	1.87	10954	61.48	11002	11002.00	1177.47	109.28	0.00
tsp225	10972	10972	0.00	706.70	10972	2.52	10920	76.87	10973	10969.60	1128.39	516.11	**-0.01**
pr226	4893	4893	0.00	1183.10	4890	4.83	4893	313.81	4893	4893.00	602.26	35.66	0.00
gr229	11482	11482	0.00	563.10	11482	6.46	11397	29.97	11482	11475.85	1019.99	444.54	0.00
gil262	2031	2031	0.00	1770.50	2030	1.35	2031	93.73	2031	2031.00	469.39	2.76	0.00
pr264	10253	10253	0.00	277.50	10166	6.42	10179	180.21	10253	10158.10	1201.48	206.18	0.00
a280	12064	12064	0.00	351.80	12048	3.39	11955	217.26	12064	12049.80	1333.67	491.79	0.00
pr299	14986	14986	0.00	7771.90	14980	3.46	14959	48.86	14986	14982.05	1284.58	471.44	0.00
lin318	15132	15132	0.00	-	15119	7.91	14960	106.15	15146	15144.10	1614.19	600.93	**-0.09**
rd400	20107	20107	0.00	5093.10	20101	9.61	20060	103.75	20102	20097.25	1829.40	812.23	0.02
Average	5755.24	5755.24	-	837.30	5745.56	2.09	5739.00	51.93	5742.98	5739.30	682.12	127.60	-

Table 6.11 – Results for the OP on the large-sized instances of Set IV.

Instances	BKS	B&C [KML18]			EA4OP [KML18]		ALNS [San19]		HGA				
		LB	*Gap*(%)	Time	*Best*	Time	*Best*	Time	*Best*	*Avg.*	Time	TMB	*Gap*(%)
fl417	20496	20496	0.00	18000.00	20494	39.61	20496	165.27	20496	20493.30	1738.74	620.28	0.00
gr431	13976	13976	0.00	18000.00	13969	50.29	13807	794.43	13979	13978.60	1475.64	471.77	**-0.02**
pr439	19613	19613	0.00	3936.10	19510	13.61	19453	765.03	19613	19599.80	1871.00	1175.60	0.00
pcb442	5869	5839	0.51	18000.00	5650	3.40	5869	1290.30	5888	5888.00	904.60	51.57	**-0.32**
d493	21740	21740	0.00	18000.00	21674	21.00	21578	785.63	21744	21688.15	2240.91	1428.08	**-0.02**
att532	26728	26728	0.00	18000.00	26728	17.20	26684	461.68	26721	26714.95	2175.10	968.04	0.03
ali535	13520	13520	0.00	15739.60	13442	73.07	13350	2346.62	13520	13396.35	1601.36	933.42	0.00
pa561	27719	27712	0.03	18000.00	27719	24.14	27445	570.88	27729	27712.50	2845.50	1612.59	**-0.04**
u574	28823	28823	0.00	18000.00	28822	26.03	28815	76.97	28827	28818.10	2337.93	1089.82	**-0.01**
rat575	28364	28364	0.00	18000.00	28334	24.68	28237	436.44	28357	28330.10	2764.61	1590.76	0.02
p654	31814	31814	0.00	18000.00	31717	123.82	31724	267.05	31798	31748.15	2747.90	1396.04	0.05
d657	32548	32548	0.00	13485.10	32534	33.00	32378	304.43	32546	32523.45	3151.48	1861.76	0.01
gr666	21013	21013	0.00	18000.00	20901	132.65	20762	18000.00	21077	21069.40	2317.14	1205.05	**-0.30**
u724	34988	34988	0.00	18000.00	34921	40.93	34554	629.02	34987	34952.45	3959.01	2095.75	0.00
rat783	7829	7829	0.00	18000.00	7548	13.35	7713	8573.52	7832	7772.45	1208.69	572.72	**-0.04**
dsj1000	27357	27357	0.00	18000.00	25352	48.13	26573	18000.00	27431	27407.05	3693.85	2017.84	**-0.27**
pr1002	23527	23527	0.00	18000.00	22482	35.67	22832	11291.20	23590	23463.60	2966.59	1898.84	**-0.27**
u1060	51775	51768	0.01	18000.00	51775	150.58	51593	2079.50	51849	51795.15	6074.41	3741.43	**-0.14**
vm1084	38678	38678	0.00	18000.00	38228	50.34	37970	7560.86	38700	38695.65	4929.41	2658.18	**-0.06**
pcb1173	56010	55954	0.10	18000.00	56010	77.73	55618	6709.24	56018	55926.85	7185.71	5406.83	**-0.01**
d1291	4029	4029	0.00	2335.60	4024	45.07	4029	1707.97	4029	4029.00	1043.51	7.06	0.00
rl1304	57782	57782	0.00	18000.00	57545	112.18	57576	8261.13	58220	58137.95	7787.85	5404.39	**-0.75**
rl1323	65664	65476	0.29	18000.00	65664	99.81	65166	905.12	65667	65617.10	7640.51	4957.15	0.00
nrw1379	69214	69119	0.14	18000.00	69214	152.00	69150	2234.08	69184	69156.80	7689.43	3984.67	0.04
fl1400	70511	70476	0.05	18000.00	70488	287.75	70511	3310.76	70530	70528.45	7588.49	2439.46	**-0.03**
u1432	54540	54540	0.00	18000.00	53550	127.79	52742	14148.70	54490	54218.35	8482.44	6081.59	0.09
fl1577	33754	22191	34.26	18000.00	33754	200.71	31118	18000.00	34613	33808.55	4872.68	2631.43	**-2.48**
d1655	33231	29920	9.96	18000.00	31880	371.31	33231	18000.00	34203	33968.00	4756.13	3447.54	**-2.84**
vm1748	82126	81778	0.42	18000.00	82126	265.55	81786	18000.00	82461	82399.45	12942.79	8541.65	**-0.41**
u1817	37457	31800	15.10	18000.00	36416	418.80	37457	18000.00	38576	38106.95	5505.37	4524.52	**-2.90**
rl1889	83875	71527	14.72	18000.00	83081	363.35	83875	15860.70	84827	84708.10	14813.47	10162.37	**-1.12**
d2103	37124	31045	16.37	18000.00	34192	465.36	37124	18000.00	37825	37399.55	5371.52	3715.71	**-1.85**
u2152	55397	48472	12.50	18000.00	54744	906.84	55397	18000.00	57972	57435.30	9685.72	8004.51	**-4.44**
u2319	110995	110995	0.00	18000.00	110960	438.26	110555	18000.00	111327	111146.50	18000.42	15008.89	**-0.30**
pr2392	50944	45407	10.87	18000.00	50902	285.26	50944	18000.00	54000	53252.75	8483.59	7303.01	**-5.66**
pcb3038	101173	91831	9.23	18000.00	101173	800.13	99612	18000.00	104367	104010.45	18000.18	15932.87	**-3.06**
fl3795	80069	71328	10.92	18000.00	80069	4496.09	76916	18000.00	94492	92311.60	17333.96	16339.19	**-15.26**
fnl4461	85088	84098	1.16	18000.00	85088	1490.80	83032	18000.00	92721	91987.95	17380.33	14834.60	**-8.23**
rl5915	279430	279116	0.11	18000.00	279277	8438.60	279430	18000.00	281337	280645.75	18000.03	17349.20	**-0.68**
rl5934	137838	-	-	-	137838	4037.07	134787	18000.00	158854	157125.89	18000.55	17000.64	**-13.23**
pla7397	142399	106131	25.47	18000.00	142399	6667.36	136820	18000.00	154773	153488.55	18000.06	16700.21	**-7.99**
Average	52500.66	47789.00	-	17087.41	52195.10	767.54	51874.02	9256.99	55540.73	55255.05	7062.81	5296.76	-

Table 6.12 – Results for the PCTSP on the medium-sized instances of Set I.

Instances	B&C [BGP09]		HGA-Giant				HGA				
	UB	Time	*Best*	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
st70	260*	0.85	260	273.30	1266.53	664.74	260	260.00	792.75	5.85	0.00
eil76	235*	0.94	220	230.00	734.33	377.61	213	213.30	488.85	120.07	**-3.18**
pr76	41248*	2.39	41248	41248.30	2110.90	686.59	41248	41248.00	1262.36	13.95	0.00
gr96	20688*	38.05	20688	20688.00	2126.95	464.85	20697	20697.00	1477.65	18.70	0.04
rat99	581*	14.30	581	582.45	1682.12	642.50	581	582.00	968.70	295.95	0.00
kroA100	9184*	9.11	9184	9342.90	2045.73	440.68	9184	9184.00	1446.20	20.41	0.00
kroB100	9096*	5.72	9096	9184.60	2122.75	571.52	9096	9098.05	1199.66	274.31	0.00
kroC100	9457*	18.26	9457	9701.65	1934.45	665.54	9457	9457.00	1441.24	18.38	0.00
kroD100	8719*	6.30	8997	9434.85	2138.77	606.55	8719	8719.00	1430.61	37.08	0.00
kroE100	9097*	6.87	9097	9249.40	2148.35	770.65	9097	9097.00	1543.35	23.68	0.00
rd100	3168*	6.53	3210	3243.15	2056.05	624.37	3168	3168.00	1538.52	94.76	0.00
eil101	232*	4.36	248	257.90	1035.23	376.38	232	232.20	741.61	277.42	0.00
lin105	5920*	168.02	5954	6001.45	1986.56	432.48	5920	5920.00	1495.24	26.85	0.00
pr107	18311*	6.87	18311	18315.80	2159.51	1041.38	18311	19313.10	1506.69	525.56	0.00
pr124	22998*	13.30	22998	23183.20	2320.54	600.97	22998	22998.00	1417.57	18.90	0.00
bier127	26347*	4.49	26347	26752.15	2354.50	895.53	26347	26347.00	1361.66	33.11	0.00
ch130	2408*	8.64	2426	2499.30	2221.70	714.48	2408	2408.00	1347.44	43.04	0.00
pr136	46167*	71.88	47087	47363.85	2189.83	929.18	46167	46167.00	1591.33	233.84	0.00
gr137	29575*	10.62	29575	29593.70	2215.35	639.47	29575	29575.00	1667.00	5.24	0.00
pr144	27424*	84.26	28061	28077.55	2265.45	690.53	27424	27424.00	1767.58	44.92	0.00
ch150	2760*	22.95	2792	2913.70	2201.50	890.90	2760	2760.30	1603.15	455.81	0.00
kroA150	11496*	2137.76	11649	12051.80	2286.16	975.07	11496	11496.00	1699.46	84.79	0.00
kroB150	11357*	36.53	11452	11956.95	2124.50	875.65	11357	11357.00	1850.63	48.11	0.00
pr152	36333*	68.85	36606	36850.00	2270.58	824.73	36333	36333.00	1995.90	118.36	0.00
u159	18511*	570.01	18689	18902.80	2419.91	748.44	18511	18511.00	1764.78	26.31	0.00
rat195	1112*	156.26	1129	1143.85	2096.24	763.96	1112	1112.45	1543.24	741.92	0.00
d198	6913*	1366.88	6929	6948.10	2341.96	748.18	6913	6913.00	1991.81	38.38	0.00
kroA200	12372*	118.79	12898	13630.75	2434.66	1440.98	12372	12380.65	2186.29	974.14	0.00
kroB200	12338*	351.33	12747	13319.45	2402.80	1184.53	12338	12338.00	1777.11	54.89	0.00
gr202	13790*	328.51	13894	14072.85	2397.94	708.81	13790	13796.15	1708.05	585.02	0.00
ts225	57995	14400.00	57535	58461.45	2529.29	765.36	57535	57535.00	1998.88	6.50	0.00
tsp225	1721*	317.29	1822	1881.30	2272.71	870.35	1721	1721.75	1948.02	897.45	0.00
pr226	36720*	5429.52	36935	38738.25	2463.89	1079.71	36720	37151.00	1900.28	980.77	0.00
gr229	39875*	154.54	40822	42451.30	2518.79	916.35	39875	40144.75	1819.58	888.08	0.00
gil262	986*	165.14	1130	1186.50	2633.00	1285.05	991	994.55	2048.30	1002.03	0.51
pr264	22644*	532.23	22919	23268.90	2879.28	1740.95	22903	25727.60	2100.00	676.60	1.14
a280	1231*	303.65	1286	1326.65	2359.74	776.42	1252	1259.05	1922.74	709.23	1.71
pr299	23089	14400.00	23023	23426.00	2852.53	1176.44	22514	22522.80	2552.63	791.74	**-2.21**
lin318	15913*	2355.21	16418	16942.30	2652.22	948.33	15913	15913.00	2350.49	563.66	0.00
rd400	6284*	2110.73	6948	7317.45	3448.83	1987.45	6284	6316.15	2634.51	1678.07	0.00
fl417	5754	14400.00	5562	5624.00	3547.23	875.99	5449	5450.85	2771.75	994.36	**-2.03**
gr431	35222*	14285.70	35245	35747.50	3395.01	1295.71	35222	35224.60	2664.16	921.48	0.00
pr439	35297*	1483.28	36727	37401.65	3305.44	1200.81	35297	35350.65	2529.08	748.10	0.00
pcb442	22281	14400.00	23496	24007.00	3537.49	1244.83	22281	22301.10	3041.85	1485.59	0.00
d493	13582*	1943.26	14229	14448.15	3346.13	1587.33	13582	13600.95	3256.53	994.78	0.00
att532	8943*	10280.10	9289	9491.20	3787.90	1613.14	10433	10593.00	3270.04	1813.74	16.67
Average	16209.41	2230.44	16417.74	16711.59	2383.07	899.16	16218.61	16324.17	1813.38	443.74	-

Table 6.13 – Results for the PCTSP on the large-sized instances of Set I.

Instances	HGA-Giant				HGA				
	Best	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
ali535	47890	52262.80	4519.04	2758.00	42756	42984.10	3721.87	2148.39	**-10.72**
u574	15780	16551.30	4189.39	3013.66	14671	14700.85	3916.90	1981.10	**-7.03**
rat575	3270	3348.85	4099.71	1331.54	3023	3036.85	3545.65	1864.71	**-7.55**
p654	16552	17626.60	3941.21	1956.80	16173	16173.00	3259.72	804.99	**-2.29**
d657	21784	22598.00	4461.48	2880.53	20889	20904.65	3385.33	2084.09	**-4.11**
gr666	84119	86308.20	4358.81	2372.07	78410	80987.20	3552.02	1650.69	**-6.79**
u724	18543	19301.60	4991.53	4470.48	16692	16749.25	4003.05	1612.22	**-9.98**
rat783	4325	4411.10	5277.20	2495.65	3938	3979.55	4783.58	1768.90	**-8.95**
dsj1000	6997800	7075698.35	7245.65	5423.50	6940600	6956310.00	6896.05	4090.00	**-0.82**
pr1002	118568	122205.35	6048.96	5157.00	106138	108198.10	5336.69	3625.03	**-10.48**
u1060	100381	102363.10	7231.86	5815.26	88335	88665.00	5427.29	2880.01	**-12.00**
vm1084	76512	90154.95	7887.30	6953.26	65255	65256.90	5796.05	1498.21	**-14.71**
pcb1173	26934	27469.60	6878.13	5312.04	24916	24989.15	7144.23	3451.92	**-7.49**
d1291	24049	24600.05	7618.76	4238.72	23276	23380.85	6738.42	3042.44	**-3.21**
rl1304	114795	123217.65	9369.78	6994.19	100463	101120.10	8019.14	4149.27	**-12.48**
rl1323	132231	138641.90	9195.54	8503.08	107724	108437.25	8577.82	5991.34	**-18.53**
nrw1379	25518	25954.85	10504.95	6806.16	23831	23934.25	9366.99	3989.30	**-6.61**
fl1400	8084	8216.20	8644.44	4470.06	8336	8343.45	9068.82	4034.91	**3.12**
u1432	76281	78190.90	8709.27	4461.42	72688	72908.40	9032.67	3971.92	**-4.71**
fl1577	9941	10083.85	9384.07	4445.69	9728	9739.85	6949.05	4495.42	**-2.14**
d1655	29662	30511.45	11214.47	9003.36	28321	28730.45	8465.60	4938.04	**-4.52**
vm1748	112141	130574.70	16281.23	15758.16	82916	83133.20	10188.62	5078.04	**-26.06**
u1817	28613	29363.45	13185.74	8177.49	26490	26824.50	11088.22	6072.33	**-7.42**
rl1889	160227	167929.95	13974.32	11564.46	113498	114168.45	13340.98	7308.52	**-29.16**
d2103	36513	36972.85	11210.51	6603.01	34286	34287.90	13037.04	4268.58	**-6.10**
u2152	32478	33222.95	15191.68	10582.24	30649	30921.55	15537.73	7965.56	**-5.63**
u2319	118786	119444.20	15661.34	6877.61	116000	116000.00	16073.73	496.72	**-2.35**
pr2392	181451	185710.90	13677.79	5550.13	164029	164955.25	17622.28	10371.24	**-9.60**
pcb3038	68022	69834.65	18000.38	16520.10	62174	62818.70	18000.42	17591.35	**-8.60**
fl3795	13594	15375.20	18000.17	16308.70	12741	13404.40	18000.27	17055.30	**-6.27**
fnl4461	93107	94070.45	18000.19	17465.75	81399	81720.40	18000.59	17867.45	**-12.57**
rl5915	315805	323273.25	18000.65	13592.38	216241	218312.65	18001.11	17855.01	**-31.53**
rl5934	316957	323398.55	18000.26	10025.79	218703	222194.10	18000.77	17881.14	**-31.00**
pla7397	8837800	8896016.67	18000.63	15546.70	8296170	8328854.00	18001.35	17803.95	**-6.13**
Average	537309.21	544261.89	10381.07	7453.97	507395.85	509327.19	9761.18	6226.12	-

Table 6.14 – Results for the PCTSP on the medium-sized instances of Set II.

Instances	B&C [BGP09]		HGA-Giant				HGA				
	UB	Time	*Best*	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
st70	247*	1.31	247	254.60	1157.24	431.77	247	247.00	456.84	2.95	0.00
eil76	200*	6.44	202	206.20	574.44	244.88	200	200.00	290.26	49.88	0.00
pr76	38330*	22.13	38850	38977.00	2034.23	802.75	38330	38330.00	976.65	15.46	0.00
gr96	19380*	34.60	19380	19380.00	2215.10	55.86	19380	19380.00	1234.05	10.74	0.00
rat99	518*	61.50	526	535.60	1665.93	725.69	518	518.40	592.64	210.63	0.00
kroA100	8519*	29.70	8795	8975.55	2111.43	906.26	8519	8519.00	1190.93	95.89	0.00
kroB100	7794*	41.70	7821	8148.30	2333.69	977.55	7794	7794.00	1177.52	25.59	0.00
kroC100	9060*	41.16	9296	9421.35	2355.78	675.91	9060	9060.00	1417.05	10.46	0.00
kroD100	8267*	30.74	8459	8561.40	2377.51	786.14	8267	8267.00	1242.05	295.46	0.00
kroE100	7644*	17.90	8180	8663.35	2166.03	579.36	7644	7644.00	1239.45	19.25	0.00
rd100	2892*	22.29	2932	3009.95	2238.05	688.31	2892	2892.00	921.78	57.41	0.00
eil101	211*	9.11	221	230.40	1044.23	267.20	211	212.55	451.24	171.27	0.00
lin105	5614*	716.02	5802	5825.85	2069.88	648.43	5614	5622.15	1308.90	245.13	0.00
pr107	26372*	76.69	26485	26639.75	2371.05	1158.75	26372	26372.00	1433.83	111.59	0.00
pr124	23150*	162.39	23868	24103.75	2344.45	581.15	23150	23150.00	1267.54	26.94	0.00
bier127	24478*	37.55	24992	25129.05	2606.16	848.59	24478	24478.00	1276.40	30.21	0.00
ch130	2220*	83.66	2366	2435.55	2382.69	1007.95	2220	2220.00	1314.72	504.62	0.00
pr136	40241	14400.00	40636	41808.25	2156.29	1033.55	40023	40023.00	1403.00	202.39	**-0.54**
gr137	28242*	366.15	28242	28251.70	2292.34	753.94	28242	28242.00	1752.22	16.38	0.00
pr144	27073*	284.38	27449	28700.55	2697.84	1275.77	27073	27073.00	1450.82	29.21	0.00
ch150	2476*	541.81	2648	2740.70	2468.95	972.60	2476	2478.10	1360.78	235.18	0.00
kroA150	9968*	60.85	10715	11038.15	2540.79	974.39	9968	9968.00	1521.98	33.11	0.00
kroB150	10278*	469.95	10719	10939.35	2502.28	751.03	10278	10439.50	1657.29	72.20	0.00
pr152	34474*	249.75	34710	34912.80	2384.66	1117.21	34474	34474.30	1762.78	346.99	0.00
u159	17161*	763.28	18222	18597.75	2291.02	948.67	17161	17161.00	1617.72	63.57	0.00
rat195	988*	112.81	1031	1046.30	2280.34	916.64	990	994.25	1045.83	520.83	0.20
d198	6653*	2579.62	6676	6705.95	2458.00	729.38	6653	6653.00	1604.25	419.63	0.00
kroA200	11219*	2278.69	12027	12624.50	2626.21	1387.48	11219	11251.60	1898.65	850.61	0.00
kroB200	11250*	415.38	12799	13325.25	2749.84	1116.39	11250	11250.00	1811.84	180.79	0.00
gr202	12804*	753.52	13274	13391.90	2712.00	998.64	12804	12808.10	1600.15	635.79	0.00
ts225	53102*	907.77	54975	56442.90	2530.42	918.77	53102	53102.00	1583.29	229.16	0.00
tsp225	1585*	1803.93	1723	1782.85	2517.27	661.19	1585	1589.45	1463.25	842.61	0.00
pr226	36190*	8186.06	37088	38052.90	2426.58	885.13	36190	36775.45	1947.51	686.58	0.00
gr229	35856*	3478.96	36615	37188.75	2703.52	1180.40	35856	36020.80	1960.93	407.65	0.00
gil262	865*	165.21	1043	1095.30	2942.29	1356.09	865	865.55	1463.20	639.99	0.00
pr264	23660	14400.00	22790	23118.90	2758.50	1253.41	25080	25099.60	2121.82	634.38	10.05
a280	1143	14400.00	1217	1262.25	2643.18	1115.98	1131	1138.70	1363.13	556.05	**-1.05**
pr299	20613	14400.00	21636	22019.90	2677.80	1171.38	20534	20591.45	2247.42	1298.94	**-0.38**
lin318	14909*	3394.98	15404	16223.55	2748.88	1212.81	14909	14925.95	2345.94	953.60	0.00
rd400	5590*	3102.53	7082	7226.35	3394.91	1067.25	5590	5781.40	2517.79	1409.09	0.00
fl417	5971	14400.00	5466	5542.15	3552.72	1282.62	5354	5359.15	2366.97	858.14	**-2.05**
gr431	31725	14400.00	33331	33932.05	3792.43	2189.49	31725	31725.00	2453.09	835.34	0.00
pr439	33110	14400.00	34534	35038.95	3943.30	1696.64	33079	33086.10	2653.69	668.86	**-0.09**
pcb442	19165	14400.00	21878	22446.90	3990.55	1727.80	19162	19188.05	2909.42	1160.58	**-0.02**
d493	12835	14400.00	14240	14554.30	3811.67	1316.84	12687	12719.25	2915.55	1106.52	**-1.15**
att532	8231	14400.00	9068	9200.75	4335.16	1312.68	9792	9939.30	3210.29	1526.12	18.96
Average	15266.80	3811.10	15775.22	16080.64	2542.99	971.97	15307.57	15339.76	1604.40	419.65	-

Table 6.15 – Results for the PCTSP on the large-sized instances of Set II.

Instances	HGA-Giant				HGA				
	Best	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
ali535	47600	50836.75	5481.66	2164.89	40838	41073.30	3416.66	2160.47	**-14.21**
u574	15790	16337.25	4802.40	1522.94	13660	13738.10	3456.81	2078.83	**-13.49**
rat575	3005	3073.95	4746.15	2022.61	2700	2712.00	2724.57	1257.38	**-10.15**
p654	16233	16492.95	4757.14	2166.05	15461	15469.85	3351.91	1702.63	**-4.76**
d657	21075	21497.50	5872.18	1932.54	18950	18979.90	3755.76	1820.76	**-10.08**
gr666	83808	86094.80	5538.10	2295.56	75702	76431.60	3826.03	1844.19	**-9.67**
u724	18070	18984.40	4889.32	1300.49	14924	15030.60	4356.68	2933.25	**-17.41**
rat783	4081	4150.45	5136.55	1833.39	3446	3513.55	3926.83	1968.81	**-15.56**
dsj1000	6658210	6698721.50	6578.54	5468.90	6428930	6473253.50	6368.08	4346.50	**-3.44**
pr1002	119676	122904.35	6145.16	1718.06	98795	100118.00	6100.34	3884.73	**-17.45**
u1060	97788	100108.30	7116.67	1771.21	81534	82636.45	6000.21	3785.21	**-16.62**
vm1084	69869	78092.75	9083.64	4339.40	63684	63744.74	6639.36	3508.90	**-8.85**
pcb1173	26350	27363.60	9777.27	3449.71	22982	23223.75	6765.56	3839.71	**-12.78**
d1291	23583	24179.10	8895.48	4675.84	22148	22327.05	5478.72	3323.13	**-6.08**
rl1304	113099	118157.60	8620.93	3407.41	95589	96046.30	6363.72	2790.89	**-15.48**
rl1323	121711	129922.50	9484.58	3661.33	102312	103271.80	6746.93	4499.24	**-15.94**
nrw1379	25049	25339.80	12061.47	5763.69	20805	21200.75	8473.57	4805.47	**-16.94**
fl1400	8064	8389.95	11851.43	8180.52	7732	7780.40	6437.19	3320.17	**-4.12**
u1432	73071	73931.15	9896.33	3153.81	58418	59232.45	7718.68	4745.64	**-20.05**
fl1577	9836	10023.90	9579.35	4623.14	9111	9174.95	8257.34	5592.10	**-7.37**
d1655	30854	31228.80	12543.25	5283.63	26257	26735.05	9320.19	7625.40	**-14.90**
vm1748	92731	102265.20	14105.31	8951.64	80034	80407.85	10982.57	6402.71	**-13.69**
u1817	29390	30224.75	12691.13	5159.59	24316	24679.65	9936.50	7717.81	**-17.26**
rl1889	141203	148699.15	16072.90	5702.52	110226	111197.95	12595.92	8933.69	**-21.94**
d2103	35451	36533.90	14148.48	6486.04	32935	32968.20	11825.39	5348.26	**-7.10**
u2152	33168	33655.30	15866.23	6010.47	27543	27942.90	10232.30	8366.51	**-16.96**
u2319	110241	112362.63	17030.77	6024.11	84351	85185.30	12510.91	9243.60	**-23.48**
pr2392	180615	183115.50	18000.32	7757.63	152816	156057.50	16755.98	16166.88	**-15.39**
pcb3038	67488	69128.40	18000.44	9210.16	55810	56828.40	18000.24	17887.34	**-17.30**
fl3795	14267	15973.30	18000.34	12775.48	11859	12732.70	18000.24	17682.58	**-16.88**
fnl4461	90189	91179.85	18000.37	12372.52	72006	73024.85	18000.60	17851.83	**-20.16**
rl5915	276743	290102.75	18000.48	16072.21	209848	212486.30	18001.09	17859.98	**-24.17**
rl5934	278156	293190.80	18000.33	16426.86	213500	215103.75	18000.86	17781.86	**-23.24**
pla7397	8641000	8641000.00	18000.20	13202.50	7376900	7466082.50	18001.12	17870.27	**-14.63**
Average	516984.24	520978.32	11140.44	5790.79	461062.41	465599.76	9186.14	7086.67	**-10.82**

Table 6.16 – Results for the PCTSP on the medium-sized instances of Set III.

Instances	B&C [BGP09]		HGA-Giant				HGA				
	UB	Time	*Best*	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
st70	308*	7.71	309	311.65	1113.25	375.41	308	308.65	525.64	228.30	0.00
eil76	204*	9.05	206	208.85	671.13	266.01	204	204.00	375.40	14.80	0.00
pr76	42200*	24.02	42955	43378.25	2157.69	917.24	42200	42200.00	1261.54	8.22	0.00
gr96	22491*	50.48	22340	22615.75	2297.87	999.25	22316	22354.75	1407.77	413.90	**-0.11**
rat99	579*	55.05	600	611.05	1453.40	531.32	580	582.65	826.28	284.86	0.17
kroA100	8325*	17.47	8325	8444.70	2237.71	1008.02	8325	8325.00	1189.84	32.95	0.00
kroB100	8768*	42.17	8768	8829.25	2203.32	812.81	8768	8774.95	1222.64	403.58	0.00
kroC100	9283*	86.21	9410	9546.70	2222.38	923.35	9283	9363.90	1440.03	179.19	0.00
kroD100	8998*	57.52	8998	9063.60	2090.52	774.45	8998	8998.00	1278.01	24.81	0.00
kroE100	9313*	41.76	9398	9437.75	2273.53	722.27	9313	9313.00	1215.78	22.28	0.00
rd100	3377*	51.72	3377	3394.10	2010.36	813.90	3377	3419.00	1368.30	317.12	0.00
eil101	223*	17.55	230	232.05	1094.94	451.81	224	224.55	525.92	171.42	0.45
lin105	6547*	423.40	6667	6706.10	2172.05	845.92	6547	6547.00	1426.18	78.27	0.00
pr107	27198	14400.00	27198	27258.10	2400.69	1008.50	27184	27184.00	1097.56	24.53	**-0.05**
pr124	26375*	206.93	26785	27137.35	2512.35	1040.17	26375	26375.00	1184.75	10.29	0.00
bier127	42358*	4654.12	42930	43118.30	2501.51	868.81	42359	42360.40	1816.47	644.71	0.00
ch130	2305*	60.85	2338	2352.75	2408.65	632.41	2305	2312.55	1326.79	557.86	0.00
pr136	42179*	4564.78	43227	43905.45	2719.20	1166.28	42179	42188.10	1524.84	482.31	0.00
gr137	34023	14400.00	33714	34140.45	2598.89	842.07	33270	33403.95	1840.08	724.72	**-1.32**
pr144	30033	14400.00	30123	30402.00	2574.98	1030.72	29746	29746.00	1553.85	215.12	**-0.96**
ch150	2675*	132.36	2706	2740.70	2541.85	939.59	2675	2678.50	1328.33	134.24	0.00
kroA150	9409*	78.72	9750	9957.95	2601.10	1229.45	9409	9409.00	1412.88	189.20	0.00
kroB150	10392*	256.38	10763	10927.35	2428.56	617.70	10564	10564.00	1625.80	150.05	1.66
pr152	40937	14400.00	41488	41936.60	2224.23	1079.51	40599	40599.00	1511.65	27.00	**-0.83**
u159	17631*	328.30	17670	17803.75	2324.98	852.79	17631	17670.70	1705.66	616.38	0.00
rat195	999*	776.12	1086	1112.55	2285.45	1181.38	1013	1049.85	1257.77	458.17	1.40
d198	7388*	1974.04	7435	7472.80	2525.87	758.28	7388	7388.00	2067.25	329.20	0.00
kroA200	11987*	803.92	12293	12787.70	2760.52	847.09	12075	12104.85	1904.79	962.35	0.73
kroB200	10752*	1398.61	10888	11157.40	2858.80	1039.78	10752	10752.00	1729.95	378.08	0.00
gr202	14377*	5085.50	14806	14903.80	2780.78	928.50	14546	14558.75	1829.80	858.27	1.18
ts225	53414	14400.00	53325	53438.85	2402.22	702.62	53325	53325.00	1419.70	240.27	0.00
tsp225	1649	14400.00	1707	1749.30	2520.54	893.41	1649	1652.55	1416.66	569.65	0.00
pr226	39091	14400.00	39296	39669.05	2505.70	929.80	38874	38912.25	1946.43	875.65	**-0.56**
gr229	46791*	4004.14	47146	48360.10	3190.54	1254.60	46749	47162.40	2337.96	672.78	**-0.09**
gil262	961*	387.02	1026	1057.85	2796.33	893.21	966	970.00	1462.00	580.02	0.52
pr264	23264	14400.00	23230	24015.65	3264.81	1951.50	23093	23108.45	1539.55	220.56	**-0.59**
a280	1084*	4324.35	1085	1094.90	2415.33	994.18	1087	1088.80	1230.21	572.81	0.28
pr299	20317*	5129.46	21324	21577.95	2922.90	1517.35	20317	20448.00	2148.44	1495.82	0.00
lin318	16401*	6867.39	17798	18404.75	3380.24	1080.94	16401	16402.15	2370.42	871.77	0.00
rd400	5700*	2602.41	6253	6558.75	3699.86	1367.74	5877	5895.65	2376.49	857.93	3.11
fl417	5740	14400.00	5553	5620.95	4028.67	2096.99	5368	5377.45	1103.21	357.33	**-3.33**
gr431	56484	14400.00	63656	65237.30	4405.86	1619.90	55817	57198.60	3233.26	1842.90	**-1.18**
pr439	35771	14400.00	37236	37795.45	3940.71	1968.45	35788	35814.35	2473.97	967.82	0.05
pcb442	19632	14400.00	21316	21976.10	3715.95	1426.35	19666	20028.25	2413.00	1819.01	0.17
d493	13480	14400.00	14137	14542.65	3513.02	1479.74	13507	13517.30	2902.48	995.09	0.20
att532	10258	14400.00	11953	12056.35	5744.27	2534.88	10315	10477.60	3437.85	1944.92	0.56
Average	17427.63	5350.42	17887.48	18153.28	2641.16	1048.18	17376.35	17442.15	1621.59	517.97	0.03

Table 6.17 – Results for the PCTSP on the large-sized instances of Set III.

Instances	HGA-Giant				HGA				
	Best	*Avg.*	Time	TMB	*Best*	*Avg.*	Time	TMB	*Gap*(%)
ali535	65661	66246.25	5730.36	2094.07	68710	69764.6	3977.863	2677.96	4.64
u574	16157	16397.70	4795.63	2385.67	13730	13822.35	3007.96	1315.915	**-15.02**
rat575	2917	2956.90	5530.21	2681.86	2658	2701.3	2660.836	1695.456	**-8.88**
p654	16085	16289.70	4901.00	2363.43	15853	15881.9	3031.258	1607.091	**-1.44**
d657	23671	24151.15	5338.83	1786.25	21380	21875.45	4052.302	2953.046	**-9.68**
gr666	95945	97270.30	7417.76	4435.51	92942	93215.35	4476.175	2486.513	**-3.13**
u724	16562	17093.85	5561.84	2368.25	14054	14093.9	3523.882	1816.394	**-15.14**
rat783	3681	3829.80	7990.99	3744.49	3503	3608	3702.949	1751.809	**-4.84**
dsj1000	6126650	6336921.60	6543.15	5799.36	6099840	6181059	5648.652	4855.394	**-0.44**
pr1002	111941	114649.10	11067.47	6027.71	92882	93979.3	5509.153	4359.783	**-17.03**
u1060	82282	86070.05	7319.40	2661.46	73035	73956.45	5231.196	3551.189	**-11.24**
vm1084	63126	66508.45	8700.17	4304.68	57945	58226.8	5199.252	3100.454	**-8.21**
pcb1173	24908	25581.65	13603.76	7447.96	22969	23934.6	6636.45	5696.322	**-7.78**
d1291	25119	25887.70	8792.17	4146.81	23049	23128.55	7140.085	3542.74	**-8.24**
rl1304	106362	115222.50	12498.09	6158.64	81770	82141.55	6307.316	4078.16	**-23.12**
rl1323	111527	117009.25	10944.81	4032.25	90419	90907.53	6129.159	4270.05	**-18.93**
nrw1379	22078	22430.05	18000.09	11323.12	21446	22760.75	7500.976	6377.507	**-2.86**
fl1400	7301	7421.05	18000.11	7555.35	6975	7007.15	5764.467	2576.688	**-4.47**
u1432	57658	59432.30	11420.94	5401.35	51171	51515.65	5986.278	3438.962	**-11.25**
fl1577	9801	10273.10	18000.14	8630.82	8967	8984.75	6507.709	3432.093	**-8.51**
d1655	31294	32498.55	14776.31	6702.76	27553	27646.1	8515.859	4979.303	**-11.95**
vm1748	103042	125572.90	18000.19	8477.87	67744	67934.45	8315.873	3773.694	**-34.26**
u1817	24718	25525.95	16373.74	6680.61	21427	21681.15	7770.203	5462.698	**-13.31**
rl1889	136123	143098.40	16635.66	7346.77	101257	101533	9976.649	6185.451	**-25.61**
d2103	30504	31208.05	18000.16	9900.11	29254	29261.45	8875.615	2896.752	**-4.10**
u2152	27756	28669.35	18000.13	8352.88	23677	24148.85	7911.774	5968.639	**-14.70**
u2319	95562	98274.15	18000.17	11252.63	86288	86996.95	11012.27	6800.371	**-9.70**
pr2392	152349	155399.85	18000.20	8465.84	151021	152901.5	13270.02	11648.99	**-0.87**
pcb3038	57885	59404.50	18000.18	9279.09	51095	54014.25	17992.01	17107.99	**-11.73**
fl3795	14806	15673.40	18000.43	7880.81	11850	11986.3	18002.67	14258.02	**-19.96**
fnl4461	78257	80511.60	18000.27	9824.28	77167	78417.1	18000.35	17879.18	**-1.39**
rl5915	263766	290963.05	18000.64	10836.60	177708	180451.5	18000.44	17735.67	**-32.63**
rl5934	284844	308088.90	18000.47	9457.43	183885	186586.2	18000.89	17772.43	**-35.44**
pla7397	6368310	6556739.50	18000.64	13159.04	6364860	6448766	18000.74	17789.99	**-0.05**
Average	431136.71	278919.72	12880.77	6557.82	418767.18	424261.46	8401.15	6348.31	-

6.4 Computational results on the SDVRP instances

Detailed comparative results between the proposed SplitMA and the reference algorithms on the four sets of benchmark instances are provided in Tables 6.18–6.27. Following [SSO15], we provide for the instances of Set I the results using both real and rounded costs (the distance matrices of these instances with round costs are obtained from http://dimacs.rutgers.edu/programs/challenge/vrp/vrpsd/). For the other benchmark sets, we report real value costs like in the literature.

Table 6.18 – Results for the SDVRP-LF on the instances of Set I.

Instances	LB	BKS	iVNDiv		RGTS			SplitILS			SplitMA				
			Best	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap(%)*	Time	TMB
eil22	-	375.28	375.28	4.19	375.28	375.28	0.00	375.28	375.28	0.14	375.28	375.28	0.00	0.13	0.02
eil23	525.65	568.56	569.75	3.42	598.56	568.56	0.00	568.56	568.56	0.12	568.56	568.56	0.00	0.11	0.04
eil30	-	512.72	512.72	14.47	519.70	525.33	210.00	512.72	512.72	0.32	512.72	512.72	0.00	0.22	0.10
eil33	-	837.06	853.10	14.03	843.64	843.64	29.00	837.06	837.06	0.45	837.06	837.06	0.00	46.98	0.41
eil51	518.26	524.61	524.61	54.91	524.93	531.24	11.00	524.61	524.61	1.63	524.61	524.61	0.00	0.50	0.49
eilA76	809.67	823.89	851.24	83.28	860.86	-	37.00	823.89	825.22	27.25	823.89	823.89	0.00	122.50	19.77
eilB76	985.42	1009.04	1059.57	79.00	1023.23	1023.32	23.00	1009.04	1011.19	44.98	1009.04	1011.20	0.00	140.38	52.55
eilC76	723.55	738.67	753.29	148.20	746.34	774.20	23.00	738.67	739.83	15.68	738.67	738.67	0.00	122.60	13.84
eilD76	672.54	687.60	699.35	140.83	702.26	702.26	31.00	687.60	688.37	9.92	686.70	687.24	**-0.13**	117.77	27.89
eilA101	803.62	826.14	852.74	319.33	849.98	851.23	61.00	826.14	826.26	36.59	826.14	826.70	0.00	148.83	29.26
eilB101	1055.40	1076.26	1139.27	185.84	1112.15	1112.29	73.00	1076.26	1078.58	101.26	1076.01	1076.93	**-0.02**	169.50	74.25
S51D1	457.10	459.50	471.92	40.53	459.50	459.93	12.00	459.50	459.50	1.07	459.50	459.50	0.00	0.35	0.33
S51D2	700.40	708.42	731.01	28.34	723.97	723.32	1.00	708.42	709.54	9.98	708.42	708.60	0.00	98.05	23.32
S51D3	938.50	948.01	1001.22	14.70	970.67	970.89	4.00	948.01	949.96	14.15	947.97	947.97	**0.00**	104.49	9.85
S51D4	1549.70	1561.01	1680.66	16.53	1614.10	1614.90	14.00	1561.01	1563.25	59.96	1560.88	1561.21	**-0.01**	246.21	140.36
S51D5	1326.61	1333.67	1389.40	13.94	1381.68	1385.31	3.00	1333.67	1333.85	32.41	1333.67	1334.47	0.00	145.26	40.99
S51D6	2165.64	2169.10	2218.23	16.83	2213.93	2215.41	2.00	2169.10	2174.71	83.79	2169.10	2170.60	0.00	275.95	94.04
S76D1	592.60	598.94	606.47	476.27	629.62	629.62	101.00	598.94	598.98	4.54	598.94	598.94	0.00	101.67	4.38
S76D2	1071.30	1087.99	1143.36	46.94	1113.43	1113.43	10.00	1087.99	1089.69	74.51	1087.40	1088.53	**-0.05**	147.69	66.14
S76D3	1407.54	1427.81	1490.08	53.34	1459.96	1461.20	15.00	1427.81	1429.01	88.72	1425.73	1428.31	**-0.15**	164.97	58.95
S76D4	2059.80	2079.76	2173.61	51.84	2103.05	2103.05	14.00	2079.76	2080.76	173.55	2079.74	2079.84	**0.00**	217.01	104.34
S101D1	716.80	726.59	749.19	2125.58	791.21	791.55	123.00	726.59	728.44	14.16	726.59	726.62	0.00	135.55	28.87
S101D2	1358.90	1383.35	1443.44	217.91	1415.92	1417.40	21.00	1383.35	1386.45	129.94	1377.89	1384.74	**-0.39**	198.08	88.04
S101D3	1853.10	1876.97	1988.78	146.61	1907.92	1907.92	19.00	1876.97	1881.26	277.62	1874.84	1880.13	**-0.11**	245.04	142.05
S101D5	2767.60	2792.01	2984.48	104.05	2896.00	2898.50	14.00	2792.01	2795.73	696.64	2789.81	2798.39	**-0.08**	876.06	646.71
Average	-	1085.32	1130.51	176.04	1113.516	-	-	1085.32	1086.75	75.98	**1084.77**	**1086.03**	-	153.03	66.68
Best#	-	-	0	-	0	0	-	0	4	-	**10**	**15**	-	-	-
p-value	-	5.46E-03	2.67E-05	-	2.70E-05	2.35E-05		5.46E-03	1.01E-02	-	-	-	-	-	-

Table 6.19 – Results for the SDVRP-LF on the instances of Set I with rounded costs.

Instances	LB	BKS	iVNDiv		SplitILS			SplitMA				
			Best	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
eil22	375.00	375	375	4.19	375	375	0.13	375	375.00	0.00	33.38	0.04
eil23	569.00	569	570	3.42	569	569	0.09	569	569.05	0.00	31.77	0.16
eil30	510.00	510	510	14.47	510	510	0.3	510	510.00	0.00	44.28	0.12
eil33	834.70	835	851	14.03	835	835	0.39	835	835.00	0.00	43.80	0.13
eil51	521.00	521	521	54.91	521	521.55	1.63	521	521.50	0.00	61.05	1.13
eilA76	807.60	818	847	83.28	818	820.45	25.68	818	824.30	0.00	96.83	49.36
eilB76	981.40	1002	1055	79	1002	1005.8	38.05	1002	1006.90	0.00	106.99	38.11
eilC76	717.80	733	746	148.2	733	733.55	15.17	733	737.40	0.00	88.71	20.60
eilD76	666.10	681	695	140.83	681	683	11.02	682	685.50	0.15	87.72	15.16
eilA101	799.80	814	843	319.33	815	815.85	32.7	817	819.30	0.37	106.37	12.39
eilB101	1040.60	1061	1122	185.84	1061	1065.4	75.43	1061	1075.10	0.00	120.25	61.02
S51D1	454.40	458	466	40.53	458	458	1.21	458	458.00	0.00	55.13	0.28
S51D2	694.20	703	725	28.34	703	704.65	8.32	703	703.00	0.00	81.68	14.26
S51D3	935.17	942	994	14.7	943	944.2	13.58	942	942.00	0.00	95.09	21.03
S51D4	1547.00	1551	1672	16.53	1552	1555.55	47.34	1551	1551.00	0.00	353.89	87.78
S51D5	1325.34	1328	1385	13.94	1328	1329.15	33.46	1328	1328.00	0.00	194.67	36.55
S51D6	2153.00	2153	2211	16.83	2163	2165.7	65.68	2156	2156.11	0.14	280.29	124.06
S76D1	592.00	592	600	476.27	592	592.45	4.75	592	593.25	0.00	76.83	2.87
S76D2	1061.10	1081	1138	46.94	1081	1083.35	59.2	1081	1081.80	0.00	139.35	40.61
S76D3	1395.90	1419	1485	53.34	1419	1422.05	8.07	1420	1420.20	0.07	162.39	57.44
S76D4	2046.10	2071	2160	51.84	2071	2074.3	148.48	2072	2072.95	0.05	265.37	127.91
S101D1	716.00	716	740	2125.58	716	718.4	14.17	716	719.00	0.00	91.57	13.08
S101D2	1337.10	1364	1426	217.91	1364	1370.95	116.33	1360	1369.74	**-0.29**	140.02	75.89
S101D3	1832.20	1859	1974	146.61	1859	1868.75	233.36	1858	1862.95	**-0.05**	199.14	114.72
S101D5	2737.10	2770	2970	104.05	2772	2779.65	579.68	2767	2775.56	**-0.11**	989.28	877.67
Avgerage	-	1077.04	1123.24	176.04	1077.64	1080.07	61.37	**1077.08**	**1079.70**	-	157.83	71.70
Best#	-	-	0	-	**3**	9	-	**3**	**12.00**	-	-	-
p-value	-	8.52E-01	4.00E-05	-	3.42E-01	4.34E-01	-	-	-	-	-	-

Table 6.20 – Results for the SDVRP-LF on the instances of Set II.

Instances	BKS	SS		iVNDiv		SplitILS			SplitMA				
		Best	Time	*Best*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
p01-50	524.61	524.61	49.70	524.61	54.91	524.61	524.61	1.87	524.61	524.61	0.00	73.29	0.62
p01-50D1	460.79	460.79	51.80	471.92	33.70	460.79	460.79	1.16	460.79	460.79	0.00	59.53	0.35
p01-50D2	741.06	741.06	66.40	766.19	19.77	741.06	741.26	9.87	741.06	741.06	0.00	88.95	2.19
p01-50D3	982.77	997.83	87.10	1039.89	18.16	982.77	983.70	18.44	982.77	982.77	0.00	110.82	29.23
p01-50D4	1456.00	1554.38	92.60	1522.43	16.36	1456.00	1456.87	46.74	1456.00	1456.00	0.00	162.84	14.05
p01-50D5	1467.47	1532.19	92.40	1540.39	15.33	1467.47	1467.47	48.93	1467.47	1467.47	0.00	145.46	13.50
p01-50D6	2154.21	2312.48	5.80	2215.34	18.70	2154.21	2154.51	83.85	2154.21	2154.63	0.00	289.48	144.16
p02-75	823.89	829.01	166.50	851.24	83.28	823.89	824.77	30.84	823.89	823.89	0.00	119.36	19.36
p02-75D1	596.25	596.99	144.00	597.46	303.77	596.25	596.25	5.00	596.25	596.25	0.00	97.13	5.57
p02-75D2	1064.49	1071.87	143.80	1099.47	73.05	1064.49	1066.87	53.42	1064.49	1065.41	0.00	145.60	50.85
p02-75D3	1393.11	1463.60	126.80	1478.67	67.80	1393.11	1393.11	101.77	1393.11	1393.18	0.00	159.46	48.46
p02-75D4	2081.38	2182.34	119.90	2200.51	71.11	2081.38	2084.62	219.74	2074.57	2080.37	**-0.33**	268.47	130.09
p02-75D5	2112.19	2228.90	11.10	2238.98	80.30	2112.19	2113.38	267.72	2104.37	2112.46	**-0.37**	272.09	150.94
p02-75D6	3179.20	3387.86	10.50	3304.24	58.05	3179.20	3181.30	441.77	3173.48	3178.53	**-0.18**	446.40	323.80
p03-100	826.14	829.45	276.10	852.74	319.33	826.14	826.39	40.81	826.14	826.70	0.00	147.77	28.78
p03-100D1	726.81	726.81	272.10	745.35	2194.23	726.81	730.01	17.72	726.81	726.81	0.00	140.47	32.32
p03-100D2	1376.09	1397.50	305.10	1425.90	190.53	1376.09	1380.28	182.16	1373.85	1381.60	**-0.16**	199.30	103.36
p03-100D3	1823.17	1908.02	225.20	1956.13	154.47	1823.17	1827.47	326.55	1822.25	1826.76	**-0.05**	219.96	101.25
p03-100D4	2751.13	2894.21	177.90	2865.86	126.52	2751.13	2754.52	629.59	2745.81	2750.08	**-0.19**	335.31	212.80
p03-100D5	2813.82	2986.33	17.00	2941.64	103.94	2813.82	2817.05	737.35	2812.04	2814.61	**-0.06**	344.25	197.63
p03-100D6	4294.12	4576.13	38.30	4429.21	94.98	4294.12	4298.50	731.49	4291.58	4294.52	**-0.06**	577.05	378.35
p04-150	1024.59	1045.22	527.10	1074.11	1361.16	1024.59	1026.60	251.66	1023.23	1024.32	**-0.13**	228.02	127.47
p04-150D1	866.31	871.26	743.30	891.98	3461.44	866.31	866.31	119.63	866.31	866.31	0.00	204.75	17.56
p04-150D2	1861.63	1937.20	326.60	1978.01	878.55	1861.63	1866.48	1055.54	1862.22	1869.21	0.03	294.79	187.48
p04-150D3	2528.51	2649.97	21.30	2671.62	625.83	2528.51	2531.79	1514.55	2525.51	2530.36	**-0.12**	399.21	269.29
p04-150D4	3988.06	4062.88	50.40	4165.18	671.36	3988.06	3997.49	1986.49	3980.33	3988.04	**-0.19**	886.49	686.40
p04-150D5	3986.49	4185.68	23.00	4165.18	675.39	3986.49	3996.85	2076.38	3980.33	3988.04	**-0.15**	889.04	688.41
p04-150D6	6231.01	6479.46	30.50	6482.11	584.84	6231.01	6233.76	1660.06	6225.41	6238.81	**-0.09**	2093.71	2013.53
p05-199	1289.40	1324.73	588.30	1368.67	3284.64	1289.40	1296.37	1594.46	1287.51	1295.99	**-0.15**	316.82	199.75
p05-199D1	1017.30	1023.14	1874.80	1073.55	15505.22	1017.30	1018.40	438.21	1017.28	1018.42	**0.00**	292.70	153.93
p05-199D2	2307.82	2433.17	32.10	2464.65	1457.16	2307.82	2313.37	2440.32	2306.31	2317.16	**-0.07**	425.89	353.13
p05-199D3	3153.01	3291.96	31.20	3411.38	2173.84	3153.01	3163.89	3895.07	3147.31	3160.63	**-0.18**	806.19	774.46
p05-199D4	4844.58	5074.57	50.70	5184.57	3650.59	4844.58	4855.82	3806.84	4840.46	4849.00	**-0.09**	1410.50	1283.91
p05-199D5	5061.25	5265.01	327.30	5363.65	3026.22	5061.25	5070.77	4570.46	5061.31	5069.56	0.00	1811.51	1669.95
p05-199D6	8045.18	8323.72	215.00	8329.55	2124.66	8045.18	8047.68	4718.09	8022.22	8030.53	**-0.29**	3920.84	3863.79
p06-120	1037.88	1042.12	270.30	1201.83	3414.41	1037.88	1043.41	90.06	1037.88	1037.88	0.00	171.02	46.75
p06-120D1	975.96	976.57	370.90	1087.80	3952.67	975.96	976.42	46.16	975.96	975.96	0.00	173.93	26.43
p06-120D2	2703.75	2742.60	380.80	2806.92	558.56	2703.75	2708.51	762.81	2702.50	2705.77	**-0.05**	277.22	165.90
p06-120D3	3907.27	3979.67	329.00	4026.53	358.56	3907.27	3910.03	1543.98	3906.96	3911.81	**0.01**	444.97	347.11
p06-120D4	6201.66	6357.33	20.60	6364.87	458.91	6201.66	6215.87	1975.24	6194.24	6197.24	**-0.12**	936.03	763.29
p06-120D5	6372.58	6481.09	20.50	6545.50	469.17	6372.58	6375.64	2289.79	6328.42	6330.36	**-0.69**	1165.84	931.02
p06-120D6	10001.95	10158.32	20.40	10302.16	636.72	10001.95	10005.18	2209.90	10001.70	10006.23	**0.00**	2832.47	2639.81
p07-100	819.56	819.56	192.40	824.78	126.08	819.56	819.56	27.99	819.56	819.56	0.00	136.50	1.34
p07-100D1	632.63	636.00	166.50	673.54	1207.42	632.63	633.11	14.38	632.63	632.63	0.00	150.36	16.89
p07-100D2	1413.85	1418.81	206.30	1428.27	123.00	1413.85	1413.91	130.70	1413.85	1413.91	0.00	185.34	40.73
p07-100D3	1967.41	1995.34	266.50	2007.11	107.47	1967.41	1967.93	260.65	1967.41	1967.47	0.00	236.64	91.15
p07-100D4	3087.75	3166.31	272.70	3156.31	96.98	3087.75	3088.96	435.09	3088.23	3088.78	0.02	319.06	159.89
p07-100D5	3125.47	3248.76	16.00	3225.63	110.05	3125.47	3126.22	619.62	3125.39	3125.81	**0.00**	369.35	184.86
p07-100D6	4902.81	5065.26	13.80	5028.78	178.19	4902.81	4907.00	823.19	4901.06	4902.75	**-0.04**	647.66	432.65
Average	2591.93	2678.74	201.40	2701.48	1130.15	2591.93	2595.12	925.59	**2588.92**	**2592.27**	-0.08	539.39	410.71
Best#	-	0	-	0	-	3	10	-	**26**	**32**	-	-	-
p-value	2.23E-06	1.74E-09	-	1.18E-09	-	2.23E-06	2.75E-04	-	-	-	-	-	-

Table 6.21 – Results for the SDVRP-LF on the instances of Set III.

Instances	LB	BKS	RGTS			SplitILS			SplitMA				
			Best	*Avg.*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
p01-50	-	524.61	529.23	535.39	13.00	524.61	524.61	1.83	524.61	524.61	0.00	72.67	0.60
p01-50D1	459.50	459.50	466.86	473.32	18.00	459.50	459.50	1.21	459.50	459.50	0.00	59.96	0.29
p01-50D2	756.71	756.71	784.60	789.83	3.00	756.71	760.52	14.55	756.71	758.02	0.00	95.68	44.36
p01-50D3	996.93	1005.75	1025.04	1036.50	1.00	1005.75	1005.93	21.48	1005.75	1005.75	0.00	113.82	19.37
p01-50D4	1485.00	1488.27	1503.33	1538.25	1.00	1488.27	1489.05	52.71	1487.18	1487.62	-0.07	184.82	61.84
p01-50D5	1474.10	1481.71	1503.21	1513.15	8.00	1481.71	1484.62	42.90	1481.71	1481.89	0.00	157.79	70.90
p01-50D6	2149.05	2156.14	2195.67	2202.50	4.00	2156.14	2160.60	84.35	2155.80	2155.80	**-0.02**	245.13	49.59
p02-75	-	823.89	864.64	879.35	10.00	823.89	824.39	30.31	823.89	823.89	0.00	118.45	19.08
p02-75D1	616.58	617.85	629.08	637.00	21.00	617.85	620.19	5.72	617.85	617.85	0.00	103.20	4.01
p02-75D2	1093.56	1110.43	1146.21	1161.75	8.00	1110.43	1112.70	54.11	1109.24	1110.48	**-0.11**	146.02	75.07
p02-75D3	1483.17	1502.05	1550.35	1584.59	12.00	1502.05	1503.42	110.02	1502.05	1503.52	0.00	161.38	59.92
p02-75D4	2270.44	2301.61	2398.40	2412.78	14.00	2301.61	2304.89	283.77	2302.12	2306.06	0.02	321.12	174.07
p02-75D5	2192.25	2219.52	2240.04	2251.50	13.00	2219.52	2222.58	261.25	2219.11	2220.02	**-0.02**	238.88	112.85
p02-75D6	3192.10	3223.06	3259.36	-	6.00	3223.06	3226.79	377.02	3217.51	3221.30	**-0.17**	416.63	282.12
p03-100	-	826.14	845.98	858.20	34.00	826.14	826.45	42.16	826.14	826.70	0.00	147.76	28.85
p03-100D1	753.12	760.00	804.86	834.16	130.00	760.00	760.70	22.00	760.00	760.00	0.00	151.50	50.93
p03-100D2	1435.23	1458.46	1491.82	1497.82	32.00	1458.46	1462.37	200.43	1458.46	1460.90	0.00	202.81	128.34
p03-100D3	1971.43	1997.76	2062.53	2019.50	51.00	1997.76	2001.83	366.31	1996.76	2002.79	**-0.05**	243.98	113.71
p03-100D4	3043.27	3090.65	3171.59	3182.40	54.00	3090.65	3094.91	746.44	3085.69	3088.52	**-0.16**	381.22	260.34
p03-100D5	2945.76	2991.22	3091.25	3111.23	54.00	2991.22	2991.89	756.18	2986.27	2991.17	**-0.17**	364.90	237.24
p03-100D6	4316.42	4387.32	4465.03	4474.00	75.00	4387.32	4389.19	719.27	4378.33	4384.70	**-0.20**	656.07	438.25
p04-150	-	1023.87	1059.71	1069.89	457.00	1023.87	1026.48	243.93	1023.23	1024.32	**-0.06**	228.72	128.20
p04-150D1	896.03	921.47	979.72	998.25	424.00	921.47	923.74	164.58	921.20	921.79	**-0.03**	224.56	116.43
p04-150D2	1986.79	2017.00	2093.21	2102.50	159.00	2017.00	2021.78	1156.99	2016.93	2025.54	**0.00**	313.34	180.33
p04-150D3	2811.64	2849.66	2943.54	2979.02	184.00	2849.66	2856.41	1699.36	2849.59	2853.04	**0.00**	431.38	315.14
p04-150D4	4474.18	4543.18	4652.10	4610.04	255.00	4543.18	4550.63	2467.51	4533.82	4547.78	**-0.21**	1094.75	992.48
p04-150D5	4269.77	4336.80	4460.22	4508.16	252.00	4336.80	4342.45	2366.91	4332.75	4341.43	**-0.09**	918.83	796.82
p04-150D6	6287.09	6396.68	6511.46	6511.46	200.00	6396.68	6402.63	2180.59	6378.28	6393.31	**-0.29**	2175.27	1853.53
p05-199	-	1285.79	1368.81	1401.30	698.00	1285.79	1292.79	1672.72	1287.51	1295.99	0.13	315.59	198.30
p05-199D1	1042.37	1074.18	1158.06	1151.59	989.00	1074.18	1080.65	629.08	1073.57	1081.15	-0.06	294.34	165.21
p05-199D2	2423.64	2481.44	2570.97	2570.97	324.00	2481.44	2487.28	2846.16	2478.37	2488.61	**-0.12**	497.44	402.65
p05-199D3	3420.17	3472.79	3592.77	3578.04	225.00	3472.79	3481.37	3015.92	3469.90	3479.66	**-0.08**	621.69	548.97
p05-199D4	5425.69	5526.28	5798.39	5798.39	220.00	5526.28	5530.56	5799.52	5521.61	5531.00	**-0.08**	3511.05	3433.92
p05-199D5	5306.11	5404.44	5556.01	5556.01	198.00	5404.44	5415.31	5706.50	5398.15	5414.40	-0.12	3460.31	3427.53
p05-199D6	8062.24	8188.47	8319.35	8319.35	241.00	8188.47	8195.06	3528.41	8181.44	8197.54	**-0.09**	4548.54	4470.13
p11-120	-	1037.88	1043.89	1080.30	1231.00	1037.88	1038.68	85.14	1037.88	1037.88	0.00	172.08	46.63
p11-120D1	1023.37	1043.19	1099.30	1120.10	1176.00	1043.19	1043.21	93.35	1042.80	1042.94	**-0.04**	169.38	76.68
p11-120D2	2879.63	2899.91	2939.41	2952.60	99.00	2899.91	2905.28	898.52	2898.25	2902.33	**-0.06**	318.13	226.02
p11-120D3	4162.99	4219.01	4301.53	4308.53	176.00	4219.01	4220.59	2260.39	4215.98	4218.70	**-0.07**	474.16	322.23
p11-120D4	6808.07	6856.11	6967.53	6967.53	301.00	6856.11	6863.96	3363.54	6849.73	6858.08	**-0.09**	1374.52	1220.46
p11-120D5	6584.11	6674.97	6770.14	6770.14	148.00	6674.97	6678.58	2306.51	6639.95	6645.59	**-0.52**	1072.92	800.01
p11-120D6	10111.11	10132.50	10132.50	10133.20	42.00	10215.90	10218.78	2006.24	10192.00	10196.90	0.59	2644.68	2223.42
Average	-	2799.24	2865.42	2865.38	203.83	2801.23	2804.84	1159.19	**2797.56**	**2802.12**	-	701.08	575.64
Best#	-	-	1	1	-	3	10	-	**25**	**29**	-	-	-
p-value	-	6.37E-05	7.86E-08	9.69E-08	-	5.01E-06	3.23E-04	-	-	-	-	-	-

Table 6.22 – Results for the SDVRP-LF on the instances of Set IV.

Instances	LB	BKS	HGA		RGTS			SplitILS			SplitMA				
			Best	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
SD1	228.28	228.28	228.28	0.27	228.28	228.28	0.00	228.28	228.28	0.05	228.28	228.28	0.00	10.98	0.03
SD2	708.28	708.28	708.28	1.95	708.28	708.28	0.00	708.28	708.28	0.58	708.28	708.28	0.00	53.74	0.06
SD3	430.40	430.40	430.58	1.94	430.58	430.58	0.00	430.58	430.58	0.59	430.58	430.58	0.04	41.80	0.06
SD4	631.05	631.05	631.05	6.24	633.98	633.98	0.00	631.05	631.05	2.16	631.05	631.05	0.00	84.20	0.22
SD5	1390.57	1390.57	1390.57	14.20	1401.28	1401.72	3.00	1390.57	1390.57	5.90	1390.57	1390.57	0.00	168.71	0.44
SD6	831.24	831.24	833.58	14.97	846.16	861.12	2.00	831.24	831.24	5.62	831.24	831.24	0.00	121.49	0.48
SD7	3640.00	3640.00	3640.00	28.61	3640.00	3640.00	3.00	3640.00	3640.00	13.74	3640.00	3640.00	0.00	237.60	0.26
SD8	5068.28	5068.28	5068.28	48.26	5068.28	5068.28	2.00	5068.28	5068.28	24.07	5068.28	5068.28	0.00	221.65	2.12
SD9	2044.19	2044.20	2054.84	48.91	2044.73	2058.03	1.00	2044.20	2044.43	35.86	2044.20	2044.20	0.00	215.89	2.73
SD10	2684.88	2684.88	2746.54	114.16	2701.55	2709.12	6.00	2684.88	2684.88	81.76	2684.88	2684.88	0.00	298.34	4.87
SD11	13275.00	13280.00	13280.00	231.64	13280.00	13280.00	15.00	13280.00	13280.00	136.43	13280.00	13280.00	0.00	455.49	5.45
SD12	7175.80	7213.61	7279.97	227.11	7213.62	7213.62	19.00	7213.61	7216.34	179.19	7213.61	7213.61	0.00	436.65	22.08
SD13	10053.60	10110.57	10110.57	421.95	10129.52	10129.52	61.00	10110.58	10110.58	168.07	10110.60	10110.60	0.00	526.65	12.42
SD14	10588.20	10715.53	10786.52	718.65	10783.00	10783.00	41.00	10715.53	10722.73	432.26	10715.50	10716.32	**0.00**	666.98	415.02
SD15	14908.50	15093.85	15160.04	1278.35	15151.06	15158.30	110.00	15093.85	15102.85	658.54	15089.60	15091.21	**-0.03**	939.34	559.89
SD16	3379.33	3379.33	3433.83	1225.88	3481.21	3481.21	54.00	3395.11	3395.16	580.27	3381.25	3381.26	0.06	1163.46	500.74
SD17	26317.20	26493.56	26559.92	1722.20	26512.51	26512.51	130.00	26493.56	26499.23	484.43	26493.60	26493.60	0.00	1090.71	247.79
SD18	14029.20	14197.80	14302.22	1735.83	14293.49	14293.49	61.00	14197.80	14202.85	676.77	14194.70	14203.31	**-0.02**	865.62	550.21
SD19	19707.20	19989.95	20152.53	3093.17	20131.94	20154.32	310.00	19989.95	20000.54	1261.95	19991.90	20003.86	0.01	1160.30	813.57
SD20	39252.80	39641.91	39706.51	6208.16	39701.96	39703.32	560.00	39641.91	39648.42	1518.12	39635.50	39638.21	**-0.02**	2500.54	1487.16
SD21	11271.00	11271.00	11461.20	10565.70	11365.16	11369.31	371.00	11344.96	11357.62	4326.99	11281.90	11315.20	0.10	2393.82	2242.46
Average	-	**9002.11**	9045.97	1319.44	9035.55	9038.95	83.29	9006.39	9009.23	504.45	9002.17	9004.98	-	650.19	327.05
Best#	-	-	1	-	0	0	-	2	3	-	**4**	**8**	-	-	-
p-value	-	3.66E-04	3.09E-03	-	5.35E-04	5.35E-04		3.33E-01	1.15E-01	-	-	-	-	-	-

Table 6.23 – Results for the SDVRP-UF on the instances of Set I.

Instances	LB	BKS	TSVBA		FBTS		SplitILS			SplitMA				
			Best	Time	*Best*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
eil22	375.28	375.28	375.28	2.58	375.30	4.00	375.28	375.28	0.15	375.28	375.28	0.00	0.13	0.02
eil23	568.56	568.56	569.75	1.59	568.60	4.00	568.56	568.56	0.13	568.56	568.56	0.00	0.11	0.04
eil30	505.01	505.01	505.01	7.45	519.00	7.00	505.01	505.01	0.24	505.01	505.01	0.00	0.23	0.23
eil33	837.05	837.06	843.64	8.38	837.10	10.00	837.06	837.06	0.51	837.06	837.06	0.00	44.48	0.37
eil51	524.61	524.61	527.67	49.84	528.00	23.00	524.61	524.61	1.79	524.61	524.61	0.00	0.62	0.62
eilA76	809.58	823.89	853.20	145.78	842.70	191.00	823.89	824.92	30.76	823.89	823.89	0.00	104.54	12.03
eilB76	984.13	1009.04	1034.21	91.36	1017.10	289.00	1009.04	1012.07	51.83	1009.04	1011.22	0.00	118.58	64.89
eilC76	722.76	738.67	761.55	151.13	754.30	73.00	738.67	739.89	16.96	738.67	738.67	0.00	102.90	15.59
eilD76	674.17	687.60	695.96	122.52	701.10	57.00	687.60	689.36	11.16	686.70	687.43	**-0.13**	98.84	20.79
eilA101	804.40	826.14	844.21	295.22	838.80	194.00	826.14	826.58	38.90	826.14	826.70	0.00	123.61	37.67
eilB101	1055.59	1076.26	1112.15	173.13	1096.10	280.00	1076.26	1079.15	110.61	1076.26	1077.47	0.00	146.98	72.87
S51D1	459.50	459.50	468.79	13.56	464.80	13.00	459.50	459.50	1.24	459.50	459.50	0.00	0.33	0.31
S51D2	708.41	708.42	718.69	31.66	711.90	121.00	709.29	709.49	11.20	708.42	708.51	0.00	86.84	29.75
S51D3	941.03	947.97	969.78	18.75	952.80	215.00	948.06	950.12	15.74	947.97	947.97	0.00	94.70	7.83
S51D4	1560.87	1560.88	1628.20	19.77	1587.80	134.00	1562.01	1563.29	56.28	1560.88	1561.05	0.00	194.41	113.89
S51D5	1333.66	1333.67	1362.19	15.39	1348.80	127.00	1333.67	1333.67	36.69	1333.67	1333.85	0.00	136.04	34.41
S51D6	2163.22	2169.10	2236.16	14.38	2202.20	81.00	2169.10	2177.78	62.55	2169.10	2170.32	0.00	278.82	130.03
S76D1	598.93	598.94	613.70	252.28	615.90	33.00	598.94	598.94	4.86	598.94	598.94	0.00	85.69	6.07
S76D2	1066.88	1087.40	1128.15	60.44	1103.60	329.00	1087.40	1089.45	69.36	1087.40	1088.99	0.00	131.38	76.99
S76D3	1406.85	1427.86	1472.92	51.13	1449.80	314.00	1427.86	1429.26	96.50	1426.78	1429.05	**-0.07**	148.34	69.86
S76D4	2053.66	2079.76	2180.13	53.56	2108.60	299.00	2079.76	2081.16	188.38	2079.74	2079.77	**0.00**	201.13	83.08
S101D1	716.92	726.59	749.93	860.31	745.70	223.00	726.59	728.45	15.93	726.59	726.59	0.00	109.53	16.20
S101D2	1356.78	1378.43	1409.03	219.52	1394.60	327.00	1378.43	1386.03	151.66	1377.01	1383.72	**-0.10**	172.17	120.33
S101D3	1845.07	1874.81	1947.62	132.19	1913.30	325.00	1874.81	1880.62	317.29	1874.65	1880.39	**-0.01**	222.64	141.64
S101D5	2758.21	2791.22	2910.71	131.16	2858.80	374.00	2791.22	2795.36	572.13	2789.61	2791.59	**-0.06**	318.60	161.87
Average	-	1084.67	1116.75	116.92	1101.47	161.88	1084.75	1086.62	74.51	**1084.46**	**1085.45**	-	116.87	48.70
Best#	-	-	0	-	0	-	0	1	-	**6**	**16**	-	-	-
p-value	-	5.51E-02	2.07E-05	-	1.23E-05	-	4.82E-03	5.46E-04	-	-	-	-	-	-

Table 6.24 – Results for the SDVRP-UF on the instances of Set I with rounded costs.

Instances	LB	BKS	MAPM		TSVBA		SplitILS			SplitMA				
			Best	Time	*Best*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap(%)*	Time	TMB
eil22	375.00	375	375	4.11	375	2.58	375	375.00	0.15	375	375.00	**0.00**	43.72	0.02
eil23	569.00	569	569	5.47	570	1.59	569	569.00	0.11	569	569.00	0.00	41.61	0.24
eil30	503.00	503	503	5.7	503	7.45	503	503.00	0.23	503	503.00	0.00	51.83	0.08
eil33	835.00	835	835	5.19	844	8.38	835	835.00	0.45	835	835.00	0.00	58.09	0.09
eil51	521.00	521	521	7.28	526	49.84	521	521.00	1.75	521	521.00	0.00	79.40	9.86
eilA76	792.71	818	828	35.94	847	145.78	818	821.75	24.63	818	820.60	0.00	129.57	32.56
eilB76	957.60	1002	1019	13.09	1027	91.36	1002	1007.05	37.68	1002	1005.90	0.00	144.39	67.67
eilC76	714.24	733	738	14.75	754	151.13	733	733.75	14.75	733	733.35	0.00	117.30	34.35
eilD76	667.93	682	682	23.12	691	122.52	682	683.05	10.39	680	682.70	**-0.29**	111.44	51.22
eilA101	792.40	814	818	25.25	834	295.22	814	816.20	32.61	814	816.65	0.00	131.91	50.99
eilB101	1017.77	1061	1082	21.81	1104	173.13	1061	1064.00	78.42	1061	1063.60	0.00	156.53	74.92
S51D1	458.00	458	458	8.77	465	13.56	458	458.00	1.17	458	458.00	0.00	73.38	0.37
S51D2	703.00	703	707	7.44	715	31.66	703	704.75	8.12	703	703.15	0.00	109.07	20.49
S51D3	933.07	943	945	7.84	966	18.75	943	944.05	13.06	942	942.00	**-0.11**	131.09	16.09
S51D4	1547.44	1553	1578	11.98	1621	19.77	1553	1556.50	39.25	1551	1551.00	**-0.13**	538.46	71.39
S51D5	1326.73	1328	1351	16.72	1357	15.39	1328	1329.25	32.07	1328	1328.00	0.00	292.13	32.91
S51D6	2153.00	2153	2182	9.92	2228	14.38	2163	2166.15	52.95	2156	2156.00	0.14	515.77	111.52
S76D1	592.00	592	592	15.23	606	252.28	592	592.30	4.75	592	592.20	0.00	97.51	13.56
S76D2	1040.67	1082	1089	30.5	1124	60.44	1082	1083.15	53.6	1080	1081.30	**-0.18**	192.15	65.90
S76D3	1379.57	1420	1427	12.89	1466	51.13	1420	1423.05	67.81	1418	1420.00	**-0.14**	235.31	97.06
S76D4	2034.70	2073	2117	8.76	2170	53.56	2073	2074.95	144.89	2071	2071.90	**-0.10**	389.04	142.81
S101D1	714.87	716	717	49.75	741	860.31	716	718.35	14.76	716	718.05	0.00	116.78	56.03
S101D2	1301.93	1366	1372	31.72	1398	219.52	1366	1371.40	112.47	1360	1365.65	**-0.44**	202.81	103.46
S101D3	1803.51	1864	1891	33.98	1936	132.19	1864	1868.05	236.05	1858	1862.10	**-0.32**	295.32	183.00
S101D5	2709.48	2770	2854	18.66	2897	131.16	2770	2779.10	439.49	2765	2767.80	**-0.18**	595.48	310.54
Average	-	1077.36	1090.00	17.03	1110.60	116.92	1077.76	1079.91	56.86	**1076.36**	**1077.72**	-	194.00	61.88
Best#		-	0	-	0	-	0	1	-	**9**	**18**	-	-	-
p-value	-	2.54E-02	1.96E-04	-	2.69E-05	-	1.95E-03	3.40E-04	-	-	-	-	-	-

Table 6.25 – Results for the SDVRP-UF on the instances of Set II.

Instances	BKS	MAPM		TSVBA		SplitILS			SplitMA				
		Best	Time	*Best*	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap(%)*	Time	TMB
p01-50	524.61	524.61	8.53	527.67	49.84	524.61	524.61	1.82	524.61	524.61	0.00	65.81	0.52
p01-50D1	460.79	460.79	12.38	466.74	19.69	460.79	460.79	1.17	460.79	460.79	0.00	52.48	0.31
p01-50D2	741.06	751.41	10.22	753.98	23.17	741.06	741.26	9.72	741.06	741.06	0.00	80.89	1.85
p01-50D3	982.79	988.31	12.49	1023.24	17.72	982.79	983.59	18.04	982.77	983.13	**0.00**	99.79	14.50
p01-50D4	1456.00	1467.06	21.42	1530.81	19.11	1456.00	1457.37	42.86	1456.00	1456.00	0.00	154.69	8.24
p01-50D5	1467.47	1477.01	24.53	1505.38	19.09	1467.47	1467.47	49.42	1467.47	1467.47	0.00	137.18	10.08
p01-50D6	2150.97	2154.35	22.91	2219.32	24.41	2150.97	2152.95	51.94	2150.00	2150.15	**-0.05**	235.87	115.36
p02-75	823.89	823.89	35.72	853.20	145.78	823.89	824.80	30.52	823.89	823.89	0.00	104.77	12.02
p02-75D1	596.25	600.06	18.75	614.09	136.14	596.25	596.25	4.98	596.25	596.25	0.00	83.10	4.51
p02-75D2	1064.49	1074.46	34.14	1085.70	97.17	1064.49	1066.36	53.25	1064.49	1065.37	0.00	125.18	58.26
p02-75D3	1393.11	1413.80	37.38	1458.59	67.66	1393.11	1393.11	97.13	1393.11	1393.20	0.00	139.04	18.85
p02-75D4	2081.38	2102.58	46.11	2164.74	61.81	2081.38	2084.91	191.06	2076.92	2082.27	**-0.21**	251.73	135.55
p02-75D5	2111.83	2132.16	51.78	2182.33	55.17	2111.83	2114.03	212.38	2105.15	2112.19	**-0.31**	222.52	133.15
p02-75D6	3178.47	3200.35	27.48	3278.33	86.27	3178.47	3181.28	412.86	3175.61	3177.67	**-0.09**	420.01	234.40
p03-100	826.14	829.44	34.59	844.21	295.22	826.14	826.39	40.01	826.14	826.70	0.00	125.41	37.52
p03-100D1	726.81	726.81	37.12	741.60	1944.09	726.81	730.80	16.67	726.81	726.81	0.00	115.70	29.32
p03-100D2	1376.22	1392.85	78.06	1416.35	160.95	1376.22	1380.23	187.76	1373.85	1380.81	**-0.17**	171.11	81.05
p03-100D3	1823.58	1845.30	28.39	1886.70	145.05	1823.58	1827.81	313.83	1823.29	1826.80	**-0.02**	197.78	86.17
p03-100D4	2749.53	2780.95	84.38	2874.86	125.28	2749.53	2753.99	647.44	2745.64	2749.10	**-0.14**	314.73	210.32
p03-100D5	2813.52	2858.87	100.16	2929.29	134.84	2813.52	2817.05	737.91	2811.62	2815.30	**-0.07**	327.31	196.40
p03-100D6	4294.12	4312.95	55.75	4435.56	185.55	4294.12	4299.40	737.85	4292.05	4296.29	**-0.05**	573.55	416.70
p04-150	1023.66	1042.37	103.69	1079.55	2217.67	1023.66	1026.89	250.37	1023.23	1024.44	**-0.04**	190.31	85.11
p04-150D1	866.31	875.61	100.27	891.10	2640.95	866.31	866.31	120.92	866.31	866.31	0.00	168.47	19.19
p04-150D2	1861.63	1878.71	147.89	1929.91	755.08	1861.63	1866.95	1041.64	1865.12	1869.29	0.19	255.43	176.79
p04-150D3	2527.96	2561.65	224.89	2647.17	470.34	2527.96	2531.50	1445.25	2523.87	2530.27	**-0.16**	341.16	263.57
p04-150D4	3988.64	4045.87	244.91	4151.90	451.95	3988.64	3996.55	1901.05	3979.53	3985.62	**-0.23**	610.69	517.32
p04-150D5	3985.76	4045.87	244.86	4151.90	449.34	3985.76	3995.25	1836.14	3979.53	3985.62	**-0.15**	607.89	515.09
p04-150D6	6232.37	6267.48	401.62	6416.12	678.94	6232.37	6234.56	1543.69	6223.33	6235.82	**-0.14**	1396.79	1306.96
p05-199	1286.92	1311.59	353.84	1339.49	4514.28	1286.92	1293.71	1298.10	1283.27	1293.19	**-0.28**	258.95	174.95
p05-199D1	1017.28	1018.71	356.22	1069.24	11.215.52	1017.28	1018.59	431.97	1017.28	1018.99	0.00	236.00	76.64
p05-199D2	2305.70	2340.14	347.14	2408.16	1544.36	2305.70	2313.04	2296.08	2301.06	2316.42	**-0.20**	371.31	309.70
p05-199D3	3156.02	3191.25	436.20	3296.69	1216.69	3156.02	3163.26	3316.93	3146.79	3156.56	**-0.29**	532.41	438.04
p05-199D4	4843.83	4941.22	725.69	5066.24	108.63	4843.83	4855.49	3739.98	4836.17	4843.71	**-0.16**	1390.29	1251.93
p05-199D5	5063.89	5155.36	749.94	5281.55	119.04	5063.89	5072.74	4222.85	5054.50	5065.53	**-0.18**	1322.26	1218.41
p05-199D6	8037.88	8081.58	571.70	8333.61	153.12	8037.88	8048.57	4616.79	8022.89	8033.28	**-0.19**	3832.93	3738.14
p06-120	1037.88	1041.20	50.92	1051.24	1944.19	1037.88	1039.13	81.50	1037.88	1037.98	0.00	141.58	43.15
p06-120D1	975.96	976.57	72.98	990.59	2736.34	975.96	976.57	44.82	975.96	975.96	0.00	141.15	29.09
p06-120D2	2707.52	2720.38	144.19	2744.74	463.97	2707.52	2710.15	704.44	2702.26	2705.74	**-0.19**	243.22	193.95
p06-120D3	3907.27	3934.39	163.14	4010.80	340.53	3907.27	3909.28	1487.97	3909.11	3910.94	0.05	411.09	331.21
p06-120D4	6195.37	6318.37	196.14	6308.76	418.98	6195.37	6219.01	1805.91	6194.55	6197.79	**-0.01**	894.16	787.48
p06-120D5	6373.24	6424.71	271.39	6511.08	436.80	6373.24	6376.25	2303.70	6329.30	6331.17	**-0.68**	1103.03	948.83
p06-120D6	10003.99	10063.47	298.08	10186.06	30.32	10003.99	10005.29	2161.59	10003.80	10006.26	**0.00**	2985.43	2721.87
p07-100	819.56	819.56	42.89	819.60	75.33	819.56	819.56	27.67	819.56	819.56	0.00	121.23	1.19
p07-100D1	632.63	649.73	34.97	658.99	461.75	632.63	636.76	12.28	632.63	633.90	0.00	115.49	28.64
p07-100D2	1413.85	1417.28	43.27	1441.48	98.31	1413.85	1413.99	128.52	1413.85	1413.85	0.00	163.04	43.11
p07-100D3	1967.41	1994.59	51.31	2010.00	84.50	1967.41	1968.09	265.71	1967.41	1968.08	0.00	213.56	85.31
p07-100D4	3088.47	3113.72	52.13	3157.48	97.58	3088.47	3089.41	416.34	3087.93	3088.73	**-0.02**	305.53	178.16
p07-100D5	3125.47	3155.69	91.31	3200.62	96.39	3125.47	3125.98	568.97	3125.29	3125.53	**-0.01**	339.51	157.26
p07-100D6	4903.00	4919.48	180.11	4996.88	152.92	4903.00	4906.56	799.40	4901.06	4903.27	**-0.04**	614.40	429.07
Average	2591.68	2616.83	152.73	2672.32	553.59	2591.68	2595.18	872.02	2588.43	2591.03	-0.08	744.91	611.60
Best#	-	0	-	0	-	2	9	**-**	**28**	**34**	**-**	**-**	-
p-value	1.81E-06	1.74E-09	-	1.11E-09	-	1.81E-06	3.63E-05	-	-	-	-	-	-

Table 6.26 – Results for the SDVRP-UF on the instances of Set III.

Instances	LB	BKS	ABHC	FBTS		SplitILS			SplitMA				
				Best	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
p01-50	-	524.61	524.61	532.00	18.00	524.61	524.61	1.88	524.61	524.61	0.00	65.49	0.61
p01-50D1	459.50	459.50	-	461.00	31.00	459.50	459.50	1.16	459.50	459.50	0.00	54.10	0.24
p01-50D2	754.45	757.15	776.42	759.80	307.00	757.15	761.12	13.24	756.71	758.08	**-0.06**	88.35	38.35
p01-50D3	999.06	1005.75	1012.56	1026.50	210.00	1005.75	1005.75	20.70	1005.75	1005.75	0.00	105.14	23.10
p01-50D4	1487.16	1487.18	1489.64	1552.10	134.00	1488.58	1488.89	43.04	1487.18	1488.01	0.00	152.96	49.01
p01-50D5	1474.34	1481.71	1488.28	1498.10	151.00	1481.71	1483.36	44.11	1481.71	1482.30	0.00	148.07	82.21
p01-50D6	2149.42	2155.80	2174.54	2191.41	107.60	2156.14	2161.31	78.49	2155.80	2155.80	0.00	229.40	71.22
p02-75	-	823.89	829.89	827.40	320.00	823.89	825.06	29.69	823.89	823.89	0.00	104.76	12.03
p02-75D1	612.45	617.85	-	637.00	44.00	617.85	619.59	5.70	617.85	617.85	0.00	86.26	3.36
p02-75D2	1095.65	1109.62	1123.97	1118.10	325.00	1109.62	1112.11	53.26	1109.24	1110.45	**-0.03**	130.05	45.96
p02-75D3	1482.50	1502.05	1508.73	1525.70	318.00	1502.05	1503.57	108.37	1502.05	1502.91	0.00	145.45	70.57
p02-75D4	2272.05	2298.58	2340.09	2358.80	322.00	2298.58	2301.85	207.22	2296.98	2298.01	**-0.07**	266.02	177.53
p02-75D5	2195.44	2219.97	2243.93	2280.30	406.00	2219.97	2224.06	265.00	2217.63	2219.51	**-0.11**	221.39	134.90
p02-75D6	3192.55	3223.40	3266.78	3259.00	200.00	3223.40	3226.20	378.95	3216.67	3219.88	**-0.21**	410.77	268.82
p03-100	-	826.14	826.14	847.40	188.00	826.14	826.45	40.52	826.14	826.70	0.00	124.75	37.31
p03-100D1	749.42	760.00	-	792.20	87.00	760.00	760.46	21.65	760.00	760.02	0.00	124.23	35.02
p03-100D2	1437.78	1458.46	1478.59	1476.90	326.00	1458.46	1462.68	179.69	1458.46	1461.61	0.00	170.89	69.89
p03-100D3	1971.34	1996.76	2035.91	2023.20	318.00	1996.76	2002.23	362.02	1996.76	2002.33	0.00	223.39	112.41
p03-100D4	3042.93	3085.69	3145.33	3181.30	339.00	3085.69	3093.64	736.52	3085.69	3089.11	0.00	366.24	274.55
p03-100D5	2945.42	2989.30	3014.08	3044.10	325.00	2989.30	2992.75	742.88	2990.34	2991.23	0.03	355.86	162.60
p03-100D6	4334.44	4387.32	4447.47	4441.70	300.00	4387.32	4389.43	675.01	4378.33	4384.69	**-0.20**	659.74	364.76
p04-150	-	1023.87	1028.42	1081.60	426.00	1023.87	1027.28	233.73	1023.23	1024.44	**-0.06**	188.78	84.70
p04-150D1	895.46	921.91	-	953.00	369.00	921.91	923.69	161.90	921.20	922.06	**-0.08**	183.29	75.86
p04-150D2	1986.34	2016.97	2055.18	2060.40	375.00	2016.97	2021.36	1109.88	2016.93	2026.05	**0.00**	268.00	166.31
p04-150D3	2811.98	2849.66	2912.08	2910.80	394.00	2849.66	2857.28	1518.82	2849.66	2853.02	0.00	374.35	261.46
p04-150D4	4474.92	4545.46	4638.74	4681.70	389.00	4545.46	4550.85	2410.48	4537.82	4548.81	**-0.17**	884.75	768.45
p04-150D5	4267.33	4334.71	4435.95	4483.40	372.00	4334.71	4341.15	2357.52	4328.77	4339.85	**-0.14**	722.75	662.86
p04-150D6	6284.76	6395.41	6467.17	6459.80	300.00	6395.41	6402.15	1926.20	6380.51	6391.51	**-0.23**	1463.21	1320.52
p05-199	-	1289.89	1302.89	1342.50	477.00	1289.89	1293.24	1355.25	1287.18	1293.19	**-0.21**	258.40	174.50
p05-199D1	1042.37	1074.18	-	1126.60	449.00	1074.18	1080.64	626.12	1074.06	1081.46	**-0.01**	237.69	154.62
p05-199D2	2423.99	2478.40	2540.06	2525.00	418.00	2478.40	2486.54	2661.25	2476.06	2485.54	**-0.09**	389.77	335.06
p05-199D3	3420.23	3471.41	3581.66	3542.50	429.00	3471.41	3480.76	3014.82	3469.18	3477.51	**-0.06**	559.50	477.57
p05-199D4	5422.95	5521.57	5669.26	5700.70	500.00	5521.57	5529.06	4349.61	5515.50	5519.99	**-0.11**	1666.05	1477.85
p05-199D5	5304.09	5409.76	5541.09	5585.10	438.00	5409.76	5417.75	4524.33	5398.71	5409.49	**-0.20**	1386.87	1318.93
p05-199D6	8062.14	8192.03	8297.71	8255.40	300.00	8192.03	8195.67	3258.29	8176.30	8190.21	**-0.19**	3462.49	3422.14
p11-120	-	1037.88	1042.12	1048.30	177.00	1037.88	1043.38	84.84	1037.88	1037.98	0.00	141.91	43.34
p11-120D1	1023.39	1043.19	-	1119.20	344.00	1043.19	1043.22	109.76	1042.80	1042.88	**-0.04**	139.01	66.96
p11-120D2	2867.79	2898.50	2913.09	2953.10	344.00	2898.50	2907.07	895.11	2898.25	2900.15	**-0.01**	294.04	199.85
p11-120D3	4156.68	4219.01	4270.38	4298.40	345.00	4219.01	4220.79	1957.37	4216.10	4219.56	**-0.07**	447.07	355.21
p11-120D4	6780.19	6854.09	6890.39	7206.20	358.00	6854.09	6865.23	3442.31	6850.78	6856.60	**-0.05**	1412.36	1171.14
p11-120D5	6593.28	6658.52	6671.04	6858.10	354.00	6673.95	6678.11	2354.02	6639.96	6645.18	**-0.28**	1019.33	780.82
p11-120D6	10113.55	10204.81	10233.37	10285.70	300.00	10204.81	10216.80	2279.57	10193.20	10197.45	**-0.11**	2718.38	2305.72
Average	-	2800.28	-	2864.56	300.82	2800.69	2804.92	1062.86	**2797.27**	**2801.08**	-	534.55	420.44
Best#	-	-	0	0	-	1	4	**-**	**25**	**35**	**-**	**-**	-
p-value	-	9.46E-06	-	1.65E-08	-	4.20E-06	8.68E-07	-	-	-	-	-	-

Table 6.27 – Results for the SDVRP-UF on the instances of Set IV.

Instances	LB	BKS	TSVBA		SplitILS			SplitMA				
			Best	Time	*Best*	*Avg.*	Time	*Best*	*Avg.*	*Gap*(%)	Time	TMB
SD1	228.28	228.28	228.28	0.00	228.28	228.28	0.05	228.28	228.28	0.00	7.34	0.41
SD2	708.28	708.28	708.28	0.02	708.28	708.28	0.63	708.28	708.28	0.00	37.30	0.06
SD3	430.58	430.58	430.58	0.03	430.58	430.58	0.62	430.58	430.58	0.00	32.63	0.06
SD4	631.05	631.05	631.05	0.08	631.05	631.05	2.26	631.05	631.05	0.00	73.23	0.22
SD5	1390.57	1390.57	1390.57	0.13	1390.57	1390.57	6.07	1390.57	1390.57	0.00	149.36	0.64
SD6	831.24	831.24	831.24	0.14	831.24	831.24	5.81	831.24	831.24	0.00	118.25	0.54
SD7	3639.97	3640.00	3640.00	0.09	3640.00	3640.00	14.12	3640.00	3640.00	0.00	215.58	0.25
SD8	5068.28	5068.28	5068.28	0.14	5068.28	5068.28	24.93	5068.28	5068.28	0.00	208.20	2.62
SD9	2044.18	2044.20	2071.03	0.36	2044.20	2044.20	38.78	2044.20	2044.20	0.00	204.59	2.75
SD10	2684.86	2684.88	2747.83	0.89	2684.88	2684.88	101.10	2684.88	2684.88	0.00	279.78	5.62
SD11	13280.00	13280.00	13280.00	0.41	13280.00	13280.00	152.42	13280.00	13280.00	0.00	445.85	5.24
SD12	7135.27	7213.61	7213.62	0.84	7213.61	7216.60	210.71	7213.61	7213.61	0.00	431.13	50.94
SD13	9992.74	10110.58	10110.58	1.20	10110.58	10110.58	189.45	10110.60	10110.60	0.00	507.67	10.88
SD14	10502.76	10717.53	10802.87	2.31	10717.53	10723.79	479.85	10715.50	10716.60	**-0.02**	602.92	310.92
SD15	14787.05	15094.48	15153.45	3.20	15094.48	15105.90	731.98	15089.60	15091.78	**-0.03**	863.71	500.17
SD16	3379.33	3379.33	3446.43	7.59	3381.26	3394.48	930.72	3381.25	3381.25	0.06	1100.32	458.89
SD17	26166.80	26493.56	26493.56	7.27	26496.06	26499.32	577.29	26493.60	26493.96	0.00	934.31	350.03
SD18	13892.74	14202.53	14323.04	27.95	14202.53	14205.07	834.60	14194.70	14203.06	**-0.06**	803.83	560.56
SD19	19584.84	19995.69	20157.10	11.95	19995.69	20007.52	1524.67	19991.30	20004.14	**-0.02**	1058.17	856.28
SD20	38901.37	39635.51	39722.86	11.02	39635.51	39647.61	1563.38	39635.50	39637.02	**0.00**	1601.22	1057.46
SD21	11254.83	11271.06	11458.76	111.56	11345.68	11365.37	5034.56	11294.50	11307.54	0.21	2156.59	2087.82
Average	-	9002.44	9043.31	8.91	9006.20	9010.17	591.62	**9002.74**	**9004.62**	-	563.43	298.21
Best#	-	-	0	-	0	1	-	**5**	**9**	**-**	**-**	-
p-value	-	8.53E-01	1.62E-02	-	3.47E-02	1.14E-02	-	-	-	-	-	-

List of Publications

Published/accepted papers

— Pengfei HE, Jin-Kao HAO. Iterated two-phase local search for the colored traveling salesmen problem. *Engineering Applications of Artificial Intelligence* 97 (2021): 104018.

— Pengfei HE, Jin-Kao HAO, Qinghua WU. Grouping memetic search for the colored traveling salesmen problem. *Information Sciences* 570 (2021): 689-707.

— Pengfei HE, Jin-Kao HAO. Hybrid search with neighborhood reduction for the multiple traveling salesman problem. *Computers & Operations Research* 142 (2022): 105726.

Submitted papers

— Pengfei HE, Jin-Kao HAO. Memetic search for the minmax multiple traveling salesman problem with single and multiple depots. *European Journal of Operational Research*, Under major revision, June, 2022.

— Pengfei HE, Jin-Kao HAO. General edge assembly crossover driven memetic search for split delivery vehicle routing. *Transportation Science*, Under major revision, June, 2022.

— Pengfei HE, Jin-Kao HAO, Qinghua WU. A hybrid genetic algorithm for undirected traveling salesman problems with profits. *Networks*, Under major revision, July, 2022.

— Pengfei HE, Jin-Kao HAO, Qinghua WU. A hybrid genetic algorithm for the Hamiltonian p-median problem. *IEEE Transactions on Evolutionary Computation*, Under review, June, 2022.

Reference

[AV21] Luca Accorsi and Daniele Vigo, « A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems », *in*: *Transportation Science (2021) https://doi.org/10.1287/trsc.2021.1059* (2021) (cit. on pp. 69–71).

[AH10] Rafael E Aleman and Raymond R Hill, « A tabu search with vocabulary building approach for the vehicle routing problem with split demands », *in*: *International Journal of Metaheuristics* 1.*1* (2010), pp. 55–80 (cit. on pp. 31–33, 109, 110).

[AZH09] Rafael E Aleman, Xinhui Zhang, and Raymond R Hill, « A ring-based diversification scheme for routing problems », *in*: *International Journal of Mathematics in Operational Research* 1.*1-2* (2009), pp. 163–190 (cit. on pp. 32, 33).

[AZH10] Rafael E Aleman, Xinhui Zhang, and Raymond R Hill, « An adaptive memory algorithm for the split delivery vehicle routing problem », *in*: *Journal of Heuristics* 16.*3* (2010), pp. 441–473 (cit. on p. 31).

[App+02] David Applegate, William Cook, Sanjeeb Dash, and André Rohe, « Solution of a min-max vehicle routing problem », *in*: *INFORMS Journal on Computing* 14.*2* (2002), pp. 132–143.

[App+06] David L Applegate, Robert E Bixby, Vasek Chvátal, and William J Cook, *The Traveling Salesman Problem: a Computational Study*, Princeton University Press, 2006 (cit. on p. 18).

[ABS14] Claudia Archetti, Nicola Bianchessi, and M Grazia Speranza, « Branch-and-cut algorithms for the split delivery vehicle routing problem », *in*: *European Journal of Operational Research* 238.*3* (2014), pp. 685–698 (cit. on pp. 30, 31, 98, 109–111).

[ABS11] Claudia Archetti, Nicola Bianchessi, and Maria Grazia Speranza, « A column generation approach for the split delivery vehicle routing problem », *in*: *Networks* 58.*4* (2011), pp. 241–254.

[AS04] Claudia Archetti and M Grazia Speranza, « Vehicle routing in the 1-skip collection problem », *in*: *Journal of the Operational Research Society* 55.*7* (2004), pp. 717–727 (cit. on p. 98).

[ASS08] Claudia Archetti, M Grazia Speranza, and Martin WP Savelsbergh, « An optimization-based heuristic for the split delivery vehicle routing problem », *in*: *Transportation Science* 42.*1* (2008), pp. 22–31 (cit. on pp. 31, 33).

[AS12] Claudia Archetti and Maria Grazia Speranza, « Vehicle routing problems with split deliveries », *in*: *International Transactions in Operational Research* 19.*1-2* (2012), pp. 3–22 (cit. on p. 30).

[ASH06] Claudia Archetti, Maria Grazia Speranza, and Alain Hertz, « A tabu search algorithm for the split delivery vehicle routing problem », *in*: *Transportation Science* 40.*1* (2006), pp. 64–73 (cit. on pp. 30–33, 105).

[AGS19] Florian Arnold, Michel Gendreau, and Kenneth Sörensen, « Efficiently solving very large-scale routing problems », *in*: *Computers & Operations Research* 107 (2019), pp. 32–42 (cit. on p. 68).

[AS19a] Florian Arnold and Kenneth Sörensen, « Knowledge-guided local search for the vehicle routing problem », *in*: *Computers & Operations Research* 105 (2019), pp. 32–46 (cit. on pp. 43, 45, 46, 68, 70, 71).

[AS19b] Florian Arnold and Kenneth Sörensen, « What makes a VRP solution good? The generation of problem-specific knowledge for heuristics », *in*: *Computers & Operations Research* 106 (2019), pp. 280–288 (cit. on p. 114).

[BM85] E Balas and G Martin, « ROLL-A-ROUND: Software package for scheduling the rounds of a rolling mill », *in*: *Copyright Balas and Martin Associates* 104 (1985) (cit. on pp. 25, 82).

[Bal89] Egon Balas, « The prize collecting traveling salesman problem », *in*: *Networks: An International Journal* 19.*6* (1989), pp. 621–636 (cit. on pp. 25, 82).

[BQ64] Michel L Balinski and Richard E Quandt, « On an integer program for a delivery problem », *in*: *Operations Research* 12.*2* (1964), pp. 300–304 (cit. on p. 16).

[Bea83] John E Beasley, « Route first—cluster second methods for vehicle routing », *in*: *Omega* 11.*4* (1983), pp. 403–408 (cit. on p. 34).

[Bee+18] Onne Beek, Birger Raa, Wout Dullaert, and Daniele Vigo, « An efficient implementation of a static move descriptor-based local search heuristic », *in*: *Computers & Operations Research* 94 (2018), pp. 1–10 (cit. on p. 69).

[Bek06] Tolga Bektas, « The multiple traveling salesman problem: an overview of formulations and solution procedures », *in*: *Omega* 34.*3* (2006), pp. 209–219 (cit. on pp. 18, 22).

[BMM00] José-Manuel Belenguer, MC Martinez, and E Mota, « A lower bound for the split delivery vehicle routing problem », *in*: *Operations Research* 48.*5* (2000), pp. 801–810 (cit. on pp. 30, 32, 98, 111).

[BGN14] Leonardo Berbotto, Sergio Garcıéa, and Francisco J Nogales, « A randomized granular tabu search heuristic for the split delivery vehicle routing problem », *in*: *Annals of Operations Research* 222.*1* (2014), pp. 153–173 (cit. on pp. 31, 32, 110).

[BGP09] Jean-François Bérubé, Michel Gendreau, and Jean-Yves Potvin, « A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem », *in*: *Networks: An International Journal* 54.*1* (2009), pp. 56–67 (cit. on pp. 27, 28, 82, 90, 92, 143, 145, 147).

[Bie+93] Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson, « A note on the prize collecting traveling salesman problem », *in*: *Mathematical Programming* 59.*1* (1993), pp. 413–420 (cit. on p. 25).

[BY20] Andreas Bortfeldt and Junmin Yi, « The split delivery vehicle routing problem with three-dimensional loading constraints », *in*: *European Journal of Operational Research* 282.*2* (2020), pp. 545–558.

[BPR07] Mourad Boudia, Christian Prins, and Mohamed Reghioui, « An effective memetic algorithm with population management for the split delivery vehicle routing problem », *in*: *International Workshop on Hybrid Metaheuristics*, Springer, 2007, pp. 16–30 (cit. on pp. 30–32, 106–108, 110, 114).

[BDM10] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim, « A memetic algorithm for the team orienteering problem », *in*: *4OR* 8.*1* (2010), pp. 49–70 (cit. on pp. 34, 94).

[BRC07] Evelyn C Brown, Cliff T Ragsdale, and Arthur E Carter, « A grouping genetic algorithm for the multiple traveling salesperson problem », *in*: *International Journal of Information Technology & Decision Making* 6.*02* (2007), pp. 333–347 (cit. on pp. 22–24).

[CVH08] Ann Melissa Campbell, Dieter Vandenbussche, and William Hermann, « Routing for relief efforts », *in*: *Transportation Science* 42.*2* (2008), pp. 127–145 (cit. on p. 22).

[CCM08] Vicente Campos, Angel Corberán, and Enrique Mota, « A scatter search algorithm for the split delivery vehicle routing problem », *in*: *Advances in computational intelligence in transport, logistics, and supply chain management*, Springer, 2008, pp. 137–152 (cit. on pp. 31, 32, 110).

[Cam+14] Vicente Campos, Rafael Martıé, Jesús Sánchez-Oro, and Abraham Duarte, « GRASP with path relinking for the orienteering problem », *in*: *Journal of the Operational Research Society* 65.*12* (2014), pp. 1800–1813 (cit. on pp. 26, 27, 86).

[Car+09] John Carlsson, Dongdong Ge, Arjun Subramaniam, Amy Wu, and Yinyu Ye, « Solving min-max multi-depot vehicle routing problem », *in*: *Lectures on Global Optimization* 55 (2009), pp. 31–46 (cit. on pp. 22, 24).

[CR06] Arthur E Carter and Cliff T Ragsdale, « A new approach to solving the multiple traveling salesperson problem using genetic algorithms », *in*: *European Journal of Operational Research* 175.*1* (2006), pp. 246–257 (cit. on pp. 18, 20, 22–24).

[Cat+14] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Thibaut Vidal, « A memetic algorithm for the multi trip vehicle routing problem », *in*: *European Journal of Operational Research* 236.*3* (2014), pp. 833–848 (cit. on pp. 34, 40).

[CGW96] I-Ming Chao, Bruce L Golden, and Edward A Wasil, « A fast and effective heuristic for the orienteering problem », *in*: *European Journal of Operational Research* 88.*3* (1996), pp. 475–489 (cit. on p. 27).

[CL08] Antonio Augusto Chaves and Luiz Antonio Nogueira Lorena, « Hybrid metaheuristic for the prize collecting travelling salesman problem », *in*: *European Conference on Evolutionary Computation in Combinatorial Optimization*, Springer, 2008, pp. 123–134 (cit. on p. 27).

[CK21] Omar Cheikhrouhou and Ines Khoufi, « A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy », *in*: *Computer Science Review* 40 (2021), p. 100369 (cit. on pp. 9, 22).

[Che+17] Ping Chen, Bruce Golden, Xingyin Wang, and Edward Wasil, « A novel approach to solve the split delivery vehicle routing problem », *in*: *International Transactions in Operational Research* 24.*1-2* (2017), pp. 27–41 (cit. on pp. 31, 32, 110).

[CHD10] Ping Chen, Hou-Kuan Huang, and Xing-Ye Dong, « Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem », *in*: *Expert Systems with Applications* 37.*2* (2010), pp. 1620–1627 (cit. on p. 43).

[CGW07] Si Chen, Bruce Golden, and Edward Wasil, « The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results », *in*: *Networks: An International Journal* 49.*4* (2007), pp. 318–329 (cit. on pp. 31, 33).

[CSR21] Glaubos Chémaco, Luidi Simonetti, and Isabel Rosseti, « A Branch-and-Cut and MIP-based heuristics for the Prize-Collecting Travelling Salesman Problem », *in*: *RAIRO: Recherche Opérationnelle* 55 (2021), p. 719 (cit. on p. 27).

[Cro58] Georges A Croes, « A method for solving traveling-salesman problems », *in*: *Operations Research* 6.*6* (1958), pp. 791–812 (cit. on p. 86).

[DFJ54] George Dantzig, Ray Fulkerson, and Selmer Johnson, « Solution of a large-scale traveling-salesman problem », *in*: *Journal of the Operations Research Society of America* 2.*4* (1954), pp. 393–410 (cit. on pp. 16, 18).

[DR59] George B Dantzig and John H Ramser, « The truck dispatching problem », *in*: *Management Science* 6.*1* (1959), pp. 80–91 (cit. on p. 9).

[DLV10] Ulrich Derigs, B Li, and Ulrich Vogel, « Local search-based metaheuristics for the split delivery vehicle routing problem », *in*: *Journal of the Operational Research Society* 61.*9* (2010), pp. 1356–1364 (cit. on pp. 30–32, 110).

[DM02] Elizabeth D Dolan and Jorge J Moré, « Benchmarking optimization software with performance profiles », *in*: *Mathematical Programming* 91.*2* (2002), pp. 201–213 (cit. on pp. 35, 53).

[DDC18] Xueshi Dong, Wenyong Dong, and Yongle Cai, « Ant colony optimisation for coloured travelling salesman problem by multi-task learning », *in*: *IET Intelligent Transport Systems* 12.*8* (2018), pp. 774–782 (cit. on pp. 19, 21, 48, 50, 52).

[Don+19] Xueshi Dong, Qing Lin, Min Xu, and Yongle Cai, « Artificial bee colony algorithm with generating neighbourhood solution for large scale coloured traveling salesman problem », *in*: *IET Intelligent Transport Systems* 13.*10* (2019), pp. 1483–1491 (cit. on pp. 20, 21, 50).

[DT89] Moshe Dror and Pierre Trudeau, « Savings by split delivery routing », *in*: *Transportation Science* 23.*2* (1989), pp. 141–145 (cit. on pp. 9, 30, 31, 106–108).

[DT90] Moshe Dror and Pierre Trudeau, « Split delivery routing », *in*: *Naval Research Logistics* 37.*3* (1990), pp. 383–402 (cit. on pp. 9, 30, 31, 106).

[Fal98] Emanuel Falkenauer, *Genetic Algorithms and Grouping Problems*, USA: John Wiley & Sons, Inc., 1998 (cit. on pp. 21, 40).

[FDG05] Dominique Feillet, Pierre Dejax, and Michel Gendreau, « Traveling salesman problems with profits », *in*: *Transportation Science* 39.*2* (2005), pp. 188–205 (cit. on pp. 9, 25–27, 82, 86).

[FGT98] 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 (cit. on pp. 28, 82, 90).

[FT88] Matteo Fischetti and Paolo Toth, « An additive approach for the optimal solution of the prize collecting traveling salesman problem », *in*: *Vehicle routing: Methods and studies* 231 (1988), pp. 319–343 (cit. on pp. 25, 82).

[FM91] BR Fox and MB McMahon, « Genetic operators for sequencing problems », *in*: *Foundations of genetic algorithms*, vol. 1, Elsevier, 1991, pp. 284–300 (cit. on p. 34).

[Fra+95] Paulo M França, Michel Gendreau, Gilbert Laporte, and Felipe M Müller, « The m-traveling salesman problem with minmax objective », *in*: *Transportation Science* 29.*3* (1995), pp. 267–275 (cit. on pp. 9, 21, 22).

[GBF11] Philippe Galinier, Zied Boujbel, and Michael Coutinho Fernandes, « An efficient memetic algorithm for the graph partitioning problem », *in*: *Annals of Operations Research* 191.*1* (2011), pp. 1–22 (cit. on pp. 21, 46).

[GH99] Philippe Galinier and Jin-Kao Hao, « Hybrid evolutionary algorithms for graph coloring », *in*: *Journal of Combinatorial Optimization* 3.*4* (1999), pp. 379–397 (cit. on p. 46).

[GLS98a] Michel Gendreau, Gilbert Laporte, and Frédéric Semet, « A tabu search heuristic for the undirected selective travelling salesman problem », *in*: *European Journal of Operational Research* 106.*2-3* (1998), pp. 539–545 (cit. on p. 27).

[GLS98b] Michel Gendreau, Gilbert Laporte, and Frederic Semet, « A branch-and-cut algorithm for the undirected selective traveling salesman problem », *in*: *Networks: An International Journal* 32.*4* (1998), pp. 263–273 (cit. on pp. 25, 82).

[GLV87] Bruce L Golden, Larry Levy, and Rakesh Vohra, « The orienteering problem », *in*: *Naval Research Logistics* 34.*3* (1987), pp. 307–318 (cit. on pp. 25, 27).

[GDM00] L Gomes, V Diniz, and Carlos A Martinhon, « An Hybrid GRASP+ VND Metaheuristic for the Prize-Collecting Traveling Salesman Problem », *in*: *XXXII Simpósio Brasileiro de Pesquisa Operacional* (2000), pp. 1657–1665 (cit. on p. 27).

[GGW10] Chris Groër, Bruce Golden, and Edward Wasil, « A library of local search heuristics for the vehicle routing problem », *in*: *Mathematical Programming Computation* 2.*2* (2010), pp. 79–101 (cit. on p. 32).

[GLV16] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen, « Orienteering problem: A survey of recent variants, solution approaches and applications », *in*: *European Journal of Operational Research* 255.*2* (2016), pp. 315–332 (cit. on pp. 25, 82).

[HC16] Anthony Fu-Wha Han and Yu-Ching Chu, « A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts », *in*: *Transportation Research Part E: Logistics and Transportation Review* 88 (2016), pp. 11–31 (cit. on p. 31).

[Hao12] Jin-Kao Hao, « Memetic algorithms in discrete optimization », *in*: *Handbook of memetic algorithms*, Springer, 2012, pp. 73–94 (cit. on pp. 46, 62, 71, 82, 100).

[HH21] Pengfei He and Jin-Kao Hao, « Iterated two-phase local search for the colored traveling salesmen problem », *in*: *Engineering Applications of Artificial Intelligence* 97 (2021), p. 104018 (cit. on pp. 11, 20, 41, 48, 50, 52–54).

[HH22] Pengfei He and Jin-Kao Hao, « Hybrid search with neighborhood reduction for the multiple traveling salesman problem », *in*: *Computers & Operations Research* (2022), p. 105726 (cit. on pp. 11, 21, 23, 24, 68, 73).

[HHW21] Pengfei He, Jin-Kao Hao, and Qinghua Wu, « Grouping memetic search for the colored traveling salesmen problem », *in*: *Information Sciences* 570 (2021), pp. 689–707 (cit. on pp. 10, 20, 24, 25, 72, 74, 129, 131, 132).

[He+20] Pengfei He, Jing Li, Hailong Qin, Zairui He, and Ruiyin He, « Fields distinguished by edges and middles visited by heterogeneous vehicles to minimize non-working distances », *in*: *Computers and Electronics in Agriculture* 170 (2020), p. 105273 (cit. on pp. 18, 40).

[He+18] Pengfei He, Jing Li, Dongqing Zhang, and Shan Wan, « Optimisation of the harvesting time of rice in moist and non-moist dispersed fields », *in*: *Biosystems Engineering* 170 (2018), pp. 12–23 (cit. on pp. 18, 40).

[Hel00] Keld Helsgaun, « An effective implementation of the Lin–Kernighan traveling salesman heuristic », *in*: *European Journal of Operational Research* 126.*1* (2000), pp. 106–130 (cit. on pp. 24, 33, 34, 46, 56, 68).

[HP77] Saman Hong and Manfred W Padberg, « A note on the symmetric multiple traveling salesman problem with fixed charges », *in*: *Operations Research* 25.*5* (1977), pp. 871–874 (cit. on p. 21).

[ISW09] Arif Imran, Said Salhi, and Niaz A Wassan, « A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem », *in*: *European Journal of Operational Research* 197.*2* (2009), pp. 509–518 (cit. on p. 67).

[JLE08] Mingzhou Jin, Kai Liu, and Burak Eksioglu, « A column generation approach for the split delivery vehicle routing problem », *in*: *Operations Research Letters* 36.*2* (2008), pp. 265–270 (cit. on p. 31).

[JM02] Soonchul Jung and Byung Ro Moon, « A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows. », *in*: *GECCO*, 2002, pp. 1309–1316 (cit. on p. 34).

[Kar+21] Korhan Karabulut, Hande Öztop, Levent Kandiller, and M Fatih Tasgetiren, « Modeling and optimization of multiple traveling salesmen problems: An evolution strategy approach », *in*: *Computers & Operations Research* 129 (2021), p. 105192 (cit. on pp. 23, 24, 68, 72–74, 129, 131).

[KAO15] Ali Husseinzadeh Kashan, Ali Akbar Akbari, and Bakhtiar Ostadi, « Grouping evolution strategies: An effective approach for grouping problems », *in*: *Applied Mathematical Modelling* 39.*9* (2015), pp. 2703–2720 (cit. on pp. 20, 40).

[KZ16] Morteza Keshtkaran and Koorush Ziarati, « A novel GRASP solution approach for the Orienteering Problem », *in*: *Journal of Heuristics* 22.*5* (2016), pp. 699–726 (cit. on pp. 26, 27).

[KML18] Gorka Kobeaga, Marıéa Merino, and Jose A Lozano, « An efficient evolutionary algorithm for the orienteering problem », *in*: *Computers & Operations Research* 90 (2018), pp. 42–59 (cit. on pp. 26–28, 34, 87, 89, 90, 92, 135–142).

[KML20] Gorka Kobeaga, Mariéa Merino, and Jose A Lozano, « A revisited branch-and-cut algorithm for large-scale orienteering problems », *in*: *arXiv preprint arXiv:2011.02743* (2020) (cit. on pp. 27, 82, 89, 92, 135–140).

[KFT18] László T Kóczy, Péter Földesi, and Boldizsár Tüű-Szabó, « Enhanced discrete bacterial memetic evolutionary algorithm-an efficacious metaheuristic for the traveling salesman optimization », *in*: *Information Sciences* 460 (2018), pp. 389–400 (cit. on p. 40).

[Kra+95] Slavko Krajcar, Davor Skrlec, Branko Pribicevic, and Snjezana Blagajac, « GA approach to solving multiple vehicle routing problem », *in*: *Portuguese Conference on Artificial Intelligence*, Springer, 1995, pp. 473–481 (cit. on p. 34).

[LM90] Gilbert Laporte and Silvano Martello, « The selective travelling salesman problem », *in*: *Discrete Applied Mathematics* 26.*2-3* (1990), pp. 193–207 (cit. on pp. 25, 82).

[LR94] Adrienne C Leifer and Moshe B Rosenwein, « Strong linear programming relaxations for the orienteering problem », *in*: *European Journal of Operational Research* 73.*3* (1994), pp. 517–523 (cit. on p. 82).

[Li+14] Jun Li, MengChu Zhou, Qirui Sun, Xianzhong Dai, and Xiaolong Yu, « Colored traveling salesman problem », *in*: *IEEE Transactions on Cybernetics* 45.*11* (2014), pp. 2390–2401 (cit. on pp. 9, 18, 19, 21, 40, 41, 48, 50).

[LS06] Yun-Chia Liang and Alice E Smith, « An ant colony approach to the orienteering problem », *in*: *Journal of the Chinese Institute of Industrial Engineers* 23.*5* (2006), pp. 403–414 (cit. on p. 27).

[Lin65] Shen Lin, « Computer solutions of the traveling salesman problem », *in*: *Bell System Technical Journal* 44.*10* (1965), pp. 2245–2269 (cit. on p. 16).

[LK73] Shen Lin and Brian W Kernighan, « An effective heuristic algorithm for the traveling-salesman problem », *in*: *Operations Research* 21.*2* (1973), pp. 498–516 (cit. on pp. 16, 68).

[Lóp+16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle, « The irace package: Iterated racing for automatic algorithm configuration », *in*: *Operations Research Perspectives* 3 (2016), pp. 43–58 (cit. on pp. 49, 72, 89, 109).

[LY19] Li-Chih Lu and Tai-Wen Yue, « Mission-oriented ant-team ACO for min–max MTSP », *in*: *Applied Soft Computing* 76 (2019), pp. 436–444 (cit. on p. 23).

[LBW21] Yongliang Lu, Una Benlic, and Qinghua Wu, « A highly effective hybrid evolutionary algorithm for the covering salesman problem », *in*: *Information Sciences* 564 (2021), pp. 144–162 (cit. on p. 68).

[LHW19] Yongliang Lu, Jin-Kao Hao, and Qinghua Wu, « Hybrid evolutionary search for the traveling repairman problem with profits », *in*: *Information Sciences* 502 (2019), pp. 91–108 (cit. on p. 40).

[Mar+15] Yannis Marinakis, Michael Politis, Magdalene Marinaki, and Nikolaos Matsatsinis, « A memetic-grasp algorithm for the solution of the orienteering problem », *in*: *Modelling, computation and optimization in information systems and management sciences*, Springer, 2015, pp. 105–116 (cit. on pp. 26, 27).

[Men+17] Xianghu Meng, Jun Li, Xianzhong Dai, and Jianping Dou, « Variable neighborhood search for a colored traveling salesman problem », *in*: *IEEE Transactions on Intelligent Transportation Systems* 19.*4* (2017), pp. 1018–1026 (cit. on pp. 9, 19, 50).

[Mil78] P Miliotis, « Using cutting planes to solve the symmetric travelling salesman problem », *in*: *Mathematical Programming* 15.*1* (1978), pp. 177–188 (cit. on p. 16).

[MTZ60] Clair E Miller, Albert W Tucker, and Richard A Zemlin, « Integer programming formulation of traveling salesman problems », *in*: *Journal of the ACM (JACM)* 7.*4* (1960), pp. 326–329 (cit. on pp. 16, 18).

[MH97] Nenad Mladenović and Pierre Hansen, « Variable neighborhood search », *in*: *Computers & Operations Research* 24.*11* (1997), pp. 1097–1100 (cit. on p. 67).

[ML55] G Morton and AH Land, « A Contribution to the "Travelling-Salesman" Problem », *in*: *Journal of the Royal Statistical Society: Series B (Methodological)* 17.*2* (1955), pp. 185–194 (cit. on p. 16).

[MCC07] Enrique Mota, Vicente Campos, and Ángel Corberán, « A new metaheuristic for the vehicle routing problem with split demands », *in*: *European Conference on Evolutionary Computation in Combinatorial Optimization*, Springer, 2007, pp. 121–129 (cit. on p. 31).

[Müh90] Heinz Mühlenbein, « Evolution in Time and Space - The Parallel Genetic Algorithm », *in*: *Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990*, ed. by Gregory J. E. Rawlins, Morgan Kaufmann, 1990, pp. 316–337 (cit. on p. 114).

[MS22] Pedro Munari and Martin Savelsbergh, « Compact formulations for split delivery routing problems », *in*: *Transportation Science* (2022) (cit. on pp. 30, 31, 98, 109–111).

[MR20] Chase C Murray and Ritwik Raj, « The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones », *in*: *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 368–398 (cit. on p. 9).

[NB09] Yuichi Nagata and Olli Bräysy, « Edge assembly-based memetic algorithm for the capacitated vehicle routing problem », *in*: *Networks: An International Journal* 54.*4* (2009), pp. 205–215 (cit. on pp. 34, 40, 63, 64, 67, 82, 99, 100, 104, 114).

[NBD10] 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 (cit. on pp. 34, 63, 64, 82, 99).

[NK97] Yuichi Nagata and Shigenobu Kobayashi, « Edge Assembly Crossover: A High-Power Genetic Algorithm for the Travelling Salesman Problem », *in*: *Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23, 1997*, ed. by Thomas Bäck, Morgan Kaufmann, 1997, pp. 450–457 (cit. on pp. 23, 33, 62–64, 67, 70, 82, 84, 99).

[NK13] Yuichi Nagata and Shigenobu Kobayashi, « A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem », *in*: *INFORMS Journal on Computing* 25.*2* (2013), pp. 346–363 (cit. on pp. 23, 33, 43, 46, 62–64, 67, 70, 82, 84, 86, 99, 114, 115).

[Nar+13] Koushik Venkata Narasimha, Elad Kivelevitch, Balaji Sharma, and Manish Kumar, « An ant colony optimization technique for solving min–max multi-depot vehicle routing problem », *in*: *Swarm and Evolutionary Computation* 13 (2013), pp. 63–73 (cit. on p. 24).

[NCM12] Ferrante Neri, Carlos Cotta, and Pablo Moscato, eds., *Handbook of Memetic Algorithms*, vol. 379, Studies in Computational Intelligence, Springer, 2012 (cit. on p. 62).

[OSH87] IM Oliver, DJd Smith, and John RC Holland, « Study of permutation crossover operators on the traveling salesman problem », *in*: *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987 (cit. on pp. 33, 130).

[Or76] I Or, « Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking », *in*: *PhD thesis (Department of Industrial Engineering and Management Science, Northwestern University)* (1976) (cit. on p. 44).

[Ost+17] Krzysztof Ostrowski, Joanna Karbowska-Chilinska, Jolanta Koszelew, and Pawel Zabielski, « Evolution-inspired local improvement algorithm solving orienteering problem », *in*: *Annals of Operations Research* 253.*1* (2017), pp. 519–543 (cit. on pp. 26, 27, 33).

[OKY18] Gizem Ozbaygin, Oya Karasan, and Hande Yaman, « New exact solution approaches for the split delivery vehicle routing problem », *in*: *EURO Journal on Computational Optimization* 6.*1* (2018), pp. 85–115 (cit. on pp. 30, 31, 98, 109, 111).

[PS15] Venkatesh Pandiri and Alok Singh, « Two metaheuristic approaches for the multiple traveling salesperson problem », *in*: *Applied Soft Computing* 26 (2015), pp. 74–89 (cit. on pp. 23, 24, 72, 73).

[PS18] Venkatesh Pandiri and Alok Singh, « A swarm intelligence approach for the colored traveling salesman problem », *in*: *Applied Intelligence* 48.*11* (2018), pp. 4412–4428 (cit. on pp. 19–21, 41, 48, 53).

[PSC13] Odivaney Pedro, Rodney Saldanha, and Ricardo Camargo, « A tabu search approach for the prize collecting traveling salesman problem », *in*: *Electronic Notes in Discrete Mathematics* 41 (2013), pp. 261–268 (cit. on p. 27).

[Pot96] Jean-Yves Potvin, « Genetic algorithms for the traveling salesman problem », *in*: *Annals of Operations Research* 63.*3* (1996), pp. 337–370 (cit. on p. 33).

[Pot09] Jean-Yves Potvin, « State-of-the art review-Evolutionary algorithms for vehicle routing », *in*: *INFORMS Journal on Computing* 21.*4* (2009), pp. 518–548 (cit. on pp. 33, 82, 99, 114).

[PR95] Jean-Yves Potvin and Jean-Marc Rousseau, « An exchange heuristic for routeing problems with time windows », *in*: *Journal of the Operational Research Society* 46.*12* (1995), pp. 1433–1446 (cit. on pp. 66–68, 103, 107).

[Pri04] Christian Prins, « A simple and effective evolutionary algorithm for the vehicle routing problem », *in*: *Computers & Operations Research* 31.*12* (2004), pp. 1985–2002 (cit. on pp. 32, 34, 40, 82, 99).

[RB91] Ram Ramesh and Kathleen M Brown, « An efficient four-phase heuristic for the generalized orienteering problem », *in*: *Computers & Operations Research* 18.*2* (1991), pp. 151–165 (cit. on pp. 25, 27, 82).

[Rao80] MR Rao, « A note on the multiple traveling salesmen problem », *in*: *Operations Research* 28.*3-part-i* (1980), pp. 628–632 (cit. on p. 21).

[Ras+03] Steven Rasmussen, Phillip Chandler, Jason Mitchell, Corey Schumacher, and Andrew Sparks, « Optimal vs. heuristic assignment of cooperative autonomous unmanned air vehicles », *in*: *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, p. 5586 (cit. on p. 22).

[RP06] Stefan Ropke and David Pisinger, « An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows », *in*: *Transportation Science* 40.*4* (2006), pp. 455–472 (cit. on p. 106).

[San19] Alberto Santini, « An adaptive large neighbourhood search algorithm for the orienteering problem », *in*: *Expert Systems with Applications* 123 (2019), pp. 154–167 (cit. on pp. 26, 27, 89, 90, 92, 135–142).

[Sha98] Paul Shaw, « Using constraint programming and local search methods to solve vehicle routing problems », *in*: *International conference on principles and practice of constraint programming*, Springer, 1998, pp. 417–431 (cit. on p. 105).

[Shi+18] Jianli Shi, Jin Zhang, Kun Wang, and Xin Fang, « Particle swarm optimization for split delivery vehicle routing problem », *in*: *Asia-Pacific Journal of Operational Research* 35.*02* (2018), p. 1840006 (cit. on pp. 31, 32, 110).

[SG10] John Silberholz and Bruce Golden, « The effective application of a new approach to the generalized orienteering problem », *in*: *Journal of Heuristics* 16.*3* (2010), pp. 393–415 (cit. on pp. 26, 27, 86).

[SSO15] Marcos Melo Silva, Anand Subramanian, and Luiz Satoru Ochi, « An iterated local search heuristic for the split delivery vehicle routing problem », *in*: *Computers & Operations Research* 53 (2015), pp. 234–249 (cit. on pp. 31, 32, 107–109, 149).

[SB09] Alok Singh and Anurag Singh Baghel, « A new grouping genetic algorithm approach to the multiple traveling salesperson problem », *in*: *Soft Computing* 13.*1* (2009), pp. 95–101 (cit. on pp. 20, 22, 23, 57).

[SFK97] Davor Skrlec, Minea Filipec, and Slavko Krajcar, « A heuristic modification of genetic algorithm used for solving the single depot capacited vehicle routing problem », *in*: *Proceedings Intelligent Information Systems. IIS'97*, IEEE, 1997, pp. 184–188 (cit. on p. 34).

[SLK02] Sung Hun Song, Kwan Suk Lee, and Gi Sub Kim, « A practical approach to solving a newspaper logistics problem using a digital map », *in*: *Computers & Industrial Engineering* 43.*1-2* (2002), pp. 315–330 (cit. on p. 98).

[Soy15] Banu Soylu, « A general variable neighborhood search heuristic for multiple traveling salesmen problem », *in*: *Computers & Industrial Engineering* 90 (2015), pp. 390–401 (cit. on pp. 20, 23, 46, 67, 68).

[Sun+20] Wen Sun, Jin-Kao Hao, Wenyu Wang, and Qinghua Wu, « Memetic search for the equitable coloring problem », *in*: *Knowledge-Based Systems* 188 (2020), p. 105000 (cit. on p. 46).

[SH73] Joseph A Svestka and Vaughn E Huckfeldt, « Computational experience with an m-salesman traveling salesman algorithm », *in*: *Management Science* 19.*7* (1973), pp. 790–799 (cit. on p. 21).

[Tai+97] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin, « A tabu search heuristic for the vehicle routing problem with soft time windows », *in*: *Transportation Science* 31.*2* (1997), pp. 170–186 (cit. on pp. 43, 44, 68).

[TS00] M Fatih Tasgetiren and Alice E Smith, « A genetic algorithm for the orienteering problem », *in*: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 2, IEEE, 2000, pp. 910–915 (cit. on pp. 27, 33).

[TSZ06] Reza Tavakkoli-Moghaddam, AR Saremi, and MS Ziaee, « A memetic algorithm for a vehicle routing problem with backhauls », *in*: *Applied Mathematics and Computation* 181.*2* (2006), pp. 1049–1060 (cit. on p. 34).

[Tod+17] Raca Todosijević, Saıd Hanafi, Dragan Urošević, Bassem Jarboui, and Bernard Gendron, « A general variable neighborhood search for the swap-body vehicle routing problem », *in*: *Computers & Operations Research* 78 (2017), pp. 468–479 (cit. on p. 67).

[TV14] Paolo Toth and Daniele Vigo, *Vehicle routing: problems, methods, and applications*, SIAM, 2014.

[Tsi84] Theodore Tsiligirides, « Heuristic methods applied to orienteering », *in*: *Journal of the Operational Research Society* 35.*9* (1984), pp. 797–809 (cit. on p. 27).

[Van09] Pieter Vansteenwegen, « Planning in tourism and public transportation », *in*: *4OR* 7.*3* (2009), pp. 293–296 (cit. on p. 28).

[VG19] Pieter Vansteenwegen and Aldy Gunawan, « State-of-the-art solution techniques for OP and TOP », *in*: *Orienteering Problems*, Springer, 2019, pp. 41–66 (cit. on pp. 26, 28).

[VSV11] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden, « The orienteering problem: A survey », *in*: *European Journal of Operational Research* 209.*1* (2011), pp. 1–10 (cit. on pp. 25, 26, 82).

[Vid17] Thibaut Vidal, « Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension », *in*: *Operations Research* 65.*4* (2017), pp. 992–1010 (cit. on p. 34).

[Vid22] Thibaut Vidal, « Hybrid genetic search for the CVRP: Open-source implementation and SWAP* Neighborhood », *in*: *Computers & Operations Research* 140 (2022), p. 105643 (cit. on pp. 34, 63, 71, 83, 88, 89, 100, 106, 114).

[Vid+12] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei, « A hybrid genetic algorithm for multidepot and periodic vehicle routing problems », *in*: *Operations Research* 60.*3* (2012), pp. 611–624 (cit. on pp. 34, 40, 82, 99, 108).

[Vid+13] 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 (cit. on pp. 34, 63, 82, 99).

[Vid+14] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins, « A unified solution framework for multi-attribute vehicle routing problems », *in*: *European Journal of Operational Research* 234.*3* (2014), pp. 658–673 (cit. on pp. 34, 63, 71, 82, 99, 114, 129).

[Wan+95] Qiwen Wang, Xiaoyun Sun, Bruce L Golden, and Jiyou Jia, « Using artificial neural networks to solve the orienteering problem », *in*: *Annals of Operations Research* 61.*1* (1995), pp. 111–120 (cit. on p. 27).

[WGW15] Xingyin Wang, Bruce Golden, and Edward Wasil, « The min-max multi-depot vehicle routing problem: Heuristics and computational results », *in*: *Journal of the Operational Research Society* 66.*9* (2015), pp. 1430–1441 (cit. on pp. 22, 24, 25, 68, 73, 74, 133).

[WCL17] Yongzhen Wang, Yan Chen, and Yan Lin, « Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem », *in*: *Computers & Industrial Engineering* 106 (2017), pp. 105–122 (cit. on pp. 20, 23, 24, 40, 67, 68, 72, 73).

[WC12] Joseph IV Wilck and Tom Cavalier, « A genetic algorithm for the split delivery vehicle routing problem », *in*: *American Journal of Operations Research* 2 (2012), pp. 207–216 (cit. on pp. 31, 32, 110).

[Yua+13] Shuai Yuan, Bradley Skinner, Shoudong Huang, and Dikai Liu, « A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms », *in*: *European Journal of Operational Research* 228.*1* (2013), pp. 72–82 (cit. on pp. 20, 23).

[ZK10] Emmanouil E Zachariadis and Chris T Kiranoudis, « A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem », *in*: *Computers & Operations Research* 37.*12* (2010), pp. 2089–2105 (cit. on p. 69).

[Zha+15] Zizhen Zhang, Huang He, Zhixing Luo, Hu Qin, and Songshan Guo, « An efficient forest-based tabu search algorithm for the split-delivery vehicle routing problem », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 1, 2015 (cit. on pp. 31, 32, 110).

[Zhe+22a] Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li, « Reinforced Lin-Kernighan-Helsgaun Algorithms for the Traveling Salesman Problems », *in*: *arXiv preprint arXiv:2207.03876* (2022) (cit. on pp. 20, 50).

[Zhe+22b] Jiongzhi Zheng, Yawei Hong, Wenchang Xu, Wentao Li, and Yongfu Chen, « An effective iterated two-stage heuristic algorithm for the multiple Traveling Salesmen Problem », *in*: *Computers & Operations Research* 143 (2022), p. 105772 (cit. on pp. 23, 24, 68, 72–74, 77, 129, 131, 132).

[ZSP18] Honglu Zhou, Mingli Song, and Witold Pedrycz, « A comparative study of improved GA and PSO in solving multiple traveling salesmen problem », *in*: *Applied Soft Computing* 64 (2018), pp. 564–580 (cit. on p. 23).

[ZHG18] Yangming Zhou, Jin-Kao Hao, and Fred Glover, « Memetic search for identifying critical nodes in sparse graphs », *in*: *IEEE Transactions on Cybernetics* 49.*10* (2018), pp. 3699–3712 (cit. on pp. 21, 46, 58).

[Zho+22] Yangming Zhou, Wenqiang Xu, Zhang-Hua Fu, and MengChu Zhou, « Multi-Neighborhood Simulated Annealing-Based Iterated Local Search for Colored Traveling Salesman Problems », *in*: *IEEE Transactions on Intelligent Transportation Systems* 23.*9* (2022), pp. 16072–16082 (cit. on pp. 20, 50).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

université angers

Titre : Algorithme génétique hybride pour quelques problèmes de routage de véhicules

Mot clés : Problèmes de voyageur de commerce, Problème de tournées de véhicules, Algorithme génétique hybride, Croisement d'assemblage d'arc, Optimisation combinatoire.

Résumé : Cette thèse présente des algorithmes génétiques hybrides pour quatre problèmes de routage : le problème de voyageurs de commerce colorés (CTSP), le problème de voyageurs de commerce minmax multiples (minmax mTSP), le problème de voyageurs de commerce avec bénéfices (TSP avec bénéfices) et le problème de routage de véhicules de livraison fractionnne (SDVRP). Ces problèmes sont largement présents dans des applications du monde réel et sont utiles pour modéliser de nombreux problèmes pratiques. Étant donné leur grande difficulté en terme de résolution, les métaheuristiques sont un choix pertinant pour résoudre les instances difficiles. Quatre algorithmes génétiques hybrides associés à des opérateurs de croisement dédiés et des procédures de recherche locale sont proposés pour résoudre ces problèmes. En particulier, le puissant croisement d'assemblage d'arc est étendu et généralisé pour résoudre des problèmes de routage riche. Les études expérimentales réalisées sur un large éventail d'instances de référence indiquent que les approches proposées rivalisent favorablement avec les algorithmes de l'état de l'art. Des expériences approfondies montrent le rôle des éléments clés de nos algorithmes, notamment le "croisement d'assemlage d'arc général" et la recherche locale pour le SDVRP et la préservation de la diversité pour le TSP avec profits.

Title: Hybrid genetic algorithm for routing problems

Keywords: Traveling salesman problem, Vehicle routing, Hybrid genetic algorithm, Edge assembly crossover, Combinatorial optimization.

Abstract: This thesis presents hybrid genetic algorithms for four routing problems: colored traveling salesmen problem (CTSP), minmax multiple traveling salesmen problem (minmax mTSP), traveling salesman problems with profits (TSPs with profits) and split delivery vehicle routing problem (SDVRP). These problems widely exist in real-life applications and are useful to model numerous practical problems. Given that they are computational challenge, metaheuristic algorithms are naturally presented to solve large-sized instances. Four hybrid genetic algorithms associated with dedicated crossover operators and local search procedures are proposed for these problems. In particular, the powerful edge assembly crossover is extended and generalized to solve rich routing problems. Computational studies performed on a wide range of benchmark instances indicate that the proposed approaches compete favourably with state-of-the-art algorithms. Additional experiments show the roles of the key composing ingredients of our algorithms, including the general edge assembly crossover, the local search, for the SDVRP and the diversity preservation for TSPs with profits.