Alma Mater Studiorum - Università di Bologna

Dipartimento di Ingegneria dell'Energia Elettrica e
dell'Informazione "Guglielmo Marconi"

Dottorato di Ricerca in
Ingegneria Biomedica, Elettrica e dei Sistemi

Ciclo XXXI

Settore Concorsuale: 01/A6 - RICERCA OPERATIVA
Settore Scientifico Disciplinare: MAT/09 - RICERCA OPERATIVA

Algorithms for Variants of Routing Problems

Presentata da Carlos Emilio Contreras Bolton

Coordinatore Dottorato:
Prof. Daniele Vigo

Supervisori:
Prof. Daniele Vigo
Prof. Paolo Toth
Dr. Valentina Cacchiani

Esame finale anno 2019

Abstract

In this thesis, we propose mathematical optimization models and algorithms for variants of routing problems. The first contribution consists of models and algorithms for the Traveling Salesman Problem with Time-dependent Service times (TSP-TS). We propose a new Mixed Integer Programming model and develop a multi-operator genetic algorithm and two Branch-and-Cut methods, based on the proposed model. The algorithms are tested on benchmark symmetric and asymmetric instances from the literature, and compared with an existing approach, showing the effectiveness of the proposed algorithms. The second work concerns the Pollution Traveling Salesman Problem (PTSP). We present a Mixed Integer Programming model for the PTSP and two mataheuristic algorithms: an Iterated Local Search algorithm and a Multi-operator Genetic algorithm. We performed extensive computational experiments on benchmark instances. The last contribution considers a rich version of the Waste Collection Problem (WCP) with multiple depots and stochastic demands using Horizontal Cooperation strategies. We developed a hybrid algorithm combining metaheuristics with simulation. We tested the proposed algorithm on a set of large-sized WCP instances in non-cooperative scenarios and cooperative scenarios.

Acknowledgements

I would like to thank my supervisors Prof. Paolo Toth, Dr. Valentina Cacchiani and Prof. Daniele Vigo, for the help and guidance they provided during my three years of Ph.D. studies.

I want to thank all the colleagues of the group of Operations Research of the DEI, for their help and good shared moments. I would like also to thank all of my family and friends, specially my parents and my brother for their unconditional support and love.

Finally, I would like also to thank the support of the Chilean Council of Scientic and Technological Research, CONICYT, through the grant CONICYT PFCHA/DOCTORADO BECAS CHILE/2015-72160389.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In today's competitive world, transport and logistics companies which want to reduce their expenses have to consider various costs. These costs often represent a large percentage of the total budget and involve crucial decisions, such as: the location of depots (strategic problem), the definition of vehicle routes, and also how to distribute the load of each truck to make optimal use of the available space (operational problems). Especially for distributors of goods, many of the above mentioned problems are combinatorial optimization problems. Combinatorial optimization is a branch of optimization in applied mathematics and computer science, and is related to operations research, theory of algorithms and computational complexity theory and also to other fields such as artificial intelligence and software engineering (Papadimitriou & Steiglitz, 1998).

Many complex combinatorial optimization problems cannot be solved efficiently and remain a challenge for the scientific community. This computational intractability arises because many combinatorial problems are NP-hard (Garey & Johnson, 1979). Up to date, there are still no existing algorithms to solve in polynomial time the problems previously mentioned.

One of the most studied combinatorial optimization problems is the problem of distributing products from some depots to their clients, known as the Vehicle Routing Problem (VRP). Due to its great relevance in real-life applications and also to its intrinsic difficulty, the VRP quickly attracted the attention of experts and scholars, working in operations research, management science, computer science, graph theory, and proved to be widely used in transportation systems, logistics distribution systems and express delivery system. In fact, year after year, an exorbitant amount of money is being spent daily on fuel, vehicle operation, maintenance of vehicles and labor. Therefore, it is essential to try to reduce the amount of money spent on routing and its related activities. A small improvement in routing problems can lead to huge logistics savings in absolute terms (Anbuudayasankar et al., 2014).

The aim of the VRP is to find routes for a vehicle fleet to serve a set of customers, while typically minimizing the total transportation time (Toth & Vigo, 2014). The VRP was first introduced by Dantzig & Ramser (1959), who proposed to solve the transport route optimization problem for a refinery. Since there, over the years, several variants have been studied. These variants are based on distinctions such as: service type to customers, fleet size, fleet type, demand type, and many others. These problems have many real life applications including urban waste collection (Beliën et al., 2014), vehicle routing in snow plowing operations (Perrier et al., 2008), college bus routing (Christodoulou, 2010), public-transport

vehicle routing (Ceder, 2011) or emergency vehicle dispatching systems during electrical breakdowns (Weintraub et al., 1999), just to name a few. Furthermore, there are many different techniques for solving the VRP, including exact methods, heuristics, metaheuristics, etc. (Toth & Vigo, 2014; Laporte et al., 2014; Kim et al., 2015).

The VRP is a generalization of another well-known problem: the Traveling Salesman Problem (TSP) which consists of finding a minimum cost route that goes exactly once through a given set of cities that the salesman must visit returning to the starting city (Hamiltonian tour). Therefore, in the TSP, only one vehicle is considered with infinite capacity and no additional constraints are imposed. Just like the VRP, the TSP has been extensively researched since the early 19th century. However, the origin of the name is not clear, as there is no documentation concerning a specific author. What the scientific community seems to agree on is that the name was minted in the 1930s at the Princeton University (Applegate et al., 2007).

One of the first important works on the TSP, which is also considered one of the most important milestones in the history of combinatorial optimization, was made by Dantzig et al. (1954) from the RAND Corporation's, who showed that a route of 49 cities, through each of the 48 US states and Washington DC, had the shortest route distance. The work was carried out using an ingenious combination of linear programming, graph theory and map reading. Another important result has been obtained by Karp (1972), who showed that the problem of determining if at least one Hamilton tour exists in a sparse graph is NP-complete, which implies the NP-hardness of the TSP. Therefore, as the VRP is a generalization of the TSP, the VRP is NP-hard as well (Papadimitriou & Steiglitz, 1998).

TSP has several applications such as planning, logistics, manufacture of microchips. With slight modifications, is found as a sub-problem in many areas, such as DNA sequencing (Applegate et al., 2007). In these applications, the cities represent, customers, soldering points, or DNA fragments, and the distances represent traveling times, cost, or a similarity measures between DNA fragments. The TSP also appears in Aerospace problems, such as the active space debris removal missions (Izzo et al., 2015). In many applications, additional constraints, such as limited resources or time windows, may be imposed. Furthermore, several different approaches have emerged for solving the TSP, including exact algorithms, heuristics, metaheuristics and even the Hybridization of them (Applegate et al., 2007; Anbuudayasankar et al., 2014).

As mentioned above, the TSP has several variations, one of them is the time-dependent Traveling Salesman Problem (TDTSP), which is a generalization of the TSP where the cost of any given arc depends on its position in the tour (Gouveia & Voß, 1995). In the literature (Picard & Queyranne, 1978; Vander Wiel & Sahinidis, 1996; Bigras et al., 2008), the time-dependency has been addressed in terms of travel times, associated with the visiting order of the customers, not with the arrival time or the departure time. However, this way of modeling the problem does not sound pretty realistic because the time-dependency usually varies according to the factors which naturally occur on the time of day (e.g., availability of parking spaces, traffic congestion, and so on). Recently, Taş et al. (2016) have introduced the TSP with time-dependent service times (TSP-TS) where the time-dependency

considers the service times instead of the travel times. Indeed, the service duration depends on a continuous function of the time at which service starts. Taş et al. (2016) proved some important properties and proposed a Mixed Integer Linear Programming (MILP) model, additional strengthening constraints, strengthening lower and upper bounds and also benchmark instances. In Chapter 2, we propose a new Mixed Integer Programming model for TSP-TS, incorporating exponentially many subtour elimination constraints, that are separated in a dynamic way. The proposed model is enhanced by lower and upper bounds, that improve previous bounds from the literature. In addition, we develop a multi-operator genetic algorithm and two Branch-and-Cut methods, based on the proposed model. The algorithms are tested on benchmark symmetric instances from the literature, and compared with an existing approach. The computational results show that the proposed exact methods are able to prove the optimality of the solutions found for a larger set of instances in shorter computing times. We also tested the genetic and the Branch-and-Cut algorithms on larger size symmetric instances with up to 58 nodes and on asymmetric instances with up to 45 nodes, demonstrating the effectiveness of the proposed algorithms.

It is very likely that global warming is related with human-made CO_2 emissions and the transport activities have produced around 24% of these emissions, and 17% of them are produced in highways (European Environment Agency, 2011). Hence, the trend is that more and more countries have started to adopt emission-reducing actions. This has been reflected also in the recent years with a growing number of works on the VRP which started to incorporate "green" aspects such as pollution and alternative fuels, among others. These types of works are called Green Vehicle Routing Problems (GVRP) and aim to design a route that takes into account environmental factors such as fuel consumption and CO_2 emissions (Lin et al., 2014; Eglese & Bektaş, 2014). A subcategory of the GVRP is the Pollution Routing Problem (PRP), which considers the speed as the most important factor to estimate the CO_2 emissions; minimizing at the same time the traveled distance, and also other relevant factors such as load weight and distribution, vehicle engine, vehicle design, driving style, engine size and road gradient (Eglese & Black, 2015). Thus, Bektaş & Laporte (2011) took those ideas and introduced the PRP, an extension of the VRP with time windows, whose objective is to minimize the greenhouse gases emission costs plus the labor costs. In chapter 3, motivated by these recent works on the PRP, we study the Pollution Traveling Salesman Problem (PTSP), i.e. the problem of determining a Hamiltonian tour that minimizes a function of fuel consumption (dependent on vehicle speed and load) and driver costs. We present a MILP model for the PTSP, enhanced with sub-tour elimination constraints, and two mataheuristic algorithms: an Iterated Local Search algorithm (ILS) and a Multi-operator Genetic algorithm (MGA). Extensive computational experiments on PVRP benchmark instances adapted for the PTSP, with up to 50 customers are reported.

One of the most critical logistics activities in modern cities is the waste collection, due to the considerable impact on the quality of life, urban environment, traffic flows and municipal budgets. In fact, waste management has annual per capita costs of up to 200€ only for the collection of urban waste in European

cities (Hoornweg & Bhada-Tata, 2012). An attempt to optimize urban waste collection is the Waste Collection Problem (WCP) which can be formulated as a special instance of the VRP, in which a vehicle fleet located at a central depot has to serve a set of containers (Beliën et al., 2014; Ghiani et al., 2014). However, most existing works (Kim et al., 2006; Benjamin & Beasley, 2010; Buhrkal et al., 2012; Inghels et al., 2016) address simplified versions where the container loads are considered to be known in advance and served by a single vehicle depot, without taking into account rich and realistic WCP environments, e.g. scenarios with a large number of containers, multiple vehicle depots and uncertainty in waste levels or demands. A more realistic way is to consider the waste levels of the garbage containers stochasticly, leading to more reliable routing solutions in waste management. In Chapter 4, we propose a rich version of the WCP with multiple depots and stochastic demands. We present a hybrid algorithm combining metaheuristics with simulation. Furthermore, the Simheuristic approach allows us to study the effects of cooperation among different depots, thus quantifying the potential savings this cooperation could provide to city governments and waste collection companies.

Chapter 2

Models and Algorithms for the Traveling Salesman Problem with Time-Dependent Service Times

2.1 Introduction

The Traveling Salesman Problem with Time-dependent Service times (TSP-TS) has been recently introduced in Taş et al. (2016), where Mixed Integer Programming (MIP) models and lower and upper bounds were proposed. In the TSP-TS, the time to serve a customer is not assumed to be constant, but, as it happens in some real-life applications, is defined by a function of the start time of service at the customer. Before work Taş et al. (2016), service times were considered as fixed values and directly included in the travel times. However, in practice, service times are not always constant: for example, the availability of parking lots can be different at different times of the day, or some areas can be limited to traffic in certain time periods. Therefore, the vehicle can get closer to or farther from the customers to be visited, thus requiring shorter or longer service times. Another example is the personnel availability for loading or unloading goods at customers, that can vary over time. Similarly, the amount of goods in the customer warehouses can be smaller at the beginning of the day and increase later on, thus making the service times needed to store the goods at the customers variable during the day.

In the TSP literature, most of the works that take time dependency into account consider the variability of the travel times, i.e., study the so-called Time-Dependent TSP (TDTSP). In several papers, the arc cost is dependent on the position of the arc in the TSP tour. This problem is related to single machine scheduling, in which a set of jobs needs to be scheduled, and the transition cost c_{ijt} depends on the two consecutive jobs i and j and on the position t at which job i is processed. In addition, set-up and cooling costs are considered, respectively, for the initial and final state of the machine. The goal is to determine the minimum cost sequence of jobs. In Picard & Queyranne (1978), three Integer Programming formulations are proposed, one of which is a quadratic model. A Branch-and-Bound algorithm based on a subgradient optimization procedure is

This chapter is based on the contents of: Cacchiani, Contreras-Bolton, & Toth, Models and Algorithms for the Traveling Salesman Problem with Time-Dependent Service Times. Submitted to the European Journal of Operational Research, pages 1–30, 2018.

developed. In addition, the objective of minimizing the tardiness costs is considered. In Vander Wiel & Sahinidis (1996), a Benders decomposition approach is applied to a MIP model, which is a linearization of the quadratic formulation presented in Picard & Queyranne (1978), and combined with Pareto-optimal Benders cuts. In Bigras et al. (2008), the two linear formulations proposed in Picard & Queyranne (1978) are considered, and strengthened by applying Dantzig–Wolfe decomposition and cuts for the classical TSP and the node packing problem. A Branch-and-Cut-and-Price algorithm, based on these formulations, is developed. In Abeledo et al. (2013), the polytope corresponding to a TDTSP linear formulation proposed in Picard & Queyranne (1978) is studied, and facets are identified. A Branch-and-Cut-and-Price algorithm, that includes the proposed cuts, is developed. In Miranda-Bront et al. (2014), two formulations presented in Picard & Queyranne (1978) and Vander Wiel & Sahinidis (1996) are studied and compared. Additional valid inequalities and facets are proposed, and embedded in a Branch-and-Cut approach. We refer the reader to Gouveia & Voß (1995) for a classification of the formulations of the TDTSP.

The time dependency related to the travel times is also taken into account by another group of works that, instead of considering that the arc costs depend on the position of the arc in the tour, consider that the cost (or travel time) $\tau_{ij}(t)$ of arc (i, j) depends on the time t at which the vehicle leaves node i. In Cordeau et al. (2012), the time horizon is partitioned into intervals, and the average speed value in each interval is assumed to be known. The travel time $\tau_{ij}(t)$ is then computed according to the function proposed in Ichoua et al. (2003), that considers the speed to be constant during an interval, and allows it to change when moving to the following interval, i.e., the travel speeds are constant piecewise. The travel speed along an arc and during an interval is defined in Cordeau et al. (2012) as dependent on the maximum arc travel speed, the lightest congestion factor (i.e., the best congestion during the interval) and the degradation of the congestion factor w.r.t the least congested arc. In Cordeau et al. (2012), an Integer Linear Programming (ILP) model, including subtour elimination constraints, is proposed for this problem, and valid inequalities are derived. A Branch-and-Cut algorithm is developed and tested on a set of instances having different traffic patterns. In Arigliano et al. (2018), a Branch-and-Bound algorithm is proposed for the same problem, and tested on the same instances and on larger size ones (up to 50 customers). The reported computational results show that this algorithm outperforms the approach proposed in Cordeau et al. (2012).

Finally, more recently, the problem studied in Cordeau et al. (2012) has been extended to include time window constraints. This extension was first proposed in Arigliano et al. (2015), where an ILP model and valid inequalities were proposed. In addition, a Branch-and-Cut algorithm was developed and tested on instances adapted from those considered in Cordeau et al. (2012). In Montero et al. (2017), an alternative formulation, based on the model proposed in Sun et al. (2018) for the Profitable Time-Dependent TSP with Time Windows and Pickup and Delivery, was presented, and a Branch-and-Cut algorithm was developed. The computational results show that this algorithm obtains many additional optimal solutions compared to the approach presented in Arigliano et al. (2015). The recent work Vu et al. (2018) also studies the time-dependent TSP with time windows. A

solution method, based on the representation of the problem on a time-expanded network, is proposed. The method relies on the Dynamic Discretization Discovery framework, proposed in Boland et al. (2017), that works on a partially time-expanded network, and iteratively refines it until optimality is guaranteed. This new approach is also tested on the benchmark instances considered in Arigliano et al. (2015) and obtains the optimal solution for all the instances in shorter computing times.

2.1.1 Contribution

To the best of our knowledge, the only work studying the TSP-TS is Taş et al. (2016), where compact Mixed Integer Programming (MIP) models, i.e. formulations having a polynomial number of constraints, have been proposed, together with lower and upper bounds aimed at improving the solution process. In Section 2.3, we report the main outcomes of Taş et al. (2016). In this chapter, we propose a new MIP model for the TSP-TS, that can be used with linear or quadratic service time functions, and that embeds novel improved lower and upper bounds. In this model, we consider exponentially many subtour elimination constraints, that are separated dynamically. In addition, we developed a multi-operator Genetic Algorithm (GA), that includes many crossover and mutation operators from the literature, and inserts, in the initial population, solutions generated by using the solution of the continuous relaxation of the proposed MIP model. The main contribution of this work is the development of Branch-and-Cut (B&C) algorithms, based on the presented model: in particular, we propose a B&C algorithm that includes all the improved bounds and the subtour elimination constraints separation, and a Dynamic B&C algorithm, in which the lower and upper bounds are dynamically updated at each node of the decision tree. The GA, B&C and Dynamic B&C algorithms are tested on symmetric benchmark instances from Taş et al. (2016), and on asymmetric and larger size symmetric instances from the TSPLIB library (Reinelt (1991)).

Section 2.2 presents the problem definition and introduces the used notation. In Section 2.3 we report the mathematical models and the lower and upper bounds proposed in Taş et al. (2016). Section 2.4 describes the new mathematical model and the improvements to the lower and upper bounds from the literature. Section 2.5 is devoted to the description of the solution methods, i.e. the GA, B&C and Dynamic B&C algorithms, and in Section 3.5 we report the obtained results and the comparison with the method presented in Taş et al. (2016). Finally, we conclude our chapter with some remarks in Section 3.6.

2.2 Problem Definition

TSP-TS is defined on a complete directed graph $G = (N, A)$. The set of nodes $N = \{0, 1, \ldots, n, n+1\}$ contains the set of customers $N \setminus \{0, n+1\}$ and the depot represented by nodes 0 and $n+1$ (the depot is duplicated for convenience). The set of arcs A contains one arc (i, j) for each pair of nodes and has an associated travel time t_{ij}, as in the classical Asymmetric TSP (ATSP) (Öncan et al. (2009), Roberti & Toth (2012)). In addition, each customer $i \in N \setminus \{0, n+1\}$ requires a

service time, defined as a continuous function $s_i(b_i)$, where b_i represents the start time of service at node i. We set b_0, $s_0(b_0)$, and $s_{n+1}(b_{n+1})$ to 0, since the depot does not require any delivery of goods.

In TSP-TS, we have to determine the Hamiltonian path (i.e., the path visiting each node exactly once) from node 0 to node $n+1$ that minimizes the total path duration, given by the sum of the total travel and service times. In the following, we will use the terms *"path"* or *"route"* as synonyms, since nodes 0 and $n+1$ are two copies of the same depot. Since TSP-TS generalizes ATSP, that occurs when $s_i(b_i) = 0$ $(i \in N \setminus \{0, n+1\})$, it is NP-hard. In addition, as observed in Taş et al. (2016), an optimal ATSP solution is not always optimal for TSP-TS. We next report three important properties proven in Taş et al. (2016) that are used for the definition of the mathematical models and of the lower and upper bounds.

Property 1. First-In-First-Out property (FIFO)*: If the service at customer i starts at a time b_i, any service starting at a later time at that customer cannot be completed earlier than $b_i + s_i(b_i)$. In addition, in Taş et al. (2016), the authors proved that $s_i(b_i)$ satisfies the FIFO property if and only if $\frac{ds_i(b_i)}{db_i} \geq -1$.*

Property 2. Waiting property*: If the salesman arrives at customer i before b_i' (the earliest time at which the FIFO property starts holding at that customer), waiting at that customer to begin service at time b_i' is then beneficial.*

Property 3. Arrival time property*: If all customers have the same service time function, then the waiting time required to satisfy the FIFO property can be spent at the depot.*

As in Taş et al. (2016), we will consider service time functions that satisfy the FIFO property for any value of the arrival time, and we will assume that the arrival time property holds, i.e., the start time of service at a customer coincides with the arrival time at that customer.

2.3 Mathematical Models and Bounds from the Literature

In this section, we report the MIP models and the lower and upper bounds proposed in Taş et al. (2016). The best method proposed in that chapter will be used for comparison in Section 3.5.

2.3.1 Basic Model

We present the *basic model* that is a natural formulation for the TSP-TS. For every arc $(i, j) \in A$, we introduce a binary variable x_{ij} $(i \in N, j \in N)$ assuming value 1 if node j is served immediately after node i (and 0 otherwise). In addition, for every node $i \in N$, we introduce a non-negative variable b_i representing the arrival time at node i. According to the arrival time property, b_i also corresponds to the start time of service at i. The basic model reads as follows:

$$\min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} + \sum_{i \in N \setminus \{0, n+1\}} s_i(b_i) \tag{2.1}$$

$$\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N \setminus \{n+1\}, \tag{2.2}$$

$$\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{2.3}$$

$$b_i + s_i(b_i) + t_{ij} - M(1 - x_{ij}) \leq b_j, \quad i \in N,\ j \in N, \tag{2.4}$$

$$b_i \geq 0, \quad i \in N, \tag{2.5}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N,\ j \in N. \tag{2.6}$$

The objective function (2.1) is to minimize the total duration of the Hamiltonian path from node 0 to node $n+1$, where the duration is given by the sum of the total travel times (first summation) and of the total service times (second summation). As in the classical ATSP, we impose, by constraints (2.2) and (2.3) respectively, to have exactly one outgoing arc from each node except for the depot $n+1$, and one ingoing arc to each node, except for the depot 0. The novel constraints (2.4) are specific for the TSP-TS and require that, if node j is visited immediately after node i, the start time of service b_j at node j must be at least the start time of service b_i at node i plus the service time $s_i(b_i)$ at node i plus the travel time from i to j. To deactivate these constraints when the corresponding arc (i, j) is not selected, a large positive constant M is used, whose value is determined by a lower bounding procedure described in the following. Note that these constraints guarantee that subtours will not be selected, thus making model (2.1)-(2.6) valid without explicit subtour elimination constraints. Finally, (2.5) and (2.6) define the domain of the variables.

The lower bounding procedure proposed in Taş et al. (2016) for determining the value of M is proven to be valid when all customers have the same (quadratic or linear) service time function, and works as follows. The first $n+1$ largest travel times are selected and taken in descending order. Let $L(1), \ldots, L(n+1)$ be the ordered list of the selected travel times. Then, the "path" consisting of the sequence of these travel times is considered, and the arrival (b_i) and departure (d_i) times at each node i in the path are computed:

- $b_1 = L(1); \quad d_i = b_i + s_i(b_i), \quad i = 1, \ldots, n$
- $b_i = d_{i-1} + L(i), \quad i = 2, \ldots, n+1.$

The M value is set equal to b_{n+1}, i.e., it corresponds to the arrival time at the depot. This value is sufficient to deactivate constraints (2.4) for arcs $(i, j) \in A$ having $x_{ij} = 0$.

2.3.2 Gavish and Graves Model

In Taş et al. (2016), in order to strengthen model (2.1)-(2.6), the Gavish and Graves (GG) subtour elimination constraints (Gavish & Graves (1978)) were added. Let g_{ij} $(i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\})$ be a non-negative variable denoting the

number of arcs in a path from depot 0 to arc $(i, j) \in A$. The GG constraints read as follows:

$$\sum_{j \in N \setminus \{0\}} g_{ij} - \sum_{j \in N \setminus \{0, n+1\}} g_{ji} = 1, \quad i = 1, \ldots, n, \tag{2.7}$$

$$0 \leq g_{ij} \leq n x_{ij}, \quad i = 1, \ldots, n, \ j = 1, \ldots, n+1. \tag{2.8}$$

Constraints (2.7) require that the number of arcs in a path from depot 0 to node i (whatever successor node is selected from i), must be one unit larger than the number of arcs in a path from depot 0 to the predecessor of node i (whatever node it is). Constraints (2.8) require the g_{ij} variables to be non-negative and restrict their value to be 0, if arc (i, j) is not selected, and at most n, if it is selected, since the length of the full Hamiltonian path is $n + 1$.

2.3.3 Lower and Upper Bounds

In Taş et al. (2016), three bounds were proposed:

- upper bound on the total route duration (B_1),
- lower bound on the total service time (B_2),
- lower and upper bounds on the arrival time at each customer (B_3).

The upper bound B_1 on the total route duration of the optimal solution is presented for the case in which all customers have the same service time function. It consists of applying the Nearest Neighbor Heuristic (NNH) algorithm for the ATSP. The obtained sequence of nodes $(0, a(1), a(2), \ldots, a(n), a(n+1) = n+1)$ is used to define the list of travel times $L(1), L(2), \ldots, L(n+1)$: in particular, $L(1) = t_{0,a(1)}$ and $L(k) = t_{a(k-1),a(k)}$ $(k = 2, \ldots, n+1)$. Then, arrival (b_i) and departure (d_i) times at each node i in the path are computed, as described for the computation of M. The value of the upper bound B_1 is set equal to the arrival time b_{n+1} at the depot.

The lower bound B_2 on the total service time is proposed for the case in which all customers have the same *linear* service time function $s_i(b_i) = \beta b_i + \gamma$, where $\beta, \gamma > 0$, and $s_i(b_i) > 0$ $(i \in N \setminus \{0, n+1\})$. The value of B_2 is computed by considering the first n smallest travel times. More precisely, let $L(1), L(2), \ldots, L(n)$ be the ascending ordered list of the first n smallest travel times, consider the path $(0, 1, 2, \ldots, n)$ whose arcs have the selected travel times, and compute the arrival (b_i) and departure (d_i) times at each node i in the path, as for the computation of M. The value of B_2 is given by $d_n - \sum_{i=1}^{n} L(i)$, i.e., it is obtained by considering the departure time from the last visited customer n and subtracting the total route travel time.

Finally, the upper and lower bounds B_3 on the arrival time at each customer $i \in N \setminus \{0, n+1\}$ are set, respectively, as: $b_i \leq M$ and $b_i \geq LB_3(i) =$ earliest time at which the FIFO property starts holding for customer i, which is 0 for the considered benchmark instances.

The best method proposed in Taş et al. (2016) consists of the basic model (2.1)-(2.6) (where the value of M is computed as described in Section 2.3.1), with the additional GG (polynomial) subtour elimination constraints (2.7)-(2.8) and the lower and upper bounds B_1, B_2 and B_3. We identify this method as TGJL16.

2.4 Improvements to Models and Bounds

In Section 2.4.1, we propose improved lower and upper bounds, and an improved value for M. In addition, in Section 2.4.2, we enhance the basic model by using exponentially many subtour elimination constraints (SECs). Furthermore, in Section 2.4.3, we propose a new model for the TSP-TS, which can also embed the improved bounds and the SECs.

2.4.1 Improved Bounds (IBs) and M

As described in Section 2.3, in Taş et al. (2016), three bounds were proposed to strengthen the basic model. We improve these three bounds, under the same assumptions, by proposing:

- an improved upper bound on the total route duration (IB_1),
- an improved lower bound on the total service time (IB_2),
- improved lower and upper bounds on the arrival time at each customer (IB_3).

To compute an upper bound IB_1 on the total route duration, instead of applying the NNH algorithm, we developed a multi-operator Genetic Algorithm (GA), that provides high quality solutions to the TSP-TS. GA is based on the calibration of the probabilities to be used for executing the considered crossover and mutation operators, performed according to the algorithm proposed in Contreras-Bolton & Parada (2015) for the TSP, but it also relies on the solution of the continuous relaxation of the basic model. In particular, as it will be shown in the computational results (Section 2.6.1), the GA obtains better solutions when the SECs are included in the model. The GA will be described in Section 2.5.1. It provides the sequence of nodes of the Hamiltonian path, that is used to compute the arrival and departure times at each node, and IB_1 is set equal to b_{n+1}.

The improved lower bound on the total service time IB_2 is proposed under the same assumptions considered for B_1, i.e., all customers have the same linear service time function $s_i(b_i) = \beta b_i + \gamma$, where β, $\gamma > 0$, and $s_i(b_i) > 0$ ($i \in N \setminus \{0, n+1\}$). The value of IB_2 is computed as the maximum of two lower bounds IBL_2 and IBE_2, obtained as follows. In IBL_2, instead of considering the first n smallest travel times as done in Taş et al. (2016), we compute, for each node i ($i = 0, 1, \ldots, n$) the minimum travel time τ_i of the arcs leaving node i (avoiding to use arc (j, i) if arc (i, j) is used). Then, we let $L(1), L(2), \ldots, L(n)$ be the ascending ordered list of the smallest n values τ_i, and proceed as in the computation of B_2, i.e., we compute the arrival (b_i) and departure (d_i) times at each node i and set $IBL_2 = d_n - \sum_{i=1}^{n} L(i)$. For IBE_2, in a similar way, we compute, for each node i ($i = 1, 2, \ldots, n$) the minimum travel time τ_i of the arcs entering node i (avoiding to

use arc (j, i) if arc (i, j) is used), and proceed as for the computation of IBL_2. The improved lower bound IB_2 is then set as $\max\{IBL_2, IBE_2\}$. The validity of IB_2 relies on the fact that, in any Hamiltonian path, every node (but the depot) must be visited exactly once. Therefore, we need to reach each node by selecting only one of its leaving (entering, resp.) arcs. Since the linear service time function is non-decreasing, arriving at a node earlier gives a smaller service time than arriving there later. For this reason, we can consider the smallest travel time arc leaving (entering, resp.) every node.

Improved lower and upper bounds IB_3 on the service start time at each customer i $(i = 1, 2, \ldots, n)$ can be computed by considering that every node must be reached starting from the depot 0 and must reach depot $n + 1$. Therefore, for every customer i, the start time of service (which corresponds to the arrival time at the customer thanks to the FIFO property) cannot be smaller than the time corresponding to the shortest path from the depot 0 to i. In other words, $b_i \geq sp_{0,i}$, where $sp_{0,i}$ is the value of the shortest path from depot 0 to customer i, which also includes in the computation the service times at the visited nodes, except for the service time at i. In addition, in order to determine a solution not worse than that obtained by the GA algorithm, the start time of service at each customer i cannot be larger than the total route duration found by GA, i.e., $b_i \leq IB_1$. The latter upper bound can be further improved, by considering that the vehicle must go back to the depot, i.e., the depot $n + 1$ must be reached from every customer i. This gives the following upper bound on the service start time of customer i: $b_i \leq IB_1 - sp_{i,n+1}$, where $sp_{i,n+1}$ gives the value of the shortest path from i to $n+1$, by taking into account the service times at the visited nodes, including the service time at i. Note that, for the quadratic service time function, in the computation of the shortest path $sp_{i,n+1}$ we only consider the travel times.

Finally, when all customers have the same (quadratic or linear) service time function, as assumed in Taş et al. (2016), we propose an improvement of the value of M. Recall that, in constraints (2.4), the value of M is used to deactivate the constraint, when arc (i, j) is not selected. The value of M must be such that $b_j \geq -M + b_i + s_i(b_i) + t_{ij}$, when arc (i, j) is not in the solution, i.e., $M \geq b_i + s_i(b_i) + t_{ij} - b_j$. We consider the value IB_1 of the feasible solution obtained by the GA algorithm, which represents the total route duration, i.e., the arrival time at $n + 1$. However, this value is not necessarily enough to deactivate constraints (2.4), since arc (i, j) might not be selected in the GA solution and might be an arc with a "long" travel time t_{ij}. Therefore, the value of M is chosen as dependent on the specific arc (i, j), i.e., we use M_{ij} instead of M. We set $M_{0j} = t_{0j}$ for $j = 1, \ldots, n$, and $M_{ij} = IB_1 + t_{ij}$ for $i = 1, \ldots, n$, $j = 1, \ldots, n + 1$. Note that the replacement of M with the values M_{ij} for each arc (i, j) is correct: indeed, for arcs $(0, j)$ $(j = 1, \ldots, n)$, $M_{0j} = t_{0j} \geq b_0 + s_0(b_0) + t_{0j} - b_j$, since $b_0 = s_0(b_0) = 0$; for arcs (i, j) $(i = 1, \ldots, n, j = 1, \ldots, n + 1)$, $M_{ij} = IB_1 + t_{ij} \geq b_i + s_i(b_i) + t_{ij} - b_j$, since $b_i + s_i(b_i) \leq IB_1$ (as IB_1 is an upper bound on the total route duration).

2.4.2 Subtour Elimination Constraints (SECs)

In Taş et al. (2016), additional subtour elimination constraints, namely GG constraints, were used to improve the basic model performance. The authors also tried

other compact ATSP formulations with a polynomial number of subtour elimination constraints: the Miller, Tucker and Zemlin (MTZ) (Miller et al. (1960)) and the Desrochers and Laporte (DL) (Desrochers & Laporte (1991)) formulations. However, the best performance was achieved by considering the GG constraints. In this chapter, we propose to replace the GG constraints (2.7)-(2.8) with the explicit subtour elimination constraints (SECs) proposed in Dantzig et al. (1954) for the ATSP:

$$\sum_{i\in S}\sum_{j\in N\setminus S} x_{ij} \geq 1, \quad S \subseteq N\setminus\{n+1\}, \quad 0\in S, \quad |S|\geq 2. \tag{2.9}$$

These constraints, imposed for every subset S of nodes that contains at least two nodes, the depot 0, and does not contain depot $n+1$, require to select at least one arc going from S to $N\setminus S$. Since the number of constraints (2.9) is exponential in the number of nodes, we adopt a constraint separation procedure proposed in Padberg & Rinaldi (1990).

2.4.3 New Model (NM)

The new model (NM) described in this section is based on the formulation proposed in Maffioli & Sciomachen (1997) for the ATSP with Time Windows (ATSP-TW). For sake of clarity, we start by presenting this model, and then we show how to modify it to include the service time component.

For every arc $(i,j)\in A$, let x_{ij} be a binary variable assuming value 1 if arc (i,j) is selected in the optimal route (and 0 otherwise). For every customer $i\in N$, let $[r_i,d_i]$ be the corresponding time window. Moreover, for $i\in N\setminus\{0,n+1\}$ and $j\in N\setminus\{0\}$, $i\neq j$, let y_{ij} be an additional variable with the following meaning:

- if $x_{ij}=0$ then $y_{ij}=0$;
- if $x_{ij}=1$ then y_{ij} denotes the arrival time at node i when node j is visited immediately after node i.

The ATSP-TW model presented in Maffioli & Sciomachen (1997) reads as follows:

$$\min \sum_{i\in N}\sum_{j\in N} t_{ij}x_{ij} \tag{2.10}$$

$$\sum_{j\in N\setminus\{i\}} x_{ij} = 1, \quad i\in N\setminus\{n+1\}, \tag{2.11}$$

$$\sum_{i\in N\setminus\{j\}} x_{ij} = 1, \quad j\in N\setminus\{0\}, \tag{2.12}$$

$$r_i x_{ij} \leq y_{ij} \leq d_i x_{ij}, \quad i\in N\setminus\{0,n+1\}, j\in N\setminus\{0\}, i\neq j, \tag{2.13}$$

$$\sum_{i\in N\setminus\{0,n+1\},i\neq j} y_{ij} + \sum_{i\in N\setminus\{n+1\},i\neq j} t_{ij}x_{ij} \leq \sum_{k\in N\setminus\{0\},k\neq j} y_{jk}, \quad j\in N\setminus\{0,n+1\}, \tag{2.14}$$

$$x_{ij}\in\{0,1\}, \quad i\in N,\ j\in N, \tag{2.15}$$

$$y_{ij}\geq 0, \quad i\in N\setminus\{0,n+1\},\ j\in N\setminus\{0\},\ i\neq j. \tag{2.16}$$

The objective function (2.10) calls for minimizing the total travel times. Constraints (2.11) and (2.12) require, respectively, to have an outgoing arc from every node except for the depot $n+1$, and an ingoing arc for every node except for the depot 0. Time window constraints are imposed by (2.13), where the arrival time y_{ij} at node i (when j is visited immediately after i) is required to be within the time window $[r_i, d_i]$. These constraints are also used to set y_{ij} to 0 when the arc (i,j) is not selected in the optimal route. Constraints (2.14) are used to specify the arrival time y_{ij} at every customer $j \in N \setminus \{0, n+1\}$: when node j is visited immediately after node i, the arrival time y_{jk} at customer j, no matter which node k is visited afterwards (including depot $n+1$), must be at least the arrival time y_{ij} at node i, i.e., any predecessor of customer j (except for the depot), plus the travel time t_{ij} from node i to node j. Finally, constraints (2.15) and (2.16) define the variable domains.

The new model (NM) that we propose uses, in addition to the x_{ij} and y_{ij} variables, the b_i variables defined in model (2.1)-(2.6), representing the start time of service at node $i \in N$. We first present NM for the case of linear service time function $s_i(b_i) = \beta b_i + \gamma$, where β, $\gamma > 0$, and $s_i(b_i) > 0$ ($i \in N \setminus \{0, n+1\}$), and then extend it to a quadratic service time function. Model (2.10)-(2.16) is modified to deal with time-dependent service times, as follows:

$$\min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} + \sum_{i \in N \setminus \{0,n+1\}} s_i(b_i) \tag{2.17}$$

$$\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N \setminus \{n+1\}, \tag{2.18}$$

$$\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{2.19}$$

$$b_i = \sum_{k \in N \setminus \{0\}, k \neq i} y_{ik}, \quad i \in N \setminus \{0, n+1\}, \tag{2.20}$$

$$y_{ij} \geq LB_3(i) x_{ij}, \quad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\}, i \neq j, \tag{2.21}$$

$$y_{ij} + \beta y_{ij} + \gamma x_{ij} \leq M x_{ij} \qquad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\},\ i \neq j, \tag{2.22}$$

$$\sum_{i \in N \setminus \{0,n+1\}, i \neq j} y_{ij} + \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \beta y_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} \gamma x_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} t_{ij} x_{ij} \leq \sum_{k \in N \setminus \{0\}, k \neq j} y_{jk}, \quad j \in N \setminus \{0, n+1\}, \tag{2.23}$$

$$b_i \geq 0, \quad i \in N, \tag{2.24}$$

$$x_{ij} \in \{0,1\}, \quad i \in N,\ j \in N, \tag{2.25}$$

$$y_{ij} \geq 0, \quad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\},\ i \neq j. \tag{2.26}$$

The objective function (2.17) is the same as (2.1), and minimizes the total route duration given by the sum of the total travel and service times. As in the previous models, constraints (2.18) and (2.19) impose to select an outgoing arc from every node except for the depot $n+1$ and an ingoing arc for every node except for the depot 0. Constraints (2.20) are used to define the values of the

b_i variables: the arrival time b_i at i can be expressed as the arrival time y_{ik}, no matter which successor node k is chosen. Constraints (2.21) and (2.22), in which lower and upper bounds on the arrival times at the customers are used, replace the time window constraints (2.13). In particular, we use the bounds B_3 proposed in Taş et al. (2016). Note that constraints (2.21) and (2.22) ensure that y_{ij} is set to 0 if x_{ij} is also 0. Constraints (2.23) correspond to (2.14) but also include the service time at the customers: for every customer j, when node j is visited immediately after node i, the arrival time y_{jk} at j (no matter which successor k is selected) must be at least the arrival time y_{ij} at its predecessor i (no matter which it is) plus the service time at i (given by $\beta b_i + \gamma$), plus the travel time t_{ij} between i and j. Constraints (2.24)-(2.26) restrict the variable domain.

In order to improve NM, we can insert the improved bounds IB_3 in constraints (2.21) and (2.22) as follows:

$$y_{ij} \geq sp_{0,i}x_{ij}, \quad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\}, i \neq j, \tag{2.27}$$

$$y_{ij} + \beta y_{ij} + \gamma x_{ij} \leq (IB_1 - sp_{i,n+1})x_{ij} \qquad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\},\ i \neq j. \tag{2.28}$$

In particular, the arrival time y_{ij} at node i, when j is its successor, must be at least the value of the shortest path from the depot 0 to i, if arc (i, j) is selected. In addition, the arrival time y_{ij} at node i, when j is its successor, plus the service time $s_i(b_i)$, which corresponds to $\beta b_i + \gamma$, must be at most the improved upper bound IB_1 on the total route duration minus the value of the shortest path from i to the depot $n + 1$, when arc (i, j) is selected.

If we consider a quadratic service time function defined as $s_i(b_i) = \alpha b_i^2 - \beta b_i + \gamma$, as done in Taş et al. (2016), constraints (2.22) and (2.23) are changed as follows.

$$y_{ij} + \alpha y_{ij}^2 - \beta y_{ij} + \gamma x_{ij} \leq M x_{ij} \qquad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\},\ i \neq j, \tag{2.29}$$

$$\sum_{i \in N \setminus \{0,n+1\}, i \neq j} y_{ij} + \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \alpha y_{ij}^2 - \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \beta y_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} \gamma x_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} t_{ij} x_{ij} \leq \sum_{k \in N \setminus \{0\}, k \neq j} y_{jk}, \quad j \in N \setminus \{0, n+1\}. \tag{2.30}$$

A similar improvement as in the case of the linear service time functions can be applied for the quadratic case by replacing constraints (2.29) with the following ones:

$$y_{ij} + \alpha y_{ij}^2 - \beta y_{ij} + \gamma x_{ij} \leq (IB_1 - sp_{i,n+1})x_{ij} \qquad i \in N \setminus \{0, n+1\},\ j \in N \setminus \{0\},\ i \neq j. \tag{2.31}$$

In the following, we will call NMB the model corresponding to (2.17)-(2.26), for the linear service time functions, and to (2.17)-(2.21), (2.24)-(2.26), (2.29)-(2.30), for the quadratic service time function. We will call NMI the improved model corresponding to (2.17)-(2.20), (2.23)-(2.26), (2.27)-(2.28), for the linear service

time functions, and to (2.17)-(2.20), (2.24)-(2.26), (2.27), (2.30)-(2.31), for the quadratic service time function.

2.5 Solution Methods

To solve the TSP-TS we propose the metaheuristic algorithm GA, presented in Section 2.5.1, that provides high quality solutions in very short computing times (see Section 2.6.1), and two exact B&C algorithms, described in Section 2.5.2, that use all the improved bounds (for the quadratic service times, IB_2 is not used), the improved value of M and the SECs separation. The B&C algorithms are applied using the new model NMI and, for comparison, the basic model.

2.5.1 Genetic Algorithm

Genetic algorithms are effective metaheuristic algorithms, that belong to the class of evolutionary algorithms, and are based on the evolution of a population of individuals, corresponding to solutions of the considered problem. In the proposed GA, each individual corresponds to a Hamiltonian path from depot 0 to depot $n+1$, and we represent it as a permutation of nodes in $\{1, \ldots, n\}$.

An initial population is computed by applying three algorithms, each one generating a given percentage of the population: (i) an algorithm that generates a random route (25% of the initial population): in particular, starting from the depot 0, we choose a random node, and then, from the selected node, we select the next node in a random way among the non-visited ones, and so on, until a Hamiltonian path is built; (ii) the NNH algorithm with each individual generated starting from a different randomly chosen node in N (50% of the initial population); (iii) a continuous relaxation based algorithm, that uses the optimal continuous relaxation solution x of the basic model to build a feasible Hamiltonian path (25% of the initial population). In particular, one individual of the population is obtained in a deterministic way as follows: we define the depot 0 as the starting node h, and iteratively select the node j such that $x_{hj} + x_{jh}$ has the highest value; arc (h, j) is added to the route, and j becomes the new starting node h. If, for all nodes j connected to h, $x_{hj} + x_{jh} = 0$, then we choose the arc with the smallest t_{hj}. The procedure is repeated until a complete feasible Hamiltonian path has been obtained. Then, the remaining individuals are obtained by adding randomness in the construction of the route: we start from the depot 0, defined as node h, and select the next node j according to a probability given by $x_{hj} + x_{jh} \in [0, 2]$. The selected node j becomes the new starting node h, and the procedure is repeated until a Hamiltonian path has been built. We propose two variants for the continuous relaxation based algorithm, which either consider or not the SECs separation during the solution of the relaxed model. A comparison of the two variants is reported in Section 2.6.1.

We consider many crossover and mutation operators, listed in the following. We omit their description and refer the interested reader to the corresponding reference: Order Based Crossover (OX2) (Syswerda (1991)), Distance Preserving Crossover (DPX)(Reisleben et al. (1996)), Maximal Preservative Crossover (MPX)

(Mühlenbein (1991)), Heuristic Crossover (HX) (Grefenstette et al. (1985)), Modified Inver-over Operator (MIO) (Wang et al. (2012)), Uniform Nearest Neighbor (UNN) (Buriol et al. (2004)), Exchange Mutation (EM) (Banzhaf (1990)), Scramble mutation (SM) (Syswerda (1991)), Inversion mutation (IVM) (Fogel (1993)), Insertion Mutation (ISM) (Fogel (1988)), Greedy Sub-Tour Mutation (GSTM) (Albayrak & Allahverdi (2011)) and 3-opt (we use a simplified 3-opt version, which considers all pairs of arcs and selects the third arc randomly out of ten arcs).

We performed a calibration of the probabilities to use for each crossover and mutation operator, by using the algorithm proposed in Contreras-Bolton & Parada (2015), which considers many alternative operators and finds a good combination of them. In Contreras-Bolton & Parada (2015), the authors studied how to select appropriate combinations of crossover and mutation operators, typically used for the TSP, and applied evolutionary computing to determine the best probability for each operator. The algorithm proposed in Contreras-Bolton & Parada (2015) was calibrated on a subset of the TSP-TS instances considered in Section 3.5 (three symmetric ones and three asymmetric ones) and by considering the small linear service time function $s_i = 5(10^{-3})b_i + 3(10^{-2})$ ($i \in N \setminus \{0, n+1\}$). Then, the same probabilities are used in all the computational experiments.

We consider an initial population of 150 individuals, that can contain duplicated individuals. Each individual is evaluated (by starting from depot 0), according to the objective function (2.1), that represents the total route duration. An iterative loop is then executed to evolve the population, until a terminating condition is met (in our computational experiments, 200 iterations are performed). At each iteration, the following steps are executed:

1. *Selection* is used to select an individual in the population. It is a tournament selection based on three individuals. In particular, three individuals are randomly chosen from the current population and the best one is selected.

2. *Crossover* is used to combine two individuals of the population. Once that two individuals have been selected (according to the tournament selection), one of the crossover operators, according to the probabilities reported in Table 2.1, is applied. The crossover operator is applied on the two individuals (parents) with 85% probability.

3. *Mutation* is used to modify an individual in order to add diversity to the population and to ensure the exploration of a large solution space. It is applied on the offspring generated at step 2, with 20% probability. One of the mutation operators, according to the probabilities shown in Table 2.1, is selected and applied to obtain a new individual.

4. *Evaluation* is applied to compute the quality of the obtained solutions: the offspring generated by crossover and mutation are evaluated, according to the objective function (2.1).

5. *Elitism* is applied as a last step: the offspring population created by selection, crossover and mutation replaces the original parental population, but the 10% worst offspring are replaced by the best parents in the current population.

Despite its simplicity, GA is able to find, in very short computing times, solutions of the TSP-TS with small percentage gaps with respect to the best known solution values, as shown in Section 2.6.1.

i	Crossover	Prob.	Mutation	Prob.
1	OX2	0.010	EM	0.100
2	DPX	0.400	SM	0.009
3	MPX	0.076	IVM	0.008
4	HX	0.214	ISM	0.018
5	MIO	0.100	GSTM	0.425
6	UNN	0.200	3-opt	0.440

Table 2.1: Probabilities of using each crossover or mutation operator.

2.5.2 Branch-and-Cut and Dynamic Branch-and-Cut Algorithms

We propose a B&C algorithm, in which the SECs are separated at every node of the decision tree by using the separation procedure proposed in Padberg & Rinaldi (1990) until no violated constraints exist. The B&C is based on the new model NMI, in which all the improved bounds IB_1, IB_2 and IB_3, as well as the improved value of M, are used. Note that, since IB_2 is only valid for linear service time functions, as explained in Section 2.4.1, we do not apply it on instances with a quadratic service time function. The GA algorithm is used to define the value of IB_1, and for the upper bound on the start time of service in IB_3. In the B&C algorithm, we apply the branching rules chosen by default in the CPLEX solver, and use the callback functions to separate the SECs.

Beside the B&C algorithm, we also propose a Dynamic B&C algorithm, in which the improved bounds IB_2 and IB_3 are dynamically updated during the solution process in order to take into account the branching decisions. More precisely, the two bounds are recomputed at every node of the decision tree, by considering that some arcs (i, j) have been selected in the solution ($x_{ij} = 1$) and must be chosen in the bound computation, and other arcs have been discarded ($x_{ij} = 0$) and must not be used in the bound computation. In this way, better bounds can be obtained during the exploration of the decision tree. The improved bound IB_1 is not updated, since the starting value computed at the root node is already close to the value of the best known solution, and the computing time of GA, although small, is not negligible. In IB_2, the smallest leaving (entering, resp.) arc for every node i is selected only among the allowed ones: there is no choice if variable x_{ij} is fixed to 1, and the arc is chosen only among those not set to 0. In IB_3, the shortest path computation for $sp_{0,i}$ and $sp_{i,n+1}$ takes into account the available arcs.

The B&C and the Dynamic B&C algorithms can also be applied on the basic model (2.1)-(2.6), whose continuous relaxation is strengthened by the SECs, by including all the improved bounds and the improved value of M. This alternative method will be considered in the computational experiments for comparison.

2.6 Computational Results

All the models and algorithms were implemented in C++. All the experiments were executed on an Intel(R) Core(TM) i7-6900K with 3.20 GHz and 64 GB of RAM (with a single thread), using GNU/Linux Ubuntu 16.04, and CPLEX 12.7.1 was used as solver of the MIP models and of their continuous relaxations. In the computational experiments, we considered the 22 symmetric instances introduced in Taş et al. (2016), adapted from the TSPLIB (Reinelt (1991)), and new asymmetric and larger size symmetric instances, also adapted from the TSPLIB. The 22 symmetric instances contain up to 45 nodes (and an additional node for the ending depot). In Taş et al. (2016), the original travel times were modified in order to have the same average travel time per arc in each instance. The new instances correspond to all the symmetric (resp. asymmetric) instances contained in the TSPLIB with up to 58 (resp. 45) nodes. The travel times are adjusted according to the same rule used in Taş et al. (2016).

For comparison with Taş et al. (2016) we consider the same linear and quadratic service time functions, defined as follows:

- small service times: $s_i = 5(10^{-3})b_i + 3(10^{-2})$ $(i \in N \setminus \{0, n+1\})$;
- medium service times: $s_i = 10^{-2}b_i + 6(10^{-2})$ $(i \in N \setminus \{0, n+1\})$;
- large service times: $s_i = 2(10^{-2})b_i + 1.2(10^{-1})$ $(i \in N \setminus \{0, n+1\})$;
- quadratic service times: $s_i = 4(10^{-5})b_i^2 - 4(10^{-3})b_i + 3(10^{-1})$ $(i \in N \setminus \{0, n+1\})$.

We first report, in Section 2.6.1, the computational results of the GA algorithm on the 22 symmetric instances introduced in Taş et al. (2016). Then, in Section 2.6.2, we report the comparison, in terms of the lower bounds of the continuous relaxation (Section 2.6.2) and of the integer solutions (Section 2.6.2), of the proposed methods with the best method (TGJL16) presented in Taş et al. (2016), by considering the 22 symmetric instances from Taş et al. (2016). Finally, in Section 2.6.3, we report the results obtained on the larger size symmetric instances and on the asymmetric instances.

For the B&C and Dynamic B&C algorithms, as well as for TGJL16, we set a time limit of 7200 seconds for the 22 symmetric instances from Taş et al. (2016), and of 50000 seconds for the new instances. To have a fair comparison with TGJL16, we implemented the corresponding model (which includes the GG constraints and the bounds B_1, B_2 and B_3) and run the CPLEX solver on the same computer.

2.6.1 Genetic Algorithm

We report the results obtained by the GA algorithm on the 22 symmetric instances from Taş et al. (2016) with small and medium service times in Table 2.2, and with large and quadratic service times in Table 2.3. The GA algorithm was run 10 times on each instance. In each table, we show the instance name, in which the number indicates the number of nodes (i.e., $n+1$) in the corresponding graph, and the best

known solution value (obtained by TGJL16 or by the proposed B&C and Dynamic B&C algorithms). Then, we report for the considered service time function (small and medium in Table 2.2, large and quadratic in Table 2.3), the results obtained by the GA algorithm in two variants: the first one does not consider the separation of the SECs in the solution of the continuous relaxation, used to generate a subset of the initial population, while the second one includes it. For each variant, we show, for each instance, the average and the minimum percentage gaps (computed w.r.t. the best solution value), obtained over the 10 runs, and the average computing time (expressed in seconds) over the 10 runs. Note that the computing time to obtain the minimum percentage gap is ten times the average computing time. In the last two rows, we show the averages over all instances, of the values in the corresponding columns, and the number of best known solutions found.

As we can observe, the GA algorithm finds the best known solution for all instances except one (when the large service time function is considered). The computing time is very short in all cases, being at most 1.7 seconds, on average, for the quadratic service time function with SECs. The average percentage gap over 10 runs when the SECs separation is included is always not worse than when it is neglected. The minimum percentage gap over 10 runs when the SECs separation is taken into account is not worse than when it is neglected, with the exception of one instance in the case of the small service time function. We can also observe that the largest gap among all instances w.r.t. the best known solution value is 0.966% (for instance dantzig42 and medium service time). In the computation of IB_1 and IB_3 we decided to use the minimum cost solutions obtained out of 5 runs to limit the computing times.

2.6.2 Comparison with TGJL16

This section is devoted to the comparison of the results obtained by the proposed exact algorithms with the results obtained by the best method (TGJL16) presented in Taş et al. (2016). We show in Section 2.6.2 the comparison of the lower bounds obtained by solving the continuous relaxation of the best model proposed in Taş et al. (2016) with those obtained by solving the continuous relaxation of the new model NMI, enhanced with improved bounds and SECs separation. In addition, we show in Section 2.6.2 the comparison of the upper bounds obtained by TGJL16 and by the proposed B&C and Dynamic B&C algorithms. The times required for computing M, and the lower and upper bounds are included in the total computing times shown in the following tables, both for TGJL16 and the proposed methods.

Lower Bounds

We present, in Tables 2.4, 2.5, 2.6 and 2.7, the comparison of the lower bounds obtained by TGJL16 and by the proposed model with or without the improvements of the lower and upper bounds and the SECs separation, by considering small, medium, large or quadratic service time functions. In particular, in each table, we report the results obtained by the following models:

- TGJL16: continuous relaxation of the basic model (2.1)-(2.6), enhanced with the GG constraints (2.7)-(2.8), with the computation of the M value, and

Table 2.2: GA on instances from Taş et al. (2016) with small and medium service times.

# inst	Small service times							Medium service times						
	Best	without SECs			with SECs			Best	without SECs			with SECs		
		Avg%	Min%	Time	Avg%	Min%	Time		Avg%	Min%	Time	Avg%	Min%	Time
burma14	228.83	0.000	0.000	0.129	0.000	0.000	0.133	236.44	0.000	0.000	0.103	0.000	0.000	0.105
ulysses16	271.74	0.000	0.000	0.128	0.064	0.000	0.131	279.57	0.000	0.000	0.130	0.023	0.000	0.129
gr17	238.39	0.000	0.000	0.149	0.000	0.000	0.154	245.40	0.000	0.000	0.149	0.000	0.000	0.154
gr21	237.11	0.000	0.000	0.212	0.000	0.000	0.218	249.32	0.000	0.000	0.214	0.000	0.000	0.216
ulysses22	306.44	0.005	0.000	0.229	0.004	0.000	0.234	318.06	0.164	0.000	0.223	0.382	0.000	0.232
gr24	269.09	0.000	0.000	0.274	0.000	0.000	0.297	284.93	0.000	0.000	0.272	0.000	0.000	0.288
fri26	247.99	0.000	0.000	0.333	0.000	0.000	0.334	263.01	0.000	0.000	0.326	0.000	0.000	0.338
bayg29	345.49	0.303	0.000	0.413	0.050	0.000	0.428	371.22	0.225	0.000	0.411	0.000	0.000	0.431
bays29	309.27	0.256	0.000	0.412	0.158	0.000	0.422	331.90	0.198	0.000	0.412	0.152	0.000	0.421
att30	253.85	0.000	0.000	0.440	0.000	0.000	0.458	273.10	0.000	0.000	0.439	0.000	0.000	0.454
dantzig30	324.21	0.537	0.000	0.434	0.212	0.000	0.449	349.60	0.301	0.301	0.430	0.091	0.000	0.449
eil30	323.40	0.384	0.000	0.445	0.247	0.000	0.456	349.16	0.338	0.000	0.443	0.169	0.000	0.454
gr30	283.91	0.000	0.000	0.459	0.000	0.000	0.476	305.23	0.106	0.000	0.448	0.000	0.000	0.463
hk30	324.20	0.503	0.000	0.457	0.000	0.000	0.478	347.35	0.625	0.000	0.454	0.000	0.000	0.470
swiss30	342.50	0.000	0.000	0.463	0.000	0.000	0.469	366.78	0.000	0.000	0.460	0.000	0.000	0.469
eil35	363.39	0.340	0.000	0.654	0.112	0.000	0.684	397.42	0.337	0.000	0.641	0.155	0.000	0.680
gr35	281.82	0.000	0.000	0.680	0.000	0.000	0.740	306.91	0.000	0.000	0.678	0.000	0.000	0.752
swiss35	373.60	0.000	0.000	0.709	0.000	0.000	0.727	406.92	0.000	0.000	0.699	0.000	0.000	0.725
eil40	410.35	0.183	0.000	0.911	0.023	0.000	0.963	452.89	0.327	0.000	0.896	0.164	0.000	0.947
dantzig42	257.37	0.326	0.000	1.021	0.590	0.000	1.078	285.07	0.722	0.000	0.995	0.966	0.000	1.063
swiss42	351.15	0.029	0.000	1.044	0.027	0.000	1.066	388.64	0.589	0.000	0.993	0.074	0.000	1.052
eil45	448.11	0.034	0.000	1.241	0.069	0.036	1.299	502.52	0.156	0.000	1.218	0.248	0.000	1.296
Avg.	308.74	0.132	0.000	0.511	0.071	0.002	0.532	332.34	0.186	0.014	0.501	0.110	0.000	0.527
#Best		11	22		11	21			10	21		12	22	

Table 2.3: GA on instances from Taş et al. (2016) with large and quadratic service times.

# inst	Large service times							Quadratic service times						
	Best	without SECs			with SECs			Best	without SECs			with SECs		
		Avg%	Min%	Time	Avg%	Min%	Time		Avg%	Min%	Time	Avg%	Min%	Time
burma14	252.62	0.000	0.000	0.102	0.000	0.000	0.105	224.83	0.000	0.000	0.126	0.000	0.000	0.187
ulysses16	296.28	0.000	0.000	0.127	0.000	0.000	0.129	268.14	0.025	0.000	0.153	0.025	0.000	0.303
gr17	260.34	0.000	0.000	0.149	0.000	0.000	0.154	234.82	0.000	0.000	0.178	0.000	0.000	0.367
gr21	275.96	0.000	0.000	0.213	0.000	0.000	0.217	232.77	0.000	0.000	0.269	0.000	0.000	0.471
ulysses22	343.58	0.000	0.000	0.227	0.000	0.000	0.227	301.58	0.000	0.000	0.277	0.174	0.000	0.714
gr24	320.42	0.034	0.000	0.273	0.000	0.000	0.290	263.04	0.000	0.000	0.354	0.000	0.000	1.014
fri26	297.39	0.000	0.000	0.324	0.000	0.000	0.338	239.08	0.000	0.000	0.412	0.000	0.000	0.898
bayg29	430.35	0.136	0.000	0.404	0.054	0.000	0.428	345.11	0.000	0.000	0.511	0.000	0.000	1.386
bays29	383.78	0.161	0.000	0.413	0.161	0.000	0.426	305.46	0.236	0.000	0.517	0.235	0.000	1.241
att30	316.51	0.000	0.000	0.431	0.000	0.000	0.450	246.98	0.000	0.000	0.557	0.000	0.000	1.648
dantzig30	404.54	0.727	0.692	0.412	0.566	0.000	0.440	321.89	0.554	0.554	0.565	0.218	0.000	1.955
eil30	408.23	0.211	0.000	0.440	0.145	0.000	0.438	320.74	0.403	0.000	0.550	0.293	0.000	1.334
gr30	353.89	0.000	0.000	0.447	0.000	0.000	0.471	279.94	0.000	0.000	0.579	0.000	0.000	1.349
hk30	400.88	0.746	0.000	0.430	0.000	0.000	0.441	318.63	0.667	0.000	0.580	0.000	0.000	1.835
swiss30	422.54	0.000	0.000	0.461	0.000	0.000	0.466	340.42	0.000	0.000	0.577	0.000	0.000	1.247
eil35	474.90	0.218	0.000	0.527	0.238	0.197	0.589	365.34	0.319	0.000	0.799	0.089	0.000	2.153
gr35	365.75	0.000	0.000	0.669	0.000	0.000	0.700	276.18	0.000	0.000	0.846	0.000	0.000	2.281
swiss35	485.44	0.000	0.000	0.694	0.000	0.000	0.721	378.36	0.000	0.000	0.876	0.000	0.000	2.150
eil40	556.10	0.073	0.000	0.745	0.069	0.000	0.800	421.10	0.073	0.000	1.122	0.103	0.000	3.030
dantzig42	352.36	0.418	0.000	0.926	0.000	0.000	1.028	247.25	0.090	0.000	1.248	0.668	0.000	4.538
swiss42	480.30	0.117	0.000	0.904	0.051	0.000	1.035	350.45	0.263	0.000	1.270	0.030	0.000	3.501
eil45	638.13	0.090	0.000	1.188	0.073	0.000	1.280	474.44	0.117	0.000	1.473	0.271	0.000	4.089
Avg.	387.29	0.133	0.031	0.478	0.062	0.009	0.508	307.12	0.125	0.025	0.629	0.096	0.000	1.713
#Best		11	21		14	21				12	21	12	22	

of the bounds B_1, B_2 (only for the linear service time functions) and B_3, described in Section 2.3.1;

- NMB+Bs: continuous relaxation of the model (2.17)-(2.26), for the linear service time functions, and of the model (2.17)-(2.21), (2.24)-(2.26), (2.29)-(2.30), for the quadratic service time function, with the computation of the M value, and of the bounds B_1, B_2 (only for the linear service time functions) and B_3, described in Section 2.3.1;

- NMI+IBs: continuous relaxation of the model NMI (corresponding to the model (2.17)-(2.20), (2.23)-(2.26), (2.27)-(2.28), for the linear service time functions, and to the model (2.17)-(2.20), (2.24)-(2.26), (2.27), (2.30)-(2.31), for the quadratic service time function) including in the model the improved bounds IB_1, IB_2 (only for linear service time functions) and IB_3, described in Section 2.4.1;

- NMI+IBs+SECs: NMI+IBs, combined with the separation of the SECs (described in Section 2.4.2);

- basic+IBs+SECs: continuous relaxation of model (2.1)-(2.6), with the computation of the improved M value described in Section 2.4.1, including in the model the improved bounds IB_1, IB_2 (only for linear service time functions) and IB_3, described in Section 2.4.1, and the separation of the SECs (described in Section 2.4.2).

In each table, we report, for each instance, the instance name and the best known solution value. Then, for each considered model, we report the lower bound and the corresponding computing time (expressed in seconds). In the last row, we display the averages of the values in the corresponding columns.

By looking at the results, we can see that the lower bounds obtained by NMB+Bs are worse than those obtained by TGJL16, since the model NMB is used without any enhancement. When the improved bounds are inserted, the lower bounds increase significantly and become, on average, better than those of TGJL16 for all the service time functions. It is evident that using the exponentially many SECs, which are dynamically separated at the root node, gives another considerable improvement, although it comes at the expenses of larger computing times. However, the computing times are still short, with the exception of the quadratic service time function, which makes the TSP-TS problem harder to solve. We can also observe that if we use the basic model instead of NMI, and include all the proposed improvements, the obtained lower bounds are slightly worse, although the computing times are larger for NMI than for the basic model. However, as it will be shown in Section 2.6.2, both the B&C and the Dynamic B&C algorithms have a better performance when model NMI is used. Finally, we can also see that the lower bounds obtained by NMI+IBs+SECs are rather close to the best known solution values. The case of quadratic service time function shows the largest gap between the best known solution values and the lower bounds, especially because IB_2 cannot be applied in this case. We can conclude that both the improved bounds and the separation of the exponentially many SECs help to find better lower bounds in all cases.

Table 2.4: Comparison of Lower Bounds with small service times on instances from Taş et al. (2016).

# inst	Best	TGJL16		NMB+Bs		NMI+IBs		NMI+IBs+SECs		basic+IBs+SECs	
		LB	Time	LB	Time	LB	Time	LB	Time	LB	Time
burma14	228.83	204.04	0.07	190.25	0.01	206.02	0.05	226.28	0.05	225.65	0.04
ulysses16	271.74	235.39	0.00	221.70	0.01	237.97	0.05	268.66	0.05	267.83	0.04
gr17	238.39	198.86	0.00	189.73	0.01	207.35	0.04	237.29	0.05	236.27	0.03
gr21	237.11	213.89	0.01	208.98	0.01	216.81	0.07	233.48	0.06	232.75	0.04
ulysses22	306.44	247.10	0.00	227.90	0.02	251.49	0.05	299.86	0.07	297.53	0.04
gr24	269.09	234.59	0.01	222.27	0.02	235.55	0.08	265.12	0.20	265.12	0.04
fri26	247.99	220.17	0.01	216.87	0.02	221.74	0.12	244.59	0.18	244.59	0.05
bayg29	345.49	315.78	0.02	307.47	0.02	314.93	0.14	339.71	0.23	339.71	0.05
bays29	309.27	277.44	0.02	268.42	0.02	276.65	0.15	302.66	0.21	302.66	0.06
att30	253.85	194.16	0.01	178.06	0.02	191.55	0.12	246.30	0.16	245.95	0.06
dantzig30	324.21	271.53	0.01	232.61	0.02	271.68	0.16	316.13	0.13	313.39	0.06
eil30	323.40	295.20	0.02	291.92	0.02	295.30	0.16	317.34	0.31	317.34	0.06
gr30	283.91	238.55	0.02	231.10	0.02	240.07	0.20	276.92	0.19	276.44	0.06
hk30	324.20	272.37	0.01	258.26	0.02	269.43	0.14	317.09	0.19	316.70	0.06
swiss30	342.50	308.29	0.02	302.84	0.02	309.66	0.16	335.10	0.18	334.52	0.06
eil35	363.39	329.39	0.04	325.34	0.03	329.16	0.23	352.72	0.70	352.72	0.09
gr35	281.82	229.32	0.03	219.34	0.03	229.53	0.20	273.48	0.31	273.07	0.09
swiss35	373.60	318.26	0.04	314.53	0.03	319.18	0.23	363.35	0.35	363.17	0.08
eil40	410.35	365.76	0.05	359.75	0.06	365.11	0.30	398.99	0.97	398.99	0.11
dantzig42	257.37	213.92	0.03	193.50	0.05	214.08	0.41	248.58	0.50	248.06	0.12
swiss42	351.15	280.76	0.03	273.90	0.05	282.84	0.33	340.61	0.92	340.61	0.13
eil45	448.11	397.54	0.06	389.35	0.09	396.72	0.45	435.40	1.81	435.40	0.15
Avg.	308.74	266.47	0.02	255.64	0.03	267.40	0.17	301.80	0.36	301.29	0.07

Table 2.5: Comparison of Lower Bounds with medium service times on instances from Taş et al. (2016).

# inst	Best	TGJL16		NMB+Bs		NMI+IBs		NMI+IBs+SECs		basic+IBs+SECs	
		LB	Time	LB	Time	LB	Time	LB	Time	LB	Time
burma14	236.44	207.89	0.01	193.98	0.01	210.33	0.05	231.08	0.05	229.91	0.04
ulysses16	279.57	239.02	0.01	225.21	0.01	242.11	0.04	272.98	0.05	272.00	0.03
gr17	245.40	202.78	0.01	193.55	0.01	212.76	0.04	242.89	0.05	241.07	0.03
gr21	249.32	220.70	0.01	215.73	0.01	223.48	0.06	240.92	0.06	240.33	0.04
ulysses22	318.06	252.18	0.01	232.83	0.02	257.26	0.05	305.53	0.07	303.14	0.04
gr24	284.93	245.19	0.02	232.77	0.02	246.72	0.09	276.64	0.13	276.64	0.04
fri26	263.01	229.06	0.02	225.74	0.02	232.89	0.10	255.83	0.17	255.83	0.05
bayg29	371.22	335.54	0.02	327.16	0.03	334.48	0.13	359.48	0.21	359.48	0.05
bays29	331.90	293.82	0.02	284.71	0.02	292.77	0.15	319.04	0.18	319.04	0.06
att30	273.10	204.06	0.01	187.90	0.02	201.68	0.15	256.90	0.17	256.66	0.06
dantzig30	349.60	285.16	0.01	246.05	0.01	284.92	0.16	330.72	0.14	327.90	0.06
eil30	349.16	316.39	0.02	313.07	0.02	316.31	0.13	338.53	0.26	338.53	0.06
gr30	305.23	250.89	0.02	243.37	0.02	252.78	0.13	289.96	0.20	289.56	0.06
hk30	347.35	286.35	0.02	272.15	0.02	283.90	0.15	331.72	0.17	331.46	0.06
swiss30	366.78	321.90	0.01	316.40	0.01	324.80	0.13	350.37	0.18	349.89	0.06
eil35	397.42	356.91	0.03	352.82	0.04	356.56	0.21	380.26	0.71	380.26	0.09
gr35	306.91	244.39	0.03	234.33	0.03	244.20	0.20	288.72	0.41	288.39	0.09
swiss35	406.92	337.99	0.03	334.23	0.03	340.45	0.24	384.76	0.38	384.67	0.08
eil40	452.89	400.83	0.05	394.77	0.06	400.00	0.38	434.06	0.81	434.06	0.10
dantzig42	285.07	230.60	0.03	210.07	0.05	230.67	0.45	266.17	0.51	265.77	0.12
swiss42	388.64	304.39	0.03	297.46	0.04	307.74	0.32	366.10	0.96	366.10	0.12
eil45	502.52	441.73	0.06	433.45	0.10	440.49	0.44	479.59	1.59	479.59	0.15
Avg.	332.34	282.17	0.02	271.26	0.03	283.51	0.17	318.28	0.34	317.74	0.07

Table 2.6: Comparison of Lower Bounds with large service times on instances from Taş et al. (2016).

# inst	Best	TGJL16		NMB+Bs		NMI+IBs		NMI+IBs+SECs		basic+IBs+SECs	
		LB	Time	LB	Time	LB	Time	LB	Time	LB	Time
burma14	252.62	216.01	0.01	201.89	0.00	218.56	0.05	240.11	0.05	238.86	0.04
ulysses16	296.28	246.74	0.01	232.71	0.00	250.69	0.04	281.55	0.05	280.82	0.03
gr17	260.34	211.21	0.01	201.81	0.00	223.01	0.04	253.88	0.05	251.31	0.03
gr21	275.96	235.57	0.01	230.53	0.01	239.67	0.06	257.23	0.06	256.78	0.04
ulysses22	343.58	263.27	0.01	243.66	0.01	268.09	0.05	317.25	0.08	315.30	0.04
gr24	320.42	268.84	0.01	256.22	0.01	271.69	0.09	302.38	0.13	302.38	0.05
fri26	297.39	248.99	0.01	245.62	0.01	258.26	0.11	281.36	0.17	281.36	0.05
bayg29	430.35	380.70	0.02	372.16	0.02	379.12	0.14	404.63	0.22	404.63	0.05
bays29	383.78	331.23	0.02	321.95	0.02	329.66	0.14	356.44	0.23	356.44	0.06
att30	316.51	226.76	0.01	210.46	0.01	225.01	0.14	281.20	0.18	281.09	0.06
dantzig30	404.54	316.18	0.01	276.68	0.02	313.91	0.13	362.62	0.14	360.77	0.06
eil30	408.23	365.14	0.02	361.74	0.02	364.70	0.14	387.28	0.30	387.28	0.06
gr30	353.89	279.03	0.02	271.39	0.02	281.24	0.12	319.57	0.23	319.29	0.05
hk30	400.88	318.33	0.02	303.98	0.02	317.08	0.14	365.34	0.16	365.22	0.06
swiss30	422.54	352.92	0.01	347.34	0.01	359.11	0.12	384.96	0.17	384.66	0.06
eil35	474.90	421.85	0.04	417.68	0.03	421.23	0.25	445.25	0.54	445.25	0.09
gr35	365.75	279.71	0.03	269.49	0.02	278.94	0.21	324.30	0.48	324.09	0.08
swiss35	485.44	384.08	0.03	380.27	0.02	389.97	0.30	434.59	0.39	434.59	0.09
eil40	556.10	485.75	0.07	479.58	0.06	484.57	0.37	518.99	0.97	518.99	0.12
dantzig42	352.36	271.00	0.03	250.27	0.04	271.20	0.36	308.58	0.46	308.41	0.11
swiss42	480.30	361.62	0.04	354.55	0.04	368.08	0.38	427.52	0.96	427.52	0.12
eil45	638.13	551.55	0.07	543.11	0.07	549.56	0.44	589.41	1.74	589.41	0.15
Avg.	387.29	318.93	0.02	307.87	0.02	321.06	0.17	356.57	0.35	356.11	0.07

Table 2.7: Comparison of Lower Bounds with quadratic service times on instances from Taş et al. (2016).

# inst	Best	TGJL16		NMB+Bs		NMI+IBs		NMI+IBs+SECs		basic+IBs+SECs	
		LB	Time	LB	Time	LB	Time	LB	Time	LB	Time
burma14	224.83	200.33	0.04	186.20	0.05	201.89	0.12	221.93	0.55	221.54	0.49
ulysses16	268.14	231.91	0.05	217.68	0.05	233.41	0.13	264.50	1.39	263.87	0.81
gr17	234.82	195.12	0.04	185.64	0.05	201.74	0.13	232.14	1.94	231.68	0.99
gr21	232.77	207.48	0.06	202.47	0.13	210.05	0.35	226.50	2.36	225.58	1.76
ulysses22	301.58	242.31	0.07	221.60	0.14	245.49	0.33	293.87	5.80	292.29	2.54
gr24	263.04	224.71	0.08	211.55	0.20	224.12	0.42	255.55	6.24	254.40	4.56
fri26	239.08	211.92	0.10	208.46	0.25	211.75	0.50	235.09	8.58	234.25	4.38
bayg29	345.11	297.66	0.12	288.43	0.32	297.52	0.59	322.72	15.62	321.60	7.46
bays29	305.46	262.43	0.12	252.48	0.35	261.74	0.71	288.22	11.25	287.64	6.61
att30	246.98	185.09	0.15	168.01	0.41	181.79	0.56	236.94	16.63	236.12	10.68
dantzig30	321.89	259.00	0.15	216.58	0.41	257.48	0.60	303.84	17.85	300.34	12.84
eil30	320.74	275.86	0.12	272.25	0.39	276.50	0.59	299.43	10.25	298.00	7.64
gr30	279.94	227.24	0.14	219.08	0.37	229.37	0.56	267.25	16.17	264.36	7.19
hk30	318.63	259.56	0.13	244.35	0.37	256.74	0.53	306.28	17.21	303.18	11.84
swiss30	340.42	295.79	0.14	289.72	0.39	296.40	0.64	323.65	11.17	320.35	6.97
eil35	365.34	304.68	0.22	300.03	0.68	304.80	0.88	328.98	30.96	328.00	17.04
gr35	276.18	215.71	0.24	204.69	0.61	216.45	1.20	261.99	34.68	259.20	15.55
swiss35	378.36	300.50	0.27	296.40	0.72	300.73	1.04	346.96	23.35	343.68	16.10
eil40	421.10	335.17	0.34	328.11	0.96	334.69	1.44	369.47	50.31	368.00	28.89
dantzig42	247.25	199.25	0.42	177.45	1.08	197.93	1.19	233.94	61.28	232.35	47.28
swiss42	350.45	259.97	0.39	252.34	1.28	260.23	1.09	319.94	59.89	318.00	34.79
eil45	474.44	360.15	0.46	350.78	1.42	358.26	1.31	398.72	27.68	397.00	44.77
Avg.	307.12	252.36	0.18	240.65	0.48	252.69	0.68	288.09	19.60	286.43	13.24

Integer Solutions

Tables 2.8, 2.9, 2.10 and 2.11 contain the comparison of the solutions obtained by TGJL16 and by the B&C and Dynamic B&C algorithms, with small, medium,

large or quadratic service time functions. The results obtained by the Dynamic B&C algorithm are shown in the cases of applying the algorithm to both the NMI and the basic models. We consider the following algorithms:

- TGJL16: the basic model (2.1)-(2.6), enhanced with the GG constraints (2.7)-(2.8), with the computation of the M value and the bounds B_1, B_2 (only for the linear service time functions) and B_3, solved by CPLEX;
- NMI B&C: the model (2.17)-(2.20), (2.23)-(2.26), (2.27)-(2.28), for the linear service time functions, and the model (2.17)-(2.20), (2.24)-(2.26), (2.27), (2.30)-(2.31), for the quadratic service time function, with the computation of the improved M value and of the improved bounds IB_1, IB_2 (only for the linear service time functions) and IB_3 (described in Section 2.4.1), and with the separation of the SECs (described in Section 2.4.2), solved by using the CPLEX callback functions for separating the SECs at every node of the decision tree;
- NMI Dyn. B&C: the method NMI B&C in which, at every node of the decision tree, IB_2 (only for the linear service time functions) and the shortest path computations for IB_3 are dynamically updated by considering the x variables fixed by the branching;
- basic Dyn. B&C: as in the NMI Dyn. B&C algorithm, but using the basic model (2.1)-(2.6).

In each table, we report, for each instance, the instance name and the best known solution value. Then we show, for each method, the integer solution value obtained at the end of the solving process, the optimality percentage gap (i.e., the percentage gap between the best upper bound and the best lower bound found at the end of the solving process), and the corresponding computing time (expressed in seconds). At the bottom of each table, we report the averages of the values shown in the corresponding columns, except for the computing time: indeed, not all methods solve the same subset of instances. Therefore, to have a fair comparison, we show the average computing time by considering only the instances solved by a subset of methods. More precisely, we call: (i) Avg. TGJL16 the average computing time over the subset of instances solved by TGJL16; (ii) Avg. NMI B&C the average computing time over the subset of instances solved by the NMI B&C algorithm; (iii) Avg. basic Dyn. B&C the average computing time over the subset of instances solved by the basic Dyn. B&C algorithm. Note that not all these averages are reported in every table, since, in some cases, the subset of instances solved by the various methods coincides. In addition, we show at the bottom of each table, the number of instances solved, by each method, to proven optimality. For the case of the quadratic service time function (Table 2.11), we show an additional row, called Avg. Feas., since TGJL16 is not able to find a feasible solution for a subset of instances.

In Table 2.8, in which the small service times are considered, we can see that all methods find the optimal solution for all the 22 instances, and the computing times of the proposed B&C and Dynamic B&C algorithms are about one order of magnitude shorter than those of TGJL16.

When the medium service times are considered (Table 2.9), not all the instances can be solved to optimality, and the Dynamic B&C algorithm based on NMI obtains the largest number of optimal solutions (20 out of 22), while TGJL16 can solve only 14 instances to proven optimality. In addition, the average percentage gaps of the proposed algorithms are always very small. By looking at the average computing times, we can see that, on the subset of 14 instances solved by all methods, the fastest one is the Dynamic B&C algorithm based on the basic model. On the contrary, if we consider the subset of instances solved by NMI B&C, the fastest algorithm turns out to be the Dynamic B&C based on the NMI formulation. The same happens if we consider the subset of instances solved by the basic Dynamic B&C algorithm.

In the case of the large service times (Table 2.10), the TSP-TS instances become harder for all methods: TGJL16 can solve to proven optimality only 6 instances, and the largest number of instances solved to proven optimality is 11 (obtained by the Dynamic B&C algorithm based on the model NMI). As for the medium service times, also in this case, the fastest algorithm on the subset of instances solved by all methods is the Dynamic B&C algorithm based on the basic model, while, when we consider the subsets of instances solved by the NMI B&C and by the basic Dyn. B&C algorithms, the Dynamic B&C algorithm based on the NMI formulation is the fastest one. In the latter cases, we can observe that the computing times are significantly reduced by using the NMI Dyn. B&C algorithm. Note that, since the 9 instances solved by the NMI B&C algorithm and the 9 instances solved by the basic Dyn. B&C algorithm are not the same, we do not have the corresponding average computing time values. In addition, we can observe that the average percentage gap obtained by the NMI Dyn. B&C algorithm is 2.59%, and is the smallest one.

Finally, in Table 2.11, we report the results obtained for the quadratic service time function. This case turns out to be the most difficult one for the TGJL16 method: indeed, for the last four instances, no feasible solution is obtained within the time limit of two hours. Both the B&C and the Dynamic B&C algorithms based on the NMI formulation are able to determine the same subset of 18 instances solved to proven optimality in similar computing times. TGJL16 solves 9 instance to proven optimality in an average computing time that is significantly larger than that of the other algorithms. If we consider the subset of instances solved by the Dynamic B&C algorithm based on the basic model and its computing time, we can see that the computing times of both the NMI B&C and the NMI Dyn. B&C algorithms are considerably shorter.

We can conclude that the proposed algorithms outperform TGJL16 in terms of number of instances solved to proven optimality and computing times. Among the proposed algorithms, the Dynamic B&C algorithm based on the NMI formulation has globally the best performance. In particular, with respect to the NMI B&C algorithm, that does not include the dynamic update of the improved bounds, the NMI Dyn. B&C algorithm is always able to solve a larger (or equal) number of instances to proven optimality (in shorter or comparable computing times). This highlights that the dynamic update of the bounds is effective. By comparing the Dynamic B&C algorithm based on the model NMI with that based on the basic model, we can see that again the former solves a larger number of instances to

proven optimality, and the computing times of the former are significantly shorter than those of the latter, thus showing the usefulness of the NMI formulation.

2.6.3 Additional Instances

In Section 2.6.3 we report the results obtained on 13 larger size symmetric instances (Table 2.12), corresponding to all the symmetric instances with up to 58 nodes contained in the TSPLIB (except the instances already considered in Taş et al. (2016)). In Section 2.6.3 we show the results obtained on all the 27 asymmetric instances (Table 2.13) with up to 45 nodes contained in the TSPLIB. In both cases, we consider small service times, and report the results of the following methods: GA+SECs, i.e., the GA in which the continuous relaxation of the basic model, combined with the SECs separation, is used for building a subset of the initial population; TGJL16 and NMI Dyn.B&C (defined as in Section 2.6.2). For the additional instances, we consider a time limit of 50000 seconds. For the GA+SECs, we consider 10 runs for each instance.

In each table, we report the instance name and the best solution value found by the three considered algorithms. Then, for GA+SECs, we show the average, over the 10 runs, percentage gap w.r.t. the best solution value, the minimum percentage gap found over the 10 runs and the average computing time over the 10 runs. In addition, we show, for the TGJL16 and the NMI Dyn. B&C algorithms, the lower bound value of the continuous relaxation at the root node and the corresponding computing time, the integer solution value obtained at the end of the solving process, the optimality percentage gap (i.e., the percentage gap between the best upper bound and the best lower bound found at the end of the solving process), and the corresponding computing time. All the computing times are expressed in seconds. At the bottom of each table, we display the averages of the values in the corresponding columns, except for the average computing time that is shown separately for comparison on the instances solved to proven optimality by both exact methods. In addition, we report, for each exact method, the number of instances solved to proven optimality.

Larger Symmetric Instances

From Table 2.12 we can observe that the GA algorithm is able to obtain the best solution for all instances but one, when the best solution over 10 runs is considered. On average, the gap is only 0.19%, and the computing time 1.26 seconds, thus confirming the effectiveness of the GA algorithm. As it was noted for the 22 instances, the lower bound obtained at the root node for the NMI Dyn.B&C algorithm is much larger than that of TGJL16, even though it requires longer computing times. As regard as the integer solutions found, TGJL16 solves to proven optimality only 6 instances, while the Dynamic B&C algorithm based on the NMI formulation solves 12 (out of 13) instances. In addition, the computing time of the latter is much shorter than that of the former.

Table 2.8: Comparison of Upper Bounds with small service times on instances from Taş et al. (2016).

# inst	Best	TGJL16			NMI B&C			NMI Dyn. B&C			basic Dyn. B&C		
		UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time
burma14	228.83	228.83	0.00	0.32	228.83	0.00	0.31	228.83	0.00	0.37	228.83	0.00	0.19
ulysses16	271.74	271.74	0.00	2.18	271.74	0.00	0.41	271.74	0.00	0.33	271.74	0.00	0.30
gr17	238.39	238.39	0.00	1.44	238.39	0.00	0.25	238.39	0.00	0.33	238.39	0.00	0.19
gr21	237.11	237.11	0.00	0.38	237.11	0.00	0.43	237.11	0.00	0.33	237.11	0.00	0.37
ulysses22	306.43	306.43	0.00	19.84	306.43	0.00	1.29	306.43	0.00	1.29	306.43	0.00	1.93
gr24	269.09	269.09	0.00	1.99	269.09	0.00	0.91	269.09	0.00	0.96	269.09	0.00	0.67
fri26	247.99	247.99	0.00	2.62	247.99	0.00	0.83	247.99	0.00	0.83	247.99	0.00	0.56
bayg29	345.49	345.49	0.00	8.33	345.49	0.00	2.15	345.49	0.00	2.42	345.49	0.00	1.70
bays29	309.27	309.27	0.00	18.85	309.27	0.00	3.64	309.27	0.00	4.49	309.27	0.00	4.51
att30	253.85	253.85	0.00	223.61	253.85	0.00	2.50	253.85	0.00	3.37	253.85	0.00	9.95
dantzig30	324.21	324.21	0.00	165.18	324.21	0.00	2.04	324.21	0.00	2.34	324.21	0.00	6.11
eil30	323.40	323.40	0.00	3.42	323.40	0.00	1.94	323.40	0.00	2.02	323.40	0.00	1.22
gr30	283.91	283.91	0.00	14.00	283.91	0.00	1.30	283.91	0.00	1.37	283.91	0.00	1.89
hk30	324.20	324.20	0.00	168.18	324.20	0.00	2.30	324.20	0.00	2.46	324.20	0.00	11.25
swiss30	342.50	342.50	0.00	7.65	342.50	0.00	1.36	342.50	0.00	1.41	342.50	0.00	1.20
eil35	363.38	363.38	0.00	44.19	363.38	0.00	29.57	363.38	0.00	32.39	363.38	0.00	11.31
gr35	281.82	281.82	0.00	149.94	281.82	0.00	3.30	281.82	0.00	3.54	281.82	0.00	21.83
swiss35	373.60	373.60	0.00	26.10	373.60	0.00	2.35	373.60	0.00	2.67	373.60	0.00	7.97
eil40	410.35	410.35	0.00	217.95	410.35	0.00	69.54	410.35	0.00	82.15	410.35	0.00	49.87
dantzig42	257.37	257.37	0.00	2302.25	257.37	0.00	35.06	257.37	0.00	36.49	257.37	0.00	130.97
swiss42	351.15	351.15	0.00	2073.20	351.15	0.00	14.00	351.15	0.00	16.10	351.15	0.00	140.62
eil45	448.10	448.10	0.00	2062.93	448.10	0.00	434.35	448.10	0.00	587.63	448.10	0.00	534.56
Avg.	308.74	308.74	0.00		308.74	0.00		308.74	0.00		308.74	0.00	
#Opt.			22			22			22			22	
Avg. TGJL16				341.57			27.72			35.70			42.69

Table 2.9: Comparison of Upper Bounds with medium service times on instances from Taş et al. (2016).

# inst	Best	TGJL16			NMI B&C			NMI Dyn. B&C			basic Dyn. B&C		
		UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time
burma14	236.44	236.44	0.00	1.02	236.44	0.00	0.35	236.44	0.00	0.35	236.44	0.00	0.19
ulysses16	279.57	279.57	0.00	5.16	279.57	0.00	0.42	279.57	0.00	0.50	279.57	0.00	0.53
gr17	245.40	245.40	0.00	1.59	245.40	0.00	0.24	245.40	0.00	0.24	245.40	0.00	0.24
gr21	249.32	249.32	0.00	1.81	249.32	0.00	0.49	249.32	0.00	0.60	249.32	0.00	0.31
ulysses22	318.06	318.06	0.00	268.13	318.06	0.00	3.75	318.06	0.00	4.58	318.06	0.00	35.27
gr24	284.93	284.93	0.00	33.75	284.93	0.00	1.38	284.93	0.00	1.48	284.93	0.00	2.18
fri26	263.01	263.01	0.00	110.43	263.01	0.00	1.32	263.01	0.00	1.37	263.01	0.00	1.48
bayg29	371.22	371.22	0.00	126.62	371.22	0.00	21.95	371.22	0.00	43.84	371.22	0.00	31.50
bays29	331.90	331.90	0.00	551.96	331.90	0.00	90.11	331.90	0.00	82.24	331.90	0.00	119.10
att30	273.10	273.10	3.29	7200.00	273.10	0.00	136.82	273.10	0.00	130.81	273.10	0.00	3201.08
dantzig30	349.60	349.60	2.75	7200.00	349.60	0.00	44.95	349.60	0.00	60.86	349.60	0.00	385.31
eil30	349.16	349.16	0.00	22.01	349.16	0.00	11.38	349.16	0.00	10.86	349.16	0.00	6.10
gr30	305.23	305.23	0.00	324.07	305.23	0.00	5.11	305.23	0.00	5.52	305.23	0.00	34.00
hk30	347.35	347.35	1.05	7200.00	347.35	0.00	21.64	347.35	0.00	24.34	347.35	0.00	823.23
swiss30	366.78	366.78	0.00	350.44	366.78	0.00	6.37	366.78	0.00	6.95	366.78	0.00	9.95
eil35	397.42	397.42	0.00	2139.22	397.42	0.00	4241.00	397.42	0.00	1310.73	397.42	0.00	280.15
gr35	306.91	306.91	2.77	7200.00	306.91	0.00	81.35	306.91	0.00	81.46	306.91	0.00	2414.11
swiss35	406.92	406.92	0.00	3680.01	406.92	0.00	67.37	406.92	0.00	52.19	406.92	0.00	348.97
eil40	452.89	452.89	1.24	7200.00	452.89	1.19	7200.00	452.89	0.00	6087.80	452.89	0.00	1913.40
dantzig42	285.07	285.07	5.81	7200.00	285.07	1.77	7200.00	285.07	0.80	7200.00	285.07	3.37	7200.00
swiss42	388.64	388.64	5.26	7200.00	388.64	1.88	7200.00	388.64	0.00	6825.77	388.64	3.48	7200.00
eil45	502.52	502.52	3.09	7200.00	502.52	2.77	7200.00	502.52	2.13	7200.00	502.52	2.40	7200.00
Avg.	332.34	332.34	1.15		332.34	0.35		332.34	0.13		332.34	0.42	
#Opt.			14			18			20			19	
Avg. TGJL16				544.02			317.95			108.68			62.14
Avg. NM B&C							263.11			101.05			427.43
Avg. basic Dyn. B&C										416.14			505.64

Table 2.10: Comparison of Upper Bounds with large service times on instances from Taş et al. (2016).

# inst	Best	TGJL16			NMI B&C			NMI Dyn. B&C			basic Dyn. B&C		
		UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time
burma14	252.62	252.62	0.00	2.66	252.62	0.00	0.45	252.62	0.00	0.48	252.62	0.00	0.52
ulysses16	296.28	296.28	0.00	45.86	296.28	0.00	1.11	296.28	0.00	1.36	296.28	0.00	3.40
gr17	260.34	260.34	0.00	7.26	260.34	0.00	0.49	260.34	0.00	0.49	260.34	0.00	0.52
gr21	275.96	275.96	0.00	13.82	275.96	0.00	1.88	275.96	0.00	1.83	275.96	0.00	1.08
ulysses22	343.58	343.57	4.83	7200.00	343.57	0.00	58.29	343.57	0.00	46.59	343.57	0.00	1776.91
gr24	320.42	320.42	0.00	1285.79	320.42	0.00	35.36	320.42	0.00	26.27	320.42	0.00	70.38
fri26	297.39	297.39	4.03	7200.00	297.39	0.00	20.74	297.39	0.00	21.93	297.39	0.00	62.84
bayg29	430.35	430.35	2.82	7200.00	430.35	1.94	7200.00	430.35	0.00	6926.77	430.35	0.80	7200.00
bays29	383.78	383.78	4.72	7200.00	383.78	4.04	7200.00	383.78	3.01	7200.00	383.78	4.03	7200.00
att30	316.51	316.51	10.42	7200.00	316.51	6.78	7200.00	316.51	5.35	7200.00	316.51	7.68	7200.00
dantzig30	404.54	404.54	9.67	7200.00	404.54	4.24	7200.00	404.54	3.29	7200.00	404.54	7.06	7200.00
eil30	408.23	408.23	0.00	2437.02	408.23	0.00	5455.40	408.23	0.00	798.22	408.23	0.00	301.15
gr30	353.89	353.89	6.82	7200.00	353.89	0.00	5785.61	353.89	0.00	1978.32	353.89	2.19	7200.00
hk30	400.88	400.88	8.29	7200.00	400.88	3.20	7200.00	400.88	2.62	7200.00	400.88	5.99	7200.00
swiss30	422.54	422.54	6.77	7200.00	422.54	1.74	7200.00	422.54	0.00	1616.20	422.54	0.00	6408.20
eil35	474.90	474.90	3.87	7200.00	474.90	4.14	7200.00	474.90	3.19	7200.00	474.90	3.19	7200.00
gr35	365.75	365.75	10.45	7200.00	365.75	7.06	7200.00	365.75	5.37	7200.00	365.75	9.02	7200.00
swiss35	485.44	485.44	9.84	7200.00	485.44	6.95	7200.00	485.44	5.33	7200.00	485.44	7.31	7200.00
eil40	556.10	556.10	5.19	7200.00	556.10	5.32	7200.00	556.10	4.68	7200.00	556.10	4.62	7200.00
dantzig42	352.36	352.36	12.55	7200.00	352.36	10.61	7200.00	352.36	9.43	7200.00	352.36	10.61	7200.00
swiss42	480.30	480.29	11.77	7200.00	480.29	9.18	7200.00	480.29	8.31	7200.00	480.29	9.14	7200.00
eil45	638.13	638.13	6.81	7200.00	638.13	6.68	7200.00	638.13	6.41	7200.00	638.13	6.00	7200.00
Avg.	387.29	387.29	5.40		387.29	3.27		387.29	2.59		387.29	3.53	
#Opt.			6			9			11			9	
Avg. TGJL16				632.07			915.78			138.11			62.84
Avg. NM B&C							1262.15			319.50			
Avg. basic Dyn. B&C										279.26			958.33

Table 2.11: Comparison of Upper Bounds with quadratic service times on instances from Taş et al. (2016).

# inst	Best	TGJL16			NMI B&C			NMI Dyn. B&C			basic Dyn. B&C		
		UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time	UB	Gap%	Time
burma14	224.83	224.83	0.00	2.79	224.83	0.00	3.64	224.83	0.00	3.65	224.83	0.00	1.09
ulysses16	268.14	268.14	0.00	9.94	268.14	0.00	4.25	268.14	0.00	4.11	268.14	0.00	1.95
gr17	234.82	234.82	0.00	10.26	234.82	0.00	5.83	234.82	0.00	5.88	234.82	0.00	1.89
gr21	232.77	232.77	0.00	6.58	232.77	0.00	8.18	232.77	0.00	8.00	232.77	0.00	4.29
ulysses22	301.58	301.58	0.00	666.92	301.58	0.00	18.89	301.58	0.00	19.71	301.58	0.00	14.41
gr24	263.04	263.04	0.00	45.21	263.04	0.00	17.90	263.04	0.00	18.04	263.04	0.00	19.75
fri26	239.08	239.08	0.00	28.36	239.08	0.00	21.93	239.08	0.00	21.92	239.08	0.00	25.24
bayg29	345.11	345.11	4.06	7200.00	345.11	0.00	292.44	345.11	0.00	441.98	345.11	1.06	7200.00
bays29	305.46	305.46	2.12	7200.00	305.46	0.00	145.85	305.46	0.00	179.04	305.46	0.00	6409.08
att30	246.98	246.98	1.15	7200.00	246.98	0.00	40.64	246.98	0.00	43.63	246.98	0.00	428.38
dantzig30	321.89	321.89	3.94	7200.00	321.89	0.00	153.15	321.89	0.00	240.26	321.89	0.00	2594.89
eil30	320.74	320.74	2.39	7200.00	320.74	0.00	398.16	320.74	0.00	499.42	320.74	0.00	5381.88
gr30	279.94	279.94	0.00	2913.02	279.94	0.00	42.25	279.94	0.00	44.07	279.94	0.00	90.91
hk30	318.63	318.63	3.16	7200.00	318.63	0.00	49.71	318.63	0.00	52.49	318.63	0.00	2626.19
swiss30	340.42	340.42	0.00	1819.48	340.42	0.00	40.83	340.42	0.00	41.58	340.42	0.00	226.65
eil35	365.34	365.34	8.14	7200.00	365.34	4.85	7200.00	365.34	4.61	7200.00	365.34	7.64	7200.00
gr35	276.18	276.18	8.93	7200.00	276.18	0.00	113.79	276.18	0.00	129.96	276.18	3.14	7200.00
swiss35	378.36	387.31	12.86	7200.00	378.36	0.00	1839.16	378.36	0.00	2174.10	378.36	3.79	7200.00
eil40	421.10	-	-	7200.00	421.10	8.41	7200.00	421.10	8.52	7200.00	421.64	10.72	7200.00
dantzig42	247.25	-	-	7200.00	247.25	0.00	676.00	247.25	0.00	999.18	247.25	3.99	7200.00
swiss42	350.45	-	-	7200.00	350.45	3.81	7200.00	350.45	3.95	7200.00	350.45	7.62	7200.00
eil45	474.44	-	-	7200.00	474.44	13.24	7200.00	476.22	13.27	7200.00	476.22	16.76	7200.00
Avg.	307.12	-	-		307.12	1.38		307.20	1.38		307.22	2.49	
Avg. Feas.		292.90	2.60		292.41	0.27		292.41	0.26		292.41	0.87	
#Opt.			9			18			18			14	
Avg. TGJL16				611.40			18.19			18.55			42.91
Avg. NM B&C							215.15			273.72			
Avg. basic Dyn. B&C							84.41			67.94			1273.33

Table 2.12: Comparison of Lower and Upper Bounds with small service times on larger symmetric instances.

# inst	Best	GA+SECs			TGJL16					NMI Dyn.B&C				
		Avg%	Min%	Time	LB	Time	UB	Gap%	Time	LB	Time	UB	Gap%	Time
att35	271.36	0.00	0.00	0.74	238.07	0.02	271.36	0.00	76.89	264.96	0.73	271.36	0.00	3.05
dantzig35	296.02	0.14	0.00	0.66	242.80	0.03	296.02	0.00	419.88	285.84	0.56	296.02	0.00	10.42
hk35	347.51	0.00	0.00	0.64	303.97	0.04	347.51	0.00	112.29	337.31	0.65	347.51	0.00	6.13
att40	296.33	0.00	0.00	0.90	247.62	0.04	296.33	0.00	5293.41	288.89	0.88	296.33	0.00	9.26
hk40	356.28	0.00	0.00	0.93	304.76	0.04	356.28	0.00	1404.32	343.39	0.90	356.28	0.00	31.04
att45	318.94	0.06	0.00	1.24	258.23	0.04	318.94	2.09	50000.00	309.84	0.95	318.94	0.00	473.42
hk45	365.41	0.00	0.00	1.27	301.77	0.04	365.41	0.00	30916.40	353.13	1.63	365.41	0.00	487.68
att48	336.75	0.89	0.00	1.42	271.45	0.04	338.97	5.58	50000.00	326.51	1.93	336.75	0.00	3311.93
gr48	383.13	0.62	0.00	1.44	311.41	0.05	383.13	3.70	50000.00	362.01	2.40	383.13	0.00	13732.86
hk48	371.86	0.01	0.00	1.57	320.20	0.05	371.86	1.25	50000.00	358.56	1.93	371.86	0.00	3621.10
eil51	442.53	0.70	0.17	1.70	392.49	0.17	442.53	0.25	50000.00	429.21	4.25	442.53	0.00	24122.59
berlin52	439.76	0.00	0.00	1.95	356.02	0.03	439.76	6.31	50000.00	416.05	2.05	439.76	1.30	50000.00
brazil58	386.48	0.05	0.00	1.95	273.53	0.07	386.48	5.18	50000.00	369.57	2.13	386.48	0.00	43579.73
Avg.	354.80	0.19	0.01	1.26	294.02	0.05	354.97	1.87		341.94	1.62	354.80	0.10	
#Opt.								6					12	
Avg. TGJL16									6370.53					91.26

Asymmetric Instances

A similar behavior as that observed in Table 2.12 can also be found in Table 2.13. The GA algorithm finds several best solutions in short computing time (0.42 seconds on average). The solution found is proved to be optimal for 26 instances (out of 27) by the Dynamic B&C algorithm, while TGJL16 can prove the optimality of only 16 solutions. In addition, the computing time to prove the optimality of the solutions is much shorter for the NMI Dyn.B&C algorithm (on average about 37 seconds versus 4935 seconds). Therefore, we can conclude that the proposed algorithm turns out to be very effective on the asymmetric instances as well.

2.7 Conclusions and Future Research

We studied the Traveling Salesman Problem with Time-dependent Service times (TSP-TS), which considers the service time at each customer as a continuous function of the start time of service. We proposed a new formulation for the problem and included explicit subtour elimination constraints, dynamically separated. In addition, we proposed an upper bound on the total route duration, obtained by a multi-operator Genetic Algorithm, an improved lower bound on the total service time, and new lower and upper bounds on the start time of service at each customer. These ingredients are included in Branch-and-Cut algorithms, one of which exploits the dynamic update of the bounds during the solving process. The proposed algorithms are tested on benchmark instances from the literature and compared to an existing method. The results show that the optimality of the solutions found can be proved for a larger set of instances in shorter computing times. Additional computational experiments on larger size symmetric instances with up to 58 nodes and on asymmetric instances with up to 45 nodes show the effectiveness of the proposed algorithm.

Future research will focus on extending the proposed methods to other variants of the TSP that embed the time-dependency. In addition, the problem with time-dependent service times could be generalized to deal with other features, such as more vehicles or time window constraints.

Table 2.13: Comparison of Lower and Upper Bounds with small service times on asymmetric instances.

# inst	Best	GA+SECs			TGJL16					NMI Dyn.B&C				
		Avg%	Min%	Time	LB	Time	UB	Gap%	Time	LB	Time	UB	Gap%	Time
br17	88.60	0.00	0.00	0.16	27.01	0.00	88.60	0.00	2.00	87.18	0.09	88.60	0.00	1.53
ft30	338.54	0.40	0.00	0.23	278.97	0.02	338.54	0.00	1005.68	323.87	0.15	338.54	0.00	14.12
ftv30a	298.60	0.14	0.00	0.28	268.60	0.02	298.60	0.00	11.03	289.10	0.11	298.60	0.00	1.12
ftv30b	326.36	0.34	0.00	0.34	303.34	0.02	326.36	0.00	12.95	316.91	0.14	326.36	0.00	1.05
ftv30c	305.95	0.00	0.00	0.30	268.66	0.02	305.95	0.00	101.96	291.18	0.22	305.95	0.00	1.96
p30	201.96	0.01	0.00	0.27	144.48	0.01	201.96	2.51	50000.00	196.18	0.21	201.96	0.00	46.82
ry30p	303.02	0.46	0.00	0.33	247.13	0.02	303.02	0.00	993.95	291.65	0.18	303.02	0.00	11.04
ftv33	328.01	0.00	0.00	0.44	292.87	0.02	328.01	0.00	21.88	319.75	0.33	328.01	0.00	1.92
ft35	394.62	1.55	0.00	0.30	337.02	0.02	394.62	0.00	46230.96	376.45	0.25	394.62	0.00	270.68
ftv35	358.07	1.41	0.56	0.43	322.67	0.03	358.07	0.00	2452.24	342.86	0.42	358.07	0.00	35.33
ftv35a	341.12	0.94	0.00	0.40	306.64	0.02	341.12	0.00	384.90	325.36	0.29	341.12	0.00	11.78
ftv35b	351.27	1.61	1.36	0.42	320.41	0.03	351.27	0.00	647.08	335.42	0.37	351.27	0.00	7.50
ftv35c	343.30	0.00	0.00	0.52	308.72	0.03	343.30	0.00	667.67	330.51	0.38	343.30	0.00	16.12
p35	229.24	0.38	0.00	0.39	160.62	0.01	238.05	9.19	50000.00	222.10	0.39	229.24	0.00	46.63
ry35p	326.29	0.00	0.00	0.43	297.81	0.03	326.29	0.00	108.50	320.52	0.37	326.29	0.00	2.28
ftv38	380.31	0.84	0.40	0.51	342.64	0.04	380.31	0.00	20789.05	363.64	0.44	380.31	0.00	127.61
ft40	436.93	0.76	0.00	0.36	374.07	0.03	436.93	2.82	50000.00	416.86	0.45	436.93	0.00	3593.72
ftv40a	365.59	0.00	0.00	0.54	329.82	0.03	365.59	0.00	2599.55	350.24	0.60	365.59	0.00	53.58
ftv40b	372.98	0.00	0.00	0.54	341.56	0.04	372.98	0.00	2939.59	358.27	0.48	372.98	0.00	40.31
ftv40c	377.38	1.22	0.53	0.53	337.17	0.03	377.38	1.95	50000.00	358.50	0.45	377.38	0.00	953.51
ry40p	365.59	0.07	0.02	0.61	320.17	0.03	365.59	0.77	50000.00	352.77	0.68	365.59	0.00	58.44
p43	288.45	0.16	0.00	0.44	43.40	0.02	288.45	0.67	50000.00	287.52	0.50	288.45	0.22	50000.00
ftv44	395.91	1.99	0.56	0.53	354.54	0.06	395.91	3.25	50000.00	370.78	0.45	395.91	0.00	8936.01
ft45	473.37	1.30	0.00	0.45	402.27	0.04	473.37	3.88	50000.00	450.34	0.58	473.37	0.00	7537.42
ftv45b	408.17	2.55	0.60	0.55	359.25	0.06	408.17	3.65	50000.00	385.63	1.17	408.17	0.00	30052.82
ftv45c	371.13	0.88	0.86	0.91	326.22	0.05	371.13	3.20	50000.00	351.23	0.90	371.13	0.00	2081.32
ry45p	394.23	1.37	0.00	0.68	338.33	0.03	394.23	2.34	50000.00	379.70	0.93	394.23	0.00	383.69
Avg.	339.44	0.63	0.17	0.42	287.20	0.03	339.77	1.27		325.72	0.43	339.44	0.01	
#Opt.								16					26	
Avg. TGJL16									4935.56					37.37

Chapter 3

Algorithms for the Pollution Traveling Salesman Problem

3.1 Introduction

Nowadays, environmental issues are becoming more important. Emissions from vehicles traveling on roads are one of the main causes of pollution. Therefore, reducing carbon emissions is one of the most important goals that have to be taken into account in vehicle routing problems (Bektaş & Laporte, 2011; Lin et al., 2014). In Bektaş & Laporte (2011), the Pollution Routing Problem (PRP) was introduced: it is a variant of the Vehicle Routing Problem (VRP) in which the goal is not only to minimize the travel distance, but also the amount of green-house emissions, fuel, travel times and their costs. The authors proposed a Mixed Integer Linear Programming (MILP) model, and analyzed trade-offs between various performance measures of vehicle routing, such as distance, load, emissions and costs. In Demir et al. (2012), the MILP model was extended to allow for low travel speeds, and an effective adaptive large neighborhood search heuristic was proposed for the PRP. A matheuristic approach was proposed in Kramer et al. (2015) for PRP and other green VRP variants. The matheuristic approach combines local search-based metaheuristic with MILP for the PRP and obtains better performance than the previously proposed algorithms.

Motivated by these recent works on the PRP, we study the Pollution Traveling Salesman Problem (PTSP), i.e. the problem of determining a Hamiltonian tour that minimizes a function of fuel consumption (dependent on vehicle speed and load) and driver costs. More precisely, we refer to the PRP as modelled in Demir et al. (2012) and consider the single vehicle case. The PTSP is formally described in Section 3.2, where we also present a MILP model, enhanced with explicit sub-tour elimination constrains, for its exact solution. The main contributions of this work are two mataheuristic algorithms: an Iterated Local Search (ILS) algorithm (Lourenço et al., 2010) and a Multi-operator Genetic Algorithm (MGA), presented in Sections 3.3 and 3.4, respectively. Both algorithms are able to find good solutions for PTSP instances in very short computing times. In Section 3.5, we report computational experiments on instances, with up to 50 customers, proposed in

This chapter is based on the contents of: Cacchiani, Contreras-Bolton, Escobar, Escobar-Falcon, Linfati, & Toth, An iterated local search algorithm for the pollution traveling salesman problem. In P. Daniele, & L. Scrimali (Eds.) New Trends in Emerging Complex Real Life Problems: ODS, Taormina, Italy, September 10–13, 2018 , (pp. 83–91). Cham: Springer International Publishing, 2018.

Demir et al. (2012) and adapted here for the single vehicle case: in particular, we compare the results obtained by the ILS and MGA algorithms with those found by a Cut-and-Branch algorithm, based on the enhanced MILP model presented in Section 3.2. Finally, we draw some conclusions in Section 3.6.

3.2 Problem Description and Formulation

The PTSP is defined on a complete directed graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{0, \ldots, n\}$ is the set of nodes, 0 is a depot and $\mathcal{A}$ is the set of arcs. The distance from node i to node j is denoted by d_{ij}. The set $\mathcal{N}_0 = \mathcal{N} \setminus \{0\}$ is the customer set. Each customer $i \in \mathcal{N}_0$ has a non-negative demand q_i, and a service time t_i. We define $D = \sum_{i \in \mathcal{N}_0} q_i$ as the capacity of the vehicle, and f_d as the driver wage per unit time. We consider a discretized speed function defined by $|\mathcal{R}|$ non-decreasing speed levels $\bar{v}^r$ $(r \in \mathcal{R})$, where each $r \in \mathcal{R}$ corresponds to a speed interval in the range $[v^l, v^u]$ (where v^l and v^u are, respectively, the lower and upper speed limits).

Table 3.1: Parameters used in the PTSP model.

Notation	Description Typical	Values
w	Curb-weight (kilogram)	6350
ξ	Fuel-to-air mass ratio	1
k	Engine friction factor (kilojoule/rev/liter)	0.2
N	Engine speed (rev/second)	33
V	Engine displacement (liters)	5
g	Gravitational constant (meter/second^2)	9.81
C_d	Coefficient of aerodynamic drag	0.7
ρ	Air density (kilogram/meter^3)	1.2041
A	Frontal surface area (meter^2)	3.912
C_r	Coefficient of rolling resistance	0.01
η_{tf}	Vehicle drive train efficiency	0.4
η	Efficiency parameter for diesel engines	0.9
f_d	Driver wage per (£/second)	0.0022
κ	Heating value of a typical diesel fuel (kilojoule/gram)	44
ψ	Conversion factor (gram/second to liter/second)	737
v^l	Lower speed limit (meter/second)	5.5 (or 20 kilometer/hour)
v^u	Upper speed limit (meter/second)	25 (or 90 kilometer/hour)

We adopt the fuel consumption expression proposed in Demir et al. (2012), which extends the one presented in Bektaş & Laporte (2011) to allow for speeds lower than 40 kilometer/hour, and refer the interested reader to these papers for explanations of how this expression is derived. For a given arc $(i, j) \in \mathcal{A}$ of length d_{ij}, traversed at speed v by a vehicle carrying load $M = w + f_{ij}$, where w is the weight of the empty vehicle (curb weight) and f_{ij} is the load carried by the vehicle on this arc, the fuel consumption can be expressed as:

$$F(v) = \lambda k N V d_{ij}/v + \lambda \beta \gamma d_{ij} v^2 + \lambda w \gamma \alpha_{ij} d_{ij} + \lambda \gamma \alpha_{ij} f_{ij} d_{ij} \tag{3.1}$$

where $\lambda = \xi/\kappa\psi$ and $\gamma = 1/1000\eta_{tf}\eta$ are constants, $\beta = 0.5 C_d \rho A$ is a vehicle specific constant, $\alpha_{ij} = \tau + g \sin\theta_{ij} + g C_r \cos\theta_{ij}$ is an arc specific constant depending on the road angle θ_{ij}, the acceleration τ (meter/second^2), and all other parameters

and values, taken from Demir et al. (2012), are reported in Table 3.1. In particular, the first two terms of (3.1) represent the speed-induced energy requirements, while the last two terms represent the load-induced energy requirements.

The PTSP calls for determining the minimum cost Hamiltonian tour that departs from the depot and visits each customer exactly once by serving its demand, where the cost is given by the sum of fuel consumption and driver wage. We introduce the following decisional variables: (i) binary variables x_{ij} assuming value 1 if arc $(i,j) \in \mathcal{A}$ is traversed; non-negative variables f_{ij} representing the amount of flow (i.e. the load on the vehicle) on arc $(i,j) \in \mathcal{A}$; (iii) binary variables z_{ij}^r assuming value 1 if arc $(i,j) \in \mathcal{A}$ is traversed at speed level $r \in \mathcal{R}$. The MILP model for the PTSP reads as follows:

$$\text{Min} \sum_{(i,j)\in\mathcal{A}} \lambda k N V d_{ij} \sum_{r\in\mathcal{R}} z_{ij}^r/\bar{v}^r + \sum_{(i,j)\in\mathcal{A}} \lambda\beta\gamma d_{ij} \sum_{r\in\mathcal{R}} z_{ij}^r(\bar{v}^r)^2 \tag{3.2}$$

$$+ \sum_{(i,j)\in\mathcal{A}} \lambda w\gamma\alpha_{ij} d_{ij} x_{ij} + \sum_{(i,j)\in\mathcal{A}} \lambda\gamma\alpha_{ij} d_{ij} f_{ij} \tag{3.3}$$

$$+ \quad f_d\Big(\sum_{(i,j)\in\mathcal{A}} \sum_{r\in\mathcal{R}} (d_{ij}/\bar{v}^r) z_{ij}^r + \sum_{i\in\mathcal{N}_0} t_i\Big) \tag{3.4}$$

$$\text{subject to} \tag{3.5}$$

$$\sum_{j\in\mathcal{N}_0} f_{0j} = D \tag{3.6}$$

$$\sum_{j\in\mathcal{N}_0} f_{j0} = 0 \tag{3.7}$$

$$\sum_{j\in\mathcal{N}} x_{ij} = 1, \ \forall i \in \mathcal{N} \tag{3.8}$$

$$\sum_{i\in\mathcal{N}} x_{ij} = 1, \ \forall j \in \mathcal{N} \tag{3.9}$$

$$\sum_{j\in\mathcal{N}} f_{ji} - \sum_{j\in\mathcal{N}} f_{ij} = q_i, \ \forall i \in \mathcal{N}_0 \tag{3.10}$$

$$q_j x_{ij} \le f_{ij} \le (D - q_i)\, x_{ij}, \forall\, (i,j) \in \mathcal{A} \tag{3.11}$$

$$\sum_{r\in\mathcal{R}} z_{ij}^r = x_{ij}, \ \forall\, (i,j) \in \mathcal{A} \tag{3.12}$$

$$x_{ij} \in \{0,1\}, \ \forall\, (i,j) \in \mathcal{A} \tag{3.13}$$

$$f_{ij} \ge 0, \ \forall\, (i,j) \in \mathcal{A} \tag{3.14}$$

$$z_{ij}^r \in \{0,1\}, \ \forall\, (i,j) \in \mathcal{A}, \forall r \in \mathcal{R} \tag{3.15}$$

The objective function consists of three main components to be minimized: (3.2) and (3.3) represent the fuel consumption, as defined in (3.1), by taking into account, respectively, the energy required by speed variations and the energy used to curry the curb weight and the load on the vehicle, while (3.4) corresponds to the driver wage, where the term in the external brackets is the total tour duration which depends on the speeds on the used arcs and on the service times at the customers. Constraints (3.6) and (3.7) ensure, respectively, that the vehicle leaves full and returns empty at the depot. Constraints (3.8) and (3.9) guarantee that each node is visited exactly once. Constraints (3.10) and (3.11) define the load of the vehicle on each visited arc (and implicitly forbid subtours). Finally, constraints (3.12) link the x and z variables by imposing that exactly one speed level is chosen for each used arc $(i,j) \in \mathcal{A}$, and constraints (3.13)-(3.15) define the

variable domains.

We add to the model (3.2)-(3.15) the explicit subtour elimination constraints (SECs), as proposed in Dantzig et al. (1954) for the Asymmetric TSP:

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N} \setminus \mathcal{S}} x_{ij} \geq 1, \quad \mathcal{S} \subset \mathcal{N}, \quad \mathcal{S} \neq \emptyset. \tag{3.16}$$

Model (3.2)-(3.16) is used as benchmark in the computational experiments to evaluate the performance of the proposed ILS and MGA algorithms. In particular, we solve model (3.2)-(3.16) by a Cut-and-Branch algorithm, in which the SECs are separated, at the root node, by using the separation procedure proposed in Padberg & Rinaldi (1990), and the MILP solver CPLEX is used to obtain integer solutions.

3.3 Iterated Local Search Algorithm

The pseudo-code of the ILS algorithm is reported in Algorithm 3.1. The first step (lines 1–7) consists of iteratively solving the Linear Programming (LP) relaxation of model (3.2)-(3.16), by applying the separation procedure proposed in Padberg & Rinaldi (1990) to derive a set $\mathcal{S}$ containing a sub-tour: if one exists, then the corresponding cut (3.16) is added to the model and the LP-relaxation is solved again. Once the optimal solution x of the LP-relaxation has been derived, it is used to build a feasible tour, as follows. Initially, we define the depot 0 as the starting node h. Then, iteratively, we select the node j such that $x_{hj} + x_{jh}$ has the highest value: arc (h, j) is added to the tour, and j becomes the new starting node h. If, for all nodes j connected to h, $x_{hj} + x_{jh} = 0$, then we choose the arc with the smallest d_{hj}. The procedure is repeated until we obtain a complete feasible tour. We call s^* the locally optimal solution and s^{**} the current best solution. The following loop (lines 9–29) is made of three phases: *perturbation*, *local search* and *acceptance criterion*.

To perturb the current best solution s^{**} (lines 10–14) we apply, with probability 80% a double-bridge move, and a scramble tour move otherwise. The former move consists of randomly removing four edges (A, B), (C, D), (E,F), (G, H) and reconnecting them as (A, F), (G, D), (E, B), (C, H). The latter move corresponds to randomly choose a path of the tour and randomly mixing its nodes. After perturbation, we obtain solution s'.

The following phase (lines 15–24) is a local search procedure that is applied to the locally optimal solution s^*: with probability 80% we apply a 2-opt move, otherwise we perform an exchange improvement. The former move consists of executing the 2-opt procedure by using as arc costs only d_{ij} $(i, j) \in \mathcal{A}$, and the procedure is stopped at the first improvement. The latter move requires exchanging two nodes of the tour: if an improvement is obtained, then the exchange is performed, otherwise the original tour is kept. This procedure is executed $|\mathcal{N}|$ times. After applying the local search procedure, we obtain solution s''. Then, we choose to store in s^* the best solution between s' and s'', by considering function ϕ that gives the value of the PTSP objective function (3.2)-(3.4).

The last phase (lines 25–27) is the acceptance criterion (check-history): if s^{**} has not been improved for 10 iterations, then we apply an additional local search step to s^{**} by executing the 2-opt procedure. It uses as arc costs d_{ij} $(i,j) \in \mathcal{A}$, but each time an improvement is possible, it checks if the PTSP objective function value improves too, and accepts the change only in this case.

Finally, the best solution between s^* and s^{**} is stored in s^{**}. The termination condition is reached when I iterations are performed ($I = 5000$ in our computational experiments).

Algorithm 3.1 Iterated Local Search

1: **repeat**
2: $x \leftarrow$ solve LP-relaxation of (3.2)-(3.16)
3: $\mathcal{S} \leftarrow$ separation-procedure(x)
4: **if** $\mathcal{S} \neq \emptyset$ **then**
5: LP-relaxation of (3.2)-(3.16) $\leftarrow$ add-cut($\mathcal{S}$)
6: **end if**
7: **until** $\mathcal{S} = \emptyset$
8: $s^*, s^{**} \leftarrow$ build-feasible-tour(x)
9: **repeat**
10: **if** $rnd(0,1) < 0.8$ **then**
11: $s' \leftarrow$ double-bridge-move(s^{**})
12: **else**
13: $s' \leftarrow$ scramble-subtour(s^{**})
14: **end if**
15: **if** $rnd(0,1) < 0.8$ **then**
16: $s'' \leftarrow$ 2-opt-move(s^*)
17: **else**
18: $s'' \leftarrow$ exchange-improvement(s^*)
19: **end if**
20: **if** $\phi(s') < \phi(s'')$ **then**
21: $s^* \leftarrow s'$
22: **else**
23: $s^* \leftarrow s''$
24: **end if**
25: **if** check-history($\phi(s^{**})$) **then**
26: $s^* \leftarrow$ 2-opt-improvement(s^{**})
27: **end if**
28: $s^{**} \leftarrow$ store if $\phi(s^*) < \phi(s^{**})$
29: **until** termination condition

3.4 Multi-operator Genetic Algorithm

Genetic Algorithms are effective metaheuristic algorithms and have been successfully applied to solve ATSP and several its variants (Potvin, 1996; Larrañaga et al., 1999; Moon et al., 2002; Snyder & Daskin, 2006; Yuan et al., 2013; Groba et al., 2015). Generally, these algorithms use only single operators for crossover and

mutation, disregarding the potential synergy of multi-operators. However, the crossover and mutation operators can complement each other, generating a synergy which provides better results than those obtained by using single operators (Li et al., 2014; Contreras-Bolton & Parada, 2015). Taking the advantage of the multi-operators, we use an approach based on a Multi-operator Genetic Algorithm (MGA).

In this section, we describe MGA in Algorithm 3.2 and its components: the representation, the generation of the initial population, the evaluation function, the crossover and mutation genetic operators, and the genetic parameters, are presented in the subsections 3.4.1, 3.4.2, 3.4.3, 3.4.4 and 3.4.5.

In the initial step, a population of individuals is generated and evaluated using the instructions in lines 1–11, note that each initialization method has equal probability to occur. Subsequently, the main loop of the algorithm, presented in lines 12–21, is responsible for generating a new population from the current one via a selection of several crossover and mutation operators, as described in lines 15, 16 and 17. The selection is made in a tournament of three individuals (Eiben & Smith, 2015). Elitism is also implemented, where the best current parents replace 10% of the worst individuals generated in each generation (line 20).

Algorithm 3.2 Genetic Algorithm

Ensure: best individual

1: **for** $t \leftarrow 0$ **to** numbers of individuals **do**
2: **if** $rnd(0,1) < 0.33$ **then**
3: $I_t \leftarrow$ random()
4: **else if** $0.33 \geq rnd(0,1) < 0.66$ **then**
5: $I_t \leftarrow$ nearest-neighbor-heuristic()
6: **else**
7: $I_t \leftarrow$ LP-based-heuristic()
8: **end if**
9: $I_t \leftarrow$ 2-opt-improvement(I_t)
10: evaluation of individual I_t
11: **end for**
12: **for** $g \leftarrow 1$ **to** $\leq$ maximum number of generations **do**
13: **for** $t \leftarrow 1$ **to** numbers of individuals - 1 **do**
14: $(j, k) \leftarrow$ selection()
15: $(I'_t, I'_{t+1}) \leftarrow$ crossover(I_j, I_k, OX2 or DPX or HX or UNN)
16: $I'_t \leftarrow$ mutation(I'_t, EM or GSTM or 2opt or 3opt)
17: $I'_{t+1} \leftarrow$ mutation(I'_{t+1}, EM or GSTM or 2opt or 3opt)
18: evaluation of individual I'_t and I'_{t+1}
19: **end for**
20: generate new population g with elitism(I, I')
21: **end for**

3.4.1 Representation and fitness function

A permutation representation is used, where each individual corresponds to a Hamiltonian tour. As fitness function is used the objective function (3.2, 3.3, 3.4),

but we compute the objective function for each speed level $r \in \mathcal{R}$ taking as fitness value the smallest of their computed values.

3.4.2 Initial population

An initial population is computed by using three different ways which will be described in the following. Each one has the same probability to occur. Then we apply the 2-opt procedure (described in Section 3.3) to generate individuals and to try to improved them.

- Random: The tour is generated randomly.
- Nearest-Neighbor-Heuristic: The tour is generated by applying the Nearest Neighbor Heuristic (NNH) (Flood, 1956), that randomly starts from one of the nodes in $\mathcal{N}$.
- LP-based-heuristic: The heuristic is described in Algorithm 3.3. The first step (lines 2–8) is the procedure described in the lines 1–7 of Algorithm 3.2. Then, the second step (lines 11–24) generates a feasible solution by using a randomized NNH, where the randomization occurs at line 14.

Algorithm 3.3 LP-based-heuristic

1: $T \leftarrow \emptyset$
2: **repeat**
3: $x \leftarrow$ solve LP-relaxation of (3.2)-(3.16)
4: $\mathcal{S} \leftarrow$ separation-procedure(x)
5: **if** $\mathcal{S} \neq \emptyset$ **then**
6: LP-relaxation of (3.2)-(3.16) $\leftarrow$ add-cut($\mathcal{S}$)
7: **end if**
8: **until** $\mathcal{S} = \emptyset$
9: $h = 0$
10: $T \leftarrow \{h\}$
11: **repeat**
12: $max \leftarrow 0$
13: **for** $k \in \mathcal{N} \backslash T$ **do**
14: **if** $x_{hk} + x_{kh} > max$ **and** rnd(0, 1) < 0.5 **then**
15: $max \leftarrow x_{hk} + x_{kh}$
16: $j \leftarrow k$
17: **end if**
18: **end for**
19: **if** $max = 0$ **then**
20: $j \leftarrow$ node corresponding to the smallest d_{hj}.
21: **end if**
22: $T \leftarrow T \cup \{j\}$
23: $h \leftarrow j$
24: **until** obtains a complete feasible tour

3.4.3 Crossover operators

Four different crossover operators are used: Order Based Crossover (OX2) (Syswerda, 1991), Distance Preserving Crossover (DPX) (Reisleben et al., 1996), Heuristic Crossover (HX) (Grefenstette et al., 1985) and Uniform Nearest Neighbor (UNN) (Buriol et al., 2004), with probability of 10%, 45%, 20% and 25%, respectively.

- OX2: Randomly select a set of nodes from a parent and according to the order of the nodes in the selected positions of this parent impose the set on the other parent.
- DPX: the nodes contained in the first parent are copied into the offspring and all the arcs not in common with the second parent are deleted, leading to a set of disconnected paths. These paths are then reconnected without using any of the arcs that are contained in only one of the parents. In particular, given a path that ends at node i, the nearest available neighbor node k among the initial nodes of the remaining paths is taken and arc (i, k) is added to the tour, unless (i, k) was contained in one of the two parents. The procedure is repeated until all paths have been reconnected in a tour.
- HX: first, a random node is selected to be the current node of the offspring. Then, the cheapest one of the at most four (undirected) edges connecting the current node to an unvisited node from the parent tours is chosen. If none of the parental edges leads to an unvisited node a random edge is selected. This is repeated until a complete tour has been constructed.
- UNN: initially, all arcs in common between both parents are copied into the offspring. The remaining arcs are inserted as follows: for each node i a true or false value is generated randomly with the same probability. If the generated value is true (resp. false), then the arc which links i to the next node in parent A (resp. parent B) is copied into the offspring, if no restriction is violated. If a violation occurs in any of the two cases, then the arc of the other parent is considered. The resulting tour fragments are patched using the NNH algorithm.

3.4.4 Mutation operators

Four different mutation operators are used: exchange mutation (EM), Greedy Sub Tour Mutation (GSTM), 2-opt and 3-opt, with probability of 10%, 35%, 20% and 35%, respectively. EM is based on the operator proposed in Banzhaf (1990), which was described at line 18 of Algorithm 3.1. GSTM combines classical operators (such as "simple inversion mutation" and "scramble mutation") and greedy techniques by using different parameters. The parametrical structure of the operator prevents a stuck of the local solutions and reaches local solutions faster by greedy search methods and making random perturbation on these solutions like classical mutation operators and then applies them again a greedy method, for more details see (Albayrak & Allahverdi, 2011). The 3-opt operator was implemented by considering all the pairs of edges, and by selecting the third edge by randomly attempting only ten edges, instead of considering all the remaining edges.

3.4.5 Genetic parameters

The parameters involved in MGA are population size, maximum number of generation runs, and crossover and mutation probabilities. The adequate definition of the parameters is directly related to the computational performance of the evolutionary algorithms. Based on the existing literature (Eiben et al., 1999, 2007) and preliminary computational experiments, we set the following parameters: the size of the population was set at 150 individuals, the number of generations at 150, the crossover probability at 0.9, and the mutation probability at 0.2.

3.5 Computational Experiments

We used the sets od benchmark instances with 10, 15, 20, 25 and 50 customers, proposed in Demir et al. (2012) for the PRP, and adapted them to the PTSP. In particular, to make the instances feasible for a single vehicle, we removed the time window constraints for every customer, and used a vehicle with capacity $D = \sum_{i \in \mathcal{N}_0} q_i$. In addition, we generated new sets of instances with 30, 35, 40 and 45 customers, obtained from the benchmark instances with 50 customers by considering, respectively, the first 30, 35, 40 and 45 customers. Each set of instances contains 20 instances. The algorithms were implemented in C++, and all experiments were executed on an Intel Core i7-6900K with 16-Core 3.20GHz and 66 GB RAM (single thread). We used CPLEX 12.7.1 as LP and MILP solver, and set a time limit of two hours for the Cut-and-Branch algorithm.

In Table 3.2, we report the computational results obtained on instances with up to 50 customers by the exact Cut-and-Branch (C&B) algorithm and by the ILS and MGA algorithms. We wish to mention that we also solved model (3.2)-(3.15) by applying directly CPLEX, but the C&B algorithm is able to find some additional optimal solutions and has smaller average gaps and computing times. Therefore, we only report the results obtained by the C&B algorithm. Each row of Table 3.2 corresponds to a set of instances and shows average results over the 20 instances in the set. For the C&B algorithm, we report the integer solution value (UB) and the lower bound (LB) obtained at the end of the C&B solution process, the final percentage gap (gap) between UB and LB, the number of obtained optimal solutions (#opt), and the computing time (time) expressed in seconds. For the ILS and MGA algorithms, we executed 10 runs for each instance in every set, and we report the average and the minimum results obtained over the 10 runs. In particular, we show the solution value (val), the percentage gap (gap) with respect to UB (if negative, then an improvement has been obtained by ILS or MGA), the number of best solutions (#B) found (i.e. solutions having the same value or a smaller value than that of the solutions found by the C&B algorithm), and the computing time (time) in seconds. It is to note that the computing time for obtaining the "minimum" result over the 10 runs is ten times the value reported in column time for the corresponding "average" result. Finally, the last row reports, for each column, the average value over all the sets of instances.

We observe that the C&B algorithm is capable to obtain the optimal solution for all the instances with up to 25 customers in very short computing times (on average about 22 seconds). All but one instance with 30 customers are solved,

Table 3.2: Comparison among the Cut-and-Branch, ILS and MGA.

Cust.	BKS	C&B					MGA							ILS						
		UB	LB	gap	#opt	time	minimum			average				minimum			average			
							val	#B	gap	val	#B	gap	time	val	#B	gap	val	#B	gap	time
10	150.64	150.64	150.64	0.00	20	0.29	150.64	20	0.00	150.65	19	0.01	0.07	150.64	20	0.00	150.64	19	0.00	0.02
15	215.69	215.69	215.69	0.00	20	0.58	215.69	20	0.00	215.69	19	0.00	0.13	215.69	20	0.00	215.69	18	0.00	0.06
20	288.56	288.56	288.56	0.00	20	2.61	288.56	20	0.00	288.56	19	0.00	0.21	288.56	20	0.00	288.60	17	0.02	0.11
25	311.45	311.45	311.45	0.00	20	22.19	311.45	20	0.00	311.61	18	0.04	0.37	311.45	20	0.00	311.92	14	0.12	0.21
30	417.52	417.52	417.23	0.05	19	911.08	417.52	20	0.00	417.57	17	0.01	0.63	417.52	20	0.00	417.85	10	0.08	0.42
35	493.21	493.21	488.17	1.00	12	4226.34	493.21	20	0.00	493.33	13	0.02	0.89	493.31	19	0.02	493.76	10	0.11	0.58
40	563.13	563.13	548.05	2.67	3	6495.88	563.01	20	-0.02	563.19	12	0.01	1.20	563.01	20	-0.02	563.48	6	0.06	0.79
45	644.00	644.22	615.61	4.40	0	7200.00	643.89	19	-0.01	644.62	9	0.10	1.51	644.05	19	0.01	645.47	6	0.23	0.99
50	720.13	720.66	680.46	5.55	0	7200.00	719.80	19	-0.04	720.38	10	0.04	1.97	719.77	20	-0.04	721.05	5	0.14	1.28
Avg.	422.70	422.79	412.87	1.518	12.67	2895.44	422.64	19.78	-0.008	422.84	15.11	0.026	0.78	422.67	19.78	-0.004	423.16	11.67	0.083	0.50

Table 3.3: Comparison between MGA + ILS and ILS + MGA.

Cust.	BKS	MGA + ILS							ILS + MGA						
		minimum			average				minimum			average			
		val	#B	gap	val	#B	gap	time	val	#B	gap	val	#B	gap	time
10	150.643	150.64	20	0.00	150.64	20	0.00	0.07	150.64	20	0.00	150.64	20	0.00	0.07
15	215.687	215.69	20	0.00	215.69	20	0.00	0.11	215.69	20	0.00	215.69	19	0.00	0.12
20	288.557	288.56	20	0.00	288.56	19	0.00	0.18	288.56	20	0.00	288.58	19	0.01	0.20
25	311.451	311.45	20	0.00	311.47	19	0.01	0.27	311.45	20	0.00	311.56	16	0.03	0.30
30	417.516	417.55	19	0.01	417.66	16	0.04	0.39	417.52	20	0.00	417.62	16	0.03	0.40
35	493.212	493.21	20	0.00	493.36	14	0.03	0.53	493.23	19	0.00	493.44	12	0.05	0.51
40	563.125	563.01	20	-0.02	563.08	17	-0.01	0.69	563.01	20	-0.02	563.30	12	0.03	0.69
45	644.002	643.86	20	-0.02	644.27	11	0.05	0.85	643.86	20	-0.02	644.57	8	0.09	0.87
50	720.131	719.77	20	-0.04	720.41	11	0.04	1.06	719.80	19	-0.04	720.98	5	0.12	1.09
Avg.	422.70	422.64	19.89	-0.008	422.79	16.33	0.017	0.46	422.64	19.78	-0.008	422.93	14.11	0.039	0.47

even if the computing time increases (on average about 911 seconds). As expected, instances that contain more customers are more difficult: in particular, no instance with 45 or more customers can be solved to optimality within the time limit of two hours. The average percentage gap is rather small for instances with up to 40 customers, while it increases up to 5.55% for instances with 50 customers.

When we consider the average results obtained, for each instance, by the ILS algorithm over the 10 runs, we can see that several best solutions can be found (for more than half of the instances on average), and that the average gap from UB is very small: only 0.083% on average. The computing time is very short, being 0.50 seconds on average. Given the very short computing time, the ILS algorithm can be executed 10 times for every instance to select the solution found in the best of the 10 runs. In this case, the best solutions can be found, on average, for 19.78 (over 20) instances in 5 seconds, i.e., the ILS algorithm obtains the best solution for 98.9% of the instances, and the global average gap is slightly negative (-0.004%). In addition, we notice that for instances with 40 and 50 customers, the average gap is negative, meaning that the ILS algorithm is able to obtain better solutions than those obtained by the C&B algorithm in much larger computing times. As for the performance of the MGA algorithm, by considering the average results obtained over 10 runs, we can see that the number of best solutions found and the average gap from UB are slightly greater (15.11) and smaller (0.026%), respectively, than those of ILS. The computing time is slightly greater than that of ILS, being 0.78 seconds on average. In the case of the minimum results, the best solutions can be found, on average, for 19.78 instances in around 8 seconds, and the gap is slightly smaller than that of ILS (-0.008%).

In Table 3.3, we report additional computational results obtained by a new algorithm (called MGA + ILS) which combines MGA and ILS: in particular, we remove the part that iteratively solves the LP-relaxation of model (3.2)-(3.16), that takes more or less half of the spent computing times, and we solve only one LP-relaxation. Once the optimal solution x of the LP-relaxation has been obtained, it is used to build a feasible solution as described in the ILS algorithm. Then, the feasible tour is given to MGA, and the best solution found by MGA is improved by the second phase of ILS. The additional algorithm ILS + MGA is analogous to algorithm MGA + ILS, but the two phases (initialization and improvement) are taken, respectively from ILS and MGA. We observe that the performance of ILS + MGA is in general very similar to that of the best previously presented algorithm (i.e. MGA): the average gap (0.039%) is slightly greater and the number of best solutions found is slightly smaller, but the computing time is a bit shorter (0.47 seconds on average). Finally, MGA + ILS is in all the aspects slightly better than all the considered algorithms, being more robust (the average gap is 0.017%) and faster (the computing time is 0.46 seconds).

3.6 Conclusions

We proposed two mataheuristic algorithms to solve the PTSP: ILS and MGA. The ILS algorithm starts by building a feasible tour: it is computed by using the Linear Programming (LP) solution of a Mixed Integer Linear Programming (MILP)

model for the PTSP, that contains exponentially many sub-tour elimination constraints. Then, the ILS algorithm loops between three phases: perturbation, local search and acceptance criterion. The MGA is implemented with four crossover and mutation operators, that allow the generation of good solutions within short computing times. In addition, its multi-operator nature can avoid local optima traps in the search of the global optimum.

We tested both algorithms on instances with up to 50 customers adapted from those proposed in Demir et al. (2012) for the Pollution Routing Problem. To evaluate the performance of ILS and MGA we developed a Cut-and-Branch algorithm, in which subtour elimination constraints are added at the root node. The obtained results show that the Cut-and-Branch algorithm is able to obtain the optimal solution for instances with up to 30 customers. However, no instance with 45 or more customers can be solved to optimality within two hours of time limit. The MGA and ILS algorithms are very effective, as they are able to obtain (on average in 8 and 5 seconds, respectively) the best solution for about 99% of the instances. In addition, we combined MGA and ILS into a new mataheuristic algorithm (called MGA + ILS) that obtains the best performance among the considered algorithms. Future work will be devoted to develop exact methods that combine ILS or MGA with the Cut-and-Branch algorithm in order to solve instances with a larger number of customers. In addition, our algorithms could be embedded into the algorithms proposed for the solution of the PRP.

Chapter 4

An algorithm to solve the Multi-depot Waste Collection Problem with Stochastic Demands

4.1 Introduction

Uncollected garbage, pollution and traffic jams through extensive vehicle usage as well as high operative costs are some of the main problems arising through sub-optimal waste collection. For this reason, waste management is among the most important municipal services, especially in urban areas facing increasing waste production and growing population numbers (United Nations Population Fund, 2011; Malakahmad et al., 2014; Son, 2014). In an attempt to optimize urban waste collection, the waste collection problem (WCP) as essential part of managing the garbage supply chain has been addressed by different researchers in the past as special instance of the well-known vehicle routing problem (VRP) in which a vehicle fleet located at a central depot has to serve a number of customers (Beliën et al., 2014; Ghiani et al., 2014).

The VRP is one of the most studied combinatorial optimization problems. Given a fleet of capacitated vehicles located at a central depot n_0, a set of n customers $I = \{n_1, n_2, \ldots, n_n\}$ with a given demand $d_i \geq 0$ at each customer $i \in I$ that has to be served. The travel cost between any two nodes in $V = I \cup \{n_0\}$ (e.g. distance, time, etc.) is known and represented by $c_{ij}(i, j \in V)$. While the objective function is to minimize costs, problem constraints include that every customer is served by only one vehicle, all routes start and end at the central depot and that no vehicle can stop twice at the same customer (Toth & Vigo, 2014). Instead of delivering demand (i.e. waste levels), the aim of the WCP is to empty its clients (i.e. waste containers) according to a specific routing plan. As can be seen in Figure 4.1, collection vehicles need to empty themselves before returning to the depot (Route 1), or once the vehicle capacity is reached (Route 2). Note that multiple garbage disposal trips on a single route are possible. The collected waste is disposed at one of m landfills, included in the original node-set V by adding the

sub-set $L = \{n_{(n+1)}, n_{(n+2)}, \ldots, n_{(n+m)}\}$ to the original number of nodes.

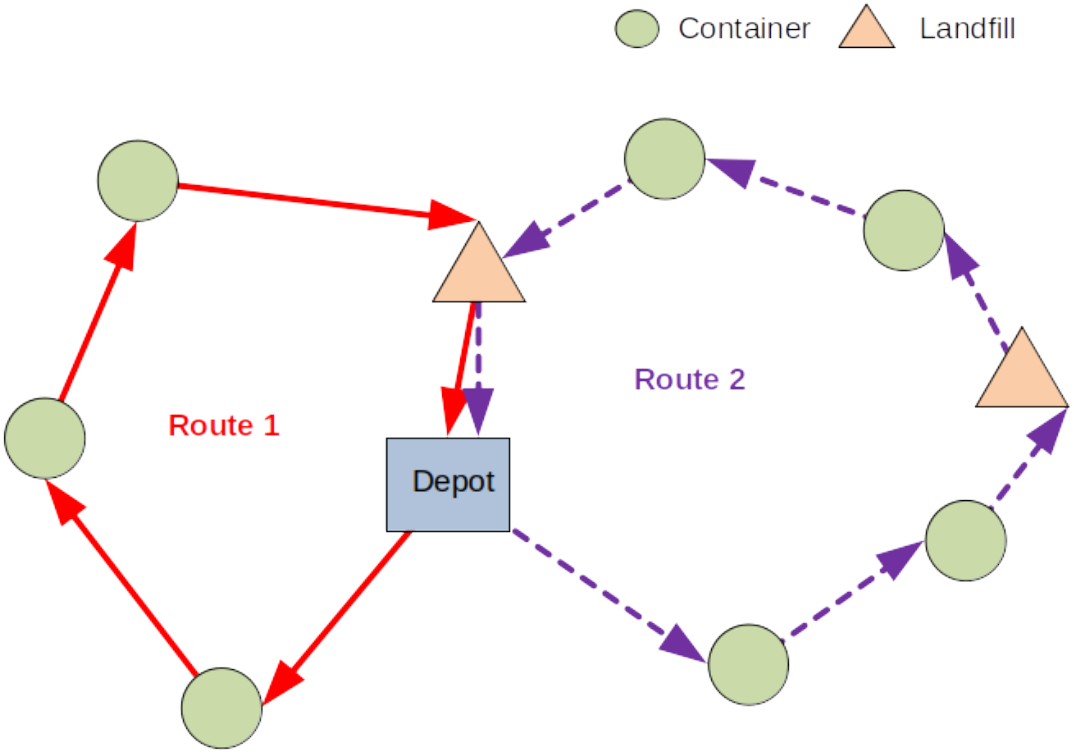


Figure 4.1: Example of a WCP solution with 2 routes.

Despite the importance of managing garbage and different approaches to optimize waste collection route planning, there is a lack of scientific publications considering rich and realistic WCP environments, e.g.: scenarios with a large number of clients, multiple vehicle depots, and uncertainty in waste levels or demands. In order to tackle the WCP with stochastic demands (WCPSD), this chapter proposes a hybrid algorithm based on the simheuristic framework described in Grasas et al. (2016), which has proven to be an effective simulation-optimization approach for different kinds of stochastic combinatorial optimization problems. In this chapter, Monte Carlo simulation is integrated into a metaheuristic framework in order to estimate the expected cost and cost variability of promising routing plans initially generated for the deterministic version of the problem. Our simheuristic algorithm also makes use of biased (oriented) randomization techniques, similar to those proposed in Faulin et al. (2008), in order to guide the random process that generates new solutions. As discussed in Juan et al. (2013b), biased-randomized algorithms are easily parallelizable and able to provide real-time solutions to complex vehicle routing problems with stochastic demands.

As far as we know, the algorithm introduced here is the first one addressing the multi-depot WCP with stochastic demands (MDWCPSD), which is closely related to real-life waste collection activity. Additionally, our approach allows managers

This chapter is based on the contents of: Gruler, Fikar, Juan, Hirsch, & Contreras-Bolton, Supporting multi-depot and stochastic waste collection management in clustered urban areas via simulation–optimization. Journal of Simulation, 11 (1), 11–19, 2017.

to quantify the potential cost savings derived from using horizontal cooperation (HC) strategies, i.e. the sharing of information and resources among different waste management service providers.

This chapter is structured as follows: Section 4.2 provides an overview on related works regarding simulation-optimization, waste collection, and horizontal cooperation. The algorithm to solve the MDWCPSD is introduced in Section 4.3, while Section 4.4 describes the computational experiments used to validate our findings. The obtained results are discussed in Section 4.5. Finally, Section 4.6 summarizes the highlights of this work.

4.2 Related studies

4.2.1 Combining simulation with metaheuristics

The need for advanced optimization methods to solve real-life problem settings, characterized by stochastic inputs and probability constraints, in combination with enhanced computational possibilities, has led to the development of different hybrid simulation-optimization approaches. In this context, the combination of simulation techniques with optimization methods in which simulation acts as an evaluation function for the optimization-outputs has proven to be a fast, reliable, and easy-to use approach in the solution of large sized optimization problems under uncertainty (Andradóttir, 2006).

A general methodology combining metaheuristics and simulation, so called simheuristics, is discussed in detail by (Juan et al., 2015a,b). A particular adaptation in the case of the iterated local search (ILS) metaheuristic is given in Grasas et al. (2016). Through a close integration between simulation and optimization, this approach has been successfully applied to different problem settings. Related to transportation and logistics, Juan et al. (2011) test the effects on safety stocks for VRPs under uncertainty. A simheuristic approach to solve the complex inventory routing problem combining routing and inventory management decisions is suggested and analysed in Juan et al. (2014). Also, regarding other application fields, the combination of metaheuristics with simulation has been implemented, for example, in production planning (Figueira et al., 2013), stochastic scheduling (Juan et al., 2014), or in the improvement of internet computing services (Cabrera et al., 2014).

4.2.2 The waste collection problem

The WCP formulated as optimization problem has been widely discussed in the scientific literature. In an extensive literature review on nearly 80 publications on garbage collection since 1970, Beliën et al. (2014) summarize theoretical and practical research work on the WCP as location and routing problems, classified according to different dimensions (e.g. type of waste or solution methodologies). Concerning the WCP as routing problem in the collection of household waste, an insertion metaheuristic for large sized (deterministic) WCP settings with a single depot and multiple landfills is presented by Kim et al. (2006), while Benjamin &

Beasley (2010) combine tabu with variable neighbourhood search. Further concerning the solution of deterministic WCPs, Buhrkal et al. (2012) suggest an adaptive large neighbourhood search metaheuristic with destroy and repair techniques.

The WCP with stochastic demands with small customer sets based on a case study in Malaysia is addressed by Ismail & Loh (2009) by applying an ant colony optimization metaheuristic. In this study, waste levels are using a discrete probability distribution and revealed once collection vehicles reach the designated pick-up points defined as a priori solution, which is then undergoing recourse actions when necessary. A similar approach to the same problem (with up to 50 customers) is solved by (Ismail & Irhamah, 2008) using a hybrid approach based on a genetic algorithm and tabu search.

4.2.3 Multi-depot VRPs

VRPs with multiple depots (MDVRP) have been subject to research for many decades. Hereby, customers are allocated to different depots in a combinatorial assignment problem, before each depot-node assignment combination is solved as VRP. Pisinger & Ropke (2007) propose an adaptive large neighbourhood search metaheuristic which they test on variance instance obtaining promising results, while requiring the fine-tuning of as much as 14 parameters. Vidal et al. (2012) use a hybrid genetic algorithm to solve different VRP variants, among them the MDVRP. The integration of biased (oriented) randomization techniques inside an ILS framework designed to solve routing problems with multiple depots is discussed in Juan et al. (2015b). For more general overviews over solution methods to the MDVRP, the reader is addressed to Kumar & Panneerselvam (2012) and Karakatič & Podgorelec (2015).

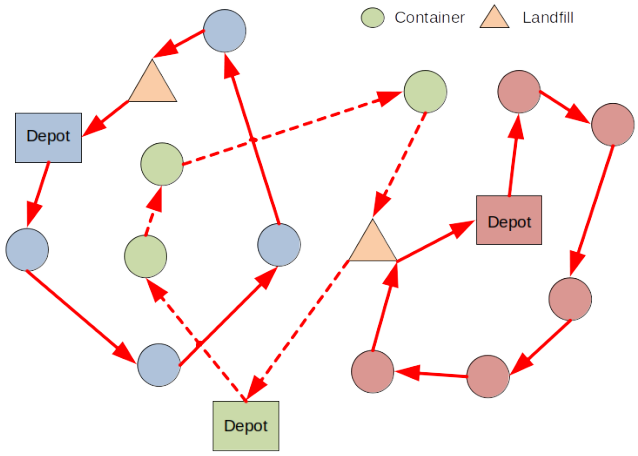
While the MDVRP is a widely studied problem, the literature is scarce for the special case of waste collection routing, even non-existent for the MDWCPSD. A closely related problem setting to the MDWCP is discussed by Crevier et al. (2007), who apply an adaptive memory principle combined with integer programming and tabu search metaheuristics to consider the MDVRP with inner-route replenishment points. Hemmelmayr et al. (2013) consider multiple depots and landfills in a case study for node routing for the collection of waste from public waste delivery points to landfills by applying a hybrid approach. Combining variable neighbourhood search and dynamic programming, the authors solve small problem instances with up to 75 collection points. Additionally, Ramos et al. (2014) improve the economic and environmental impact of recyclable waste collection by solving the WCP with multiple depots as a mixed integer-linear programming model.

4.2.4 Horizontal cooperation

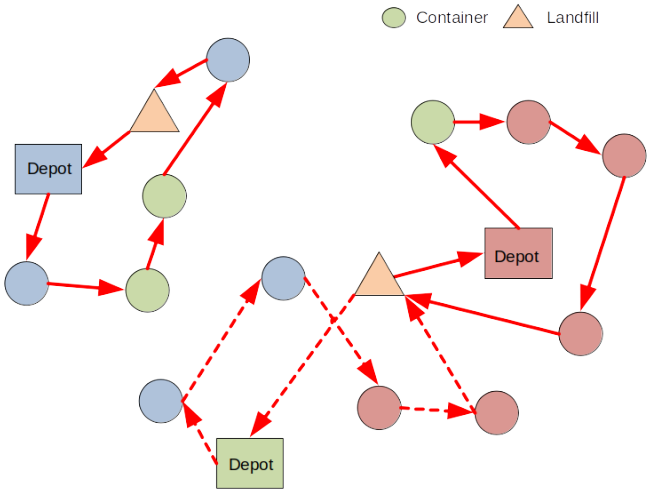
In many supply chains from different sectors, Horizontal Cooperation (HC) is being discovered as a tool to reduce overall transportation costs, vehicle usage and the environmental impact of transportation activities. HC is defined by Bahinipati et al. (2009) as "business agreement between two or more companies at the same level in the supply chain or network in order to allow ease of work and co-operation towards achieving a common objective". By sharing information and facilities, companies

hope for the effects of economies of scale through better resource usage. In this context, HC yields several potential applications concerning the development of sustainable logistics networks in different sectors, e.g., by public service providers (Ballot & Fontane, 2010; Joyce & Drumaux, 2014; Pérez-Bernabeu et al., 2015).

A simple example of the benefits of HC in waste collection can be seen in Figure 4.2. In the non-cooperative scenario shown on (a), waste management service providers (each represented by a single depot) do not share information about waste containers to be emptied and plan their respective collection routes individually. It can be intuitively observed that the cooperative scenario (b) in which containers, landfills, and depots are shared leads to lower routing costs, less traffic, higher service levels and a lower environmental impact through the waste transportation activities. Applications of HC in waste collection can be thought of especially in large clustered cities with various waste management service providers responsible for different areas of a municipality and highly clustered metropolitan areas (e.g. the Ruhr-area in Germany or greater Barcelona).



(a) Non-cooperative



(b) Cooperative

Figure 4.2: Scenarios in waste management.

4.3 Overview of our simulation-optimization algorithm

Our methodology to tackle the complex MDWCPSD is based on the SimILS general framework described in Grasas et al. (2016). This is a simheuristic framework combining biased (oriented) randomization, the ILS metaheuristic, and Monte Carlo simulation. Thus, at different stages of the algorithm, Monte Carlo simulation is hereby applied to consider the performance of a predefined deterministic solution in a stochastic environment, as will be explained below in more detail. The rest of this section provides additional details of our approach and on the particular simheuristic algorithm proposed.

4.3.1 Simheuristics to consider WCP uncertainty

In contrast to a deterministic WCP setting in which input variables are considered to be known during the route planning phase, actual garbage loads to collect are only revealed once a collection vehicle reaches a pick up point in reality. Given a WCPSD, the first step of the simheuristic procedure is to transform the stochastic problem setting into its deterministic counterpart. That is, the expected waste levels of each container on a waste collection route are used as deterministic input variables. This deterministic WCP is then solved using an efficient optimization approach. Therefore, we apply a randomized version of the well-known Clarke-and-Wright savings (CWS) heuristic (Clarke & Wright, 1964), as discussed in Juan et al. (2013a). By adapting this multi-start routing algorithm to the WCP, a number of predefined (a priori) solutions are constructed during `detSolTime` seconds. This procedure is shown in Algorithm 4.1.

Algorithm 4.1 Generation of solutions

```
1:  procedure generateEliteSolutions(Map)
2:      S ← {∅}                                          ▷ set of elite solutions
3:      while terminate(nDetSol, detSolTime) = false do
4:          sol ← solveBiasedRandomizedCWS( )
5:          if isEliteSolution(sol, d) = true then   ▷ scriptsize save d best deterministic sol
6:              S ← S ∪ sol
7:          end if
8:      end while
9:      for i ← 1 to d do                          ▷ run short simulation on promising sol
10:         evaluateReliability(S_i, Var, nIterShort)
11:     end for
12:     S ← resortSolutions(S)
13:     return S                                          ▷ return elite solutions
14: end procedure
```

After the generation of multiple solutions, a set of d deterministic solutions is tested for its behaviour in a stochastic environment by repeatedly sampling random waste levels using Monte Carlo simulation (Raychaudhuri, 2008). Note that not all deterministic solutions are tested at this point to keep the computational efforts reasonable, i.e., only the most 'promising' deterministic solutions are sent to the

simulation stage. Hereby, the waste levels of each container to be served are modelled using a log-normal distribution with waste level variance `Var` and the expected waste level at each container as mean. During `nIterSim` (usually several hundred/thousands) simulation runs, a tested predefined solution is completed considering the uncertainty of pick-up amounts by only revealing the actual waste level at a container once a collection vehicle reaches it. Note hereby that the order of containers to visit is not changed. However, limited vehicle capacities will lead to route failures when collected garbage exceeds the initially planned amount, making additional landfill trips necessary. Our solution algorithm penalizes route failures with an additional round trip to the closest landfill, before the planned route is continued.

By summing total route failure costs in all simulation runs and dividing them by `nIterSim` an unbiased estimation of expected route failures can be obtained. At this stage, it often happens that the most promising deterministic WCP solutions turn out to be less competitive in a stochastic environment due to high expected additional routing costs. Beside expected route failure costs as quality indicator of the predefined route under uncertainty, the described procedure allows for an estimation of the solution reliability by considering the proportion of routes in which a route failure occurs during the simulation runs. Finally, the solutions are re-ranked according to the total of routing and expected route failure costs. If necessary at this point, a longer simulation run can be run with the most promising stochastic solutions to get more reliable results of a number of stochastic elite solutions.

4.3.2 Combination of simulation with oriented randomization and ILS

In order to consider multiple depots in the planning of waste collection routes with stochastic waste levels, we integrate oriented randomization (Faulin et al., 2008) and ILS at different stages of the simulation-optimization approach, as shown in Algorithm 4.2. Consider a WCP with multiple depots (V_d), the first step in the solution process is to solve the node-depot assignment problem, defining which waste container is emptied from which depot. For this purpose, a priority list of containers for each depot is constructed according to a distance based criterion. After calculating the distance of each waste container to each depot $k \in V_d$, a priority list based on the marginal savings μ_i^k of emptying container i from depot k compared to serving it from the best alternative depot k^*, such that $\mu_i^k = c_i^{k^*} - c_i^k$ is build. Next, the nodes are randomly assigned to the depots according to a round-robin criterion. That is, the depots iteratively 'choose' an (unassigned) container to serve.

As suggested in Juan et al. (2015b), the process of assigning nodes to a depot is randomized but using a skewed distribution instead of a uniform one. After sorting the priority list of containers, a skewed geometric distribution is applied in such a way that containers with a higher position in the priority list are more likely to be selected by the respected depot (thus making the randomized selection process 'biased' or 'oriented'). The exact probabilities of each container to be included in a specific node-depot collection map are defined through a distribution parameter

p ($0 < p < 1$). This randomized process allows the construction of `nMaps` different node-depot allocation maps. We evaluate the competitiveness of each map by solving the deterministic WCP setting for depot and its assigned nodes, before the WCP costs for each depot are summed to obtain a preliminary MDWCP solution. After all maps have been constructed the d most promising node-depot allocations are defined.

Algorithm 4.2 Multi-depot simheuristic approach

Require: V_d, I, L; ▷ depots, customers and landfills

$S \leftarrow \{\emptyset\}$ ▷ set of elite solutions
$D \leftarrow \{\emptyset\}$ ▷ set of depot-node allocation maps
$D \leftarrow$ GENERATEINITIALMAPS(`nMaps`) ▷ generate node-depot maps
$D \leftarrow$ GENERATEELITESOLUTIONS(D) ▷ run simheuristic on each map
while TIME( ) < `maxTImeILS` **do**
    $D' \leftarrow$ PERTURBATE(D') ▷ perturbate elite maps
    $S' \leftarrow$ GENERATEELITESOLUTONS(D')
    if ACCEPTNEWSOLUTION(S, S')) = `true` **then**
        $D \leftarrow D'$ ▷ update current map
        $S \leftarrow S'$ ▷ update elite solutions
    end if
end while
for $i \leftarrow 1$ **to** d **do** ▷ run extensive simulation on promising sol
    EVALUATERELIABILITY(S_i, `Var`, `nIterShort`)
end for
$S \leftarrow$ RESORTSOLUTIONS(S)
return S ▷ return elite solutions

With each promising solution, a short simulation run with `nIterShort` iterations is started to evaluate the performance of the predefined routing plan in a stochastic environment. Based on the deterministic routing and route failure costs, a stochastic initial solution is defined and used as initial upper limit (`currentBest`) in the ILS process started in the following. Based on the work of Grasas et al. (2016), the node-depot allocation maps are hereby perturbed by applying a destroy-and-repair method to the current best solution during a time based termination criteria (`maxTimeILS`). That is, during each perturbation, p^* percent of containers are exchanged between different node-depot allocations before it is re-simulated in a short simulation to assess the quality of the newly constructed map and compare it to the current best.

Whenever a new best solution considering total costs is found, the new solution is listed as promising (in `promisingSolsStoch`) and replaces the current best. In order to escape local minima in the solution search, an acceptance criterion is implemented allowing a solution worsening if the last iteration from x to x' (such that $f(x') < f(x)$) was an improvement and difference between the current best and new solution is not greater than the latest improvement ($|f(x'') - f(x')| < f(x) - f(x')$). Once the ILS is finished, the m best solutions in `promisingSolsStoch` undergo a long simulation run to define the expected route failure costs and the solution reliabilities more closely. Finally, a list of elite MDWCPSD with the respective total costs and solution reliabilities are returned by the process.

4.4 Computational experiments

In order to test our approach, we adopted the benchmarks provided by Kim et al. (2006). The original set of instances consider different sized (102-2100 nodes) deterministic WCPs with a single depot, multiple landfills and capacitated vehicles. To make them suitable for the MDWCPSD, we use the deterministic load levels at each container as expected waste levels and mean of the log-normal distribution from which waste levels are modelled during the simulation process. Furthermore, we consider that each landfill is a disposal site and vehicle depot at the same time, leading to different MDWCP instances. In order to limit depot capacities, not more than `maxDemand` percent of total waste to be collected is assigned to a single depot, in order to keep the workload balanced and limit the vehicles per depot. It has to be noted here that time windows for the waste containers and vehicle depots are not considered in this work.

The MDWCP can be seen as cooperative scenario in which different waste management service providers apply HC in which containers, depots and landfills are shared to reduce garbage collection costs. We quantify the benefits of HC by comparing the MDWCPSD costs to a non-cooperative scenario in which each depot serves its own customers and no resources are shared. For each depot of a problem instance, nodes are assigned through a greedy round-robin criterion to create a number of single-depot WCPs. That is, a priority list of nodes based on marginal savings (as described earlier) is established. At each assignment iteration however, the current depot chooses the node with the highest savings potential to include on the planned collection route, leading to a single-depot WCP in which waste containers are assigned to vehicle depots according to a distance-based criterion. In essence, this leads to one possible node-depot allocation map with multiple depots, whereby the routing costs are estimated through the procedure described in Algorithm 4.1. However, as the node-depot assignment problem is redundant in the non-cooperative scenario without HC, it is not possible to consider different container allocations and their respective routing costs.

The algorithm is implemented as Java application on a personal computer with an Intel BYT-M 4 Core processor with 2.66 GHz and 4GB RAM, using the following parameters:

- `nMaps`: 100 (initial node-depot assignment maps created)
- `Var`: 0.05 (waste level variance)
- d: 5 (promising deterministic solutions)
- m: 5 (promising stochastic solutions)
- `nIterSimShort`: 500 runs
- `nIterSimLong`: 5,000 runs
- `detSolTime`: 5 seconds
- `maxTimeILS`: 300 seconds
- p^*: 0.3% (nodes exchanged during perturbation)

- p: 0.4 (distribution parameter for biased randomization)
- `maxDemand`: 60%

In Table 4.1, the best found solutions for the cooperative and non-cooperative scenario of each instance (and one row with the total over all tested problem settings) are listed. The first two columns of the table correspond to problem name and scenario. The remaining columns correspond to the the number of vehicles used vehicles, opened depots, fixed costs, variable (route-failure) costs, and total costs, respectively. Furthermore, the percentage difference of total costs for each scenario is outlined.

Table 4.1: Table of results.

Instances	Scenarios	Vehicles	Depots	Fixed Costs	Variable Costs	Total Costs
	Cooperative	4	3	154.46	0.00	154.46
Kim102	Non-Cooperative	3	3	190.99	0.28	191.27
	%-Difference					-0.19
	Cooperative	2	2	495.47	17.59	513.06
Kim277	Non-Cooperative	2	2	503.52	12.54	516.06
	%-Difference					-0.01
	Cooperative	6	3	194.99	1.38	196.37
Kim335	Non-Cooperative	8	5	304.88	1.77	306.65
	%-Difference					-0.36
	Cooperative	11	2	74.61	0.85	75.45
Kim444	Non-Cooperative	11	2	79.00	0.91	79.91
	%-Difference					-0.06
	Cooperative	9	9	925.02	5.57	930.59
Kim804	Non-Cooperative	20	20	1579.69	3.66	1583.35
	%-Difference					-0.41
	Cooperative	17	2	2133.22	55.36	2188.58
Kim1051	Non-Cooperative	18	3	226.68	46.74	2273.42
	%-Difference					-0.04
	Cooperative	9	4	902.00	12.80	914.80
Kim1351	Non-Cooperative	9	4	909.28	27.61	936.89
	%-Difference					-0.02
	Cooperative	12	3	1098.30	21.57	1119.87
Kim1599	Non-Cooperative	13	3	1307.04	19.53	1326.57
	%-Difference					-0.16
	Cooperative	9	5	1105.75	14.28	1120.03
Kim1932	Non-Cooperative	9	5	1119.67	17.21	1136.88
	%-Difference					-0.01
	Cooperative	12	8	1560.26	9.00	1569.26
Kim2100	Non-Cooperative	12	8	1573.71	10.08	1583.79
	%-Difference					-0.01
	Cooperative	91	41	8644.08	138.40	8782.47
Total	Non-Cooperative	105	55	9794.46	140.33	9934.79
	%-Difference					-0.12

4.5 Discussion of results

The obtained results concerning total costs, used vehicles, and opened depots for the cooperative scenario (i.e. the MDWCPSD solution) and the non-cooperative counterpart are outlined in Figure 4.3. Concerning the total expected costs, the routing costs of each instance can be improved, with an average gap of -12% over all instances. Next to possible route savings, HC leads to less used vehicles (91 compared to 105 in the non-cooperative scenario over all instances) and less opened depots (41 to 55). While the objective function consider in this chapter was to reduce total routing costs, the number of used vehicles and opened depots can also be an important driver for HC in practice, as municipalities need to consider acquisition- and maintenance-costs for collection vehicle. Moreover, opening costs and running expenses of vehicle depots should not be neglected. Looking at the results more closely, the large differences on the positive effects of HC concerning total costs can be explained with the topology, i.e., the geographical distribution of the containers with respect to the potential depots. Obviously, instances in which only scattered depot-node allocation maps can be established tend to yield much better results in cooperative scenarios.

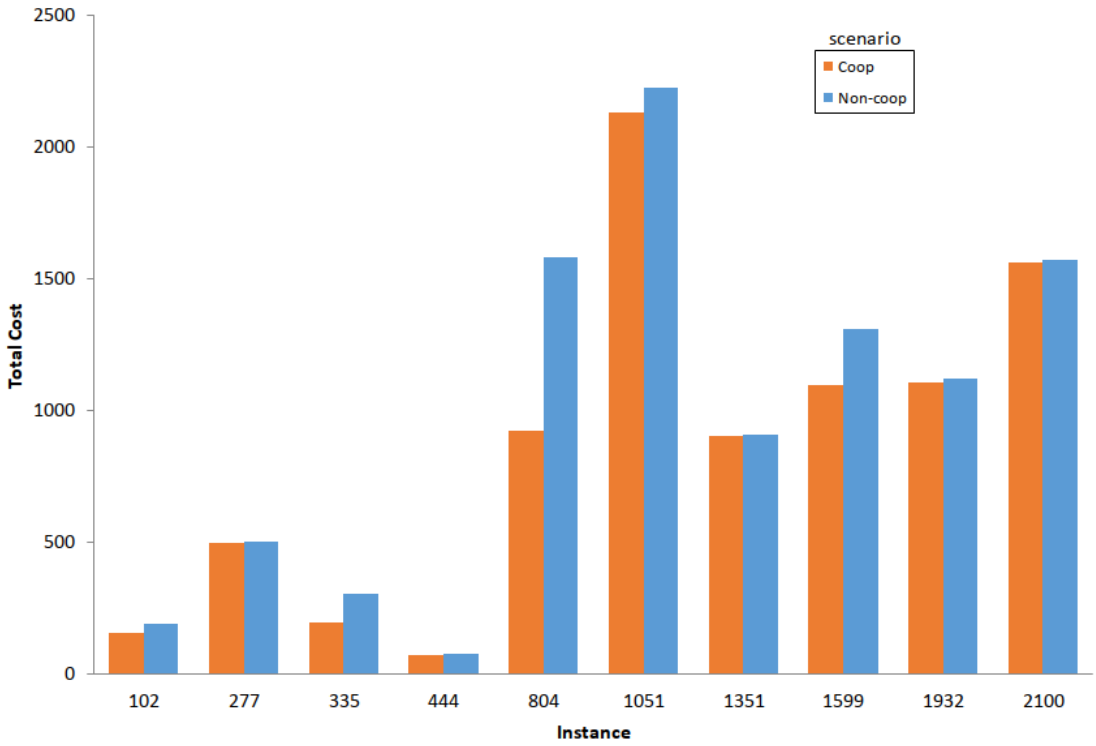


Figure 4.3: Comparison of total costs between cooperative and non-cooperative scenarios.

4.6 Conclusions

This chapter has presented a simulation-optimization approach to support efficient waste collection management under scenarios characterized by the existence

of multiple depots and stochastic demands (waste levels). Despite the fact that these scenarios are common in clustered urban areas and large cities, they had never been analysed before in the scientific literature. This is probably due to their inherent complexity, which requires the use of hybrid simulation-optimization methods. Thus, our approach relies on the combination of biased (oriented) randomization techniques, metaheuristics, and Monte Carlo simulation. The proposed hybrid algorithm was tested on a set of large-sized WCP instances. Furthermore, the possible benefits of horizontal cooperation among different waste management service providers are analysed. These benefits refer to savings in total costs, number of used vehicles, and number of opened depots. Although it is evident that these savings are highly dependent on the specific topology and characteristics of each instance, the computational experiments show that our algorithm is able to quantify them for all the proposed benchmarks, thus supporting the idea that sharing resources (i.e., containers, landfills, depots, and/or vehicles) can provide significant savings in those cases in which depots and their assigned containers are geographically scattered.

Bibliography

Abeledo, H., Fukasawa, R., Pessoa, A., & Uchoa, E. (2013). The time dependent traveling salesman problem: polyhedra and algorithm. *Mathematical Programming Computation*, *5*(1), 27–55.

Albayrak, M., & Allahverdi, N. (2011). Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms. *Expert Systems with Applications*, *38*(3), 1313–1320.

Anbuudayasankar, S. P., Ganesh, K., & Mohapatra, S. (2014). *Models for Practical Routing Problems in Logistics: Design and Practices*. Springer International Publishing.

Andradóttir, S. (2006). An overview of simulation optimization via random search. In S. G. Henderson, & B. L. Nelson (Eds.) *Simulation*, vol. 13 of *Handbooks in Operations Research and Management Science*, (pp. 617–631). Elsevier.

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press.

Arigliano, A., Calogiuri, T., Ghiani, G., & Guerriero, E. (2018). A branch-and-bound algorithm for the time-dependent travelling salesman problem. *Networks*, *72*, 382–392.

Arigliano, A., Ghiani, G., Grieco, A., & Guerriero, E. (2015). Time dependent traveling salesman problem with time windows: Properties and an exact algorithm. Tech. rep., Optimization Online.

Bahinipati, B. K., Kanda, A., & Deshmukh, S. (2009). Horizontal collaboration in semiconductor manufacturing industry supply chain: An evaluation of collaboration intensity index. *Computers & Industrial Engineering*, *57*(3), 880–895.

Ballot, E., & Fontane, F. (2010). Reducing transportation co2 emissions through pooling of supply networks: perspectives from a case study in french retail chains. *Production Planning & Control*, *21*(6), 640–650.

Banzhaf, W. (1990). The "molecular" traveling salesman. *Biological Cybernetics*, *64*(1), 7–14.

Bektaş, T., & Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B: Methodological*, *45*(8), 1232–1250.

Beliën, J., De Boeck, L., & Van Ackere, J. (2014). Municipal solid waste collection and management problems: A literature review. *Transportation Science*, *48*(1), 78–102.

Benjamin, A., & Beasley, J. (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, *37*(12), 2270–2280.

Bigras, L.-P., Gamache, M., & Savard, G. (2008). The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization*, *5*(4), 685–699.

Boland, N., Hewitt, M., Marshall, L., & Savelsbergh, M. (2017). The continuous-time service network design problem. *Operations Research*, *65*(5), 1303–1321.

Buhrkal, K., Larsen, A., & Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia - Social and Behavioral Sciences*, *39*, 241–254.

Buriol, L., França, P. M., & Moscato, P. (2004). A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, *10*(5), 483–506.

Cabrera, G., Ángel Juan, Lázaro, D., Marquès, J. M., & Proskurnia, I. (2014). A simulation-optimization approach to deploy internet services in large-scale systems with user-provided resources. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, *90*(6), 644–659.

Cacchiani, V., Contreras-Bolton, C., Escobar, J., Escobar-Falcon, L. M., Linfati, R., & Toth, P. (2018a). An iterated local search algorithm for the pollution traveling salesman problem. In P. Daniele, & L. Scrimali (Eds.) *New Trends in Emerging Complex Real Life Problems: ODS, Taormina, Italy, September 10–13, 2018*, (pp. 83–91). Cham: Springer International Publishing.

Cacchiani, V., Contreras-Bolton, C., & Toth, P. (2018b). Models and algorithms for the traveling salesman problem with time-dependent service times. *Submitted to European Journal of Operational Research*, (pp. 1–30).

Ceder, A. (2011). Public-transport vehicle scheduling with multi vehicle type. *Transportation Research Part C: Emerging Technologies*, *19*(3), 485–497.

Christodoulou, S. E. (2010). Traffic modeling and college-bus routing using entropy maximization. *Journal of Transportation Engineering*, *136*(2), 102–109.

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

Contreras-Bolton, C., & Parada, V. (2015). Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem. *PLOS ONE*, *10*(9), 1–25.

Cordeau, J.-F., Ghiani, G., & Guerriero, E. (2012). Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, *48*(1), 46–58.

Crevier, B., Cordeau, J.-F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, *176*(2), 756–773.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, *2*(4), 393–410.

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

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*(2), 346–359.

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, *10*(1), 27–36.

Eglese, R., & Bektaş, T. (2014). Green vehicle routing. In P. Toth, & D. Vigo (Eds.) *Vehicle Routing: Problems, Methods, and Applications*, (pp. 437–458). Philadelphia, PA, USA: SIAM.

Eglese, R., & Black, D. (2015). Optimizing the routing of vehicles. In A. McKinnon, M. Browne, M. Piecyk, & A. Whiteing (Eds.) *Green logistics: Improving the Environmental Sustainability of Logistics*, (pp. 229–242). Kogan Page, 3rd ed.

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *3*(2), 124–141.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., & Smith, J. E. (2007). Parameter Control in Evolutionary Algorithms. In F. Lobo, C. Lima, & Z. Michalewicz (Eds.) *Parameter Setting in Evolutionary Algorithms*, vol. 54 of *Studies in Computational Intelligence*, (pp. 19–46). Springer Berlin Heidelberg.

Eiben, A. E., & Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, 2nd ed.

European Environment Agency (2011). Laying the foundations for greener transport – TERM 2011: transport indicators tracking progress towards environmental targets in Europe. Tech. rep., Office for Official Publ. of the European Communities, Copenhagen, Denmark.

Faulin, J., Gilibert, M., Juan, A. A., Vilajosana, X., & Ruiz, R. (2008). Sr-1: A simulation-based algorithm for the capacitated vehicle routing problem. In *2008 Winter Simulation Conference*, (pp. 2708–2716).

Figueira, G., Furlan, M., & Almada-Lobo, B. (2013). Predictive production planning in an integrated pulp and paper mill. *IFAC Proceedings Volumes*, *46*(9), 371–376.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, *4*(1), 61–75.

Fogel, D. (1988). An evolutionary approach to the traveling salesman problem. *Biological Cybernetics*, *60*(2), 139–144.

Fogel, D. (1993). Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and Systems*, *24*(1), 27–36.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY, USA: W. H. Freeman & Co., 1 ed.

Gavish, B., & Graves, S. (1978). The travelling salesman problem and related problems. *Technical Report GR-078-78, Operations Research Center, MIT*.

Ghiani, G., Laganà, D., Manni, E., Musmanno, R., & Vigo, D. (2014). Operations research in solid waste management: A survey of strategic and tactical issues. *Computers & Operations Research*, *44*, 22–32.

Gouveia, L., & Voß, S. (1995). A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, *83*(1), 69–82.

Grasas, A., Juan, A. A., & Lourenço, H. R. (2016). Simils: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, *10*(1), 69–77.

Grefenstette, J. J., Gopal, R., Rosmaita, B. J., & Gucht, D. V. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, (pp. 160–168). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Groba, C., Sartal, A., & Vázquez, X. H. (2015). Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *Computers & Operations Research*, *56*, 22–32.

Gruler, A., Fikar, C., Juan, A. A., Hirsch, P., & Contreras-Bolton, C. (2017). Supporting multi-depot and stochastic waste collection management in clustered urban areas via simulation–optimization. *Journal of Simulation*, *11*(1), 11–19.

Hemmelmayr, V., Doerner, K. F., Hartl, R. F., & Rath, S. (2013). A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics*, *19*(2), 129–156.

Hoornweg, D., & Bhada-Tata, P. (2012). What a waste: a global review of solid waste management. Tech. rep., World Bank, Washington, DC.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, *144*(2), 379–396.

Inghels, D., Dullaert, W., & Vigo, D. (2016). A service network design model for multimodal municipal solid waste transport. *European Journal of Operational Research*, *254*(1), 68–79.

Ismail, Z., & Irhamah, I. (2008). Solving the vehicle routing problem with stochastic demands via hybrid genetic algorithm-tabu search. *Journal of Mathematics and Statistics*, *4*(3), 161–167.

Ismail, Z., & Loh, S. (2009). Ant colony optimization for solving solid waste collection scheduling problems. *Journal of Mathematics and Statistics*, *5*(3), 199–205.

Izzo, D., Getzner, I., Hennes, D., & Simões, L. F. (2015). Evolving solutions to tsp variants for active space debris removal. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, (pp. 1207–1214). New York, NY, USA: ACM.

Joyce, P., & Drumaux, A. (2014). *Strategic management in public organizations*. New York: Routledge.

Juan, A., Faulin, J., Grasman, S., Riera, D., Marull, J., & Mendez, C. (2011). Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C: Emerging Technologies*, *19*(5), 751–765.

Juan, A. A., Barrios, B. B., Vallada, E., Riera, D., & Jorba, J. (2014). A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, *46*, 101–117.

Juan, A. A., Faulin, J., Ferrer, A., Lourenço, H. R., & Barrios, B. (2013a). Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP*, *21*(1), 109–132.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015a). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, *2*, 62–72.

Juan, A. A., Faulin, J., Jorba, J., Caceres, J., & Marquès, J. M. (2013b). Using parallel & distributed computing for real-time solving of vehicle routing problems with stochastic demands. *Annals of Operations Research*, *207*(1), 43–65.

Juan, A. A., Pascual, I., Guimarans, D., & Barrios, B. (2015b). Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. *International Transactions in Operational Research*, *22*(4), 647–667.

Karakatič, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, *27*, 519–532.

Karp, R. (1972). Reducibility among combinatorial problems. In R. Miller, J. Thatcher, & J. Bohlinger (Eds.) *Complexity of Computer Computations*, The IBM Research Symposia Series, (pp. 85–103). Springer US.

Kim, B.-I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, *33*(12), 3624–3642.

Kim, G., Ong, Y., Heng, C. K., Tan, P. S., & Zhang, N. A. (2015). City vehicle routing problem (city vrp): A review. *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 1654–1666.

Kramer, R., Subramanian, A., Vidal, T., & dos Anjos F. Cabral, L. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, *243*(2), 523–539.

Kumar, S. N., & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, *4*(3), 66–74.

Laporte, G., Ropke, S., & Vidal, T. (2014). Heuristics for the Vehicle Routing Problem. In P. Toth, & D. Vigo (Eds.) *Vehicle Routing: Problems, Methods, and Applications*, (pp. 87–116). Philadelphia, PA, USA: SIAM.

Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, *13*(2), 129–170.

Li, K., Fialho, A., Kwong, S., & Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *18*(1), 114–130.

Lin, C., Choy, K., Ho, G., Chung, S., & Lam, H. (2014). Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, *41*(4, Part 1), 1118–1138.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated local search: Framework and applications. In M. Gendreau, & J.-Y. Potvin (Eds.) *Handbook of Metaheuristics*, (pp. 363–397). Boston, MA: Springer US.

Maffioli, F., & Sciomachen, A. (1997). A mixed-integer model for solving ordering problems with side constraints. *Annals of Operations Research*, *69*, 277–297.

Malakahmad, A., Bakri, P. M., Mokhtar, M. R. M., & Khalil, N. (2014). Solid waste collection routes optimization via gis techniques in ipoh city, malaysia. *Procedia Engineering*, *77*, 20–27.

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

Miranda-Bront, J., Méndez-Díaz, I., & Zabala, P. (2014). Facets and valid inequalities for the time-dependent travelling salesman problem. *European Journal of Operational Research*, *236*(3), 891–902.

Montero, A., Méndez-Díaz, I., & Miranda-Bront, J. J. (2017). An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers & Operations Research*, *88*, 280–289.

Moon, C., Kim, J., Choi, G., & Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*, *140*(3), 606–617.

Mühlenbein, H. (1991). Parallel genetic algorithms, population genetics and combinatorial optimization. In J. Becker, I. Eisele, & F. Mündemann (Eds.) *Parallelism, Learning, Evolution*, vol. 565 of *Lecture Notes in Computer Science*, (pp. 398–406). Springer Berlin Heidelberg.

Öncan, T., Altınel, İ. K., & Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, *36*(3), 637–654.

Padberg, M., & Rinaldi, G. (1990). An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, *47*(1), 19–36.

Papadimitriou, C., & Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications.

Pérez-Bernabeu, E., Juan, A. A., Faulin, J., & Barrios, B. B. (2015). Horizontal cooperation in road transportation: a case illustrating savings in distances and greenhouse gas emissions. *International Transactions in Operational Research*, *22*(3), 585–606.

Perrier, N., Langevin, A., & Amaya, C.-A. (2008). Vehicle routing for urban snow plowing operations. *Transportation Science*, *42*(1), 44–56.

Picard, J.-C., & Queyranne, M. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, *26*(1), 86–110.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403–2435.

Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, *63*(3), 337–370.

Ramos, T. R. P., Gomes, M. I., & Barbosa-Póvoa, A. P. (2014). Economic and environmental concerns in planning recyclable waste collection systems. *Transportation Research Part E: Logistics and Transportation Review*, *62*, 34–54.

Raychaudhuri, S. (2008). Introduction to monte carlo simulation. In *2008 Winter Simulation Conference*, (pp. 91–100).

Reinelt, G. (1991). TSPLIB–A traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376–384.

Reisleben, B., Merz, P., & Freisleben, B. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, (pp. 616–621).

Roberti, R., & Toth, P. (2012). Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. *EURO Journal on Transportation and Logistics*, *1*(1-2), 113–133.

Snyder, L. V., & Daskin, M. S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, *174*(1), 38–53.

Son, L. H. (2014). Optimizing municipal solid waste collection using chaotic particle swarm optimization in gis based environments: A case study at danang city, vietnam. *Expert Systems with Applications*, *41*(18), 8062–8074.

Sun, P., Veelenturf, L., Hewitt, M., & Van Woensel, T. (2018). The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *116*, 1–24.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In L. Davis (Ed.) *Handbook of Genetic Algorithms*, (pp. 332–349). New York: Van Nostrand Reinhold.

Taş, D., Gendreau, M., Jabali, O., & Laporte, G. (2016). The traveling salesman problem with time-dependent service times. *European Journal of Operational Research*, *248*(2), 372–383.

Toth, P., & Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM, 2nd ed.

United Nations Population Fund (2011). *State of World Population 2011*. New York, USA: UNFPA.

Vander Wiel, R. J., & Sahinidis, N. V. (1996). An exact solution approach for the time-dependent traveling-salesman problem. *Naval Research Logistics (NRL)*, *43*(6), 797–820.

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

Vu, D., Hewitt, M., Boland, N., & Savelsbergh, M. (2018). Solving time dependent traveling salesman problems with time windows. Tech. rep., Optimization Online.

Wang, Y., Sun, J., Li, J., & Gao, K. (2012). A Modified Inver-over Operator for the Traveling Salesman Problem. In D.-S. Huang, Y. Gan, P. Gupta, & M. Gromiha (Eds.) *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, vol. 6839 of *Lecture Notes in Computer Science*, (pp. 17–23). Springer Berlin Heidelberg.

Weintraub, A., Aboud, J., Fernandez, C., Laporte, G., & Ramirez, E. (1999). An emergency vehicle dispatching system for an electric utility in Chile. *Journal of the Operational Research Society*, *50*(7), 690–696.

Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, *228*(1), 72–82.