European Journal of Operational Research 283 (2020) 825–843

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Models and algorithms for the Traveling Salesman Problem with Time-dependent Service times

Valentina Cacchiani, Carlos Contreras-Bolton, Paolo Toth[*]

DEI, University of Bologna, Viale Risorgimento 2, Bologna I-40136 Italy

ARTICLE INFO

Article history:
Received 13 November 2018
Accepted 20 November 2019
Available online 26 November 2019

Keywords:
Travelling Salesman
Time-dependent Service times
Mathematical model
Branch-and-Cut algorithm
Genetic algorithm

ABSTRACT

The Traveling Salesman Problem with Time-dependent Service times (TSP-TS) is a generalization of the Asymmetric TSP, in which the service time at each customer is given by a (linear or quadratic) function of the corresponding start time of service. TSP-TS calls for determining a Hamiltonian tour (i.e. a tour visiting each customer exactly once) that minimizes the total tour duration, given by the sum of travel and service times. We propose a new Mixed Integer Programming model for TSP-TS, that is enhanced by lower and upper bounds that improve previous bounds from the literature, and by incorporating exponentially many subtour elimination constraints, that are separated in a dynamic way. 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 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. In addition, we tested the genetic algorithm on symmetric and asymmetric instances with up to 200 nodes.

1. Introduction

The Traveling Salesman Problem with Time-dependent Service times (TSP-TS) has been recently introduced in Taş, Gendreau, Jabali, and Laporte (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 the work by 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 states of the machine. The goal is to determine the minimum cost sequence of jobs. In Picard and Queyranne (1978), three Integer Programming formulations are proposed: the first one is a generalization of a TSP formulation in which costs are time-dependent, the second one refers to the Quadratic Assignment Problem (QAP) and the third one is based on a layered graph. In the latter formulation, the set of nodes of the layered graph contains the two copies of the depot 0 and $n+1$, and nodes (i, t) for each node i and position t. As observed by one of the reviewers of our paper, under the hypothesis that all customers have the same service time function, TSP-TS can be formulated as a TDTSP on the layered graph

[*] Corresponding author.
E-mail addresses: valentina.cacchiani@unibo.it (V. Cacchiani), carlos.contrerasbolton@unibo.it (C. Contreras-Bolton), paolo.toth@unibo.it (P. Toth).

https://doi.org/10.1016/j.ejor.2019.11.046
0377-2217/© 2019 Elsevier B.V. All rights reserved.

proposed in Picard and Queyranne (1978). We show, in the Appendix (available as Supplementary Material online), how the layered graph formulation can be used to model TSP-TS. A Branch-and-Bound algorithm based on a subgradient optimization procedure is developed. In addition, the objective of minimizing the tardiness costs is considered. In Vander Wiel and Sahinidis (1996), a Benders decomposition approach is applied to a MIP model, which is a linearization of the quadratic formulation presented in Picard and Queyranne (1978), and combined with Pareto-optimal Benders cuts. In Bigras, Gamache, and Savard (2008), the two linear formulations proposed in Picard and 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, Fukasawa, Pessoa, and Uchoa (2013), the polytope corresponding to the layered-graph based TDTSP linear formulation proposed in Picard and 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, Méndez-Díaz, and Zabala (2014), two of the formulations presented in Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1996) are studied and compared. Additional valid inequalities and facets are proposed, and embedded in a Branch-and-Cut approach. Recently, Kinable, Cire, and van Hoeve (2017) proposed a method for solving time-dependent sequencing problems, i.e., sequencing problems in which setup times between two jobs depend on the position of the jobs in the ordering. These problems include the TDTSP, the TDTSP with time windows and the time-dependent sequential ordering problem. A hybrid method that combines a constraint programming model with two relaxations is proposed. We refer the reader to Gouveia and 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, Ghiani, and Guerriero (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, Gendreau, and Potvin (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, Calogiuri, Ghiani, and Guerriero (2018), a Branch-and-Bound algorithm is proposed for the same problem, and tested on the same instances and on larger size ones (with 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, Ghiani, Grieco, Guerriero, and Plana (2019), 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, Méndez-Díaz, and Miranda-Bront (2017), an alternative formulation, based on the model proposed in Sun, Veelenturf, Dabia, and Van Woensel (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. (2019). The recent work Vu, Hewitt, Boland, and Savelsbergh (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, Hewitt, Marshall, and Savelsbergh (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. (2019) and obtains the optimal solution for all the instances in shorter computing times.

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 variables and constraints, have been proposed, together with lower and upper bounds aimed at improving the solution process. In this paper, 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 MIP model. The main contribution of this work consists of the design and development of two Branch-and-Cut (B&C) algorithms. Both of them are 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. In addition, we propose 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 instances with up to 45 nodes and larger size symmetric instances with up to 58 nodes from the TSPLIB library (Reinelt (1991)). Finally, the GA is tested on symmetric and asymmetric instances from the TSPLIB library with up to 200 nodes.

Section 2 presents the problem definition and introduces the used notation. In Section 3 we report the mathematical models and the lower and upper bounds proposed in Taş et al. (2016). Section 4 describes the new mathematical model and the improvements to the lower and upper bounds from the literature. Section 5 is devoted to the description of the solution methods, i.e. the GA, B&C and Dynamic B&C algorithms, and in Section 6 we report the obtained results and the comparison with the method presented in Taş et al. (2016). Finally, we conclude our paper with some remarks in Section 7.

2. Problem definition

TSP-TS is defined on a complete directed graph $G = (N, A)$. The set of nodes $N = \{0, 1, \dots, n, n+1\}$ 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, Altinel, and Laporte (2009), Roberti and 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 vehicle 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.

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 paper will be used for comparison in Section 6.

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:

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

$$\sum_{j\in N\setminus\{i\}} x_{ij} = 1, \quad i \in N \setminus \{n+1\}, \tag{2}$$

$$\sum_{i\in N\setminus\{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{3}$$

$$b_i + s_i(b_i) + t_{ij} - M(1 - x_{ij}) \leq b_j, \quad i \in N, \ j \in N, \tag{4}$$

$$b_i \geq 0, \quad i \in N, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N, \ j \in N. \tag{6}$$

The objective function (1) minimizes 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) and (3) respectively, to have exactly one outgoing arc from each node except for the depot $n+1$, and one ingoing arc for each node, except for the depot 0. The novel constraints (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 subtours to be infeasible, thus making model (1)-(6) valid without explicit subtour elimination constraints. Finally, (5) and (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 $L(1), \dots, L(n+1)$ 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 as follows:

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

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 (4) for arcs $(i, j) \in A$ having $x_{ij} = 0$.

3.2. Gavish and Graves model

In Taş et al. (2016), in order to strengthen model (1)–(6), the polynomially many Gavish and Graves (GG) subtour elimination constraints (proposed in Gavish and Graves (1978) for the ATSP) 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:

$$\sum_{j\in N\setminus\{0\}} g_{ij} - \sum_{j\in N\setminus\{0,n+1\}} g_{ji} = 1, \quad i = 1, \dots, n, \tag{7}$$

$$0 \leq g_{ij} \leq n x_{ij}, \quad i = 1, \dots, n, \ j = 1, \dots, n+1. \tag{8}$$

Constraints (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 (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 $n+1$.

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 $(0, a(1), a(2), \dots, a(n), a(n+1) = n+1)$ 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 β, $\gamma > 0$, 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 path travel time. Since the linear service time function is non-decreasing, arriving at a node earlier gives a smaller service time than arriving there later. As a consequence, a valid lower bound on the total service time can be obtained by considering the path defined by the n smallest travel times. A more formal proof of the validity of B_2 can be found in Taş et al. (2016).

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: $b_i \le M$ and $b_i \ge LB_3(i)$ = 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 (1)-(6) (where the value of M is computed as described in Section 3.1), with the additional GG (polynomially many) subtour elimination constraints (7)-(8) and the lower and upper bounds B_1, B_2 and B_3. We identify this method as TGJL16.

4. Improvements to models and bounds

In Section 4.1, we propose improved lower and upper bounds, and an improved value for M. In addition, in Section 4.2, we enhance the basic model by using exponentially many explicit subtour elimination constraints (SECs). Furthermore, in Section 4.3, we propose a new model for the TSP-TS, which can also embed the improved bounds and the SECs.

4.1. Improved bounds (IBs) and M

As described in Section 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 for the TSP-TS. Algorithm 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 and 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 6.1), GA obtains better solutions when the SECs are included in the model. Algorithm GA will be described in Section 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 proceed as follows. For each node i $(i = 0, 1, \ldots, n)$, we compute the minimum and second minimum travel time arcs (i, min_i) and $(i, smin_i)$, respectively, leaving node i, and set: $\tau_i = t_{i,min_i} = \min\{t_{ij} : j = 1, \ldots, n, j \neq i\}$ and $\tau'_i = t_{i,smin_i} = \min\{t_{ij} : j = 1, \ldots, n, j \neq i, j \neq min_i\}$. Then, we execute the following four steps:

1. sort the n minimum travel time arcs (i, min_i) $(i = 1, \ldots, n)$ in ascending order according to the travel times τ_i. Let ord_h $(h = 1, \ldots, n)$ be the initial node of the h^{th} ordered arc.
2. for each node i $(i = 1, \ldots, n)$ set $f_i = 0$;
   For $h = 1, \ldots, n$:
   (a) set $i = ord_h$, $j = min_i$;
   (b) if $f_j = 1$ and $min_j = i$ and $\tau'_j > \tau_i$ then set $L(h+1) = \min\{\tau'_j, \tau'_i\}$
   (c) else set $L(h+1) = \tau_i$, $f_i = 1$.
3. order the n travel times $L(h)$ $(h = 2, \ldots, n+1)$ according to ascending values.
4. set $L(1) = \tau_0$, consider the path $(0, 1, \ldots, n)$ whose arcs have the travel times $L(1), L(2), \ldots, L(n)$, and proceed as in the computation of B_2, i.e., compute the arrival (b_i) and departure (d_i) times at each node i and set $IBL_2 = d_n - \sum_{i=1}^{n} L(i)$.

At step 2, for each node i $(i = 1, \ldots, n)$, f_i is equal to 1 if the value τ_i, corresponding to the travel time of arc (i, min_i), has been assigned to $L(h+1)$, with $i = ord_h$, i.e., if arc (i, min_i) is used in the computation of IBL_2; in this case, the reverse arc (min_i, i) is not used in the following iterations of the loop. Otherwise, the value f_i keeps its initial value 0; in this case the reverse arc (min_i, i) can possibly be used.

For IBE_2, in a similar way, we compute, for each node i $(i = 1, 2, \ldots, n)$, the minimum and second minimum travel time arcs (min_i, i) and $(smin_i, i)$, respectively, entering node i, and set $\tau_i = t_{min_i,i} = \min\{t_{ji} : j = 0, \ldots, n, j \neq i\}$ and $\tau'_i = t_{smin_i,i} = \min\{t_{ji} : j = 0, \ldots, n, j \neq i, j \neq min_i\}$. Then we execute the same four steps described above for the computation of IBL_2, where, in step 1, we consider arcs (min_i, i) in place of arcs (i, min_i) $(i = 1, \ldots, n)$, and ord_h $(h = 1, \ldots, n)$ as the final node of the h^{th} ordered arc, and, in step 4, we consider the path $(0, 1, \ldots, n)$ whose arcs have the travel times $L(2), L(3), \ldots, L(n+1)$. 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, as previously mentioned, 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. In particular, in the computation of IBL_2, we need to leave each node i $(i = 1, \ldots, n)$, but the last node of the Hamiltonian path, by selecting only one of its leaving arcs (i, j), with $j = 1, \ldots, n$. In addition, the first arc of the Hamiltonian path must leave the initial depot 0: so, as first arc of the path, we select the minimum travel time arc leaving the depot. For the last node of the path, we consider the node say l, for which the associated τ_l has the largest value, given by $L(n+1)$. In the computation of IBE_2, we need to enter each node i $(i = 1, \ldots, n)$ by selecting only one of its entering arcs (j, i), with $j = 0, 1, \ldots, n$.

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 the 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 (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 best available value of the total route duration, i.e., the arrival time at node $(n+1)$. Of course, any upper bound on the total route duration could be used instead of IB_1. However, this value is not necessarily enough to deactivate constraints (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 replacing M by M_{ij} for each arc $(i, j) \in A$ is correct by considering that:

- 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).

4.2. Subtour elimination constraints (SECs)

In Taş et al. (2016), additional subtour elimination constraints, namely the GG constraints, are used to improve the basic model performance. The authors also evaluate two other compact ATSP formulations with a polynomial number of subtour elimination constraints, obtained, respectively, by using the Miller, Tucker and Zemlin (MTZ) (Miller, Tucker, & Zemlin (1960)) constraints and the Desrochers and Laporte (DL) (Desrochers & Laporte (1991)) constraints to improve the Linear Programming (LP) relaxation. These constraints are incorporated in the basic model as valid inequalities. However, the best performance was achieved by considering the GG constraints. In this paper, we propose to replace the GG constraints (7), (8) with the explicit subtour elimination constraints (SECs) proposed in Dantzig, Fulkerson, and Johnson (1954) for the ATSP:

$$\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \quad S \subseteq N \setminus \{n+1\}, \quad 0 \in S, \quad |S| \geq 2. \tag{9}$$

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 a node in S to a node not in S. Since the number of constraints (9) is exponential in the number of nodes, we impose in the model only a subset of these constraints, and iteratively add violated constraints detected by applying the constraint separation procedure proposed in Padberg and Rinaldi (1990).

4.3. A new model (NM) for the TSP-TS

The new model (NM) described in this section is based on the formulation proposed in Maffioli and 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 and Sciomachen (1997) reads as follows:

$$\min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} \tag{10}$$

$$\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N \setminus \{n+1\}, \tag{11}$$

$$\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{12}$$

$$r_i x_{ij} \leq y_{ij} \leq d_i x_{ij}, \quad i \in N \setminus \{0, n+1\}, j \in N \setminus \{0\}, i \neq j, \tag{13}$$

$$\sum_{i \in N \setminus \{0, n+1\}, i \neq j} y_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} t_{ij} x_{ij} \leq \sum_{k \in N \setminus \{0\}, k \neq j} y_{jk}, \quad j \in N \setminus \{0, n+1\}, \tag{14}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N, \ j \in N, \tag{15}$$

$$y_{ij} \geq 0, \quad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j. \tag{16}$$

The objective function (10) calls for minimizing the total travel times. Constraints (11) and (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 (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 (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 (15) and (16) define the variable domains.

The new proposed model (NM) uses, in addition to the x_{ij} and y_{ij} variables, the b_i variables defined in model (1)-(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 (10)-(16) is modified to deal with linear time-dependent service times, as follows:

$$\min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} + \sum_{i \in N \setminus \{0, n+1\}} s_i(b_i) \tag{17}$$

$$\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N \setminus \{n+1\}, \tag{18}$$

$$\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{19}$$

$$b_i = \sum_{k \in N \setminus \{0\}, k \neq i} y_{ik}, \quad i \in N \setminus \{0, n+1\}, \tag{20}$$

$$y_{ij} \geq LB_3(i) x_{ij}, \quad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, i \neq j, \tag{21}$$

$$y_{ij} + \beta y_{ij} + \gamma x_{ij} \leq M x_{ij} \qquad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j, \tag{22}$$

$$\sum_{i \in N \setminus \{0,n+1\}, i \neq j} y_{ij} + \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \beta y_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} \gamma x_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} t_{ij} x_{ij} \leq \sum_{k \in N \setminus \{0\}, k \neq j} y_{jk}, \quad j \in N \setminus \{0, n+1\}, \tag{23}$$

$$b_i \geq 0, \quad i \in N, \tag{24}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N, \ j \in N, \tag{25}$$

$$y_{ij} \geq 0, \quad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j. \tag{26}$$

The objective function (17) is the same as (1), and minimizes the total route duration given by the sum of the total travel and service times. As in the previous models, constraints (18) and (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 (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 (21) and (22), in which lower and upper bounds on the arrival times at the customers are used, replace the time window constraints (13). In particular, we use the bounds B_3 proposed in Taş et al. (2016). Note that constraints (21) and (22) ensure that y_{ij} is set to 0 if x_{ij} is also 0. Constraints (23) correspond to (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 (24)–(26) restrict the variable domain. It can be observed that the constraints (23) can be written as equality constraints when the *arrival time property* (see Property 3 in Section 2) holds. We performed computational experiments, on the instances with the linear service time functions proposed in Taş et al. (2016), by replacing the constraints (23) with the corresponding equality constraints. Similar performances were obtained by the proposed Dynamic B&C algorithm (described in Section 5.2) in both cases: namely, the same number of instances was solved to optimality within the given time limit, and the average optimality gap was slightly larger when solving the model with the equality constraints than with the constraints (23). Therefore, we do not report the computational results obtained by the Dynamic B&C algorithm with the equality constraints.

In order to improve NM, we can insert the improved bounds IB_3 in constraints (21) and (22) as follows:

$$y_{ij} \geq sp_{0,i} x_{ij}, \quad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, i \neq j, \tag{27}$$

$$y_{ij} + \beta y_{ij} + \gamma x_{ij} \leq (IB_1 - sp_{i,n+1}) x_{ij} \qquad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j. \tag{28}$$

In particular, if arc (i, j) is selected, 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. In addition, when arc (i, j) is selected, 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 $n+1$.

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 (22) and (23) are changed as follows.

$$y_{ij} + \alpha y_{ij}^2 - \beta y_{ij} + \gamma x_{ij} \leq M x_{ij} \qquad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j, \tag{29}$$

$$\sum_{i \in N \setminus \{0,n+1\}, i \neq j} y_{ij} + \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \alpha y_{ij}^2 - \sum_{i \in N \setminus \{0,n+1\}, i \neq j} \beta y_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} \gamma x_{ij} + \sum_{i \in N \setminus \{n+1\}, i \neq j} t_{ij} x_{ij} \leq \sum_{k \in N \setminus \{0\}, k \neq j} y_{jk}, \quad j \in N \setminus \{0, n+1\}. \tag{30}$$

A similar improvement as in the case of the linear service time functions can be applied for the quadratic case by replacing constraints (29) with the following ones:

$$y_{ij} + \alpha y_{ij}^2 - \beta y_{ij} + \gamma x_{ij} \leq (IB_1 - sp_{i,n+1}) x_{ij} \qquad i \in N \setminus \{0, n+1\}, \ j \in N \setminus \{0\}, \ i \neq j. \tag{31}$$

In the following, we will call NMB the model corresponding to (17)-(26), for the linear service time functions, and to (17)-(21), (24)-(26), (29)-(30), for the quadratic service time function. We will call NMI the improved model corresponding to (17)-(20), (23)-(26), (27)-(28), for the linear service time functions, and to (17)-(20), (24)-(26), (27), (30)-(31), for the quadratic service time function.

5. Solution methods

To solve the TSP-TS we propose the metaheuristic algorithm GA, presented in Section 5.1, that provides high quality solutions in very short computing times (see Section 6.1), and two exact B&C algorithms, described in Section 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.

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 algorithm, 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 considering only the nodes in $N \setminus \{n+1\}$ and by applying three algorithms, each one generating a given percentage of the population:

- an algorithm (random route algorithm) that generates a random route (25% of the initial population): in particular, starting from the depot 0, we randomly choose a node, and then, from the chosen node, we select the next node in a random way among the non-visited ones, and so on, until a Hamiltonian path is built;

- the randomized NNH algorithm with each individual generated starting from a different randomly chosen node (not necessarily the depot), and finishing when all the nodes in $N \setminus \{n+1\}$ have been visited (50% of the initial population);
- a continuous relaxation based algorithm, that uses the optimal continuous relaxation solution x of the basic model with the improved bound IB_2 to build a feasible Hamiltonian path (25% of the initial population).

We use the basic model, instead of model NMI, in order to reduce the global computing time of algorithm GA. In particular, the first 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 non-visited 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 the non-visited nodes j we have $x_{hj} + x_{jh} = 0$, 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 non-visited node j according to a probability given by $x_{hj} + x_{jh}$. If, for all the non-visited nodes j, we have $x_{hj} + x_{jh} = 0$, then we choose the arc with the smallest t_{hj}. 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 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, Merz, & Freisleben (1996)), Maximal Preservative Crossover (MPX) (Mühlenbein (1991)), Heuristic Crossover (HX) (Grefenstette, Gopal, Rosmaita, & Gucht (1985)), Modified Inver-over Operator (MIO) (Wang, Sun, Li, & Gao (2012)), Uniform Nearest Neighbor (UNN) (Buriol, França, & Moscato (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 executing the algorithm proposed in Contreras-Bolton and Parada (2015), which considers many alternative operators and finds a good combination of them. In that paper, 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 and Parada (2015) was calibrated on a subset of the TSP-TS instances considered in Section 6 (three symmetric ones and three asymmetric ones) and by considering the small linear service time function $s_i = 5(10^{-3})b_i + 3(10^{-2}) \quad (i \in N \setminus \{0, n+1\})$. 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 (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:

Table 1
Probabilities of using each crossover or mutation operator.

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

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. When the two individuals have been selected (according to the tournament selection), one of the crossover operators, according to the probabilities reported in Table 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 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 (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 6.1.

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 and Rinaldi (1990) until no violated constraints exist. The B&C algorithm 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 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. In particular, the two bounds are recomputed at every node α of the decision tree, by taking into account that, for the currently considered node α, 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 may not be used in the computation of the bounds. In this way, the bounds can be tightened using local information from the node. More precisely, for the currently considered node α, let IA (resp. EA) be the set of arcs $(i, j) \in A$ such that the variable x_{ij} has been fixed to 1 (resp. to 0) at the previous levels of the decision tree. Then an updated travel time matrix (t'_{ij}) is defined as follows: (i) for $i \in N \setminus \{n+1\}$ and $j \in N \setminus \{0\}$ set $t'_{ij} = t_{ij}$; (ii) for

$(i, j) \in EA$ set $t'_{ij} = \infty$; (iii) for $(i, j) \in IA$ set $t'_{ik} = \infty$ for $k \in N \backslash \{0, j\}$ and $t'_{kj} = \infty$ for $k \in N \setminus \{n+1, i\}$. The updated improved bounds IB_2 and IB_3 at node α are computed by considering matrix (t'_{ij}) as the current travel time matrix. 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.

The B&C and the Dynamic B&C algorithms can also be applied on the basic model (1)-(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 for comparison in the computational experiments.

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 three sets of instances:

- Set 1: the 22 symmetric instances introduced in Taş et al. (2016), adapted from the TSPLIB (Reinelt (1991)), containing up to 45 nodes (and an additional node for the ending depot),
- Set 2: all the instances of TSPLIB up to 58 nodes not considered in Set 1 (symmetric) and up to 45 nodes (asymmetric),
- Set 3: all the instances of TSPLIB up to 200 nodes not considered in Set 2 (symmetric and asymmetric). Note that asymmetric instances with up to 200 nodes contain at most 170 nodes.

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 travel times of the new instances were 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 consider the instances of Set 1, and show in Sections 6.1 and 6.2 the corresponding results. In Section 6.1, we report the computational results of the GA algorithm on the 22 symmetric instances introduced in Taş et al. (2016). Then, in Section 6.2, we report the comparison, in terms of the lower bounds of the continuous relaxation (Section 6.2.1) and of the integer solutions (Section 6.2.2), 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). In Section 6.3, we report the results obtained by GA and by the exact algorithms on the instances of Set 2. Finally, in Section 6.4, we report the results obtained by GA on the instances of Set 3.

For the B&C and Dynamic B&C algorithms, as well as for TGJL16, we set a time limit of 7200 seconds for the instances of Set 1, and of 50000 seconds for the instances of Set 2. 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 solver CPLEX 12.7.1 on the same computer. As mentioned in the introduction, the TSP-TS can be formulated using the layered graph model proposed in Picard and Queyranne (1978), as a TDTSP. The state-of-the-art method for solving TDTSP is the Branch-and-Cut-and-Price algorithm proposed in Abeledo et al. (2013): however, we could not report a comparison with the methods that we propose, since the code is no longer available from the authors at the time of writing.

6.1. Genetic algorithm on instances of Set 1

We report the results obtained by the GA algorithm on the 22 symmetric instances of Set 1 introduced in Taş et al. (2016) with small and medium service times in Table 2, and with large and quadratic service times in Table 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, large and quadratic in Table 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 and instance, we report 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 minimum percentage gap is obtained by executing 10 runs of the GA algorithm. Thus, the corresponding computing time is ten times the average computing time (and is not reported in the tables). In the last two rows, we show the averages of the values in the corresponding columns computed over all instances and report the number of best known solutions found.

As we can observe, when considering the best out of 10 runs, at least one of the GA algorithms (without or with SECs) finds the best known solution for all instances. In addition, when considering the best out of 10 runs, the GA with SECs finds the best known solution for all instances except two (one with the small and one with the large service time functions). 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 average percentage gap among all instances w.r.t. the best known solution value is 0.97% (for instance dantzig42 and medium service time). In the computation of the values of IB_1 and IB_3 used in the B&C algorithms, we decided to use the minimum cost solutions obtained by the GA algorithm with SECs out of 5 runs instead of 10 to limit the computing times.

As shown in Tables 2 and 3, the variant of the GA algorithm that includes SECs is very effective. To show the importance of using the continuous relaxation (CR) solution to build the initial population, as explained in Section 5.1, we tested the GA algorithm without CR, i.e., by building 50% of the initial population by the random route algorithm and 50% by the randomized NNH algorithm, and compared the obtained results with those of GA with SECs. The detailed results are reported, in the Appendix (available as Supplementary Material online), for small and medium service times as well as for large and quadratic service times. The results show that a smaller number of best solutions and a slightly larger average gap are obtained when CR is not considered.

6.2. Comparison with TGJL16 on Instances of Set 1

This section is devoted to the comparison of the results obtained by the proposed exact algorithms with the results obtained

Table 2
GA on instances of Set 1 (Taş et al. (2016)) with small and medium service times.

# inst	Small service times							Medium service times						
		without SECs			with SECs				without SECs			with SECs		
	Best	Avg%	Min%	Time	Avg%	Min%	Time	Best	Avg%	Min%	Time	Avg%	Min%	Time
burma14	228.83	0.00	0.00	0.13	0.00	0.00	0.13	236.44	0.00	0.00	0.10	0.00	0.00	0.11
ulysses16	271.74	0.00	0.00	0.13	0.06	0.00	0.13	279.57	0.00	0.00	0.13	0.02	0.00	0.13
gr17	238.39	0.00	0.00	0.15	0.00	0.00	0.15	245.40	0.00	0.00	0.15	0.00	0.00	0.15
gr21	237.11	0.00	0.00	0.21	0.00	0.00	0.22	249.32	0.00	0.00	0.21	0.00	0.00	0.22
ulysses22	306.43	0.01	0.00	0.23	0.00	0.00	0.23	318.06	0.16	0.00	0.22	0.38	0.00	0.23
gr24	269.09	0.00	0.00	0.27	0.00	0.00	0.30	284.93	0.00	0.00	0.27	0.00	0.00	0.29
fri26	247.99	0.00	0.00	0.33	0.00	0.00	0.33	263.01	0.00	0.00	0.33	0.00	0.00	0.34
bayg29	345.49	0.30	0.00	0.41	0.05	0.00	0.43	371.22	0.23	0.00	0.41	0.00	0.00	0.43
bays29	309.27	0.26	0.00	0.41	0.16	0.00	0.42	331.90	0.20	0.00	0.41	0.15	0.00	0.42
att30	253.85	0.00	0.00	0.44	0.00	0.00	0.46	273.10	0.00	0.00	0.44	0.00	0.00	0.45
dantzig30	324.21	0.54	0.00	0.43	0.21	0.00	0.45	349.60	0.30	0.30	0.43	0.09	0.00	0.45
eil30	323.40	0.38	0.00	0.45	0.25	0.00	0.46	349.16	0.34	0.00	0.44	0.17	0.00	0.45
gr30	283.91	0.00	0.00	0.46	0.00	0.00	0.48	305.23	0.11	0.00	0.45	0.00	0.00	0.46
hk30	324.20	0.50	0.00	0.46	0.00	0.00	0.48	347.35	0.63	0.00	0.45	0.00	0.00	0.47
swiss30	342.50	0.00	0.00	0.46	0.00	0.00	0.47	366.78	0.00	0.00	0.46	0.00	0.00	0.47
eil35	363.38	0.34	0.00	0.65	0.11	0.00	0.68	397.42	0.34	0.00	0.64	0.16	0.00	0.68
gr35	281.82	0.00	0.00	0.68	0.00	0.00	0.74	306.91	0.00	0.00	0.68	0.00	0.00	0.75
swiss35	373.60	0.00	0.00	0.71	0.00	0.00	0.73	406.92	0.00	0.00	0.70	0.00	0.00	0.73
eil40	410.35	0.18	0.00	0.91	0.02	0.00	0.96	452.89	0.33	0.00	0.90	0.16	0.00	0.95
dantzig42	257.37	0.33	0.00	1.02	0.59	0.00	1.08	285.07	0.72	0.00	1.00	0.97	0.00	1.06
swiss42	351.15	0.03	0.00	1.04	0.03	0.00	1.07	388.64	0.59	0.00	0.99	0.07	0.00	1.05
eil45	448.11	0.03	0.00	1.24	0.07	0.04	1.30	502.52	0.16	0.00	1.22	0.25	0.00	1.30
Avg.	308.74	0.13	0.00	0.51	0.07	0.00[a]	0.53	332.34	0.19	0.01	0.50	0.11	0.00	0.53
#Best		11	22		11	21			10	21		12	22	

[a] The gap is $\geq 0.001\%$ and $\leq 0.005\%$

Table 3
GA on instances of Set 1 (Taş et al. (2016)) with large and quadratic service times.

# inst	Large service times							Quadratic service times						
		without SECs			with SECs				without SECs			with SECs		
	Best	Avg%	Min%	Time	Avg%	Min%	Time	Best	Avg%	Min%	Time	Avg%	Min%	Time
burma14	252.62	0.00	0.00	0.10	0.00	0.00	0.11	224.83	0.00	0.00	0.13	0.00	0.00	0.19
ulysses16	296.28	0.00	0.00	0.13	0.00	0.00	0.13	268.14	0.03	0.00	0.15	0.03	0.00	0.30
gr17	260.34	0.00	0.00	0.15	0.00	0.00	0.15	234.82	0.00	0.00	0.18	0.00	0.00	0.37
gr21	275.96	0.00	0.00	0.21	0.00	0.00	0.22	232.77	0.00	0.00	0.27	0.00	0.00	0.47
ulysses22	343.57	0.00	0.00	0.23	0.00	0.00	0.23	301.58	0.00	0.00	0.28	0.17	0.00	0.71
gr24	320.42	0.03	0.00	0.27	0.00	0.00	0.29	263.04	0.00	0.00	0.35	0.00	0.00	1.01
fri26	297.39	0.00	0.00	0.32	0.00	0.00	0.34	239.08	0.00	0.00	0.41	0.00	0.00	0.90
bayg29	430.35	0.14	0.00	0.40	0.05	0.00	0.43	345.11	0.00	0.00	0.51	0.00	0.00	1.39
bays29	383.78	0.16	0.00	0.41	0.16	0.00	0.43	305.46	0.24	0.00	0.52	0.24	0.00	1.24
att30	316.51	0.00	0.00	0.43	0.00	0.00	0.45	246.98	0.00	0.00	0.56	0.00	0.00	1.65
dantzig30	404.54	0.73	0.69	0.41	0.57	0.00	0.44	321.89	0.55	0.55	0.57	0.22	0.00	1.96
eil30	408.23	0.21	0.00	0.44	0.15	0.00	0.44	320.74	0.40	0.00	0.55	0.29	0.00	1.33
gr30	353.89	0.00	0.00	0.45	0.00	0.00	0.47	279.94	0.00	0.00	0.58	0.00	0.00	1.35
hk30	400.88	0.75	0.00	0.43	0.00	0.00	0.44	318.63	0.67	0.00	0.58	0.00	0.00	1.84
swiss30	422.54	0.00	0.00	0.46	0.00	0.00	0.47	340.42	0.00	0.00	0.58	0.00	0.00	1.25
eil35	474.90	0.22	0.00	0.53	0.24	0.20	0.59	365.34	0.32	0.00	0.80	0.09	0.00	2.15
gr35	365.75	0.00	0.00	0.67	0.00	0.00	0.70	276.18	0.00	0.00	0.85	0.00	0.00	2.28
swiss35	485.44	0.00	0.00	0.69	0.00	0.00	0.72	378.36	0.00	0.00	0.88	0.00	0.00	2.15
eil40	556.10	0.07	0.00	0.75	0.07	0.00	0.80	421.10	0.07	0.00	1.12	0.10	0.00	3.03
dantzig42	352.36	0.42	0.00	0.93	0.00	0.00	1.03	247.25	0.09	0.00	1.25	0.67	0.00	4.54
swiss42	480.29	0.12	0.00	0.90	0.05	0.00	1.04	350.45	0.26	0.00	1.27	0.03	0.00	3.50
eil45	638.13	0.09	0.00	1.19	0.07	0.00	1.28	474.44	0.12	0.00	1.47	0.27	0.00	4.09
Avg.	387.29	0.13	0.03	0.48	0.06	0.01	0.51	307.12	0.13	0.03	0.63	0.10	0.00	1.71
#Best		11	21		14	21			12	21		12	22	

by the best method (TGJL16) presented in Taş et al. (2016). We show in Section 6.2.1 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 the improved bounds and the SECs separation. In addition, we show in Section 6.2.2 the comparison of the integer solutions 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.

6.2.1. *Lower bounds*

We present, in Tables 4–7, 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

Table 4
Comparison of Lower Bounds with small service times on instances of Set 1 (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	306.43	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	363.38	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 5
Comparison of Lower Bounds with medium service times on instances of Set 1 (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

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 (1)–(6), enhanced with the GG constraints (7) and (8), 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 3.1.
- NMB+Bs: continuous relaxation of the model (17)-(26), for the linear service time functions, and of the model (17)-(21), (24)-(26), (29)-(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 3.1;
- NMI+IBs: continuous relaxation of the model NMI (corresponding to the model (17)-(20), (23)-(26), (27)-(28), for the linear service time functions, and to the model (17)–(20), (24)–(26), (27), (30), (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 4.1;
- NMI+IBs+SECs: NMI+IBs, combined with the separation of the SECs (described in Section 4.2);
- basic+IBs+SECs: continuous relaxation of model (1)–(6), with the computation of the improved *M* value described in Section 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 4.1, and the separation of the SECs (described in Section 4.2).

Table 6
Comparison of Lower Bounds with large service times on instances of Set 1 (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	343.57	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	480.29	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 7
Comparison of Lower Bounds with quadratic service times on instances of Set 1 (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

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 reported 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 6.2.2, 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, even though the case of quadratic service time function shows a larger computing time. One reason for this is that CPLEX has to solve the continuous relaxation of a quadratic model instead of a linear one. Overall, we can conclude that both the improved

Table 8
Comparison of Integer Solutions with small service times on instances of Set 1 (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	
AvgS		104641		341.57	4139		27.72	2705		35.70	8964		42.69
#Opt.			22			22			22			22	

bounds and the separation of the exponentially many SECs help to find better lower bounds in all cases. In particular, by comparing the percentage gap between the average of the best known solution values and the average of the lower bound values obtained by TGJL16 and by NMI+IBs+SECs, which obtains the best lower bound values, we can note that: the percentage gap is reduced from 13.69% to 2.25% for the small service times, from 15.10% to 4.23% for the medium ones, from 17.65% to 7.91% for the large ones, and from 16.53% to 6.20% for the quadratic ones.

6.2.2. Integer solutions

Tables 8–11 present 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 (1)–(6), enhanced with the GG constraints (7) and (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 (17)–(20), (23)–(26), (27) and (28), for the linear service time functions, and the model (17)–(20), (24)–(26), (27), (30)–(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 4.1), and with the separation of the SECs (described in Section 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 (1)–(6).

For all the instances of Set 1, we consider for each method a time limit of 7200 seconds. We remark that the available feasible solution obtained by GA is not used as a MIP start, but the upper bound $IB_1 + \epsilon$ (with ϵ a very small positive number) is used as a cut-off value for NMI B&C, NMI Dyn-B&C and basic Dyn-B&C. Similarly, the $B_1 + \epsilon$ is used as a cut-off value for TGJL16. We also executed our best solution method NMI Dyn-B&C on all the instances of Set 1 (with small, medium, large and quadratic service times) by using the GA solution as MIP start. Similar performances were obtained in both cases (MIP start or cut-off value): in particular, when the GA solution is used as MIP start, the average optimality gap was slightly larger, and, for the case of medium service times, one less instance was solved to optimality within the time limit. Therefore, we do not report the computational results obtained by NMI Dyn-B&C with MIP start, and consider, in all the experiments, the GA solution value as a cut-off value. In addition, we underline that, for the quadratic service time case, the model becomes quadratic, but the proposed algorithms are exact also in this case: indeed, CPLEX is used as a quadratic programming solver to solve the continuous relaxation of the obtained quadratic programming model, the SECs procedure is applied on the continuous relaxed model, and the branching is applied directly by CPLEX on the quadratic programming model.

In each table, we report, for each instance, the instance name and the best known solution value. Then we show, for each method, the best integer solution value found during the execution, 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 to optimality within the time limit the same subset of instances. Therefore, we report, in row AvgS, the average computing time over the subset of instances solved to optimality within the time limit by the corresponding method and, to have a fair comparison, we also show the average computing time by considering only

Table 9
Comparison of Integer Solutions with medium service times on instances of Set 1 (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	
AvgS		220522		544.02	35388		263.11	37690		736.63	139823		505.64
#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 10
Comparison of Integer Solutions with quadratic service times on instances of Set 1 (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	343.57	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	480.29	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	
AvgS		362003		632.07	125917		1262.15	117217		1038.04	251471		958.33
#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

the instances solved to optimality 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 subsets of instances solved by the various methods coincide. In row AvgS, we also report, in column UB, the average number of nodes explored by each method computed on the instances solved to optimality within the time limit. The numbers of nodes explored by each method

Table 11
Comparison of Integer Solutions with large service times on instances of Set 1 (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	
AvgS		142413		611.40	11787		215.15	11146		273.72	210534		1273.33
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

for each instance can be found in the Appendix (available as Supplementary Material online) for the small and medium service times, and for the large and quadratic ones. In addition, we show at the bottom of each table, the number of instances solved, by each method, to optimality within the time limit. For the case of the quadratic service time function (Table 10), 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 8, in which the small service times are considered, we can see that all methods solve to optimality within the time limit 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. In addition, the average numbers of nodes explored by NMI B&C and NMI Dyn-B&C are more than two orders of magnitude smaller than those explored by TGJL16, which is congruent with the shorter computing times and the better lower bounds achieved by the proposed methods.

When the medium service times are considered (Table 9), not all the instances can be solved to optimality within the time limit, and the Dynamic B&C algorithm based on NMI obtains the largest number of proved optimal solutions (20 out of 22), while TGJL16 can solve only 14 instances to optimality within the time limit. 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 to optimality within the time limit 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 to optimality within the time limit 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. Also in this case, the average numbers of nodes explored by NMI B&C and NMI Dyn-B&C are more than two orders of magnitude smaller than those explored by TGJL16.

In the case of the large service times (Table 11), the TSP-TS instances become harder for all methods: TGJL16 can solve to optimality within the time limit only 6 instances, and the largest number of instances solved to optimality within the time limit 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 to optimality within the time limit by all methods is the Dynamic B&C algorithm based on the basic model, while, when we consider the subsets of instances solved to optimality within the time limit 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 to optimality within the time limit by the NMI B&C algorithm and the 9 instances solved to optimality within the time limit 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. The average numbers of nodes explored by NMI B&C and NMI Dyn-B&C are about three times smaller than those explored by TGJL16.

Finally, in Table 10, we report the results obtained for the quadratic service time function. 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 optimality within the time limit in similar computing times. TGJL16 solves 9 instances to optimality within the time limit in an average computing time that is significantly larger than that of the other algorithms. If we consider the subset of instances solved to optimality within the time limit 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. Moreover, the average numbers

Table 12
Comparison of Lower Bounds and Integer Solutions with small service times on symmetric instances of Set 2.

# inst	Best	GA+SECs				TGJL16					NMI Dyn-B&C				
		Val	Avg%	Min%	Time	LB	Time	UB	Gap%	Time	LB	Time	UB	Gap%	Time
att35	271.36	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	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	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	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	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	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	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	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	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	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	443.28	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	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	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	354.85	0.19	0.01	1.26	294.02	0.05	354.97	1.87		341.94	1.62	354.80	0.10	
AvgS								1726169		6370.53			228949		7449.10
#Best			6	12				12					13		
#Opt.									6					12	
Avg. TGJL16										6370.53					91.26

of nodes explored by NMI B&C and NMI Dyn-B&C are one order of magnitude smaller than those explored by TGJL16.

We can conclude that the proposed algorithms outperform TGJL16 in terms of number of instances solved to optimality within the time limit 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 optimality within the time limit (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 optimality within the time limit, and the computing times of the former are significantly shorter than those of the latter, thus showing the usefulness of the NMI formulation.

6.3. Genetic and exact algorithms on instances of Set 2

In this section, we consider the instances of Set 2. In Section 6.3.1 we report the results obtained on 13 larger size symmetric instances, 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 6.3.2 we show the results obtained on all the 27 asymmetric instances with up to 45 nodes contained in the TSPLIB. In both cases, we consider small service times (Tables 12 and 15), as well as medium service times (Tables 13 and 16), 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 improved bound IB_2 and the SECs separation, is used for building a subset of the initial population; TGJL16 and NMI Dyn-B&C (defined as in Section 6.2.2). For these additional instances, we consider for each exact method a time limit of 50000 seconds. For the GA+SECs algorithm, 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 minimum value of the solutions obtained over 10 runs, 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 reported in the corresponding columns, except for the average computing time that is shown, in row AvgS, on the instances solved to optimality within the time limit by the corresponding method, and separately for comparison on the instances solved to optimality within the time limit by both exact methods. In row AvgS, we also report the average number of nodes explored by each method computed on the instances solved to optimality within the time limit. The numbers of nodes explored by each method for each instance can be found in the Appendix (available as Supplementary Material online) for the symmetric and the asymmetric instances. In addition, we report, for each exact method, the number of instances for which the corresponding method obtained the best known solution and the number of instances solved to optimality within the time limit.

Moreover, for the large and quadratic service times, we report the results obtained on the instances of Set 2 (symmetric in Table 14 and asymmetric in Table 17) by the GA+SECs algorithm by considering 10 runs for each instance. To provide a comparison with the results obtained by GA+SECs, we also report the upper bound UB and the lower bound LB obtained by the exact method NMI Dyn-B&C with a time limit of 20000 seconds. For GA+SECs, we show the minimum and the average values of the solutions obtained over 10 runs, and the average computing time over the 10 runs.

6.3.1. Symmetricinstances of Set 2

From Table 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. By considering the average values corresponding to each instance, the average gap is only 0.19%, and the average computing time 1.26 seconds, thus confirming the effectiveness of the GA algorithm. As it was noted for the 22 instances of Set 1 considered in Taş et al. (2016), the lower bound obtained at the root node for the NMI Dyn-B&C algorithm is much larger than that obtained by TGJL16, even though it requires longer computing times. As regard as the integer solutions found, TGJL16 solves to optimality within the time limit only 6 instances, while

Table 13
Comparison of Lower Bounds and Integer Solutions with medium service times on symmetric instances of Set 2.

# inst	Best	GA+SECs				TGJL16				NMI Dyn-B&C			
		Val	Avg%	Min%	Time	LB	UB	Gap%	Time	LB	UB	Gap%	Time
att35	294.16	294.16	0.00	0.00	0.73	268.07	294.16	0.00	12929.82	280.38	294.16	0.00	57.63
dantzig35	323.05	323.05	0.80	0.00	0.59	293.19	323.05	3.33	50000.00	302.22	323.05	0.00	1753.73
hk35	376.25	376.25	0.00	0.00	0.64	347.21	376.25	0.00	26673.53	357.23	376.25	0.00	236.61
att40	324.04	324.04	0.00	0.00	0.72	288.42	324.04	4.50	50000.00	308.08	324.04	0.00	1034.17
hk40	393.46	393.46	0.00	0.00	0.92	362.47	393.46	4.35	50000.00	368.98	393.46	0.00	18520.97
att45	351.93	351.93	0.00[a]	0.00	0.99	314.28	351.93	6.04	50000.00	333.94	351.93	1.58	50000.00
hk45	409.73	409.73	0.00	0.00	1.27	376.21	409.73	5.79	50000.00	382.58	409.73	3.26	50000.00
att48	374.30	374.30	1.32	0.00	1.37	333.04	374.30	6.17	50000.00	353.82	374.30	2.50	50000.00
gr48	428.64	428.64	1.22	0.00	1.25	376.67	431.09	8.06	50000.00	396.31	428.64	0.00	21826.02
hk48	420.36	420.36	0.00	0.00	0.89	380.58	420.36	5.93	50000.00	392.38	420.36	4.17	50000.00
eil51	503.35	503.35	0.52	0.00	1.68	472.71	503.35	3.82	23540.00*	480.24	503.35	3.14	43095.26*
berlin52	497.94	497.94	0.00	0.00	1.75	427.80	497.94	12.66	50000.00	444.91	497.94	7.88	50000.00
brazil58	429.80	429.80	0.00[a]	0.00	1.93	342.63	429.80	10.30	15876.20*	392.25	429.80	5.97	50000.00
Avg.	394.39	394.39	0.30	0.00	1.13	352.56	394.57	5.46		368.72	394.39	2.19	
AvgS							8499711		19801.67		278932		7238.19
#Best			9	13			12				13		
#Opt.								2				6	
Avg. TGJL16									19801.67				147.12

[a] The gap is $\geq 0.001\%$ and $\leq 0.005\%$

Table 14
GA with large and quadratic service times on symmetric instances of Set 2.

# inst	Large service times					Quadratic service times				
	NMI Dyn-B&C		GA+SECs			NMI Dyn-B&C		GA+SECs		
	UB	LB	Val	ValAvg	Time	UB	LB	Val	ValAvg	Time
att35	347.84	341.06	347.84	347.84	0.58	263.18	263.18	263.18	263.18	2.66
dantzig35	386.92	355.76	386.92	388.23	0.53	289.40	289.40	289.40	291.01	3.66
hk35	444.91	425.54	444.91	448.09	0.53	341.36	341.36	341.36	341.36	2.32
att40	390.04	364.66	390.04	390.04	0.71	288.41	288.41	288.41	288.41	4.02
hk40	480.81	442.14	480.81	481.69	0.58	351.77	351.77	351.77	351.77	3.60
att45	431.72	399.31	431.72	431.97	0.98	312.30	307.35	312.30	312.54	7.23
hk45	515.09	465.11	515.09	517.65	0.91	363.02	355.98	363.02	363.02	4.80
att48	468.25	427.94	468.25	472.54	1.22	331.12	317.43	331.12	335.86	8.05
gr48	534.64	483.34	534.64	538.57	1.13	393.97	350.82	390.02	392.84	6.54
hk48	533.07	483.13	533.07	536.96	0.95	370.84	352.38	370.84	371.11	8.26
eil51	656.49	610.08	656.49	663.62	1.60	471.88	400.66	471.72	473.73	7.42
berlin52	613.14	527.38	613.03	617.11	0.87	460.40	410.05	460.40	460.40	6.29
brazil58	539.07	458.59	539.07	539.09	1.96	390.56	373.50	390.56	390.56	13.30
Avg.	487.84	444.92	487.84	490.26	0.97	356.02	338.64	355.70	356.60	6.01
#Best	12		13	3		11		13	7	

the Dynamic B&C algorithm based on the NMI formulation solves 12 (out of 13) instances. In addition, the computing time of the latter to prove the optimality of the solutions is much shorter than that of the former. Analogously, the average number of nodes explored by NMI Dyn-B&C is significantly smaller than that of TGJL16.

The results reported in Table 13 show that the problem becomes more difficult when we consider the medium service times. In this case, only 2 instances can be solved to optimality within the time limit by TGJL16 and 6 by NMI Dyn-B&C. In addition, the computing time on the instances solved by NMI Dyn-B&C is significantly shorter than that of TGJL16, even when we consider all the instances solved to optimality within the time limit by each method. With medium service times, the computing times become larger and for some instances, highlighted by an asterisk, the methods run out of memory. However, also in the case of medium service times, we can observe that NMI Dyn-B&C outperforms TGJL16, and that GA+SECs always finds the best known solution. Since the performances of the exact methods deteriorate with the increase of the service times, we only report the results obtained by the GA+SECs algorithm with larger and quadratic service times, and use the upper bound and lower bound values computed by executing NMI Dyn-B&C with 20000 seconds of time limit as a comparison. The results are shown in Table 14. As it can be seen, in both cases, for all instances, the GA+SECs algorithm always finds the same solution obtained by NMI Dyn-B&C or a better one. In addition, the computing times are very short, the larger ones (6 seconds, on average) appearing in the case of quadratic service times.

6.3.2. *Asymmetricinstances of Set 2*

When we consider the results reported in Table 15 for the asymmetric instances of Set 2 with small service times, we can note a behavior similar to that observed in Table 12 for the symmetric instances. The GA algorithm finds several best solutions in short computing time (0.44 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 solved by TGJL16 is much shorter for the NMI Dyn-B&C algorithm (on average about 37 seconds versus 4935 seconds). Similarly, the average number of nodes explored by NMI Dyn-B&C is significantly smaller than that of TGJL16. Therefore, we can conclude that, with small service times, the proposed algorithm turns out to be very effective on the asymmetric instances as well.

From Table 16 we can see that, with medium service times, the performance of NMI Dyn-B&C is still much better than that

Table 15
Comparison of Lower Bounds and Integer Solutions with small service times on asymmetric instances of Set 2.

# inst	Best	GA+SECs				TGJL16					NMI Dyn-B&C				
		Val	Avg%	Min%	Time	LB	Time	UB	Gap%	Time	LB	Time	UB	Gap%	Time
br17	88.60	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	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	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	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	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	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	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	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	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	360.08	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	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	356.04	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	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	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	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	381.83	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	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	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	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	379.36	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	365.68	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	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	398.14	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	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	410.63	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	374.31	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	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	340.12	0.68	0.18	0.44	287.20	0.03	339.77	1.27		325.72	0.43	339.44	0.01	
AvgS								1480765		4935.56			55680		2088.01
#Best			7	19				26					27		
#Opt.									16					26	
Avg. TGJL16										4935.56					37.37

Table 16
Comparison of Lower Bounds and Integer Solutions with medium service times on asymmetric instances of Set 2.

# inst	Best	GA+SECs				TGJL16				NMI Dyn-B&C			
		Val	Avg%	Min%	Time	LB	UB	Gap%	Time	LB	UB	Gap%	Time
br17	92.09	92.09	0.00	0.00	0.19	68.02	92.09	0.00	2.05	89.11	92.09	0.00	2.71
ft30	364.80	364.80	0.51	0.00	0.23	309.24	364.80	4.12	50000.00	337.24	364.80	0.00	1463.17
ftv30a	321.26	321.26	0.37	0.00	0.27	294.25	321.26	0.00	330.24	304.81	321.26	0.00	12.46
ftv30b	349.82	349.82	0.00	0.00	0.33	322.75	349.82	0.00	748.19	334.28	349.82	0.00	15.59
ftv30c	331.12	331.12	0.00	0.00	0.30	296.54	331.12	0.00	25885.07	309.21	331.12	0.00	22.07
p30	212.71	212.71	0.00	0.00	0.27	161.65	212.71	9.56	50000.00	204.35	212.71	0.00	836.31
ry30p	325.09	325.09	0.14	0.00	0.30	285.80	325.09	3.38	50000.00	307.15	325.09	0.00	412.95
ftv33	355.49	355.49	0.00	0.00	0.45	315.30	355.49	0.00	4291.36	337.60	355.49	0.00	23.07
ft35	426.24	426.24	1.58	0.00	0.30	365.34	426.24	6.79	50000.00	394.74	426.24	2.07	50000.00
ftv35	390.43	392.68	1.18	0.57	0.42	345.99	390.43	6.09	50000.00	362.59	390.43	0.00	11480.23
ftv35a	372.28	372.28	1.09	0.00	0.39	332.29	372.28	4.02	50000.00	346.14	372.28	0.00	2017.69
ftv35b	380.78	380.78	1.51	0.00	0.41	342.43	380.78	2.89	50000.00	356.63	380.78	0.00	501.17
ftv35c	373.66	373.66	0.02	0.00	0.37	335.71	373.66	4.60	50000.00	349.88	373.66	0.00	3474.43
p35	243.14	243.14	0.38	0.00	0.36	175.54	244.37	11.26	50000.00	231.71	243.14	0.00	1009.46
ry35p	354.38	354.38	0.00	0.00	0.43	326.71	354.38	0.48	50000.00	340.97	354.38	0.00	29.06
ftv38	417.91	420.90	1.30	0.72	0.50	369.23	417.91	7.56	50000.00	386.74	417.91	1.97	50000.00
ft40	478.38	478.45	0.53	0.01	0.36	408.64	478.38	8.37	50000.00	438.76	478.38	4.45	50000.00
ftv40a	405.49	405.49	0.03	0.00	0.54	362.27	405.49	6.09	50000.00	374.53	405.49	1.47	50000.00
ftv40b	412.11	412.11	0.00	0.00	0.54	369.78	412.11	6.71	50000.00	380.35	412.11	1.31	50000.00
ftv40c	411.78	415.18	2.42	0.83	0.50	364.51	411.78	7.23	34262.71*	383.44	411.78	0.00	756.48
ry40p	401.18	401.27	0.21	0.02	0.60	359.92	401.18	5.69	50000.00	380.49	401.18	0.00	12146.50
p43	291.96	291.96	0.15	0.00	0.43	47.13	292.21	1.34	50000.00	289.87	291.96	0.46	47157.33*
ftv44	446.13	448.35	1.07	0.50	0.55	392.21	446.13	10.53	7821.86*	401.06	446.13	6.30	50000.00
ft45	526.36	527.04	1.80	0.13	0.46	443.93	526.36	9.95	50000.00	478.29	526.36	5.93	50000.00
ftv45b	456.53	458.77	2.37	0.49	0.54	402.23	456.53	9.19	50000.00	415.46	456.85	6.41	50000.00
ftv45c	414.52	418.88	1.45	1.05	0.89	363.62	414.52	9.38	50000.00	377.96	414.52	4.60	50000.00
ry45p	437.30	437.30	1.28	0.00	0.69	391.10	437.30	6.67	50000.00	413.71	437.30	1.23	50000.00
Avg.	370.11	370.79	0.72	0.16	0.43	316.75	370.16	5.26		345.45	370.12	1.34	
AvgS							2582474		6251.38		113910		2137.71
#Best			7	18				25			26		
#Opt.								5				16	
Avg. TGJL16									6251.38				15.18

Table 17
GA with large and quadratic service times on asymmetric instances of Set 2.

# inst	Large service times					Quadratic service times				
	NMI Dyn-B&C		GA+SECs			NMI Dyn-B&C		GA+SECs		
	UB	LB	Val	ValAvg	Time	UB	LB	Val	ValAvg	Time
br17	99.61	99.61	99.61	99.61	0.15	85.63	85.63	85.63	85.63	0.37
ft30	422.00	391.09	422.00	427.64	0.22	337.08	337.08	337.08	339.36	1.16
ftv30a	372.55	372.55	373.32	374.33	0.28	294.44	294.44	294.44	294.60	1.14
ftv30b	403.72	403.72	403.72	403.72	0.33	323.21	323.21	323.21	323.21	1.21
ftv30c	389.61	375.29	389.61	389.75	0.30	300.92	300.92	300.92	301.00	1.08
p30	237.70	237.70	237.70	237.73	0.27	194.07	192.56	194.07	194.43	1.51
ry30p	375.81	361.50	375.81	375.98	0.30	298.30	298.30	298.68	300.67	2.01
ftv33	419.49	407.90	419.49	419.49	0.45	327.11	327.11	327.11	327.11	2.00
ft35	500.69	451.01	504.28	512.05	0.30	395.85	385.90	396.25	400.18	1.43
ftv35	465.54	427.42	465.54	470.18	0.41	361.73	358.32	362.26	365.54	2.18
ftv35a	443.16	411.13	443.90	448.50	0.37	341.50	341.50	341.50	349.65	1.62
ftv35b	450.30	423.13	450.30	458.28	0.42	349.74	349.74	349.74	355.53	1.85
ftv35c	440.66	413.31	440.66	442.88	0.36	340.49	340.49	340.49	340.49	1.94
p35	276.50	270.44	276.50	276.99	0.38	219.50	218.72	219.50	219.55	2.94
ry35p	420.66	417.86	420.66	420.66	0.43	322.79	322.79	322.79	322.79	2.48
ftv38	507.81	458.48	508.51	515.25	0.48	387.19	367.29	387.19	389.86	2.35
ft40	576.25	508.96	577.28	579.77	0.36	444.83	422.55	444.84	447.40	1.66
ftv40a	501.68	495.81	501.68	501.94	0.54	371.08	354.31	371.08	371.23	2.45
ftv40b	505.97	449.81	507.02	507.02	0.54	377.82	368.25	377.82	377.82	3.31
ftv40c	494.86	450.99	494.86	513.54	0.44	376.60	363.27	381.93	386.28	1.75
ry40p	487.20	458.74	487.20	487.86	0.60	366.14	357.67	366.14	367.10	3.62
p43	300.76	296.25	300.96	301.50	0.44	288.17	285.52	288.18	288.29	4.83
ftv44	567.22	485.93	567.22	573.68	0.53	410.78	368.83	411.66	417.90	2.27
ft45	651.52	562.35	658.88	665.61	0.45	491.37	449.61	493.35	501.59	2.83
ftv45b	572.00	495.22	573.02	580.95	0.55	419.45	381.99	422.38	432.64	3.48
ftv45c	516.41	455.39	515.37	527.23	0.80	374.43	349.92	374.43	376.68	3.26
ry45p	544.02	506.58	544.02	550.95	0.67	399.62	375.40	399.53	405.49	6.08
Avg.	442.36	410.67	442.93	446.78	0.42	340.73	330.42	341.19	343.78	2.33
#Best	26		18	4		26		18	6	

of TGJL16, since the latter can solve to optimality within the time limit only 5 instances compared to 16 found by the former. In addition, the computing times of NMI Dyn-B&C are much smaller than those of TGJL16. Similar to what happens for the symmetric instances, the medium service time case is harder than the small service time one, and, for some instances, the methods run out of memory. We also observe that GA+SECs shows a very good performance, being able to obtain 18 best known solutions (over 27 instances) with an average gap of 0.16% in very short computing times.

Similar to what we did for the symmetric instances of Set 2, we also report the performance of the GA+SECs algorithm on the asymmetric instances of Set 2 with large and quadratic service times, and compare these results with those obtained by NMI Dyn-B&C with a time limit of 20000 seconds. As it can be seen from Table 17, although GA+SECs obtains a smaller number of best solution values w.r.t. NMI Dyn-B&C, the difference between the solution values is always very small, and the computing times are consistently short. More precisely, the average gap between the GA+SECs and the NMI Dyn-B&C solution values is 0.10% for the large and 0.11% for the quadratic service times.

6.4. Genetic algorithm on instances of Set 3

In this section, we consider the instances of Set 3 and briefly comment, in Section 6.4.1, on the results obtained by the GA+SECs algorithm on the symmetric instances, and, in Section 6.4.2, the corresponding results for the asymmetric instances. The detailed results are reported in the Appendix (available as Supplementary Material online) both for the symmetric and asymmetric instances with small, medium, large and quadratic service times. Instances of Set 3 are the largest ones, hence we did not execute the exact methods on them, and to evaluate the results obtained by GA+SECs we report the lower bound values obtained by NMI Dyn-B&C at the root node and the results obtained by the randomized NNH algorithm described in Section 5.1. Both GA+SECs and NNH are executed 10 times for each instance. In each table, we report the instance name, and, for the corresponding service time, the lower bound value, the best solution value found by NNH out of 10 runs, and, for GA+SECs, the average and minimum solution values obtained over the 10 runs, and the average computing time (expressed in seconds) over the 10 runs. The computing time of NNH is not reported, since it is always negligible.

6.4.1. Symmetric instances of Set 3

The results obtained for small and medium service times show that significant improvements can be found by applying GA+SECs with respect to NNH (the average minimum solution values are 468.31 versus 549.67 for the small service times, and 530.32 versus 615.04 for the medium ones), and the computing times are about 35 seconds on average. We can observe that the average solution values found by GA+SECs are not very different from the minimum ones, showing that the algorithm is rather stable. We also note that the average percentage gaps between the solution values found by GA+SECs and the lower bound values are rather large: this is not surprising since a similar behavior happens for the instances of Set 2. However, we also observe that, in the latter case, the solution values found by GA+SECs are often close to the best known or even optimal ones. Similar considerations can be done for the large and quadratic service times, although the latter case requires one order of magnitude larger computing times.

6.4.2. Asymmetric instances of Set 3

As shown in the Appendix (available as Supplementary Material online), also for these instances, GA+SECs obtains, in short computing times, significant improvements over the randomized

NNH for all the instances. The largest computing time is required by the case of quadratic service times, the average computing time being about 50 seconds, while for all the other cases it is less than 5 seconds.

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 two 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 of the proposed exact algorithm have been conducted on larger size symmetric instances with up to 58 nodes and on asymmetric instances with up to 45 nodes showing its effectiveness. Finally, the genetic algorithm has been tested on symmetric and asymmetric instances with up to 200 nodes, obtaining good quality solutions in short computing times.

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.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0025. We also acknowledge project CONICYT PFCHA/DOCTORADO BECAS CHILE/2015-72160389. We are very grateful to the reviewers for their insightful comments and, in particular, to one of them for showing the connection between TSP-TS and TDTSP.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2019.11.046

References

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.

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., & Plana, I. (2019). Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm. *Discrete Applied Mathematics, 261*, 28–39.

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

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.

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

Contreras-Bolton, C., & Parada, V. (2015). Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem. *PloS one, 10*(9), e0137724.

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.

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

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

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.

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

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.

Grefenstette, J., Gopal, R., Rosmaita, B., & Gucht, D. (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.

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

Kinable, J., Cire, A. A., & van Hoeve, W.-J. (2017). Hybrid optimization methods for time-dependent sequencing problems. *European Journal of Operational Research, 259*(3), 887–897.

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

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.

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

Öncan, T., Altinel, 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*(1–3), 19–36.

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.

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 *Proceedings of IEEE international conference on evolutionary computation* (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.

Sun, P., Veelenturf, L. P., Dabia, S., & Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research, 264*(3), 1058–1073.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In L. Davis (Ed.), *Handbook of Genetic Algorithms* (pp. 332–349). New York: Van Nostrand Reinhold, chapter 21.

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.

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

Vu, D., Hewitt, M., Boland, N., & Savelsbergh, M. (2018). Solving Time Dependent Traveling Salesman Problems with Time Windows. *Technical Report*. 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*. In *Lecture Notes in Computer Science: 6839* (pp. 17–23). Springer Berlin Heidelberg.