A matheuristic algorithm for the pollution and energy minimization traveling salesman problems

Intl. Trans. in Op. Res. xx (2021) 1–33

A Matheuristic Algorithm for the Pollution and Energy Minimization Traveling Salesman Problems

Valentina Cacchiani[a], Carlos Contreras-Bolton[b], Luis Miguel Escobar-Falcón[c] and Paolo Toth[a,*]

[a]*DEI, University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy*
[b]*Departamento de Ingeniería Industrial, Universidad de Concepción, Edmundo Larenas 219, Concepción 4070409, Chile*
[c]*Program of Systems Engineering, Universidad Libre, Belmonte Av. Las Américas, Pereira 660001, Colombia*
E-mail: valentina.cacchiani@unibo.it [V. Cacchiani]; carlos.contreras.b@udec.cl [C. Contreras-Bolton]; luism.escobarf@unilibre.edu.co [L.M. Escobar-Falcón]; paolo.toth@unibo.it [P. Toth]

Received DD MMMM YYYY; received in revised form DD MMMM YYYY; accepted DD MMMM YYYY

Abstract

The Pollution Traveling Salesman Problem (PTSP) and the Energy Minimization Traveling Salesman Problem (EMTSP) generalize the well-known Asymmetric Traveling Salesman Problem by including environmental issues and the goal of reducing carbon emissions. Both problems call for determining a Hamiltonian tour that, in the PTSP, minimizes a function of fuel consumption and driver cost (where the fuel consumption depends on the distance travelled, the vehicle speed and the vehicle load), while, in the EMTSP, minimizes a function depending on the vehicle load and the traveled distances. For both the PTSP and the EMTSP, we propose a matheuristic algorithm, that uses the solution of the Linear Programming (LP) relaxation of a Mixed Integer Linear Programming (MILP) model for the considered problem to determine good initial feasible solutions, applies a multi-operator genetic algorithm to improve these solutions, and refines the best solution found through an Iterated Local Search procedure. In order to evaluate the performance of the proposed matheuristics, we compare them with exact and heuristic algorithms from the literature on benchmark instances of both problems.

Keywords: Pollution Traveling Salesman Problem; Energy Minimization Traveling Salesman Problem; Matheuristic algorithm; Linear Programming relaxation

1. Introduction

We study two generalizations of the Asymmetric Traveling Salesman Problem (ATSP), both focusing on environmental issues: given a set of customers to be visited by a vehicle, the goal is to reduce its impact on the environment. The Pollution Traveling Salesman Problem (PTSP) was recently introduced

*Author to whom all correspondence should be addressed (e-mail: paolo.toth@unibo.it).

in Cacchiani et al. (2018): it requires to determine a Hamiltonian tour minimizing *pollution*, i.e., fuel consumption (dependent on vehicle speed and load), but also accounting for driver costs. The Energy Minimization Traveling Salesman Problem (EMTSP) was newly introduced in Wang et al. (2020): it requires to find a Hamiltonian tour minimizing *energy*: the latter is measured as the sum, over all arcs, of the products between distance and vehicle load (including curb weight).

In recent years, many works about routing problems with the goal of reducing pollution and energy consumption have appeared in the literature (see, e.g., the three surveys by Demir et al. (2014), Bektaş et al. (2019), and Moghdani et al. (2020)). Solution approaches, proposed in the latest years, include heuristic algorithms (see, e.g., Andelmin and Bartolini (2019)), matheuristic methods (see, e.g., Macrina et al. (2019)), exact approaches (see, e.g., Andelmin and Bartolini (2017), Bruglieri et al. (2019a), Bruglieri et al. (2019b), Yu et al. (2019)). These methods studied problems characterized by the goal of reducing emissions and by real-life constraints: for example, the use of alternative fuel vehicles, heterogeneous fleet, time-windows, partial battery recharging. In the next sections, we focus our literature overview only on the most relevant works for our study: we review works on the Pollution Routing Problem (PRP) and on the Energy Minimization Vehicle Routing Problem (EMVRP), in addition to those on the PTSP and on the EMTSP.

1.1. Literature overview on PRP and PTSP

The PRP is a generalization of the Vehicle Routing Problem (VRP) aiming at minimizing distance, greenhouse emissions, fuel, travel times and costs. It was introduced in Bektaş and Laporte (2011), and extended in Demir et al. (2012) to allow for low travel speeds. The PTSP corresponds to the single vehicle version of the PRP as modelled in Demir et al. (2012).

Bektaş and Laporte (2011) proposed a MILP model for the PRP: the objective function requires to minimize fuel consumption, that depends on vehicle speeds and loads, and driver costs. In this model, they discretized the speed, i.e., a set of speed levels was considered for each arc. They performed an analysis to evaluate the effects of various elements, such as time windows, demand variation, vehicle types and number of vehicles. On the contrary, Fukasawa et al. (2016b) considered the speed as a continuous decision variable within an interval. They proposed two mixed-integer convex optimization models. Beside these exact methods, the majority of the solution methods for the PRP are heuristic or metaheuristic algorithms. Demir et al. (2012) proposed an Adaptive Large Neighborhood Search (ALNS) heuristic algorithm for the PRP. It consists of two stages: the first stage applies an ALNS to the VRP with Time Windows (VRPTW), while the second one executes a speed optimization algorithm. The latter is applied on the solution computed in the first stage to determine the optimal vehicle speed along each arc of the routes. Kramer et al. (2015b) proposed a matheuristic approach for the PRP, and for two green VRP variants: the EMVRP, and the Fuel Consumption VRP. This approach consists of an Iterated Local Search algorithm based on a Set Partitioning formulation, combined with a speed optimization procedure.

Kramer et al. (2015a) studied a variant of the PRP, in which both speed and departure time from the depot have to be optimized. The flexibility of the departure times allows using additional routes and reducing costs. They proposed an exact algorithm for the optimization of speeds and departure times for a fixed route, and embedded it in the Iterated Local Search algorithm proposed in Kramer et al. (2015b). Dabia et al. (2017) proposed an exact method based on a branch-and-price algorithm for this variant. In

this algorithm, the master problem is solved by column generation. The pricing problem, consisting of a speed and departure time elementary shortest path problem with resource constraints, is solved through a labelling algorithm. A generalization of this problem is the Time-Dependent PRP (TDPRP): it takes into account the limitation on the travel speed and the increase of the pollution due to the traffic congestion during rush hours. Franceschetti et al. (2013) proposed an Integer Linear Programming (ILP) formulation for this problem. Moreover, they developed an algorithm to optimize departure times and travel speeds on a fixed route. Recently, Franceschetti et al. (2017) presented a metaheuristic ALNS algorithm for the TDPRP embedding the speed optimization procedure of Franceschetti et al. (2013).

Another extension of the PRP is the bi-objective PRP: its two distinct objectives take into account the minimization of fuel consumption and of driving time. The ALNS algorithm proposed in Demir et al. (2012) was used to solve this extension by integrating four multi-objective methods: the weighting method, the weighting method with normalization, the epsilon-constraint method and a hybrid method. The bi-objective PRP was also studied in Costa et al. (2018). They obtained an approximation of the Pareto front by a two-phase local search heuristic algorithm: the first phase solves a set of weighted sum PRPs, while the second phase consists of applying a Pareto local search procedure.

Koç et al. (2014) studied a PRP with heterogeneous fleet of vehicles: they proposed a hybrid evolutionary algorithm, combined with a speed optimization procedure. In addition, they performed analyses on the effects of the different cost components (distance, fuel and emissions, driver), and of the heterogeneous fleet. Saka et al. (2017) considered the PRP with heterogeneous fleet and customer deadlines: instead of using a speed optimization procedure to determine the optimal speed for each arc, the speed optimization problem is used to estimate the cost of moves in a local search heuristic algorithm.

The pickup and delivery extension of the PRP was studied in Bravo et al. (2019) and in Majidi et al. (2018). The former work considers the multi-objective feature of the problem and proposes an evolutionary algorithm. The latter introduces a Mixed Integer Non-Linear Programming model, and proposes an ALNS heuristic for solving the problem.

Cacchiani et al. (2018) introduced the PTSP: they presented a Mixed Integer Linear Programming (MILP) model, enhanced with explicit subtour elimination constraints. This model was solved by a Cut-and-Branch algorithm (C&B) and tested on instances with up to 50 customers. Only instances with up to 25 customers were solved to proven optimality within the time limit of two hours. In order to find solutions within shorter computing times, an Iterated Local Search algorithm (ILS) (based on the framework by Lourenço et al. (2019)) was proposed and tested on the same set of instances.

1.2. Literature overview on EMVRP and EMTSP

The EMVRP was introduced by Kara et al. (2007), who presented a MILP model for the problem. Gaur et al. (2013) considered four versions of the problem with specific characteristics (e.g., all customer demands are the same, the vehicles have infinite capacities). They presented approximation algorithms with different constant approximation factors (up to factor 4) for these versions.

Xiao et al. (2012) studied a problem very similar to the EMVRP, in which there is a fixed cost for using a vehicle. They proposed a simulated annealing algorithm for its solution. The algorithm was tested on benchmark instances of the VRP. Zachariadis et al. (2015) extended the problem to its pickup and delivery variant. They proposed a branch-and-cut procedure for the solution of small-size instances

and a metaheuristic algorithm. The latter constructs a feasible solution and iteratively applies local search moves (customer relocation, customer swap and 2-opt move). It accepts the highest-quality neighboring solution, and employs diversification to escape from local optimal solutions. Computational experiments were executed on instances of the EMVRP (taken from Xiao et al. (2012)) and of its pickup and delivery version. Tiwari and Chang (2015) developed a block recombination algorithm: customers are divided into clusters, each one representing a block, and block recombination techniques are used to determine better solutions. Fukasawa et al. (2016a) proposed an arc-load model, enhanced with cycle elimination constraints, and a set-partitioning formulation strengthened by additional constraints. They theoretically compared these formulations and the one presented in Kara et al. (2007). In addition, they developed a branch-cut-and-price algorithm to solve the set partitioning formulation. Computational experiments showed the effectiveness of the proposed method.

For what concerns the single vehicle problem, Suzuki (2011) studied a TSP with time windows, in which fuel consumption elements are taken into account: the objective function includes the vehicle payload (i.e., it minimizes the distance that the vehicle must travel with a heavy payload), and the fuel consumption while waiting at customer locations (i.e., it minimizes the sum of waiting times). A metaheuristic algorithm, based on compressed annealing, was developed for this problem, and tested on instances with up to 15 customers. The EMTSP was introduced by Wang et al. (2020): they presented a mathematical model, based on the multiple-vehicle version of the arc-load model by Fukasawa et al. (2016a), and two lower bounds. They developed an approximation algorithm based on the classic Christofides's Heuristic for the symmetric TSP. In addition, they developed three heuristic algorithms and a branch-and-bound approach. All solution methods were tested on instances with up to 30 nodes, and the results were compared to those obtained by applying CPLEX to the mathematical model. The branch-and-bound algorithm was capable of solving to optimality instances with up to 28 nodes within the time limit of one hour.

1.3. Contributions

Both the PTSP and the EMTSP are hard problems whose solution requires non negligible computing time. In this work, we propose a Matheuristic Algorithm (MA) with the aim of finding good solutions for the PTSP and the EMTSP instances in short computing times. This algorithm employs the solution of the Linear Programming (LP) relaxation of a MILP model for the considered (PTSP or EMTSP) problem to construct an initial set of feasible Hamiltonian tours. Then, it applies a multi-operator Genetic Algorithm (GA) to improve these solutions, by using effective crossover and mutation TSP operators from the literature. Finally, it refines the best solution found by applying the ILS algorithm proposed in Cacchiani et al. (2018). The contributions of this work are the following ones:

- to show the relationship between the PTSP and the EMTSP,
- to derive Hamiltonian tours to construct the initial population, by starting from the LP-solution of a MILP model,
- to improve these tours through crossover and mutation operators, and by applying the ILS algorithm as a refinement procedure,
- to fine tune all parameters of MA,
- to find better solutions than those found by the C&B and ILS algorithms, proposed in Cacchiani et al.

(2018) for the PTSP, on instances with up to 200 customers in short computing times,
- to find the best solution, in short computing times, for all (but two) instances presented in Wang et al. (2020) for the EMTSP, most of which were solved to optimality by their branch-and-bound algorithm.

Section 2 is dedicated to the PTSP, while Section 3 to the EMTSP: in both sections we formally define the considered problem and present a MILP model for it. The proposed MA algorithm is presented in Section 4. In Section 5, we report the results of the parameter tuning for MA, and extensive computational experiments on benchmark instances of both problems. Finally, we draw some conclusions in Section 6.

2. The Pollution Traveling Salesman Problem

2.1. Problem Definition

The PTSP is a single-vehicle variant of the PRP in which time-windows and capacity constraints are not considered. The PTSP is defined on a complete directed graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{0, \dots, n\}$ is the set of nodes, 0 is a depot and $\mathcal{A}$ is the set of arcs. For each arc $(i, j) \in \mathcal{A}$, the associated non-negative distance d_{ij} is given.

Set $\mathcal{N}_0 = \mathcal{N} \setminus \{0\}$ is the customer set. Each customer $i \in \mathcal{N}_0$ has a non-negative demand q_i, and a service time t_i, and must be visited exactly once. A single vehicle with capacity $D = \sum_{i \in \mathcal{N}_0} q_i$ is available for visiting all customers. The driver wage per unit time is defined as u_d.

The main novelty of the PTSP with respect to the TSP is that it takes fuel consumption into account with the aim of limiting pollution. Fuel consumption varies according to two main elements: i) the vehicle speed, that can be different on different arcs, but is kept fixed, along the same arc $(i, j) \in \mathcal{A}$, for the entire distance d_{ij}, and ii) the vehicle weight, that depends on the weight of the empty vehicle (curb weight) and on the load it is carrying.

For what concerns the vehicle speed, we consider, as in Bektaş and Laporte (2011) and in Demir et al. (2012), a discretized speed function defined by $|\mathcal{R}|$ non-decreasing speed levels: each $r \in \mathcal{R}$ corresponds to a speed interval $[v_r^l, v_r^u]$ in the range $[v^l, v^u]$ (where v^l and v^u are, respectively, the lower and upper speed limits for the considered instance). As proposed in Bektaş and Laporte (2011), for each level $r \in \mathcal{R}$, we compute the average speed $\bar{v}^r = (v_r^l + v_r^u)/2$. For each arc we can select a different speed inside the discrete set of speed levels. The weight of the vehicle is expressed, for each arc $(i, j) \in \mathcal{A}$, as the sum of its curb weight w and the load f_{ij} carried by the vehicle on this arc. More precisely, we adopt the fuel consumption expression proposed in Demir et al. (2012), which extends the one presented in Bektaş and Laporte (2011) to allow for speeds lower than 40 kilometer/hour, and accounts for several factors, such as engine features (friction, speed, efficiency), acceleration, road angle. In particular, for a given arc $(i, j) \in \mathcal{A}$ of length d_{ij}, traversed at speed v by a vehicle carrying a load $w + f_{ij}$ the fuel consumption can be expressed as (see Demir et al. (2012)):

$$F(v) = \lambda k N V d_{ij}/v + \lambda \beta \gamma d_{ij} v^2 + \lambda w \gamma \alpha_{ij} d_{ij} + \lambda \gamma \alpha_{ij} f_{ij} d_{ij} \tag{1}$$

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

(i, j), and on the acceleration τ (meter/second^2). In particular, the first two terms of (1) represent the speed-induced energy requirements, while the last two terms represent the load-induced energy requirements. The values of all the parameters, taken from Demir et al. (2012), are reported in Table A1 of the Appendix.

The PTSP calls for determining the minimum cost Hamiltonian tour that departs from the depot and visits each customer exactly once by serving its demand, where the cost is given by the sum of fuel consumption and driver wage.

2.2. The PTSP MILP model

We present the MILP model proposed in Cacchiani et al. (2018), where, in addition to the classical binary variables x_{ij} $(i, j) \in \mathcal{A}$ used to define which arcs compose the optimal Hamiltonian tour, the following variables were employed to express the fuel consumption:

- non-negative variables f_{ij} representing the load on the vehicle on arc $(i, j) \in \mathcal{A}$;
- binary variables z_{ij}^r assuming value 1 if arc $(i, j) \in \mathcal{A}$ is traversed at speed level $r \in \mathcal{R}$.

The MILP model for the PTSP reads as follows:

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

$$+ \sum_{(i,j)\in\mathcal{A}} \lambda w\gamma\alpha_{ij} d_{ij} x_{ij} + \sum_{(i,j)\in\mathcal{A}} \lambda\gamma\alpha_{ij} d_{ij} f_{ij} \tag{3}$$

$$+ \quad u_d\Big(\sum_{(i,j)\in\mathcal{A}} \sum_{r\in\mathcal{R}} (d_{ij}/\bar{v}^r) z_{ij}^r + \sum_{i\in\mathcal{N}_0} t_i\Big) \tag{4}$$

subject to

$$\sum_{j\in\mathcal{N}_0} f_{0j} = D \tag{5}$$

$$\sum_{j\in\mathcal{N}_0} f_{j0} = 0 \tag{6}$$

$$\sum_{j\in\mathcal{N}} x_{ij} = 1,\ \forall i \in \mathcal{N} \tag{7}$$

$$\sum_{i\in\mathcal{N}} x_{ij} = 1,\ \forall j \in \mathcal{N} \tag{8}$$

$$\sum_{j\in\mathcal{N}} f_{ji} - \sum_{j\in\mathcal{N}} f_{ij} = q_i,\ \forall i \in \mathcal{N}_0 \tag{9}$$

$$q_j x_{ij} \leq f_{ij} \leq (D - q_i)\, x_{ij}, \forall\, (i,j) \in \mathcal{A} \tag{10}$$

$$\sum_{r \in \mathcal{R}} z_{ij}^r = x_{ij},\ \forall\, (i,j) \in \mathcal{A} \tag{11}$$

$$\textstyle\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N} \setminus \mathcal{S}} x_{ij} \geq 1, \quad \forall \mathcal{S} \subset \mathcal{N}, \quad \{0\} \in \mathcal{S}, |\mathcal{S}| \geq 2 \tag{12}$$

$$x_{ij} \in \{0,1\},\ \forall\, (i,j) \in \mathcal{A} \tag{13}$$

$$f_{ij} \geq 0,\ \forall\, (i,j) \in \mathcal{A} \tag{14}$$

$$z_{ij}^r \in \{0,1\},\ \forall\, (i,j) \in \mathcal{A}, \forall r \in \mathcal{R} \tag{15}$$

The objective function consists of three main components to be minimized: (2) and (3) represent the fuel consumption, as defined in (1), by taking into account, respectively, the energy required by speed variations and the energy used to curry the curb weight and the load on the vehicle, while (4) corresponds to the driver wage, where the term in the external brackets is the total tour duration which depends on the speeds on the used arcs and on the service times at the customers. Constraints (5) and (6) ensure, respectively, that the vehicle leaves full and returns empty at the depot. Constraints (7) and (8) guarantee that each node is visited exactly once. Constraints (9) and (10) define the load of the vehicle on each visited arc (and implicitly forbid subtours). Constraints (11) link the x and z variables by imposing that exactly one speed level is chosen for each used arc $(i,j) \in \mathcal{A}$. Finally, constraints (12) are the explicit subtour elimination constraints proposed in Dantzig et al. (1954) for the ATSP, and constraints (13)-(15) define the variable domains.

As in the PRP (Bektaş and Laporte (2011)), time window constraints can be included in the PTSP. Let $[a_i, b_i]$ be the time window in which customer i has to be visited, and y_i an additional variable representing the time at which customer i is visited by the vehicle ($i \in \mathcal{N}_0$). The time window constraints are formally defined as:

$$a_i \leq y_i \leq b_i,\ \forall i \in \mathcal{N}_0 \tag{16}$$

$$y_i - y_j + t_i + d_{ij} \sum_{r \in \mathcal{R}} z_{ij}^r / \bar{v}^r \leq M(1 - x_{ij}),\ \forall i \in \mathcal{N},\ j \in \mathcal{N}_0,\ i \neq j, \tag{17}$$

with M a large constant value.

When constraints (16) and (17) are not included in the MILP model, the optimal speed can be determined a priori for every arc. This was observed by Kramer et al. (2015b) for the PRP with variables representing continuous speed values: in that case, for each arc $(i,j) \in \mathcal{A}$, the fuel consumption function including driver wages is a convex function (assuming that acceleration and road angle are null) and, thus, the speed value that minimizes the fuel costs can be obtained as the minimum of the function by nullifying its derivative. In our case, we deal with discrete speeds, hence the function is not convex. However, since the number of speeds is finite, we can precompute the optimal speed for every arc by enumerating all possible speeds and selecting the one achieving the smallest fuel consumption plus driver cost. Indeed, the choice of the speed is independent of the vehicle load, and, when constraints (16) and (17) are not imposed, also of the visiting sequence of the customers. We observe that, if the same lower and upper speed limits v^l and v^u and the same number of speed levels $|\mathcal{R}|$ are used for all arcs, then there is a single optimal speed value (or two, depending on the discrete set adopted). In the next section, we present the simplified model for the PTSP with Precomputed Speeds (PTSP-PS).

2.3. The PTSP with Precomputed Speeds (PTSP-PS)

Once the speeds have been optimally precomputed for all arcs, model (2)-(15) can be rewritten as:

$$\begin{aligned}
&\text{Min} \sum_{(i,j)\in\mathcal{A}} \lambda k N V \frac{d_{ij}}{\bar{v}_{ij}} x_{ij} + \sum_{(i,j)\in\mathcal{A}} \lambda\beta\gamma d_{ij}(\bar{v}_{ij})^2 x_{ij} \\
&+ \sum_{(i,j)\in\mathcal{A}} \lambda w\gamma\alpha_{ij} d_{ij} x_{ij} + \sum_{(i,j)\in\mathcal{A}} \lambda\gamma\alpha_{ij} d_{ij} f_{ij} \\
&+ \quad u_d\Big(\sum_{(i,j)\in\mathcal{A}} \frac{d_{ij}}{\bar{v}_{ij}} x_{ij} + \sum_{i\in\mathcal{N}_0} t_i\Big) \qquad (18) \\
&\text{subject to} \quad (5)-(10), \quad (12)-(14),
\end{aligned}$$

where $\bar{v}_{ij}$ represents the optimal speed to be used for travelling along arc $(i,j) \in \mathcal{A}$. In this model, variables z^r_{ij} are not present, and are replaced by variables x_{ij} in (2) and (4), in addition, constraints (11) are removed. In the following sections, we will refer to this model (instead of to model (2)-(15)), since we do not deal with the time window constraints (16) and (17).

3. The Energy Minimization Traveling Salesman Problem

3.1. Problem Definition

The EMTSP is defined on a complete directed graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{0, \ldots, n\}$ is the set of nodes, 0 is a depot and $\mathcal{A}$ is the set of arcs. For each arc $(i,j) \in \mathcal{A}$, the associated non-negative distance d_{ij} is given. Set $\mathcal{N}_0 = \mathcal{N} \setminus \{0\}$ is the customer set. Each customer $i \in \mathcal{N}_0$ has a non-negative demand q_i, and must be visited exactly once. A single vehicle with capacity $D = \sum_{i\in\mathcal{N}_0} q_i$ and curb weight w is available for visiting all customers.

The EMTSP calls for determining a Hamiltonian tour that departs from the depot and visits each customer exactly once by serving its demand, minimizing the sum, over all arcs, of the products between the distance and the vehicle load (including the curb weight).

3.2. The EMTSP MILP model

By using variables x_{ij} and f_{ij} for all arcs $(i,j) \in \mathcal{A}$, defined as in the PTSP formulation, the MILP model for the EMTSP reads as follows:

$$\text{Min} \sum_{(i,j)\in\mathcal{A}} (w d_{ij} x_{ij} + d_{ij} f_{ij}) \qquad (19)$$

$$\text{subject to} \quad (5)-(10), \quad (12)-(14). \qquad (20)$$

The EMTSP constraints coincide with the constraints of the PTSP-PS. The objective function (19)

aims at the minimization of the energy consumed by the vehicle, measured, along each arc $(i,j) \in \mathcal{A}$, as the arc length d_{ij} times the total vehicle load on that arc, that is given by the curb weight w plus its load f_{ij}.

Observation 1. The PTSP-PS is a (slight) generalization of the EMTSP.

Proof. The constraints of both problems are the same. To show that the objective function of the PTSP-PS is more general than that of the EMTSP, we rewrite objective function (18) as follows:

$$\begin{aligned}
&\text{Min} \sum_{(i,j)\in\mathcal{A}} w\,[\frac{1}{w}\,(\lambda k N V \frac{d_{ij}}{\bar{v}_{ij}} + \lambda\beta\gamma d_{ij}(\bar{v}_{ij})^2 + u_d \frac{d_{ij}}{\bar{v}_{ij}}) + \lambda\gamma\alpha_{ij} d_{ij}]\; x_{ij} \\
&+ \sum_{(i,j)\in\mathcal{A}} \lambda\gamma\alpha_{ij} d_{ij} f_{ij} \\
&+ u_d \sum_{i\in\mathcal{N}_0} t_i.
\end{aligned}$$

We observe that $u_d \sum_{i\in\mathcal{N}_0} t_i$ is a constant term and, thus, we can further rewrite the objective function as:

$$u_d \sum_{i\in\mathcal{N}_0} t_i + \text{ Min} \sum_{(i,j)\in\mathcal{A}} (w D'_{ij} x_{ij} + D''_{ij} f_{ij}) \tag{21}$$

with $D'_{ij} = \frac{1}{w}\,(\lambda k N V \frac{d_{ij}}{\bar{v}_{ij}} + \lambda\beta\gamma d_{ij}(\bar{v}_{ij})^2 + u_d \frac{d_{ij}}{\bar{v}_{ij}}) + \lambda\gamma\alpha_{ij} d_{ij}$ and $D''_{ij} = \lambda\gamma\alpha_{ij} d_{ij}$. Note that (21) has the same form as (19), but the former corresponds to the latter only when $D'_{ij} = D''_{ij}$. □

4. Matheuristic Algorithm (MA)

The proposed MA consists of three phases: in the first phase (Section 4.1) several initial Hamiltonian tours are built by starting from the solution of the LP-relaxation of the MILP model (corresponding to (18), (5)-(10), (12)-(14) for the PTSP-PS, and to (19)-(20) for the EMTSP), the second phase (Section 4.2) is the core of the solution process corresponding to the GA algorithm, and is aimed at improving the initial tours, and in the third phase (Section 4.3) an ILS refinement, proposed in Cacchiani et al. (2018), is applied to further improve the solutions computed in the previous phase.

The only two differences in the application of MA to the PTSP-PS or to the EMTSP are in the first phase, in which the appropriate model is used, and in the evaluation of the solution cost (during the GA algorithm and the ILS refinement) that is based on the objective function of the corresponding problem. In addition, in the PTSP-PS, we apply a preprocessing phase that computes, for each arc $(i,j) \in \mathcal{A}$, the optimal speed before the construction of the starting solution. This computation is executed by enumerating, for each $(i,j) \in \mathcal{A}$, all the possible speeds v^r ($r \in \mathcal{R}$), and selecting the one ($\bar{v}_{ij}$) that leads to the smallest value of the sum between the fuel consumption and the driver cost, expressed as in objective function (18).

In Section 4.4 we report a summary of MA and its pseudo-code. The parameters of MA, that will be presented in the following sections, are tuned according to a procedure described in Section 4.5, and the chosen values are reported in Section 5.1.1.

4.1. Initial Hamiltonian tours

An initial population of Hamiltonian tours is computed by using five procedures based on three different heuristic methods described in the following. Each procedure is applied with a given probability that has been fine tuned (see Section 4.5). These heuristic algorithms include an LP-based heuristic, that is executed on three different LP-solutions, a Random-Heuristic and a Nearest Neighbor-Heuristic, giving rise to the five procedures. The considered heuristic methods are the following:

- Random-Heuristic: The tour is generated randomly. This algorithm is applied with probability PRH.
- Nearest-Neighbor-Heuristic: The tour is generated by applying the Nearest Neighbor Heuristic (NNH) (Flood (1956)), that randomly starts from one of the nodes in $\mathcal{N}$. This algorithm is applied with probability $PNNH$.
- LP-based heuristic: The LP-based heuristic (described in Algorithm 2) is based on the LP-relaxation of model (18), (5)-(10), (12)-(14) for the PTSP-PS, and model (19)-(20) for the EMTSP. This heuristic generates a feasible solution by using a randomized NNH, based on the LP-solution (x_{ij}). In particular, three LP-solutions x^1, x^2 and x^3 are determined by Algorithm 1 (Computation of the LP-solutions), and each of them is used in Algorithm 2 (LP-based heuristic). The latter generates a tour based on the values assumed by the x_{ij} variables in the optimal LP-solution. In addition, it includes randomization so that more than one tour can be obtained from the same LP-solution. This algorithm is applied with probability $PLPH$ (with $PLPH = 1 - PRH - PNNH$), and the same probability is then used to select x^1, x^2 or x^3. The usefulness of the LP-based heuristic will be shown in Section 5: although many best solutions are obtained by MA even without applying the LP-based heuristic, it allows the algorithm to find better solutions for the larger size instances.

Algorithm 1 (Computation of the LP-solutions) is based on the MILP model of the considered problem, that contains the classical subtour elimination constraints (SECs) proposed in Dantzig et al. (1954) for the ATSP: to solve its LP relaxation we start with an empty set of SECs (denoted with $\mathcal{F}$ in the description of Algorithm 1), and adopt the separation procedure proposed in Padberg and Rinaldi (1990) to identify violated SECs, that are iteratively added to the LP-relaxed model. The LP-relaxed model is solved by a general purpose solver (CPLEX in our experiments). The procedure is iterated until either no violated SECs exist or k^3 iterations have been executed. Parameters k^1, k^2 and k^3 (with $k^1 < k^2 < k^3$) are used to store the LP-solutions obtained after, respectively, k^1, k^2 and k^3 SECs have been added to the model. In this way, three (generally different) LP-solutions can be used by Algorithm 2 (LP-based heuristic). If less than k^1 (k^2 or k^3, respectively) SECs are found, then we store in x^1 (x^2, x^3, respectively) the last obtained solution x. Parameters k^1, k^2 and k^3 assume different values according to the instance size.

In the description of Algorithm 2 (LP-based heuristic), x denotes the considered LP-solution, h the last visited node and T the set of visited nodes. Algorithm 2 works as follows: the tour starts from node h initialized to be the depot 0. Then, in order to find the successor node in the tour, we consider, with

Algorithm 1 Computation of the LP-solutions x^1, x^2 and x^3

$\mathcal{F} \leftarrow \emptyset$; $l \leftarrow 0$
repeat
    $l \leftarrow l + 1$
    $x \leftarrow$ solution of the LP-relaxation of the MILP model with respect to the SEC set $\mathcal{F}$
    if $l = k^1$ **then**
        $x^1 \leftarrow x$
    else if $l = k^2$ **then**
        $x^2 \leftarrow x$
    else if $l = k^3$ **then**
        $x^3 \leftarrow x$
    end if
    $\mathcal{S} \leftarrow$ violated SEC found by the separation-procedure(x)
    if $\mathcal{S} \neq \emptyset$ **then**
        add $\mathcal{S}$ to the SEC set $\mathcal{F}$
    end if
until $\mathcal{S} = \emptyset$ **or** $l = k^3$
if $l < k^1$ **then**
    $x^1 \leftarrow x$
end if
if $l < k^2$ **then**
    $x^2 \leftarrow x$
end if
if $l < k^3$ **then**
    $x^3 \leftarrow x$
end if

probability $Prinit$ (see Section 4.5), each unvisited node k, and determine the maximum value max of the quantities $x_{hk} + x_{kh}$. The unvisited node j corresponding to the maximum value max is selected as the next node in the tour, unless $max = 0$ at the end of the for loop. In that case (i.e., if no unvisited node has been evaluated due to the random value r), we select the unvisited node j with the smallest distance d_{hj}. The selected node j is inserted in the tour, and becomes the last visited node h. The procedure is repeated until we obtain a complete tour. Notice that the randomization occurs at line 7. Therefore, given an LP-solution (x_{ij}), different tours can be obtained through Algorithm 2.

An example of the application of the LP-based heuristic is reported in Figure 1: the two arcs corresponding to the solid arrows have been selected in the partial tour, and the last visited node is h. At this iteration, the procedure considers all the unvisited nodes (indicated in gray), and all the potential arcs (indicated by dotted arcs). For each unvisited node k, the procedure generates a random number r between 0 and 1: if this number is larger or equal than $Prinit$, then the node is not evaluated at this iteration. Otherwise, we compute the sum of the values of the variables $x_{hk} + x_{kh}$ corresponding to the two arcs connecting h and k. The node k (among all the evaluated gray nodes) giving the maximum value of this sum, is selected as the next node j in the tour, and then it becomes the next node h.

Algorithm 2 LP-based heuristic(x)

$h \leftarrow 0$
$T \leftarrow \{h\}$
repeat
  $max \leftarrow 0$
  for $k \in \mathcal{N} \setminus T$ **do**
    $r \leftarrow rnd(0,1)$
    if $r <$*Prinit* **then**
      $H \leftarrow x_{hk} + x_{kh}$
      if $H > max$ **then**
        $max \leftarrow H$
        $j \leftarrow k$
      end if
    end if
  end for
  if $max = 0$ **then**
    $j \in \text{argmin}_{\ell \in \mathcal{N} \setminus T}\{d_{h\ell}\}$
  end if
  $T \leftarrow T \cup \{j\}$
  $h \leftarrow j$
until a complete tour T is obtained

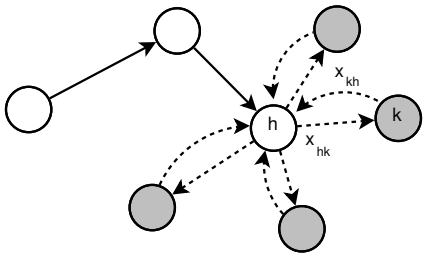

Fig. 1. Example of the application of the LP-based heuristic.

The initial population I_0 of the Hamiltonian tours is obtained by Algorithm 3 (Initialization): it starts by solving the LP-relaxed model to determine the three solutions x^1, x^2 and x^3 with Algorithm 1 (Computation of the LP-solutions), and then applies the five procedures described above to derive the initial set of the Hamiltonian tours. Once a tour has been determined, in order to improve it, a 2-opt-improvement procedure is executed. This procedure uses as arc costs the original distances d_{ij} $(i,j) \in \mathcal{A}$, but each time an improvement is possible, it checks if the value of the corresponding objective function (minimization of (18) for the PTSP-PS, and minimization of (19) for the EMTSP) is also improved, and accepts the 2-opt move only in this case.

Algorithm 3 Initialization

x^1, x^2, $x^3 \leftarrow$ Computation of the LP-solutions
$I_0 \leftarrow \emptyset$
for $t \leftarrow 1$ **to** number of individuals **do**
    $r \leftarrow rnd(0,1)$
    if $r < PRH$ **then**
        $I_{0,t} \leftarrow$ Random-Heuristic()
    else if $r < PRH + PNNH$ **then**
        $I_{0,t} \leftarrow$ Nearest-Neighbor-Heuristic()
    else
        $r' \leftarrow rnd(0,1)$
        if $r' < 0.33$ **then**
            $I_{0,t} \leftarrow$ LP-based heuristic(x^1)
        else if $r' < 0.66$ **then**
            $I_{0,t} \leftarrow$ LP-based heuristic(x^2)
        else
            $I_{0,t} \leftarrow$ LP-based heuristic(x^3)
        end if
    end if
    $I_{0,t} \leftarrow$ 2-opt-improvement($I_{0,t}$)
    $I_0 \leftarrow I_0 \cup \{I_{0,t}\}$
end for

4.2. Improvement through the multi-operator Genetic Algorithm GA

Genetic Algorithms are effective metaheuristic algorithms that have been successfully applied to solve the ATSP and several its variants (Potvin (1996), Yuan et al. (2013), Morán-Mirabal et al. (2014), Groba et al. (2015), Zhang et al. (2018)). Often these algorithms use only single operators for crossover and mutation, disregarding the potential synergy of multi-operators. However, the crossover and mutation operators can complement each other, generating a synergy which provides better results than those obtained by using single operators (see e.g., Elsayed et al. (2011), Li et al. (2013), Contreras-Bolton and Parada (2015), Mashwani et al. (2017)). We propose an approach based on a multi-operator Genetic Algorithm (GA) that effectively combines several operators from the literature.

In the following sections, we describe GA and its components. In Section 4.2.1 we present the chromosome representation and the fitness function. The crossover operators are described in Section 4.2.2, while the mutation operators are illustrated in Section 4.2.3. Finally, Section 4.2.4 presents the genetic parameters.

4.2.1. Representation and fitness function

A permutation representation is used, where each individual corresponds to a Hamiltonian tour. The objective function, consisting of (18) for the PTSP-PS, and of (19) for the EMTSP, is used as fitness function. For the PTSP-PS, the objective function requires to minimize the fuel consumption and the

driver wage, that depend on the speed and load of the vehicle in the traversed arcs. For the EMTSP, the objective function is the minimization of the sum, over all arcs, of the products between the distance and the vehicle load. In both cases, once a tour has been determined, the load over each arc is known. In addition, the optimal speed for each arc is known from the preprocessing phase. Therefore, the computation of the fitness can be done in linear time with respect to the number of customers.

4.2.2. Crossover operators

Four different crossover operators are used: Order Based Crossover (OX2) (Syswerda (1991)), Distance Preserving Crossover (DPX) (Freisleben and Merz (1996)), Heuristic Crossover (HX) (Grefenstette et al. (1985)) and Uniform Nearest Neighbor (UNN) (Buriol et al. (2004)). The probabilities of choosing each of these operators (P_{OX2}, P_{DPX}, P_{HX}, and P_{UNN}) are determined by the tuning procedure presented in Section 4.5. As it will be shown in the computational results, the contribution of each operator is useful to achieve the best solutions. A short description of each operator is reported:

- OX2: randomly select a subset of consecutive positions from the first parent and copy the corresponding nodes in the offspring. Then, copy the remaining nodes from the second parent, according to the order they have in the second parent, and connect all nodes with arcs based on the node order.
- DPX: the nodes contained in the first parent are copied into the offspring and all the arcs not in common with the second parent are deleted, leading to a set of disconnected paths (note that some paths could consist of a single node). These paths are then reconnected, by starting from a randomly chosen path, without using any of the arcs that are contained in only one of the parents. In particular, given a path that ends at node i, the nearest available neighbor node k among the initial nodes of the remaining paths is taken and arc (i, k) is added to the tour, unless (i, k) is contained in one of the two parents. The procedure is repeated until all paths have been reconnected in a tour.
- HX: first, a node is randomly selected to be the current node of the offspring. Then, the cheapest of the two arcs, in the two parents, leaving that node is selected. The procedure is repeated to extend the partial tour until a complete tour has been constructed. If the cheapest arc introduces a subtour (i.e., it connects the selected node with a visited node), then a random arc is chosen to extend the tour.
- UNN: initially, all arcs in common between both parents are copied into the offspring, leading to a set of disconnected paths. The remaining arcs are inserted as follows: for each node i, corresponding to the final node of a path, a true or false value is generated randomly with the same probability. If the generated value is true (resp. false), then the arc which links node i to the next node in parent A (resp. parent B) is copied into the offspring, if no restriction is violated (i.e., if the selected arc does not create a subtour). If a violation occurs in any of the two cases, then the arc of the other parent is considered. The resulting tour fragments are patched using the NNH algorithm.

4.2.3. Mutation operators

Three different mutation operators are used: exchange mutation (EM) (Banzhaf (1990)), Greedy Sub Tour Mutation (GSTM) (Albayrak and Allahverdi (2011)), and 3-opt. Each operator is chosen based on a probability (P_{EM}, P_{GSTM}, and P_{3opt}) that is determined by the tuning procedure presented in Section 4.5. As for the crossover operators, each mutation operator gives a contribution in achieving the best solutions. We report a short description of each operator:

- EM consists in exchanging two nodes of the tour: if an improvement is obtained, then the exchange is applied, otherwise the original tour is not changed. The procedure is repeated $|\mathcal{N}|$ times.
- GSTM combines greedy techniques (i.e., operators that apply the move that provides the maximum local gain) to reach a local minimum quickly, and the use of different parameters to favor diversification. The method determines a partial tour by randomly selecting a starting node and an ending node in the current tour with maximum distance defined in a given interval. Then, according to a given probability, the partial tour is removed from the tour, and inserted in the cheapest way to again form a tour, or perturbation operators, such as inversion or random mixing of the partial tour, are applied. The parametrical structure of the operator prevents a stuck of the local solutions, and reaches good solutions within short computing times. We refer the reader to Albayrak and Allahverdi (2011) for further details.
- 3-opt operator consists of exchanging three arcs. In particular, we consider all the pairs of arcs, and select, for each pair, the third arc randomly in a subset of ten arcs.

4.2.4. Genetic parameters

In addition to the probability of each crossover and mutation operator, the parameters involved in GA are the population size, the maximum number of generation runs, and the probability of applying crossover or mutation. The adequate definition of the parameters is directly related to the computational performance of the evolutionary algorithms. All the GA parameters where tuned in order to achieve the best performance. Details on the tuning process are presented in Section 4.5 and the parameter values are reported in Section 5.1.1.

4.3. Refinement through the Iterated Local Search ILS

The last phase of MA consists of applying the ILS algorithm proposed in Cacchiani et al. (2018), where it was used to solve the PTSP. In that case, we considered model (2)-(15), and solved to optimality its LP-relaxation. A single initial tour was constructed as in the LP-based heuristic (Algorithm 2) but without considering randomization. Then, the ILS algorithm was applied on the constructed tour.

Here, we apply ILS on the best solution T^* found by GA. For the sake of clarity, we report, in Algorithm 4, the pseudo-code of the ILS presented in Cacchiani et al. (2018). With respect to the ILS in Cacchiani et al. (2018) the difference is the starting solution on which ILS is applied: indeed, in Cacchiani et al. (2018) a single tour was computed by an LP-based heuristic without randomization, while in this work we first generate an initial population with the initialization algorithm, then apply the GA, and execute ILS on the best tour found. ILS is made of three steps: perturbation, local search and acceptance criterion.

Perturbation (lines 5–10) is applied on the current best solution, called s^{**}, that is initialized by the tour T^*. This step consists of executing a double-bridge move with probability $Ppert$ and a scramble-subtour move otherwise, and determines a new tour s'. In both cases, a random move is applied. The double-bridge move consists of randomly removing four node-disjoint edges (A, B), (C, D), (E,F), (G, H) and reconnecting them as (A, F), (G, D), (E, B), (C, H). The scramble-subtour move corresponds to randomly choosing a path of the tour and randomly changing the order of its nodes.

Local search (lines 11–16) is applied on the local optimal solution s^*, and performs a 2-opt move

with probability $Ploc$, and an exchange improvement otherwise, thus determining a new tour s''. The 2-opt move consists of iteratively exchanging two randomly chosen arcs until the first improvement is found (or all exchanges have been tried), and the move is accepted only if an improvement is obtained. The exchange improvement requires to exchange two randomly chosen nodes of the tour until the first improvement is obtained (or $|\mathcal{N}|$ node exchanges have been tried): if an improvement is obtained, then the exchange is performed, otherwise the original tour is kept. In both cases, we select the first improving move.

Finally, the local optimal tour s^* is updated (lines 17-21) as the best tour between s' and s'', according to the fitness function ϕ that gives the value of the PTSP-PS or EMTSP objective function. If s^{**} has not been improved for $Nnoimpr$ iterations (acceptance criterion, see lines 22-24), then a 2-opt-improvement is applied on s^{**}. It consists of executing a 2-opt procedure by using as cost of each arc $(i, j) \in \mathcal{A}$ the original distance d_{ij} but each time an improvement is possible, it checks if the (PTSP-PS or EMTSP) objective function value is also improved, and accepts the change only in this case. Finally, we store in s^{**} the best solution between s^* and s^{**}. The three steps of ILS are executed for $Niter$ iterations.

All the ILS parameters where tuned in order to achieve the best performance. Details on the tuning process are presented in Section 4.5 and the parameter values are reported in Section 5.1.1.

4.4. MA pseudo-code

The pseudo-code of MA is reported in Algorithm 5. A population of individuals is generated (lines 1-4) by applying Initialization (Algorithm 3), and each individual is evaluated based on the fitness function. Subsequently, the main loop of the algorithm (iterated on the maximum number of generations), presented in lines 5–25, is responsible for generating a new population from the current one. The selection (line 8) is made in a tournament of three individuals, i.e., three individuals are randomly chosen, and the best one is selected (Eiben and Smith (2015)). Then, based on the crossover and mutation probabilities $Pcross$ and $Pmut$, a new individual is possibly generated through a selection of several crossover and mutation operators, as described in lines 11 and 16. When a new individual is obtained after applying crossover and/or mutation, a 2-opt-improvement procedure (line 19) is executed to try to improve it. Elitism is then applied (line 24), where the best parents of the old population I_{g-1} replace $Pelit$ of the worst individuals generated in the current population I_g. The individual having the minimum value of the fitness function is stored as the best tour T^* and ILS is applied (line 27) starting from this tour. Finally, the best computed tour is returned. In the next section, we present the method used for tuning all the parameters.

4.5. Parameter tuning method

Given the significant number of parameters used in MA, we adopted an effective automated method for parameter tuning, called *Iterated racing for automatic algorithm configuration* (IRACE) and proposed by López-Ibáñez et al. (2016). The IRACE is a software package that includes iterated racing procedures, such as the F-race and Iterated F-race algorithms (Birattari et al. (2002), Birattari and Kacprzyk (2009), Birattari et al. (2010)) and their extensions (e.g., a restart mechanism, the use of truncated sampling

Algorithm 4 Iterated Local Search (ILS)

$T^* \leftarrow$ best tour obtained at the end of GA
$s^*, s^{**} \leftarrow T^*$
$ni := 1$
repeat
    $r' \leftarrow rnd(0,1)$
    if $r' < Ppert$ **then**
        $s' \leftarrow$ double-bridge-move(s^{**})
    else
        $s' \leftarrow$ scramble-subtour(s^{**})
    end if
    $r'' \leftarrow rnd(0,1)$
    if $r'' < Ploc$ **then**
        $s'' \leftarrow$ 2-opt-move(s^*)
    else
        $s'' \leftarrow$ exchange-improvement(s^*)
    end if
    if $\phi(s') < \phi(s'')$ **then**
        $s^* \leftarrow s'$
    else
        $s^* \leftarrow s''$
    end if
    if check-history($\phi(s^{**})$, $Nnoimpr$) **then**
        $s^* \leftarrow$ 2-opt-improvement(s^{**})
    end if
    if $\phi(s^*) < \phi(s^{**})$ **then**
        $s^{**} \leftarrow s^*$
    end if
    $ni := ni + 1$
until $ni = Niter$

distributions, an elitist racing procedure) that lead to further improvements presented in López-Ibáñez et al. (2016). The IRACE consists of iteratively applying three phases: sampling new parameter configurations according to distributions, selecting the best ones by means of a racing procedure that discards the configurations leading to the worst results, and finally updating the sampling distributions in order to increase the probability of sampling the best parameter values.

The main advantage of IRACE over the manual tuning is the large quantity of parameter combinations that can be explored. We have selected this algorithm since it is widely used in the scientific community for parameter tuning. In addition, it is a very recent update of F-race, and the authors continuously improve the corresponding software package. We refer the reader to López-Ibáñez et al. (2016) for further details.

We applied IRACE on a subset of training instances of the PTSP-PS, and then used the same parame-

Algorithm 5 Matheuristic Algorithm (MA)

execute Initialization
for $t \leftarrow 1$ to number of individuals **do**
    compute the fitness value of individual $I_{0,t}$
end for
for $g \leftarrow 1$ **to** maximum number of generations **do**
    $t \leftarrow 1$
    repeat
        $(I_{g-1,j}, I_{g-1,k}) \leftarrow$ selection()
        $r_1 \leftarrow rnd(0,1)$
        if $r_1 < Pcross$ **then**
            $I_{g,t} \leftarrow$ crossover($I_{g-1,j}$, $I_{g-1,k}$, OX2 or DPX or HX or UNN)
        else $I_{g,t} \leftarrow I_{g-1,j}$
        end if
        $r_2 \leftarrow rnd(0,1)$
        if $r_2 < Pmut$ **then**
            $I_{g,t} \leftarrow$ mutation($I_{g,t}$, EM or GSTM or 3-opt)
        end if
        if $r_1 < Pcross$ or $r_2 < Pmut$ **then**
            $I_{g,t} \leftarrow$ 2-opt-improvement($I_{g,t}$)
            compute the fitness value of individual $I_{g,t}$
        end if
        $t \leftarrow t + 1$
    until $t >$ number of individuals
    elitism(I_g, I_{g-1})
end for
find the individual $T^* \in I_g$ with the best fitness value
execute ILS on T^*
return T^*

ters for the experiments on all the PTSP-PS and EMTSP instances. We selected 20 middle to large size instances (out of 260 instances in total) of the PTSP-PS, with a number of customers ranging between 75 and 200, since MA is especially useful for instances with a relevant number of nodes. In particular, we choose the first 5 instances with 75, 100, 150 and 200 customers. Clearly, only a subset of the instances was used for parameter tuning, to avoid overfitting: indeed, the goal of the automated parameter tuning is that the configuration found in the tuning phase generalizes to similar new instances. For the EMTSP, we decided to use the same parameter setting derived for the PTSP-PS, since the two problems are rather similar (as shown in Section 3.2).

The full list of parameters used in the tuning and the corresponding final values are reported in Section 5.1.1.

5. Computational Experiments

This section presents the results obtained by MA, and the comparison of these results with those from the literature, on the PTSP-PS instances (Section 5.1) and on the EMTSP instances (Section 5.2). The algorithms were implemented in C++, and all experiments were executed on an Intel Core i7-6900K with 16-Core 3.20GHz and 66 GB RAM (single thread). We used CPLEX 12.9.0 as LP solver.

5.1. Experiments on the PTSP-PS

We considered the sets of benchmark instances with 10, 15, 20, 25, 30, 35, 40, 45 and 50 customers used in Cacchiani et al. (2018) for the PTSP. In addition, we generated instances with 75, 100, 150 and 200 customers, by adapting the instances proposed in Demir et al. (2012) (available at http://www.apollo.management.soton.ac.uk/prplib.htm) for the PRP to the PTSP. In particular, to make the instances feasible for a single vehicle, we removed the time window constraints for every customer, and used a vehicle with capacity $D = \sum_{i \in \mathcal{N}_0} q_i$. Each set of instances contains 20 instances, leading to a total of 260 instances. In all the PTSP-PS instances, the speed ranges between 20 km/h and 90 km/h (i.e., $v^l = 20$ and $v^u = 90$, as in Demir et al. (2012)), and we consider $|\mathcal{R}| = 10$ speed levels (namely a level every 7 km/h with $(\bar{v}^r) = (23.5, 30.5, 37.5.\ 44.5, 51.5, 58.5, 65.5, 72.5, 79.5, 86.5)$. The optimal speed value for all arcs and all instances is 79.5.

We first report, in the next section, the results of the parameter tuning that we performed with IRACE (see Section 4.5) on 20 instances selected as the first 5 instances with 75, 100, 150 and 200 customers. Then, we show in Section 5.1.2 the results obtained by each phase of MA, and a comparison with the exact and the heuristic algorithms proposed in Cacchiani et al. (2018). Finally, in Section 5.1.3, we show the behavior of MA with different parameter settings.

5.1.1. Experiments for parameter tuning

In Table 1, we report the list of parameters that were tuned, the list of possible values we gave to the IRACE algorithm, and the final parameter configuration. Clearly, a larger number of possible values requires a longer computing time for the execution of IRACE. In particular, the parameter tuning with the values reported in Table 1 required about 6 days of computation, therefore we had to limit the whole set of tested values.

IRACE allows to impose rules on the relationship between different parameter values. We used these rules to ensure that the sum of the probabilities ($PRH + PNNH + PLPH$) for selecting the heuristic algorithm in the initialization procedure is 1, that the sum of the crossover operator probabilities ($P_{OX2} + P_{DPX} + P_{HX} + P_{UNN}$) is 1, and that the sum of the mutation operator probabilities ($P_{EM} + P_{GSTM} + P_{3opt}$) is 1. For these parameters, we used continuous values to easily impose the rules in IRACE: indeed, to impose restrictions on discrete parameter values, one has to explicitly list all the forbidden configurations. In addition, note that both values 0 and 1 are allowed for these probabilities, so that IRACE can decide to remove some heuristic algorithms and GA operators if they are not effective, or instead to require a specific one to be always applied.

Table 1 shows the parameter name, a short description, the set of tested values or the parameter range, and the final configuration obtained through the IRACE tuning. We selected the configuration that pro-

vided, on the 20 training instances, the best comparison (in terms of gap to the best known solutions) against the ILS algorithm by Cacchiani et al. (2018), and the corresponding values are reported in the last column.

We did not include the number of generations and the population size in the parameter tuning: we observed, through preliminary tuning experiments, that the IRACE algorithm tended to choose the highest possible value for these parameters, since larger values led to better results. However, our goal was also to have short computing times. Therefore, we fixed these parameter values by an ad-hoc tuning performed on the same 20 training instances used in IRACE: the number of generations was set to 100 and the number of individuals to 150. In addition, in Section 5.1.3, we show how the MA performance changes with different numbers of generations and population sizes.

We inserted the number of ILS iterations as one of the parameters in the tuning, since the computing time of this step was very short with respect to the total computing time of MA: indeed, ILS was applied to the best solution found by GA, which was already a very good solution, and thus the ILS step was very fast.

We set $k^1 = 10$, $k^2 = 30$ and $k^3 = 50$ for instances with up to 100 customers, and $k^1 = 20$, $k^2 = 60$ and $k^3 = 100$ for the other instances. These parameters were not tuned in IRACE, but with an ad-hoc tuning, from which we decided to select different values based on the instance size: we allowed larger values for larger size instances, but limited to k^3 the total number of subtour elimination constraints, since solving the LP-relaxation is a step that requires a significant portion of the total computing time.

Table 1
Parameter tuning.

parameter	description	tested values						final
$Prinit$	randomization in LP-heur	0.3	0.4	0.5	0.6	0.7		0.7
PRH	Random-Heuristic prob.			range(0,1)				0.466
$PNNH$	Nearest-Neighbor-Heuristic prob.			range(0,1)				0.169
$PLPH$	LP-based Heuristic prob.			range(0,1)				0.365
$Pcross$	crossover prob.	0.5	0.6	0.7	0.8	0.9	1.0	0.90
$Pmut$	mutation prob.	0.1	0.2	0.3	0.4	0.5	0.6	0.50
$Pelit$	elitism prob.	0.0	0.05	0.10	0.15	0.20		0.05
P_{OX2}	OX2 crossover operator prob.			range(0,1)				0.737
P_{DPX}	DPX crossover operator prob.			range(0,1)				0.061
P_{HX}	HX crossover operator prob.			range(0,1)				0.035
P_{UNN}	UNN crossover operator prob.			range(0,1)				0.167
P_{EM}	EM mutation operator prob.			range(0,1)				0.186
P_{GSTM}	GSTM mutation operator prob.			range(0,1)				0.206
P_{3opt}	3-opt mutation operator prob.			range(0,1)				0.608
$Niter$	ILS iter.	500	1000	3000	5000	10000		10000
$Nnoimpr$	ILS iter. no improv.	10	50	100	200	500		10
$Ppert$	ILS perturbation prob.	0.5	0.6	0.7	0.8	0.9	1.0	0.80
$Ploc$	ILS local search prob.	0.5	0.6	0.7	0.8	0.9	1.0	1.0

We observe that, in the best configuration, the LP-based heuristic probability $PLPH$ is 0.365, showing that it is useful to apply this algorithm for initializing the population. Similarly, we can see that the

probability of each (crossover or mutation) operator is always different from 0, again showing that they all contribute to obtain the best solutions.

5.1.2. Results for the PTSP

In the following we first report the results obtained after each phase of the MA algorithm i.e., after the initialization (Algorithm 3) that generates the initial population, after the GA phase and after the ILS refinement (at the end of MA). The results are displayed in Table 2. Then, we report, in Table 3, the comparison of the results obtained by MA with those found by the C&B algorithm and by the ILS reported in Cacchiani et al. (2018), on instances with up to 50 customers. However, to have a fair comparison, we initialized both algorithms proposed by Cacchiani et al. (2018) with the preprocessing procedure that computes the optimal speed for every arc. For these instances, the time limit for the C&B algorithm was set to two hours in Cacchiani et al. (2018), and we use the same time limit. In addition, we report, in Table 4, the comparison of the results obtained by MA with those obtained by ILS on instances with up to 200 customers. We do not report the results obtained by the C&B algorithm on the larger instances, since its performance decreases on these instances.

In order to compare the results, in every table, for each instance set and each instance in the set, we define as *"best known solution value"* the value of the best solution found by any of the considered methods: C&B, ILS and MA for instances with up to 50 customers, and ILS and MA for the remaining instances.

Each row of the tables corresponds to a set of 20 instances and shows average results over the 20 instances in the set. For ILS and MA, for each instance in every set, we executed 10 runs and computed the *average* and the *minimum* solution values obtained over the 10 runs. To make the tables more readable, we report only in Table 2 the average of the best solution values, while, in the other tables[1], we directly show the average percentage gaps of the values of the solutions found by each method with respect to the best known solutions values.

In Table 2, for each row, we show the number of customers ($|\mathcal{N}_0|$) in the corresponding set and the average, over the 20 instances in the set, of the best known solution values (BKS), as defined above. Then, we show the results obtained *at the end* of each phase of MA: *INIT* corresponds to the initialization phase (Algorithm 3), *INIT+GA* corresponds to lines 5–25 of Algorithm 5, and *INIT+GA+ILS* corresponds to the complete MA (Algorithm 5). For each row, and for each phase of MA, and both for the *average* and the *minimum* solution values obtained in 10 runs, we report the average (over 20 instances) of the percentage gaps (*G%*) of the solution values found by the considered phase with respect to the best known solution values, and the number of best known solutions (*#B*) found. Note that, for the value of #B, when considering the average over 10 runs, we only count the instances for which the best solution was found in *all* the 10 runs (hence this column shows integer values even for the average case). In addition, we report the average computing time (*Time*) in seconds over the 10 runs. Moreover, in the last column (*TotTime*), we display, for each row, the average (over the 10 runs of the corresponding 20 instances) of the total computing time of MA (expressed in seconds). Finally, the last but one row reports, for each column, the average value over all the instance sets, and the last row shows, for each #B column, the total number of best known solutions found out of the 260 considered instances. Note

[1]Instances and solutions values are available upon request

that in some cases, even if the average value $G\%$ is 0.00, the number of best known solutions found is smaller than 20, since we report average results with two decimal digits.

Table 2
Comparison after each phase of MA.

		INIT					INIT+GA					INIT+GA+ILS (MA)					TotTime
		average			minimum		average			minimum		average			minimum		
$\lvert\mathcal{N}_0\rvert$	BKS	G%	#B	Time	G%	#B	G%	#B	Time	G%	#B	G%	#B	Time	G%	#B	
10	150.643	0.00	17	0.01	0.00	20	0.00	20	0.03	0.00	20	0.00	20	0.00	0.00	20	0.04
15	215.687	0.04	16	0.02	0.00	20	0.00	20	0.06	0.00	20	0.00	20	0.01	0.00	20	0.09
20	288.557	0.06	15	0.03	0.00	20	0.00	20	0.12	0.00	20	0.00	20	0.02	0.00	20	0.17
25	311.451	0.16	6	0.06	0.03	17	0.00	20	0.20	0.00	20	0.00	20	0.03	0.00	20	0.29
30	417.516	0.38	2	0.11	0.10	16	0.00	20	0.31	0.00	20	0.00	20	0.04	0.00	20	0.46
35	493.212	0.50	0	0.17	0.13	11	0.00	20	0.44	0.00	20	0.00	20	0.05	0.00	20	0.66
40	563.011	1.02	0	0.24	0.29	6	0.00	20	0.63	0.00	20	0.00	20	0.07	0.00	20	0.93
45	643.863	1.49	0	0.34	0.54	6	0.01	18	0.80	0.00	20	0.00	18	0.08	0.00	20	1.22
50	719.768	2.00	0	0.47	0.87	5	0.00	19	1.08	0.00	20	0.00	19	0.10	0.00	20	1.65
75	1266.405	2.99	0	1.93	1.74	0	0.02	13	3.30	0.00	20	0.01	14	0.21	0.00	20	5.45
100	1802.333	4.56	0	4.67	3.03	0	0.12	2	7.58	0.00	20	0.10	3	0.36	0.00	20	12.62
150	3142.335	6.08	0	23.69	4.54	0	0.42	0	20.16	0.01	17	0.37	0	0.77	0.00	20	44.61
200	4449.234	7.11	0	83.21	5.47	0	0.54	0	44.81	0.05	7	0.47	0	1.39	0.00	20	129.41
Avg.		2.03	4.3	8.84	1.29	9.3	0.09	14.8	6.12	0.00	18.8	0.07	14.9	0.24	0.00	20.0	15.20
Tot.			56			121		192			244		194			260	

The results reported in Table 2 show that each phase of MA is effective in improving the solutions obtained in the previous phase. For instances with up to 30 customers, the initialization phase produces very good solutions: at least 16 best known solutions are already computed in this phase. When considering larger size instances, the contribution of the GA phase is crucial for improving the initial tours. As can be seen, for all instances with up to 100 customers, all the best known solutions are obtained by the GA phase, and for instances with 150 customers a very large number (17) of best solutions is found. On instances with 200 customers, the ILS refinement phase becomes more important: it allows to find the best solution for all the instances. It is clear that the GA phase is the most effective phase, as it obtains, on average, 18.8 best known solutions for each instance set, but the ILS refinement is also useful as this value increases to 20.

When we observe the solution values obtained on average over the 10 runs, we can see that the solution qualities after the GA phase and at the end of MA are very similar: on average, 14.8 and 14.9 best known solutions for each instance set are obtained, respectively, while, as expected, the initialization phase finds a smaller number of best known solutions (on average 4.3 for each instance set). The average percentage gaps are very small both after the GA phase and at the end of MA: even for the average values over the 10 runs, the average percentage gaps are at most 0.54% and 0.47% after GA and MA, respectively.

Finally, we can observe that the total average computing time is rather short: on average about 15 seconds. It is very short for small instances with up to 50 customers, and reaches about 130 seconds, on average, for instances with 200 customers. The two phases that require more time are the initialization and the GA phases, while the ILS phase is extremely fast. We observe that, clearly, for a given instance the solution of the LP-relaxation (Algorithm 1) does not need to be re-computed for each of the 10

runs: indeed, this step can be executed only once, and the x^1, x^2 and x^3 solutions can be stored for the remaining runs, so that the computing time required to find, for each instance, the minimum solution value over the 10 runs is smaller than 10 times the computing time required for a single run. We also mention that the preprocessing phase allows to reduce the computing time of the initialization phase, thanks to the simplified model.

In Table 3, we show the comparison of MA with C&B and ILS on the instances with up to 50 customers. The columns for ILS and MA have the same meaning as in Table 2. For the C&B algorithm, we report, for each row, the average of the final percentage gaps ($G_L\%$) between the integer solution value and the corresponding lower bound obtained, for each instance of the set, at the end of the C&B solution process, the number of best known solutions (*#B*) found, the average of the percentage gaps (*G%*) between the values of the final integer solutions and of the corresponding best known solutions, and the average computing time (*Time*) expressed in seconds. As in Table 2, in some cases the number of best known solutions found is smaller than 20 although the corresponding *G%* is 0.00, due to the approximation to two decimal digits.

Table 3
Comparison on instances with up to 50 customers.

	C&B				ILS					MA				
					average			minimum		average			minimum	
$\lvert\mathcal{N}_0\rvert$	$G_L\%$	#B	G%	Time	G%	#B	Time	G%	#B	G%	#B	Time	G%	#B
10	0.00	20	0.00	0.18	0.00	20	0.01	0.00	20	0.00	20	0.04	0.00	20
15	0.00	20	0.00	0.57	0.01	17	0.02	0.00	20	0.00	20	0.09	0.00	20
20	0.00	20	0.00	1.50	0.03	17	0.04	0.00	20	0.00	20	0.17	0.00	20
25	0.00	20	0.00	9.82	0.12	13	0.07	0.00	20	0.00	20	0.29	0.00	20
30	0.00	20	0.00	347.17	0.06	11	0.12	0.00	20	0.00	20	0.46	0.00	20
35	0.58	20	0.00	3186.13	0.12	8	0.19	0.00	20	0.00	20	0.66	0.00	20
40	1.90	18	0.02	5950.54	0.09	7	0.27	0.00	19	0.00	20	0.93	0.00	20
45	3.90	16	0.10	7200.00	0.26	6	0.38	0.03	18	0.00	18	1.22	0.00	20
50	5.28	16	0.08	7200.00	0.22	3	0.52	0.00	20	0.00	19	1.65	0.00	20
Avg.	1.30	18.9	0.02	2655.10	0.10	11.3	0.18	0.00	19.7	0.00	19.7	0.61	0.00	20.0
Tot.		170				102			177		177			180

We underline that the results reported in Table 3 for the C&B algorithm and for the ILS algorithm are obtained by executing the same algorithms published in Cacchiani et al. (2018), but to have a fair comparison we have enhanced both algorithms with the preprocessing procedure that computes the optimal speed for each arc before starting the solution process. Therefore, some values shown in Table 3 are different from those reported in Cacchiani et al. (2018).

From Table 3, we observe that the C&B algorithm proves the optimality of all the solutions for instances with up to 30 customers, as indicated by the values of $G_L\%$ equal to 0.00. For larger instances, the average percentage gap with respect to the final lower bound increases, reaching 5.28% for the instances with 50 customers. In addition, while the computing times are very short for solving instances with up to 25 customers (on average at most 9.82 seconds), they increase for larger instances, and the time limit is reached for all the instances with 45 and 50 customers. By looking at columns *#B* and *G%*, we can see that the best known solution is found by the C&B algorithm for all the instances up to 35 customers, but *#B* decreases for larger instances.

ILS obtains good results if we consider the minimum solution values achieved over the 10 runs executed for each instance: indeed, it obtains 177 best known solutions out of 180 for the instance sets with up to 50 customers. From the results obtained by ILS on the averages over 10 runs, we can observe that the number of best known solutions found is close to 20 only for the instance sets with up to 20 customers, although the average gaps are small (at most 0.26%). In addition, the computing times are extremely short.

The best performance is achieved by MA for all these instance sets: MA is capable to find the best known solution for all these instances when we consider the minimum solution values achieved over the 10 runs, and achieves very good results on the averages over the 10 runs, being able to find on average 19.7 best known solutions for each instance set (with respect to the 11.3 best known solutions obtained by ILS). Moreover, the average gap is smaller (0.00%) than that of ILS (0.10%). We can observe an increase of the computing times (on average 0.61 seconds with respect to 0.18 seconds of ILS), which however are still very short compared to those of the C&B algorithm (on average 2655.10 seconds).

Table 4
Comparison on larger instances with up to 200 customers.

	ILS					MA				
	average			minimum		average			minimum	
$\|\mathcal{N}_0\|$	G%	#B	Time	G%	#B	G%	#B	Time	G%	#B
75	0.48	1	2.05	0.05	16	0.01	14	5.45	0.00	20
100	0.74	0	4.99	0.16	7	0.10	3	12.62	0.00	20
150	1.26	0	24.04	0.57	0	0.37	0	44.61	0.00	20
200	1.60	0	82.21	0.73	0	0.47	0	129.41	0.00	20
Avg.	1.02	0.3	28.32	0.38	5.8	0.24	4.3	48.02	0.00	20
Tot.		1			23		17			80

In Table 4, we report the results obtained by the ILS and MA algorithms on larger instances with up to 200 customers. The columns have the same meaning as those of Table 3 for the two compared methods ILS and MA. As observed from the results in Table 3, the performance of the C&B algorithm worsens as the number of customers increases. Therefore, for the larger instances, we only compare the ILS and MA algorithms.

When we consider the average solution values achieved over the 10 runs executed for each instance, ILS and MA can obtain a limited number of best known solutions. However, MA obtains average solution values better than those obtained by ILS, since, on average, it obtains 4.3 best known solutions with respect to 0.3, and its average gap from the best known solutions is more than four times smaller (0.24% with respect to 1.02%).

When we consider the minimum solution values achieved over the 10 runs executed for each instance, ILS still obtains good results for instances with 75 customers, as it obtains 16 best known solutions. However, on larger size instances, the performance of ILS significantly worsens: for instances with more than 100 customers, even by considering the minimum solution value over the 10 runs, we can see that no best solution is found. On the contrary, MA obtains very good results for all the sets of instances: it finds the best known solution for all the instances with up to 200 customers. The computing times are slightly larger than those of ILS, although they are still very short. The percentage gap with respect to the best known solution value is 0.0 on average with respect to 0.38% of ILS: therefore, MA is more

stable than ILS.

It is also to note that, by considering the values of G% and #B reported in Tables 3 and 4, the results corresponding to the minimum solution values (requiring the execution of 10 runs for each instance) obtained by ILS are similar to those corresponding to the average solution values obtained by MA.

In summary, we can conclude that, on the smaller instances, MA and ILS have similar performances, although MA has a better behavior when considering the average solution values over the 10 runs instead of the minimum solution values. On larger size instances, MA is definitely better than ILS, being able to achieve all the best known solutions in very short computing times. In addition, as shown in Table 2, both the initialization and the GA phases are very effective. Therefore, MA successfully improves the results from the literature, and can be effectively adopted for solving larger size instances of the PTSP-PS.

5.1.3. MA Analysis

We performed additional experiments to analyze how the results change when the values of some parameters are modified with respect to those corresponding to the best configuration obtained with IRACE. In particular, in Table 5, we show the results obtained when the LP-based heuristic is not applied (i.e., the initialization algorithm applies, with the same probability, only the Random-Heuristic and the Nearest-Neighbor-Heuristic). In addition, we show the impact on the solution quality obtained by considering different numbers of generations (50 and 150 instead of 100) and individuals (100 and 200 instead of 150) in GA. Finally, we show the outcome when a single crossover operator is applied (operator probability set to 100%) together with all the mutation ones, or when a single mutation operator is applied together with all the crossover ones: this comparison is meant to show the contribution of each operator to the final solution quality.

Each row of Table 5 corresponds to a different parameter setting, and the first row is the best configuration obtained with IRACE. Basically, each row contains the same information as the last but one row of Tables 3 and 4. For each row, we report the comparison of the results obtained by ILS and MA, in order to have, for all the MA parameter settings, the same benchmark, and to show how robust is MA with different parameter settings in comparison with the ILS method from the literature. Clearly, the outcome of ILS is not the same in every row, since the ILS results are compared with different MA results (i.e., if MA achieves better results, ILS gets worse). Both for ILS and MA, for each of the 260 instances, we executed 10 runs and computed the average and the minimum solution values obtained over the 10 runs. In each row, we report the summary of the results (i.e., the average over the 260 tested instances) obtained in the corresponding setting.

Firstly, we observe the usefulness of the LP-based heuristic. When it is not applied, all figures worsen: the average and the minimum percentage gaps increase, and not all the best known solutions are determined. However, we can also notice a significant computing time reduction, so that we could resort to this type of setting when the computing time is limited.

By looking at the results obtained when changing the number of generations and the number of individuals, we can see that a larger number of generations allows for finding slightly better average results, at the expenses of longer computing times, while a larger number of individuals is not useful. On the contrary, a smaller number of generations or individuals causes a worsening, although the results are still much better than those of ILS. In addition, the computing time decreases.

We then compare the results obtained when a single crossover operator is used (together with all the standard mutation operators). The best results are obtained when only OX2 is applied: yet, the combina-

Table 5
Comparison with other parameter settings.

	ILS					MA				
	average		minimum			average		minimum		
Setting	G%	#B	G%	#B	Time	G%	#B	G%	#B	Time
MA best	0.384	7.9	0.118	15.4	8.8	0.074	14.9	0.000	20	15.2
MA no LP-based Heur.	0.378	7.9	0.113	15.5	8.8	0.097	14.5	0.002	19.8	4.4
MA 50 gen.	0.378	7.9	0.113	15.5	8.8	0.088	14.2	0.000	19.9	11.5
MA 150 gen.	0.387	7.9	0.121	15.4	8.8	0.070	15.0	0.000	20	19.5
MA 100 ind.	0.381	7.9	0.116	15.4	8.8	0.097	14.2	0.000	20	13.8
MA 200 ind.	0.386	7.9	0.120	15.4	8.8	0.065	14.8	0.000	19.9	17.7
MA OX2 100%	0.383	7.9	0.118	15.2	8.8	0.085	14	0.000	19.9	17.5
MA DPX 100%	0.375	7.9	0.110	15.4	8.8	0.084	13.8	0.001	19.8	15.4
MA HX 100%	0.370	7.9	0.105	15.5	8.8	0.098	13.2	0.000	19.8	15.9
MA UNN 100%	0.353	7.9	0.088	15.8	8.8	0.114	14.1	0.006	19.5	10.6
MA EM 100%	0.378	7.9	0.112	15.5	8.8	0.086	13.9	0.002	19.8	10.3
MA GSTM 100%	0.380	7.9	0.115	15.4	8.8	0.084	14.1	0.001	19.9	9.9
MA 3-opt 100%	0.386	7.9	0.120	15.4	8.8	0.073	14.6	0.000	20	18.8

tion of all operators leads to explore a larger search space and to find better solutions. We also note that the average results are more affected than the minimum ones.

Finally, we compare the results obtained when a single mutation operator is used (together with all the standard crossover operators). Also in this case, the average results are more affected than the minimum ones. Slightly shorter computing times are required when using only EM or GSTM, but the average percentage gaps and the number of best solutions found worsen with respect to the best MA configurations. Therefore, we can confirm the usefulness of the combination of all the mutation operators. We can also see that MA is robust with respect to different parameter settings.

5.2. Experiments on the EMTSP

We considered the benchmark instances proposed in Wang et al. (2020). There are two sets of small-size instances[2]: the first set (set 1) contains 30 instances with the number of nodes (including the depot) in $\{16, 19, 20, 21, 23, 26\}$ that are created by adapting 10 instances of the VRP Library and considering different curb weights $w = \alpha D$, with $\alpha \in \{0.1, 0.3, 0.5\}$, while the second set (set 2) contains 45 instances with the number of nodes (including the depot) chosen from $\{15, 16, 18, 20, 22, 24, 26, 28, 30\}$. We tested all these instances with MA, and the values of the parameters set as shown in Table 1. The results are presented in the next section.

[2]All the instances are available at https://pan.baidu.com/s/1mfMFaBhgT55VFWUR7S14-Q

5.2.1. Results for the EMTSP

We report the results for set 1 in Table 6 and those for set 2 in Table 7. In both tables, we show the comparison of MA with the methods proposed in Wang et al. (2020): the authors developed an exact branch-and-bound algorithm, three heuristic algorithms, and an approximation method. The best heuristic results were obtained by an algorithm based on the Reverse Nearest Neighbor (RNN) approach, that constructs a reverse path starting backward from the depot with full vehicle load (i.e., it determines the shortest arc for heavy loads), and by the approximation method. In both tables, for each instance, we report the instance name with the corresponding α value, the solution value obtained by the branch-and-bound algorithm ($WB\&B$) and the corresponding computing time in seconds (a time limit of 3600 seconds was imposed in Wang et al. (2020)), the solution value obtained by the best among all the heuristic and approximation algorithms ($WHBest$), and the MA results. For the latter, we report, for each instance, the average and the minimum values obtained in 10 runs, the corresponding percentage gaps from the best solution value, and the average (over 10 runs) computing time (in seconds). In the last but one row, we report, for each column, the average value over all the instances, and, in the last row, we show, for each *Val.* column, the total number of best known solutions found.

Note that, in Table 6, all the instances were solved to optimality by $WB\&B$, and, in Table 7, all the instances with up to 26 nodes were solved to optimality, while for four instances with 28 or 30 nodes the time limit was reached. The computing time of the heuristic algorithms proposed by Wang et al. (2020) was not reported. For the branch-and-bound algorithm, we report the computing time provided in Wang et al. (2020). Their algorithms were implemented in C language with the Xcode 8.3.3, and run on a MacOS with 3.2 GHz processor and 8 GB memory. Therefore, the computers used to execute $WB\&B$ and MA have similar performance.

When we consider the average values achieved by MA over the 10 runs executed for each instance, we observe that the overall average percentage gap from the optimal solution values is very small (only 0.013%), and that for only 3 (out of 30) instances the optimal solution is not found. For the best value achieved by MA over the 10 runs, we can see that the overall average percentage gap reduces to 0.001%, and that for all instances but one the optimal solution was found. Compared to the best heuristic solution values reported in column $WHBest$, it is evident that MA outperforms the heuristic algorithms proposed in Wang et al. (2020). The computing time of MA is also very short, being 0.150 seconds on average, and at most 0.235 seconds.

We observe a similar behavior for the second set of instances. When we consider the average values achieved by MA over the 10 runs executed for each instance, the overall average percentage gap (0.017%) is again very small, and for only 3 (out of 45) instances the best solution is not found. The minimum value achieved by MA over the 10 runs executed for each instance corresponds to the best solution value in all cases but one, giving on average a gap of 0.008%. The average computing time is very short (0.167 seconds). The advantage of MA with respect to the existing heuristic algorithms is evident also in this case. We can conclude that, both on the PTSP-PS and the EMTSP, MA shows a very good performance in terms of solution quality and computing time.

Table 6
Comparison on the EMTSP instances (set 1).

		WB&B		WHBest	MA				
					average		minimum		
inst.	α	Val.	Time	Val.	Val.	G%	Val.	G%	Time
P-n16	0.1	22013	0.094	24251	22013	0.000	22013	0.000	0.099
	0.3	30447	0.150	33648	30447	0.000	30447	0.000	0.094
	0.5	38483	0.120	42566	38483	0.000	38483	0.000	0.092
P-n19	0.1	29874	0.834	33925	29874	0.000	29874	0.000	0.137
	0.3	41220	0.560	46449	41220	0.000	41220	0.000	0.132
	0.5	52566	0.490	58973	52566	0.000	52566	0.000	0.133
P-n20	0.1	32670	4.310	36178	32783.6	0.348	32670	0.000	0.148
	0.3	44923	2.750	49260	44923	0.000	44923	0.000	0.152
	0.5	56889	1.970	62342	56889	0.000	56889	0.000	0.144
P-n21	0.1	31815	6.240	34831	31828	0.041	31828	0.041	0.155
	0.3	43664	4.430	47731	43664	0.000	43664	0.000	0.165
	0.5	55287	3.320	60416	55287	0.000	55287	0.000	0.155
P-n23	0.1	33854	25.680	37768	33854	0.000	33854	0.000	0.168
	0.3	47227	34.790	51523	47227	0.000	47227	0.000	0.183
	0.5	59968	26.820	65518	59968	0.000	59968	0.000	0.197
E-n16	0.1	28989	0.040	28989	28989	0.000	28989	0.000	0.099
	0.3	41660	0.070	42353	41660	0.000	41660	0.000	0.090
	0.5	53900	0.080	55460	53900	0.000	53900	0.000	0.087
E-n21	0.1	44919	9.660	54544	44924	0.011	44919	0.000	0.138
	0.3	65202	12.590	73882	65202	0.000	65202	0.000	0.146
	0.5	85134	14.940	93220	85134	0.000	85134	0.000	0.148
E-n22	0.1	3491300	26.990	3943450	3491300	0.000	3491300	0.000	0.162
	0.3	4814300	19.590	5387950	4814300	0.000	4814300	0.000	0.162
	0.5	6137300	15.170	6832450	6137300	0.000	6137300	0.000	0.162
E-n23	0.1	1728187	30.980	1795564	1728187	0.000	1728187	0.000	0.168
	0.3	2799765	41.390	2806412	2799765	0.000	2799765	0.000	0.166
	0.5	3809551	32.460	3817260	3809551	0.000	3809551	0.000	0.170
E-n26	0.1	64095	1380.580	73363	64095	0.000	64095	0.000	0.214
	0.3	89742	1653.790	102815	89742	0.000	89742	0.000	0.200
	0.5	114925	1530.920	131869	114925	0.000	114925	0.000	0.235
Avg.		799662.3	162.7	864165.3	799666.7	0.013	799662.7	0.001	0.150
Tot.		30		0	27		29		

6. Conclusions

We studied the Pollution Traveling Salesman Problem (PTSP) and the Energy Minimization Traveling Salesman Problem (TSP) that generalize the Asymmetric Traveling Salesman Problem (ATSP) by considering environmental issues through the minimization of fuel consumption and energy. We proposed a matheuristic algorithm (MA), consisting of three phases. First, it effectively uses the Linear Programming (LP) solutions of a Mixed Integer Linear Programming (MILP) model (for the PTSP or the EMSTP) to generate the initial population. Then, it executes a Multi-operator Genetic Algorithm (GA) featuring four crossover and three mutation operators, applied according different probabilities, to

Table 7
Comparison on the EMTSP instances (set 2).

		WB&B		WHBest	MA				
					average		minimum		
inst.	α	Val.	Time	Val.	Val.	G%	Val.	G%	Time
15	1	59300	0.04	65213	59300	0.000	59300	0.000	0.171
	2	74705	0.12	78025	74705	0.000	74705	0.000	0.093
	3	48455	0.02	51698	48455	0.000	48455	0.000	0.097
	4	66543	0.03	71396	66543	0.000	66543	0.000	0.089
	5	81418	0.04	87136	81418	0.000	81418	0.000	0.087
16	1	71069	0.08	76669	71069	0.000	71069	0.000	0.101
	2	77669	0.17	78291	77669	0.000	77669	0.000	0.095
	3	84557	0.26	87008	84557	0.000	84557	0.000	0.094
	4	76403	0.10	81251	76403	0.000	76403	0.000	0.094
	5	62571	0.08	69622	62571	0.000	62571	0.000	0.100
18	1	95714	0.48	97231	95714	0.000	95714	0.000	0.116
	2	76621	0.37	77495	76621	0.000	76621	0.000	0.115
	3	79886	0.29	91988	79886	0.000	79886	0.000	0.121
	4	90099	0.15	92431	90099	0.000	90099	0.000	0.119
	5	81175	0.45	83270	81175	0.000	81175	0.000	0.120
20	1	101208	1.86	101208	101208	0.000	101208	0.000	0.133
	2	122715	7.09	123696	122715	0.000	122715	0.000	0.141
	3	94658	4.88	100266	94658	0.000	94658	0.000	0.143
	4	105954	6.43	106425	105954	0.000	105954	0.000	0.135
	5	80805	4.91	88566	80805	0.000	80805	0.000	0.145
22	1	111970	5.18	113859	111970	0.000	111970	0.000	0.162
	2	90974	13.185	91841	90974	0.000	90974	0.000	0.168
	3	134593	4.68	134606	134593	0.000	134593	0.000	0.166
	4	97630	5.04	97630	97630	0.000	97630	0.000	0.154
	5	148287	14.25	150949	148287	0.000	148287	0.000	0.151
24	1	102358	39.46	102859	102358	0.000	102358	0.000	0.174
	2	114742	48.95	116068	114742	0.000	114742	0.000	0.182
	3	122687	153.72	129965	122687	0.000	122687	0.000	0.190
	4	163427	12.34	163427	163427	0.000	163427	0.000	0.178
	5	158821	14.38	158821	158821	0.000	158821	0.000	0.180
26	1	124633	238.18	136037	124633	0.000	124633	0.000	0.203
	2	142410	121.38	154296	142410	0.000	142410	0.000	0.216
	3	174879	845.33	193727	174879	0.000	174879	0.000	0.206
	4	149405	186.08	162748	149405	0.000	149405	0.000	0.199
	5	114232	595.67	122917	114240.4	0.007	114232	0.000	0.204
28	1	158892	3600	179417	158892	0.000	158892	0.000	0.243
	2	148362	3270.84	159317	148362	0.000	148362	0.000	0.229
	3	148800	1015.8	161080	148800	0.000	148800	0.000	0.222
	4	131802	1745.16	151329	131802	0.000	131802	0.000	0.232
	5	146748	879.28	153243	146748	0.000	146748	0.000	0.229
30	1	178041	3600	190655	178041	0.000	178041	0.000	0.267
	2	178880	3285.01	192602	179560	0.380	178880	0.000	0.256
	3	150708	3600	162562	151263	0.368	151263	0.368	0.266
	4	186409	3600	197104	186409	0.000	186409	0.000	0.252
	5	151510	54.04	162544	151510	0.000	151510	0.000	0.259
Avg.		114727.2	599.5	121077.5	114754.9	0.017	114739.6	0.008	0.167
Tot.		45		0	42		44		

improve the initial solutions. Finally, it employs an Iterated Local Search algorithm (ILS) proposed in the literature.

All the three phases contribute to achieve the best results, although the most relevant improvements are obtained by the GA algorithm. We observed that the LP-based heuristic used in the initialization phase was decisive to obtain some of the best solutions, and that the ILS allowed to further improve the GA solutions. Extensive computational experiments were executed to tune all the parameters and to show the contribution of each crossover and mutation operator. We observed that all the ingredients are important to achieve the best results, even though we also noticed that the algorithm was robust with respect to different parameter settings.

In addition, we showed that significant improvements were obtained over the methods from the literature. We tested the proposed algorithm on 260 instances of the PTSP and on 75 instances of the EMTSP, and compared it with exact and heuristic approaches from the literature. The results showed that MA finds the best known solution for all the instances of the PTSP and for most of the instances of the EMTSP.

A future research direction is to extend MA to deal with time-window constraints at the customers: this would imply in particular to modify the MILP model, and the GA and ILS phases, that could require the use of different operators. Another research direction is to develop exact algorithms for solving the existing instances to optimality: as it can be seen from existing works, it is still very challenging to solve these problems. Exact algorithms could also benefit from a warm-start obtained by applying MA.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-17-1-0025 and FA8655-20-1-7019. We also acknowledge project CONICYT PFCHA/DOCTORADO BECAS CHILE/2015-72160389 and VRID INICIACIÓN 220.097.016-INI, Vicerrectoría de Investigación y Desarrollo (VRID), Universidad de Concepción. We wish to thank the anonymous reviewers for their insightful comments that allowed to improve the MA performance.

References

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.

Andelmin, J., Bartolini, E., 2017. An exact algorithm for the green vehicle routing problem. *Transportation Science* 51, 4, 1288–1303.

Andelmin, J., Bartolini, E., 2019. A multi-start local search heuristic for the green vehicle routing problem based on a multigraph reformulation. *Computers & Operations Research* 109, 43–63.

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

Bektaş, T., Ehmke, J.F., Psaraftis, H.N., Puchinger, J., 2019. The role of operational research in green freight transportation. *European Journal of Operational Research* 274, 3, 807–823.

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

Birattari, M., Kacprzyk, J., 2009. *Tuning metaheuristics: a machine learning perspective*, Vol. 197. Springer, Berline/Heidelberg.

Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K., et al., 2002. A racing algorithm for configuring metaheuristics. In *Gecco*, San Francisco, CA: Morgan Kaufmann Publishers, pp. 11–18.

Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T., 2010. F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*. Springer, pp. 311–336.

Bravo, M., Rojas Pradenas, L., Parada, V., 2019. An evolutionary algorithm for the multi-objective pick-up and delivery pollution-routing problem. *International Transactions in Operational Research* 26, 1, 302–317.

Bruglieri, M., Mancini, S., Pezzella, F., Pisacane, O., 2019a. A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research* 103, 109–122.

Bruglieri, M., Mancini, S., Pisacane, O., 2019b. More efficient formulations and valid inequalities for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies* 105, 283–296.

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

Cacchiani, V., Contreras-Bolton, C., Escobar, J.W., Escobar-Falcon, L.M., Linfati, R., Toth, P., 2018. An iterated local search algorithm for the pollution traveling salesman problem. In *New Trends in Emerging Complex Real Life Problems*. Springer, pp. 83–91.

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.

Costa, L., Lust, T., Kramer, R., Subramanian, A., 2018. A two-phase pareto local search heuristic for the bi-objective pollution-routing problem. *Networks* 72, 3, 311–336.

Dabia, S., Demir, E., Van Woensel, T., 2017. An exact approach for a variant of the pollution-routing problem. *Transportation Science* 51, 2, 607–628.

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

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

Demir, E., Bektaş, T., Laporte, G., 2014. A review of recent research on green road freight transportation. *European Journal of Operational Research* 237, 3, 775–793.

Eiben, Á.E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation* 3, 2, 124–141.

Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E., 2007. Parameter control in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*. Springer, pp. 19–46.

Eiben, A.E., Smith, J.E., 2015. *Introduction to evolutionary computing*, Vol. 53. Springer, Berlin.

Elsayed, S.M., Sarker, R.A., Essam, D.L., 2011. Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Computers & Operations Research* 38, 12, 1877–1896.

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

Franceschetti, A., Demir, E., Honhon, D., Van Woensel, T., Laporte, G., Stobbe, M., 2017. A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research* 259, 3, 972–991.

Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., Laporte, G., 2013. The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological* 56, 265–293.

Freisleben, B., Merz, P., 1996. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of IEEE international conference on evolutionary computation*, IEEE, pp. 616–621.

Fukasawa, R., He, Q., Song, Y., 2016a. A branch-cut-and-price algorithm for the energy minimization vehicle routing problem. *Transportation Science* 50, 1, 23–34.

Fukasawa, R., He, Q., Song, Y., 2016b. A disjunctive convex programming approach to the pollution-routing problem. *Transportation Research Part B: Methodological* 94, 61–79.

Gaur, D.R., Mudgal, A., Singh, R.R., 2013. Routing vehicles to minimize fuel consumption. *Operations Research Letters* 41, 6, 576–580.

Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D., 1985. Genetic algorithms for the traveling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, Vol. 160, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 160–168.

Groba, C., Sartal, A., Vázquez, X.H., 2015. Solving the dynamic traveling salesman problem using a genetic algorithm with

trajectory prediction: An application to fish aggregating devices. *Computers & Operations Research* 56, 22–32.

Kara, I., Kara, B.Y., Yetis, M.K., 2007. Energy minimizing vehicle routing problem. In *International Conference on Combinatorial Optimization and Applications*, Springer, pp. 62–71.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2014. The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological* 70, 239–254.

Kramer, R., Maculan, N., Subramanian, A., Vidal, T., 2015a. A speed and departure time optimization algorithm for the pollution-routing problem. *European Journal of Operational Research* 247, 3, 782–787.

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

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

Li, K., Fialho, A., Kwong, S., Zhang, Q., 2013. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 18, 1, 114–130.

Lin, C., Choy, K.L., Ho, G.T., Chung, S.H., Lam, H., 2014. Survey of green vehicle routing problem: past and future trends. *Expert systems with applications* 41, 4, 1118–1138.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43–58.

Lourenço, H.R., Martin, O.C., Stützle, T., 2019. Iterated local search: Framework and applications. In *Handbook of metaheuristics*. Springer, Boston, MA, USA, pp. 129–168.

Macrina, G., Laporte, G., Guerriero, F., Pugliese, L.D.P., 2019. An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. *European Journal of Operational Research* 276, 3, 971–982.

Majidi, S., Hosseini-Motlagh, S.M., Ignatius, J., 2018. Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery. *Soft Computing* 22, 9, 2851–2865.

Mashwani, W.K., Salhi, A., Yeniay, O., Hussian, H., Jan, M.A., 2017. Hybrid non-dominated sorting genetic algorithm with adaptive operators selection. *Applied Soft Computing* 56, 1–18.

Moghdani, R., Salimifard, K., Demir, E., Benyettou, A., 2020. The green vehicle routing problem: a systematic literature review. *Journal of Cleaner Production* p. 123691.

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

Morán-Mirabal, L., González-Velarde, J., Resende, M., 2014. Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research* 21, 1, 41–57.

Padberg, M., Rinaldi, G., 1990. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming* 47, 1-3, 19–36.

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

Saka, O.C., Gürel, S., Van Woensel, T., 2017. Using cost change estimates in a local search heuristic for the pollution routing problem. *OR spectrum* 39, 2, 557–587.

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

Suzuki, Y., 2011. A new truck-routing approach for reducing fuel consumption and pollutants emission. *Transportation Research Part D: Transport and Environment* 16, 1, 73–77.

Syswerda, G., 1991. Scheduling optimization using genetic algorithms. *Handbook of genetic algorithms* pp. 332–349.

Tiwari, A., Chang, P.C., 2015. A block recombination approach to solve green vehicle routing problem. *International Journal of Production Economics* 164, 379–387.

Wang, S., Liu, M., Chu, F., 2020. Approximate and exact algorithms for an energy minimization traveling salesman problem. *Journal of Cleaner Production* 249, 119433.

Xiao, Y., Zhao, Q., Kaku, I., Xu, Y., 2012. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & operations research* 39, 7, 1419–1431.

Yu, Y., Wang, S., Wang, J., Huang, M., 2019. A branch-and-price algorithm for the heterogeneous fleet green vehicle routing problem with time windows. *Transportation Research Part B: Methodological* 122, 511–527.

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

Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2015. The load-dependent vehicle routing problem and its pick-up and delivery extension. *Transportation Research Part B: Methodological* 71, 158–181.

Zhang, M., Qin, J., Yu, Y., Liang, L., 2018. Traveling salesman problems with profits and stochastic customers. *International Transactions in Operational Research* 25, 4, 1297–1313.

Appendix A

A.1. Fuel consumption parameters

Table A1
Parameters used in the PTSP model.

Notation	Description	Typical Values
w	Curb-weight (kilogram)	6350
ξ	Fuel-to-air mass ratio	1
k	Engine friction factor (kilojoule/rev/liter)	0.2
N	Engine speed (rev/second)	33
V	Engine displacement (liters)	5
g	Gravitational constant (meter/second^2)	9.81
C_d	Coefficient of aerodynamic drag	0.7
ρ	Air density (kilogram/meter^3)	1.2041
A	Frontal surface area (meter^2)	3.912
C_r	Coefficient of rolling resistance	0.01
η_{tf}	Vehicle drive train efficiency	0.4
η	Efficiency parameter for diesel engines	0.9
u_d	Driver wage per (£/second)	0.002222222
κ	Heating value of a typical diesel fuel (kilojoule/gram)	44
ψ	Conversion factor (gram/second to liter/second)	737
v^l	Lower speed limit (meter/second)	5.5 (or 20 kilometer/hour)
v^u	Upper speed limit (meter/second)	25 (or 90 kilometer/hour)