Computers & Operations Research 40 (2013) 1716–1728

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

A memetic algorithm for the travelling salesperson problem with hotel selection

Marco Castro [a,*], Kenneth Sörensen [a], Pieter Vansteenwegen [b,c], Peter Goos [a,d]

[a] University of Antwerp, Belgium
[b] Ghent University, Belgium
[c] KU Leuven, Belgium
[d] Erasmus University Rotterdam, The Netherlands

ARTICLE INFO

Available online 16 January 2013

Keywords:
Memetic algorithm
TSP
Hotel selection

ABSTRACT

In this paper, a metaheuristic solution procedure for the travelling salesperson problem with hotel selection (TSPHS) is presented. The metaheuristic consists of a memetic algorithm with an embedded tabu search, using a combination of well-known and problem-specific neighbourhoods. This solution procedure clearly outperforms the only other existing metaheuristic in the literature. For smaller instances, whose optimal solution is known, it is able to consistently find the optimal solution. For the other instances, it obtains several new best known solutions.

1. Introduction

The *travelling salesperson problem with hotel selection* (TSPHS) was recently introduced by Vansteenwegen et al. [37]. The motivation for this problem is that a salesperson often cannot visit all customers in a single day, due to the fact that he/she can only work for a limited number of hours per day. This implies that the salesperson needs to select a hotel each night, on top of determining the optimal sequence in which to visit all customers. Every day should start and end in one of the available hotels and, if a given day ends in a certain hotel, the next day should start in the same hotel. The primary goal of this problem is to minimise the required number of days, while the secondary goal is to minimise the total travelled length.

Throughout the paper, the term "trip" is used to indicate a sequence of customers, starting and ending in a hotel, while the term "tour" is used for a complete sequence of connected trips that, together, visits all customers.

Although this problem appears to be very similar to the (regular) travelling salesperson problem (TSP), it is inherently more difficult due to the hotel selection requirement. The selected hotels determine to a large extent the length of the total tour.

A number of applications of the TSPHS are presented in Vansteenwegen et al. [37]: the travelling salesperson who needs several days to visit all customers; a multi-day trip for a truck driver in which every day trip should start and end on an appropriate parking space; a multi-day tourist visit to a certain region; mailmen who want to split their round into a number of connected sub-rounds in order to lighten their bag. Furthermore, the TSPHS may be used to route electric vehicles in which routes are split into trips whose maximum duration is constrained by the battery charge, and batteries can be swapped or recharged at intermediate points.

The rest of the paper is organised in the following way. In Section 2, a review of the relevant literature is presented. In Section 3, a description of the problem and a modified MIP formulation is introduced, while, in Section 4, a metaheuristic procedure for the TSPHS is outlined. In Section 5, the experiments as well as a parametric analysis are presented. Finally, Section 6 provides a conclusion and some suggestions for future research.

2. Literature review

Several problems related to the TSPHS can be found in the literature. In the multiple travelling salesperson problem (mTSP) [5], a number of salespeople, all starting and ending in the same depot, are available to visit all customers. In the vehicle routing problem (VRP) [36], the objective is to minimise the total distance travelled by a number of vehicles, each limited by a given capacity. Contrary to the TSPHS, both problems only consider one depot (or hotel). The constraint that all vehicles should start and end at a single depot is relaxed in the multi-depot vehicle routing problem (MDVRP) [9,30] where several depots are available, each with a fleet of vehicles. However, in the MDVRP, each

* Corresponding author. Tel.: +32 3 265 4061; fax: +32 3 265 4901.
E-mail addresses: marco.castro@ua.ac.be (M. Castro), kenneth.sorensen@ua.ac.be (K. Sörensen), pieter.vansteenwegen@cib.kuleuven.be (P. Vansteenwegen), peter.goos@ua.ac.be (P. Goos).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.01.006

vehicle must start and end at the same depot, while, in the TSPHS, a trip may start at one hotel and end at another hotel. In location-routing problems (LRP) [28], the depots are not fixed in advance. In the basic LRP, a number of depots has to be selected from a given set, in order to minimise the total cost of using the selected depots and routing the vehicles starting from these depots. Each vehicle must return to the depot it started from. The mTSP, VRP, MDVRP and the LRP thus each have some features in common with the TSPHS, but the most important differences are that, in the TSPHS, only one vehicle or salesperson is available and all trips need to be connected.

In the context of problems with intermediate facilities (IFs), several problems related to hotel selection arise in the literature. In the periodic vehicle routing problem with intermediate facilities (PVRP-IF) [1], a depot is fixed in advance; customers are served during a work shift whose maximum duration cannot exceed an established time limit; and the vehicle may be replenished at one of the available intermediate facilities. In the waste collection vehicle routing problem with time windows (WCVRPTW) [6,22] a depot, a set of customers and a set of waste disposal facilities are available. The empty vehicles depart from the depot, collect the waste from a set of customers and are emptied at the disposal facilities. The vehicles may visit the disposal facilities as many times as needed and, at the end of the work shift, the vehicles return to the depot emptily. In the multi-depot vehicle routing problem with inter-depot routes (MDVRPI) [11], a number of depots is available and routes are designed to serve the customers. The routes may start and end at the same depot or at different depots, while the time needed by each vehicle to traverse the set of routes assigned to it stays within a certain time limit. A variant of this problem is the vehicle routing problem with intermediate replenishment facilities (VRPIRF) [35] in which a fleet of vehicles is based at a single central depot.

Several arc routing problems involving intermediate facilities also exhibit similarities with the TSPHS. In the capacitated arc routing problem with intermediate facilities (CARPIF) [17,29], an extension of the capacitated arc routing problem (CARP) [20], a set of edges in a graph represent a road network. A demand and a travel time are associated with each edge. A subset of nodes in the graph, referred to as the intermediate facilities (IFs), represents available replenishment facilities. A fleet of vehicles with homogeneous capacity is available at a central depot. Loaded vehicles depart from the depot, traverse the edges on the graph servicing demands and may be replenished at one of the available IFs. The objective is to determine the set of vehicle routes that minimises the total travelled time. Several real-world applications of this problem are mentioned in Polacek et al. [29]. The arc routing problem with intermediate facilities under capacity and length restrictions (CLARPIF) is presented as a variant of the CARPIF in Ghiani et al. [16]. In this problem, an upper bound is imposed on the length of each route.

Just like the TSPHS, which allows each hotel to be visited on multiple occasions, the PVRP-IF, WCVRPTW, MDVRPI, VRPIRF, CARPIF and CLARPIF allow multiple visits to intermediate facilities. One difference between these problems and the TSPHS is that there is no explicit upper bound on the trip length. Instead, the trip lengths are indirectly bounded by the capacity of the vehicle(s). It is true that some of the VRPs and ARPs involve an upper bound on the travel time, but this upper bound only applies to the total travel time. This is unlike the TSPHS, where the upper bound on the travel time applies to each day trip. Another important difference between the VRPs and ARPs, on the one hand, and the TSPHS, on the other hand, is that the primary objective of the TSPHS is the minimisation of the number of trips (equivalent to the number of visits to intermediate facilities), rather than minimising the duration of the entire tour.

3. Problem description

Given a non-empty set H of hotels, and a set C of customers, the TSPHS is defined on a complete graph $G=(V,A)$ where $V=H\cup C$ and $A=\{(i,j)\,|\,i,j\in V, i\neq j\}$. Each customer $i\in C$ requires a service or visiting time τ_i (with $\tau_i=0, \forall i\in H$). The time c_{ij} needed to travel from location i to j is known for all pairs of locations. The goal is to first minimise the number of connected trips required to visit all customers, and then to minimise the total travel time of the tour. The initial and final hotel of the tour (i.e., the starting point of the first trip and the end point of the final trip) are assumed to be identical and given ($i=0$). This hotel can also be used as an intermediate hotel during the tour. Furthermore, (a) each trip must start and end in one of the available hotels, (b) the travel time of each trip must not exceed a given time budget L, and (c) a trip should start in the hotel where the previous trip ended. Since there is no limit on the number of visits to a hotel, a solution to the TSPHS is not necessarily a single cycle [37].

In this paper, an IP formulation is presented that modifies the formulation of Vansteenwegen et al. [37] in two ways: (1) a weighted objective function is used to circumvent the non-linearity problem that results from the (lexicographical) ordering of the two objectives and (2) the Miller–Tucker–Zemlin sub-tour elimination constraints were replaced by the much more efficient Dantzig–Fulkerson–Johnson constraints, allowing the solver to find more optimal solutions in a smaller computing time. In order to prioritise the minimisation of the number of trips, the number of trips is multiplied by a sufficiently large number M in the objective function so that a solution involving a smaller number of trips always has a better objective function value than any other solution involving a larger number of trips.

Let x_{ij}^d be a binary variable that takes the value 1 if, in trip d, a visit to a customer or hotel i is followed by a visit to customer or hotel j, and the value 0 otherwise. Also, let the binary variable y^d take the value 1 if, in trip d, at least one customer or hotel is visited, and the value 0 otherwise. Thus, y^d will be zero if no trip is required on day d in order to visit all customers. In the IP formulation's objective function, the variables y^d and x_{ij}^d are used for the mathematical expression of the primary and secondary objective, respectively. Finally, let D be the maximum number of trips contained in the solution.

$$\min \quad M\sum_{d=1}^{D} y^d + \sum_{d=1}^{D}\left(\sum_{(i,j)\in A} c_{ij}x_{ij}^d\right) \tag{1}$$

$$\text{s. t.} \quad \sum_{d=1}^{D}\sum_{i\in V} x_{ij}^d = 1, \quad j\in C \tag{2}$$

$$\sum_{i\in V} x_{ij}^d = \sum_{i\in V} x_{ji}^d, \quad j\in C,\ d=1,\ldots,D \tag{3}$$

$$\sum_{h\in H}\sum_{j\in V\setminus\{h\}} x_{hj}^d = y^d, \quad d=1,\ldots,D \tag{4}$$

$$\sum_{h\in H}\sum_{i\in V\setminus\{h\}} x_{ih}^d = y^d, \quad d=1,\ldots,D \tag{5}$$

$$\sum_{(i,j)\in A} (c_{ij}+\tau_j)x_{ij}^d \le L, \quad d=1,\ldots,D \tag{6}$$

$$\sum_{j\in V\setminus\{0\}} x_{0j}^1 = 1 \tag{7}$$

$$\sum_{i\in V\setminus\{0\}} x_{i0}^d \ge y^d - y^{d+1}, \quad d=1,\ldots,D-1 \tag{8}$$

$$\sum_{i \in V} x_{ih}^{d} + y^{d} \geq \sum_{i \in V} x_{hi}^{d+1} + y^{d+1}, \quad h \in H,\ d = 1, \ldots, D-1 \tag{9}$$

$$\sum_{i \in V} x_{ih}^{d} - \sum_{i \in V} x_{hi}^{d+1} \leq 1 - y^{d+1}, \quad h \in H,\ d = 1, \ldots, D-1 \tag{10}$$

$$x_{ij}^{d} \leq y^{d}, \quad (i,j) \in A,\ d = 1, \ldots, D \tag{11}$$

$$y^{d} \geq y^{d+1}, \quad d = 1, \ldots, D-1 \tag{12}$$

$$\sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{K} \setminus \{i\}} x_{ij}^{d} \leq |\mathcal{K}| - 1, \quad \mathcal{K} \subset C,\ 2 \leq |\mathcal{K}| \leq |C| - 1,\ d = 1, \ldots, D \tag{13}$$

$$x_{ij}^{d} \in \{0,1\}, \quad (i,j) \in A,\ d = 1, \ldots, D \tag{14}$$

$$y^{d} \in \{0,1\}, \quad d = 1, \ldots, D \tag{15}$$

The objective function (1) lexicographically minimises the number of trips and the total distance. Constraints (2) ensure that every customer is visited once, and constraints (3) ensure the connectivity of each trip. Constraints (4) and (5) guarantee that each trip starts and ends in one of the available hotels, while constraints (6) impose an upper bound on the length of each trip. Constraints (7) and (8) ensure that the tour starts and ends at hotel 0. Constraints (9) and (10) guarantee that, if a trip ends in a given hotel, the next trip starts at the same hotel. Constraints (11) mark a trip as being used if and only if there is at least one visit to a customer or a hotel that day, and constraints (12) ensure that the trips are performed on consecutive days, starting on day 1. Constraints (13), which involve subsets $\mathcal{K}$ of the set of customers C, are the classical subtour elimination constraints applied to each trip. Note that, because a feasible solution of the TSPHS may contain cycles starting and ending at the same hotel, the subsets $\mathcal{K}$ in the subtour elimination constraints (13) involve only customers. Finally, constraints (14) and (15) are the binary constraints for variables x_{ij}^{d} and y^{d}.

4. A memetic algorithm for the TSPHS

In this section, a metaheuristic procedure to solve the TSPHS is developed. This heuristic operates on two different decision levels: the *hotel selection* level, and the *routing* level.

As stated in Section 3, a tour starts and ends at a predefined hotel ($i=0$). A key decision in the TSPHS is the selection of intermediate hotels so as to minimise both the number of visits to intermediate hotels and the travel time. This decision is referred to as "hotel selection" and requires a sequence ($H_s = \langle h_r \rangle, h_r \in H$) of intermediate hotels to be determined. This sequence may contain any number of hotels in any order, and hotels can be repeated more than once. It turns out that the hotel selection has a major impact on the quality of the final solution. A similar observation was made in Kim et al. [22] in the context of waste disposal facilities and in Vansteenwegen et al. [37]. The hotel selection's large impact is due to the fact that it determines the orientation of the tour and the way in which the trips can be constructed. In this paper, the hotel selection decision is made using a memetic algorithm (MA) [27].

The second decision level is concerned with the actual routing of the customers. In this paper, the routing is handled by a tabu search (TS) [18,19] procedure embedded in the MA as an improvement routine. The tabu search procedure uses several neighbourhood types, some of which are well-known in the vehicle routing literature. The remaining ones are developed specifically for the TSPHS.

Memetic algorithms are evolutionary algorithms in which solutions are improved using one or more local search operators. They typically maintain a pool (called "population") of "chromosomes", i.e., representations of (partial) solutions. We denote the population by P. In our memetic algorithm, a chromosome is a sequence of intermediate hotels, representing the order in which a solution visits the hotels. The quality or "fitness" of the chromosome is given directly by the quality of the corresponding TSPHS tour generated by the tabu search procedure. The main idea behind the memetic algorithm is that the population contains a variety of hotel sequences, and, hence, at least a few high-quality building blocks for ultimately forming a good TSPHS solution. The good building blocks are refined and improved generation after generation, so that better and better TSPHS solutions can be obtained from them.

A property of our memetic algorithm is that, at any point in time, all chromosomes in the population involve the same number of hotels. This number is initially set by a solution construction procedure and decreased whenever a feasible solution with a smaller number of hotels is found. Algorithm 1 details our memetic algorithm using pseudo-code. The algorithm requires two parameters to be specified: (1) $G_{\max}$: the maximum number of generations, which defines the algorithm's stopping criterion and (2) P_s: the population size.

Algorithm 1. Memetic algorithm.

```
initialisation
generation of initial population (P)
while stopping criterion not met do
    select: p1 and p2 from P
    crossover: p1 ⊗ ps → o1,o2
    for each offspring o do
        while o is not diverse enough do
            mutate o
        end while
        improve with tabu search
        replace according to established criteria
    end for
end while
```

4.1. Memetic algorithm operators

4.1.1. Construction methods

In this section, two construction methods for generating TSPHS tours are presented. The first method constructs a tour from a sequence of customers, while the second method constructs a tour from a sequence of hotels. Both methods are used by the MA at different stages of the algorithm.

Construction method `C1`: The first construction method (henceforth called `C1`) builds a feasible TSPHS tour from a high-quality TSP tour which starts and ends at the predefined starting hotel and visits all the customers, while ignoring the time limit constraint. This *giant* TSP tour is found by means of the Lin–Kernighan heuristic [26] (`LKH`), as implemented by Applegate et al. [2]. This tour, which we denote by $\Theta = \langle s_0, s_1, \ldots, s_n \rangle$ (where s_0 is the starting and ending hotel and $s_1, \ldots, s_n$ represent the n customers), is generally infeasible for the TSPHS.

Construction method `C1` then determines the best sequence of hotels by optimally partitioning the TSP tour into feasible trips. This is done by a procedure inspired by the split method of Prins [31], which, in turn, is based on the work of Beasley [4].

The partitioning procedure constructs an auxiliary graph containing at most $mn+1$ nodes, where $m = |H|$ and $n = |C|$. The first node in the auxiliary graph is the starting hotel and has index 0. All other nodes correspond to a customer–hotel combination.

We denote by i^p a node corresponding to a stay at hotel p preceded by a visit to the i-th customer in the giant tour, s_i. The final node in the auxiliary graph is n^0, representing the arrival at the ending hotel, hotel 0, after visiting the n-th customer, s_n, in the giant tour. An arc (i^p,j^q) is added to the graph if a trip departing from hotel p, visiting customers s_{i+1} to s_j, and arriving to hotel q, is feasible. The length of the arc (i^p,j^q) is the sum of the time needed to travel from i^p to j^q via the customers s_i+1 to s_j and the visiting times,

$$c_{ps_{i+1}} + \sum_{v=i+1}^{j-1} (c_{s_v s_{v+1}} + \tau_{s_v}) + \tau_{s_j} + c_{s_j q}.$$

Our split procedure then determines a path from 0 to n^0, using the arcs (i^p,j^q). Unlike the split procedure of Prins, in which a standard minimum cost path algorithm is used, our partitioning method is adapted from Dijkstra's algorithm [12] to sequentially minimise the number of arcs and the total travelled length.

Construction method `C2`: The second construction method (henceforth called `C2`) builds a TSPHS tour from a given sequence of hotels H_s. The number of trips in the resulting TSPHS solution, which we denote by k, is automatically equal to $|H_s|+1$. It is possible that a given hotel sequence results in an infeasible TSPHS solution, due to too small a number of hotels or to a poor sequence of intermediate hotels. In case an infeasible solution is obtained, the tabu search procedure for routing the customers restores the feasibility (see Section 4.2).

The `C2` method first creates a set of k empty trips, following the order imposed by H_s. Customers are added to the tour using the sequential insertion heuristic of Christofides et al. [8]. This heuristic starts by inserting customers in the first trip one by one, until no new customers can be added any more without violating the maximum length of a day trip. The heuristic then continues with the second trip, and so on. To determine the order in which the customers are added to the trips, we used the same approach as Christofides et al. [8].

It is possible that, using the approach of Christofides et al. [8], a number of customers are not included in any of the trips. These customers are added to the tour in the position that results in the smallest violation of the maximum trip length. Finally, each trip is improved using the well-known `3-opt` operator [25]. This operator removes three edges from the current solution, and reconnects the trip in the best of the eight possible ways, provided the resulting trip is better.

4.1.2. Generation of initial population

The first step of the memetic algorithm is to generate P_s chromosomes to form the initial population. For each member j of the population, represented by the hotel sequence vector H_s^j, there is an associated tour S^j.

The first member of the population, H_s^1, is generated using construction method `C1`, which guarantees a feasible TSPHS solution. That solution often is a high-quality solution and therefore provides a good upper bound, namely $|H_s^1|+1$, for the number of trips in the optimal solution. For some instances, construction method `C1` will produce a solution involving only one trip. In that case, the optimal TSPHS solution happens to be a plain TSP tour, the memetic algorithm terminates immediately and reports the solution obtained. No further optimisation of the TSP tour is required because it has been obtained using the powerful `LKH` heuristic embedded in construction method `C1`.

In case the first member of the population, H_s^1, involves more than one trip, P_s-1 chromosomes, each consisting of $|H_s^1|$ intermediate hotels, are added to the population. Due to the fact that construction method `C1` is deterministic, it cannot be used to generate solutions other than H_s^1. Therefore, the initial population is completed using construction method `C2`. That method is applied to P_s-1 random sequences of $|H_s^1|$ hotels, which results in P_s-1 different TSPHS solutions.

Finally, every solution in the population is improved using the tabu search routine (see Section 4.2).

4.1.3. Selection and crossover

After the initial population has been formed, new generations have to be produced by mating members of the population. Mating population members are named parents. In our memetic algorithm, the selection of the parents is performed using a binary tournament method. This means that two members of the population, h_1 and h_2, are selected at random, and the best of these is chosen as the first parent p_1. Next, two new population members are selected randomly, and the best of these is chosen as the second parent p_2. The random one-point crossover operator is then applied to the hotel sequences corresponding to the two parents to produce two new hotel sequences, o_1 and o_2, which are called the offspring. The crossover operator is shown in Fig. 1, where $h_{p_j}^i$ denotes the ith hotel in the hotel sequence for parent p_j.

A key feature of our crossover operator is that only a subset of hotels is swapped between the two parents, but not the customers. In other words, the sequence of customers for offspring o_1 is taken from parent p_1, whereas the sequence of customers for offspring o_2 is copied from parent p_2. The crossover operator may lead to infeasible trips, as changing the initial and/or final hotel of a trip may cause it to be longer than the maximum allowable trip time. Any infeasibility introduced by the crossover operator is remedied in the tabu search improvement procedure.

4.1.4. Population management: diversification and mutation

In order to avoid exploring solutions with hotel sequences similar, or even identical, to the ones already contained in the population, a diversification measure is used to examine the offspring generated by the crossover operation. More specifically, our memetic algorithm uses the first population management strategy suggested by Sörensen and Sevaux [33] to ensure that the population consists of a set of substantially different chromosomes. This offers the advantage that the population size can be reduced without losing diversity. Given the computationally intensive nature of the tabu search procedure embedded in our memetic algorithm, a small-sized population is of utmost importance. The use of a diversification strategy requires a measure for the distance between two solutions.

To determine the distance between two hotel selection vectors, the *edit distance* [38] is used. In the context of this paper, the edit distance is equal to the minimum number of elementary operations required to convert one hotel sequence into another, where an elementary operation is defined as the addition of a hotel, the deletion of a hotel, or the replacement of a hotel with another hotel. Denoting the edit distance between two chromosomes H_i and H_j by $d(H_i,H_j)$, the edit distance of an offspring with

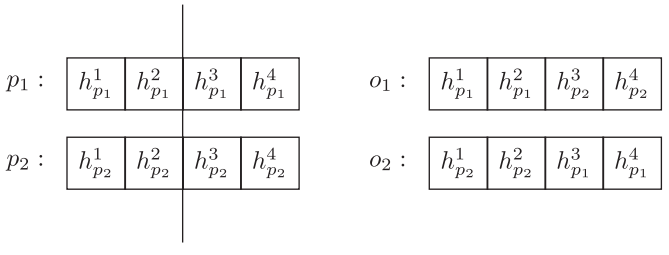

Fig. 1. One-point crossover.

hotel sequence H_o to the population P is given by

$$d_P(H_o) = \min_{i \in P} d(H_o, H_i).$$

The newly generated hotel sequence H_o is tentatively accepted if $d_P(H_o) > \Delta$, where Δ is a diversification threshold. If $d_P(H_o) \leq \Delta$, the sequence H_o is mutated until $d_P(H_o) > \Delta$. This is done by randomly replacing one of the hotels in H_o by another hotel. The threshold parameter Δ depends on the number of trips in the solution, k, and is set to $0.5k$ during the execution of the memetic algorithm.

4.1.5. Improving offspring and population update

After both offspring o_1 and o_2 have been tentatively accepted, the newly generated sequences and the corresponding tours are improved using the tabu search procedure (see Section 4.2). However, these solutions are only included in the population if they are better than the worst members of the existing population.

In order to check whether offspring o_i can be added to the population, the chromosomes are sequentially sorted in ascending order of the amount of infeasibility (the sum of all per trip violations of the time limit constraint), the number of trips, and the total tour length. Offspring o_i becomes part of the population if it is better than the current population's worst chromosome.

With this policy, it is ensured that solutions with a large amount of infeasibility are removed from the population first, as they most likely involve senseless hotel selections.

An attractive feature of this approach is that premature convergence is avoided because the population management has been used to ensure that the offspring is diverse enough.

4.1.6. Post-generation checking

After each iteration of the memetic algorithm, it is investigated whether a solution involving a smaller number of trips, say k^*, has been found. If this is the case, all members of the population involving k^*+1 or more trips are discarded, and new population members, all involving k^* trips, are generated using construction method `C2` and improved by means of the tabu search routine.

4.2. Tabu search

As was explained at the beginning of this section, the tabu search procedure handles the routing level of the heuristic. The motivation to use this metaheuristic stems from the fact that it has been very successfully used to solve other node routing problems such as the VRP and some of its variants [3,14,15,34].

Given a solution S, the tabu search described here, tries to minimise the weighted function

$$F(S) = \sum_{d=1}^{D} \sum_{(i,j) \in A} c_{ij} x_{ij}^d + \varphi \sum_{d=1}^{D} \left[\sum_{(i,j) \in A} (c_{ij} + \tau_j) x_{ij}^d - L \right]^+ \quad (16)$$

where the binary variables x_{ij}^d take the value 1 if arc (i,j) is traversed during day trip d and the value 0 otherwise, and $[z]^+ = \max(z,0)$. The weighted function considers the total travelled distance and adds a penalty to the objective function if the time limit constraint is violated. The total penalty increases with the extent to which the time limit constraint is violated. Our weighted function is inspired by the work of Gendreau et al. [14] for the CVRP.

Algorithm 2 shows the pseudo-code corresponding to our tabu search routine, which is a standard tabu search implementation. In the pseudo-code, $\hat{S}$ represents the best feasible solution found, while S^* corresponds to the solution with the best value of the weighted function $F(S)$ found so far. The neighbourhood structure $N(S)$ explored by the tabu search routine is defined by the operators `Relocate` and `Exchange` (see Section 4.2.4). The set Γ represents the list of candidate moves that may be executed at a given iteration, and the set Λ is the list of moves considered tabu. The tabu list and the candidate list form a partition of the set of all possible moves. The tabu search routine requires an initial solution S as well as the definition of five parameters:

- $i_{\max}$: the maximum number of iterations without improvement of either $\hat{S}$ or S^*;
- θ^- and θ^+: the minimum and maximum number of iterations a move is considered tabu;
- u_p: the frequency with which the penalisation factor φ is updated;
- u_c: the frequency with which each of the trips contained within the current solution are re-optimised.

Algorithm 2. Tabu search.

```
1:  procedure TABU SEARCH
2:    initialisation: time=0, Λ=∅, S*=S.
3:    if S is feasible then
4:      Ŝ=S
5:    end if
6:    while stopping criterion not met do
7:      increase iteration counter: time=time+1
8:      reset candidate list: Γ=∅
9:      for each possible move μ∈N(S) do
10:       add μ to the candidate list Γ according to tabu
          and aspiration conditions
11:     end for
12:     select best move: μ_best = arg min_{μ∈Γ} F(S⊕μ)
13:     update solution: S=S⊕μ_best
14:     update tabu list: Λ
15:     if time mod u_c = 0 then
16:       improve trips in S using 3-opt operator
17:     end if
18:     reduce number of trips using JoinTrips operator
19:     update S* and Ŝ if applicable
20:     if time mod u_p = 0 then
21:       update penalisation factor φ
22:     end if
23:   end while
24: end procedure
```

4.2.1. Local search operators

The tabu search developed in this section uses three different types of operators:

(I) the *intra-trip* `3-opt` operator to shorten the trips contained in the solution,
(II) two *inter-trip* operators, `Relocate` and `Exchange`, to reallocate one or two sets of customers, and
(III) one *trip-reduction* operator, `JoinTrips`, to join two consecutive trips.

The `Relocate` operator removes a chain of up to κ consecutive customers from one trip, and inserts it in another trip. The `Exchange` operator swaps two chains of up to κ consecutive customers from two different trips. In our tabu search procedure, the value of κ is set to 3 for both operators. A detailed description

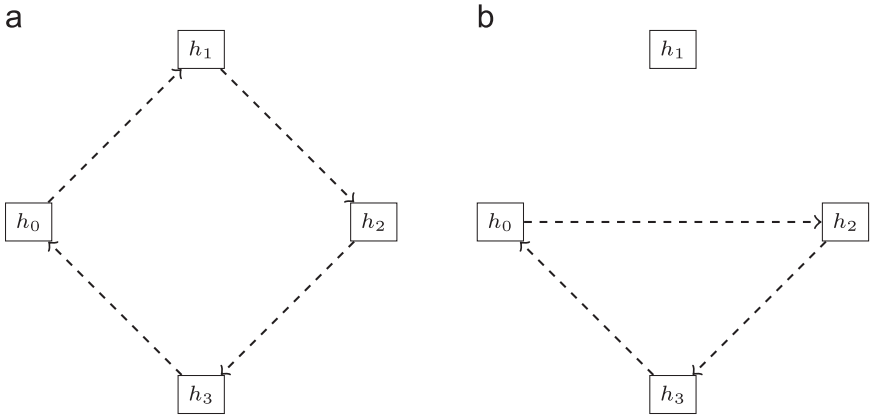

Fig. 2. Example of a `JoinTrips` move. (a) Tour before skipping h_1. (b) Tour after skipping h_1.

of the intra-trip and inter-trip operators can be found in Johnson and McGeoch [21] and Laporte et al. [23], respectively.

The `JoinTrips` operator works as follows. Given a tour involving k trips and a corresponding hotel sequence $H_s = \langle h_1, \ldots, h_{k-1} \rangle$, the operator attempts to join one or more consecutive trips, while ensuring that the resulting new solution is feasible. In the simplest case, the operator joins two consecutive trips i and $i+1$ by dropping the intermediate hotel h_i. The resulting tour then involves $k-1$ trips only and the new hotel sequence is $H_s = \langle h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_{k-1} \rangle$. Thus, when joining trips, the `JoinTrips` operator preserves the visiting order of the remaining hotels, as well as the visiting order of the customers. Fig. 2 provides a graphical illustration of the joining of two consecutive trips in a simple example involving the starting hotel h_0 and three intermediate hotels h_1, h_2 and h_3. In the figure, the squares represent the hotels, whereas the dashed arrows represent the set of customers which are visited in each trip. The original solution is shown in Fig. 2(a), while the solution obtained after joining the first and second trip by dropping hotel h_1 is shown in Fig. 2(b). The modified solution clearly involves three instead of four trips.

4.2.2. Stopping criterion

The tabu search stops if, during the past $i_{\max}$ iterations, no better solution than S^* or $\hat{S}$ has been found.

4.2.3. Tabu conditions and aspiration criteria

The tabu list Λ is a set of pairs (v,r), where v is a customer and r is a trip number, which is considered tabu. If, at any iteration, a performed move μ relocates a customer v from trip r to another trip, then the pair (v,r) is added to the tabu list Λ for the next θ iterations. The parameter θ is a random integer between two user-specified tuning parameters θ^- and θ^+.

In principle, any possible move μ involving at least one of the pairs contained in Λ is not considered in the next θ iterations. However, if the solution resulting from the forbidden move would lead to a feasible solution that is better than $\hat{S}$ in terms of the objective function (1) or to a solution that is better than S^* in terms of the weighted function (16), then the move is added to the candidate list Γ. In that case, we say that the move μ satisfies the *aspiration criterion*.

4.2.4. Exploration of the moves

The complete neighbourhood $N(S)$ structure explored by the tabu search at each iteration is the union of the neighbourhoods defined by the `Relocate` and `Exchange` operators (see Section 4.2.1). In other words, at each iteration of the tabu search procedure, both neighbourhoods are explored in the same $O(n^2)$ cycle, and the best move is identified.

4.2.5. Trip improvement and trip reduction operations

Trips contained in the current solution S are re-optimised periodically. The periodicity is controlled by the parameter u_c: every u_c iterations, the trips which have been modified by means of the `Relocate` and `Exchange` operators, are improved separately with the aid of the `3-opt` operator.

Unlike the `3-opt` operator, the trip-reduction operator `JoinTrips` is applied at each iteration. More specifically, at each iteration, tests are performed to check whether it is feasible to join two or more consecutive trips, and trips are joined whenever possible.

4.2.6. Penalisation factor update

The penalisation factor φ in the weighted function (16) is updated periodically, following the work of Gendreau et al. [14] for the CVRP. Every u_p iterations, the status of the past u_p solutions is examined. If all these solutions were feasible, then φ is reduced by a factor of 2. If the previous u_p solutions were infeasible, then φ is increased by a factor of 2. This dynamic update of the penalisation factor allows the search to switch between the space of feasible solutions and the space of infeasible solutions.

5. Computational experiments

In this section, the metaheuristic is tested on a set of existing benchmark instances. First, a brief description of the test instances is given, followed by a description of the parametric analysis performed to determine the best possible value for each parameter. Finally, the results obtained for all instances are presented. All experiments were run using an Intel Core i7-870 processor (2.93 GHz) and 4 GB of RAM.

5.1. Test instances

To test the heuristic developed in this paper, the four sets of instances developed by Vansteenwegen et al. [37] were used. All these instances were generated starting from well-known VRP and TSP benchmark instances. The instances as well as a detailed procedure for their generation are available at http://antor.ua.ac.be/tsphs.

The first set (SET 1) contains 16 instances and was created from six instances for the capacitated vehicle routing problem

with time windows (CVRPTW) from Solomon [32], involving 100 customers each, and 10 instances for the multi-depot vehicle routing problem with time windows (MDVRPTW) from Cordeau et al. [10], involving between 48 and 288 customers.

The second set of instances (SET 2) contains 52 smaller instances and was generated using 13 instances of SET 1 by using only the first 10, 15, 30 and 40 customers of the original instances. Of the 16 instances in SET 1, three instances (c201, r201 and rc201) were not included in SET 2 because their optimal solution involves a single trip. Therefore, SET 2 contains only $4 \times 13 = 52$ instances. This set of smaller instances is especially useful for comparing the performance of our heuristic with an exact method.

The third set (SET 3) consists of more complex instances involving between 51 and 1002 customers, and was generated from TSP benchmark instances in such a way that a best known solution is available. Test instances were created for 16 different TSP problems and a number of extra hotels (other than hotel 0). The visiting time was set to zero for all customers. Different subsets were generated for three different numbers of hotels (namely 3, 5 and 10).

The fourth set of instances[1] (SET 4) is obtained from the same 16 TSP instances used in SET 3, but with 10 available hotels and no best known solutions.

5.2. Parametric analysis

In this section, the results of a statistical experiment to determine the best values for the tuning parameters of the metaheuristic proposed in Section 4 are discussed. The memetic algorithm involves two tuning parameters: the population size (P_s) and the number of generations ($G_{\max}$). In addition, the embedded tabu search involves five tuning parameters, namely the maximum number of iterations without improvement ($i_{\max}$), the minimum and maximum values for the number of iterations a move is considered tabu (called θ^- and θ^+), the frequency for updating the penalisation factor φ (u_p), and the frequency for re-optimising the trips (u_c).

In order to quantify the impact of each tuning parameter on the algorithm's performance and to determine a robust parameter configuration, an experiment in which the levels of the parameters were systematically varied was conducted. In Table 1, an overview of the seven tuning parameters is provided, as well as the levels considered in the experiment. Each tuning parameter was studied at three levels.

A full factorial 3^7 experiment was conducted on each of 15 different instances, resulting in a total of $3^7 \times 15 = 32\,805$ observations. The 15 instances were obtained using a stratified sample from SET 1, SET 3 and SET 4, so as to make sure that the set of test instances involved cases with small, medium and large numbers of customers.

To quantify the impact of the seven parameters on the algorithm's performance, two type III analysis of variance (ANOVA) models were estimated, one for the number of trips and one for the tour length. The type III ANOVA models involve a random effect for each instance and fixed main effects and interaction effects of the parameters. The function of the random effect for the instances is to capture the correlation between the 3^7 test results for each instance. The assumption of random instance effects is justified because the instances can be considered a random sample from the population of all instances.

Table 1
The seven tuning parameters and the levels considered in the computational experiment.

Parameter	Levels
P_s	5, 10, 30
$G_{\max}$	5, 20, 50
$i_{\max}$	20, 50, 100
θ^-	5, 7, 10
θ^+	15, 17, 20
u_p	5, 10, 15
u_c	2, 5, 10

Table 2
Significant effects in the ANOVA models for number of trips and tour length.

Source	Prob > F
(a) *Number of trips*	
P_s	0.0138
$G_{\max}$	< 0.0001
$i_{\max}$	< 0.0001
u_p	< 0.0001
$P_s \times G_{\max}$	< 0.0001
$P_s \times u_p$	0.0236
$G_{\max} \times i_{\max}$	0.0276
$G_{\max} \times u_p$	< 0.0001
$i_{\max} \times \theta^-$	0.0158
$i_{\max} \times u_p$	0.0184
(b) *Tour length*	
u_p	< 0.0001
$P_s \times G_{\max}$	0.0187
$P_s \times \theta^+$	0.0057
$P_s \times u_p$	0.0100
$G_{\max} \times i_{\max}$	0.0411
$G_{\max} \times u_p$	0.0057
$i_{\max} \times u_p$	< 0.0001
$\theta^- \times u_c$	0.0307

Table 3
Parameter settings chosen for our MA+TS algorithm based on the parametric analysis.

Parameter	Value
P_s	5
$G_{\max}$	50
$i_{\max}$	100
θ^-	10
θ^+	20
u_p	15
u_c	5

In Table 2(a) and (b), the significant effects and interactions are shown for both performance measures. Based on these results and on the mean plots shown in Figs. A1–A4 in Appendix A, appropriate parameter settings were selected. These are shown in Table 3. The computational results presented in the next section for the four sets of instances make use of these parameter settings of.

5.3. Results

In this section, a comparison is made between the performance of the newly proposed memetic algorithm and that of the

[1] In Vansteenwegen et al. [37], an erroneous solution is given for instance lin105. This instance contains one customer which cannot be included in any trip that satisfies the time budget. As a result, no feasible solution exists for this instance. Therefore, it has been excluded from our experiments. As a result, SET 4 only contains 15 instances.

I2LS heuristic developed by Vansteenwegen et al. [37]. To this end, the CPU times for the I2LS heuristic reported in Vansteenwegen et al. [37] have been rescaled according to the procedure described in Gajpal and Abad [13]. The ratio used for the rescaling was 0.8226.

The results for SET 1 are presented in Table 4. The table's first two columns contain the name of each instance and the number of customers. Columns 3 and 4 show the number of trips and the tour length for the solution obtained by our approach (which is referred to as MA+TS in this section), while Column 5 shows the required CPU time (in seconds). Columns 6 and 7 present the best results obtained by the I2LS approach of Vansteenwegen et al. [37], and Column 8 shows the rescaled CPU time of that heuristic. Finally, Column 9, labelled "Gap (%)", presents the percentage improvement in tour length generated by the MA+TS approach vis-á-vis the I2LS approach. This column shows that the MA+TS approach was able to improve the results from the I2LS heuristic for all benchmark instances in SET 1. For two instances indicated using asterisks in the table (r101 and pr05), the MA+TS approach

Table 4
Computational results for the benchmark instances in SET 1. Asterisks indicate instances for which the MA+TS approach led to a smaller number of trips.

Instance	*N*	MA+TS			I2LS			Gap (%)
		Trips	Length	Time (s)	Trips	Length	Time (s)	
c101	100	9	9595.6	24.1	9	9685.6	0.3	0.92
r101*	100	8	1704.6	24.0	9	1801.3	0.4	5.36
rc101	100	8	1674.1	29.5	8	1724.1	0.3	2.90
c201	100	3	9560.0	16.2	3	9600.0	1.1	0.41
r201	100	2	1643.4	11.6	2	1678.0	1.5	2.06
rc201	100	2	1642.7	12.4	2	1670.0	1.3	1.63
pr01	48	2	1412.2	2.8	2	1446.0	0.2	2.33
pr02	96	3	2543.3	18.1	3	2569.3	0.9	1.01
pr03	144	4	3415.1	48.4	4	3584.1	1.3	4.71
pr04	192	5	4217.4	165.8	5	4366.3	2.3	3.41
pr05*	240	5	4958.7	331.8	6	5122.1	4.7	3.19
pr06	288	7	5963.1	327.7	7	6137.3	6.0	2.83
pr07	72	3	2070.3	13.1	3	2090.9	0.2	0.98
pr08	144	4	3372.0	64.9	4	3504.7	1.1	3.78
pr09	216	5	4420.3	228.0	5	4617.6	3.3	4.27
pr10	288	7	5940.5	409.0	7	6097.5	6.0	2.57
Avg.				108.0			1.9	2.64

Table 5
Computational results for the instances in SET 2 involving 10 customers. Values printed in bold correspond to optimal solutions.

Name	Exact		MA+TS				I2LS		
	Trips	Cost	Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Gap (%)
c101	1	**955.1**	1	**955.1**	0.0	0.00	1	955.4	0.03
r101	2	**272.8**	2	**272.8**	0.0	0.00	2	286.2	4.91
rc101	1	**237.5**	1	**237.5**	0.0	0.00	1	**237.5**	0.00
pr01	1	**426.6**	1	**426.6**	0.0	0.00	1	**426.6**	0.00
pr02	1	**661.9**	1	**661.9**	0.0	0.00	1	**661.9**	0.00
pr03	1	**553.3**	1	**553.3**	0.0	0.00	1	559.1	1.05
pr04	1	**476.4**	1	**476.4**	0.0	0.00	1	511.1	7.28
pr05	1	**528.9**	1	**528.9**	0.0	0.00	1	560.9	6.05
pr06	1	**597.4**	1	**597.4**	0.0	0.00	1	604.1	1.12
pr07	1	**670.2**	1	**670.2**	0.0	0.00	1	708.1	5.66
pr08	1	**573.4**	1	**573.4**	0.0	0.00	1	**573.4**	0.00
pr09	1	**645.5**	1	**645.5**	0.0	0.00	1	647.5	0.31
pr10	1	**461.5**	1	**461.5**	0.0	0.00	1	**461.5**	0.00
Avg.					0.0	0.00			2.03

Table 6
Computational results for the instances in SET 2 involving 15 customers. Values printed in bold correspond to optimal solutions.

Name	Exact		MA+TS				I2LS		
	Trips	Cost	Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Gap (%)
c101	2	**1452.2**	2	**1452.2**	0.0	0.00	2	1456.7	0.31
r101	2	**379.8**	2	**379.8**	0.0	0.00	2	391.5	3.08
rc101	2	**303.2**	2	**303.2**	0.0	0.00	2	306.1	0.96
pr01	1	**590.4**	1	**590.4**	0.0	0.00	1	**590.4**	0.00
pr02	1	**745.6**	1	**745.6**	0.0	0.00	1	751.1	0.74
pr03	1	**632.9**	1	**632.9**	0.0	0.00	1	649.1	2.56
pr04	1	**683.4**	1	**683.4**	0.0	0.00	1	**683.4**	0.00
pr05	1	**621.2**	1	**621.2**	0.0	0.00	1	660.4	6.31
pr06	1	**685.2**	1	**685.2**	0.0	0.00	1	**685.2**	0.00
pr07	1	**795.3**	1	**795.3**	0.0	0.00	1	812.5	2.16
pr08	1	**707.2**	1	**707.2**	0.0	0.00	1	**707.2**	0.00
pr09	1	**771.7**	1	**771.7**	0.0	0.00	1	773.6	0.25
pr10	1	**611.9**	1	**611.9**	0.0	0.00	1	**611.9**	0.00
Avg.					0.0	0.00			1.26

Table 7
Computational results for the instances in SET 2 involving 30 customers. Values printed in bold correspond to optimal solutions. The asterisk indicates an instance where the MA+TS approach suggests a solution with a smaller number of trips.

Name	Exact		MA+TS				I2LS		
	Trips	Cost	Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Gap (%)
c101[a]	3	2829.4	3	2863.2	0.0	1.19	3	2907.8	2.77
r101	3	**655.2**	3	**655.2**	0.0	0.00	3	676.2	3.20
rc101*[a]	3	610.0	3	705.5	0.1	15.65	4	747.0	22.45
pr01	1	**964.8**	1	**964.8**	0.0	0.00	1	**964.8**	0.00
pr02	2	**1078.3**	2	**1078.3**	0.0	0.00	2	1140.6	5.77
pr03	1	**952.5**	1	**952.5**	0.0	0.00	1	957.1	0.48
pr04	2	**1091.6**	2	**1091.6**	0.0	0.00	2	1149.3	5.28
pr05	1	**924.7**	1	**924.7**	0.0	0.00	1	936.3	1.25
pr06	2	**1063.2**	2	**1063.2**	0.0	0.00	2	1114.4	4.81
pr07	2	**1130.4**	2	**1130.4**	0.0	0.00	2	1158.0	2.44
pr08	2	**1006.2**	2	**1006.2**	0.0	0.00	2	1056.1	4.95
pr09	2	**1091.4**	2	**1091.4**	0.0	0.00	2	1133.1	3.82
pr10	1	**918.9**	1	**918.9**	0.0	0.00	1	927.1	0.89
Avg.					0.0	1.29			4.47

[a] Values for columns 2 and 3 correspond to the best lower bound.

Table 8
Computational results for the instances in SET 2 involving 40 customers. Values printed in bold correspond to optimal solutions.

Name	Exact		MA+TS				I2LS		
	Trips	Cost	Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Gap (%)
c101[a]	4	3817.5	4	3866.1	2.6	1.27	4	3950.0	3.47
r101[a]	4	842.9	4	862.8	2.0	2.36	4	895.5	6.24
rc101[a]	3	652.1	4	850.3	2.2	30.39	4	851.2	30.53
pr01	2	**1160.5**	2	**1160.5**	0.0	0.00	2	1223.6	5.43
pr02	2	**1336.9**	2	**1336.9**	0.0	0.00	2	1418.2	6.08
pr03	2	**1303.4**	2	**1303.4**	0.0	0.00	2	1386.9	6.40
pr04	2	**1259.5**	2	**1259.5**	0.0	0.00	2	1292.9	2.65
pr05	2	**1200.7**	2	**1200.7**	0.0	0.00	2	1200.8	0.00
pr06	2	**1242.9**	2	**1242.9**	0.1	0.00	2	1279.2	2.92
pr07	2	**1407.0**	2	**1407.0**	0.0	0.00	2	1426.5	1.38
pr08	2	**1222.2**	2	**1222.2**	0.0	0.00	2	1305.9	6.84
pr09	2	**1284.2**	2	**1284.2**	0.0	0.00	2	1287.0	0.21
pr10	2	**1200.4**	2	**1200.4**	0.0	0.00	2	1233.6	2.76
Avg.					0.53	2.61			5.76

[a] Values for columns 2 and 3 correspond to the best lower bound.

led to a reduction in the number of trips. The improvement in solution quality comes at the expense of a larger CPU time.

The results for the benchmark instances in SET 2 are presented in Tables 5–8, for 10, 15, 30 and 40 customers, respectively. To this set of instances, an exact method was applied using the TSPHS formulation presented in Section 3. The exact method used was a *straight* cutting plane method [24] implemented with the aid of the commercial solver Gurobi 4.6. The parameter D in the formulation was set equal to the number of trips in the solution produced by the MA+TS approach.

The exact method was able to find optimal solutions for 47 out of the 52 instances in SET 2. These instances are indicated using a bold font in Tables 5–8. In Vansteenwegen et al. [37], where an alternative formulation was given for the TSPHS, only 27 optimal solutions were reported for this set. Remarkably, for each of the 47 instances where an optimal solution was produced by the exact method, the MA+TS approach led to the same solution. This is unlike the I2LS approach, which produced the optimal solution in only 11 cases.

Tables 5 and 6 contain the results for the instances with 10 and 15 customers. For these numbers of customers, an optimal solution was found for each instance by the exact method and by our MA+TS approach. For instances with 10 customers, the I2LS method of Vansteenwegen et al. [37] resulted in tours that are 2.03% longer on average, with a maximum of 7.28%. For instances with 15 customers, the average gap with the optimal solution is only 1.26%, with a maximum of 6.31%, for the I2LS approach.

Tables 7 and 8 present the results for the instances with 30 and 40 customers, respectively. For these instances, it was not possible for the cutting plane method to find optimal solutions every time.

Table 7 presents the results for the instances with 30 customers. For this problem size, 11 of the 13 instances were solved to optimality by the exact method within 6 h. For instances c101 and

Table 9
Computational results for the instances in SET 3 with three extra hotels. Values printed in bold correspond to optimal solutions. Asterisks indicate instances for which the MA+TS approach led to a smaller number of trips than the I2LS approach.

Name_N	TSP	MA+TS				I2LS			
		Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Time (s)	Gap (%)
eil_51*	426	4	**426**	3.8	0.00	5	477	0.0	11.97
berlin_52*	7542	4	**7542**	3.2	0.00	5	8150	0.0	8.06
st_70*	675	4	**675**	7.0	0.00	5	754	0.0	11.70
eil_76*	538	4	**538**	27.7	0.00	5	629	0.1	16.91
pr_76	108 159	4	**108 159**	14.6	0.00	4	111 260	0.0	2.86
kroa_100*	21 282	4	**21 282**	14.2	0.00	5	22 806	0.0	7.16
kroc_100*	20 749	4	**20 749**	15.1	0.00	5	23 532	0.0	13.41
krod_100*	21 294	4	**21 294**	15.9	0.00	5	24 870	0.1	16.79
rd_100*	7910	4	**7910**	15.7	0.00	5	8869	0.1	12.12
eil_101*	629	4	**629**	16.9	0.00	5	730	0.1	16.05
lin_105*	14 379	4	**14 379**	16.4	0.00	5	16 878	0.2	17.37
ch_150*	6528	4	**6528**	35.7	0.00	5	7426	0.0	13.75
tsp_225*	3916	4	**3916**	93.4	0.00	5	4555	6.0	16.31
a_280	2579	5	2591	228.4	0.46	5	3003	9.4	16.44
pcb_442*	50 778	4	**50 778**	672.1	0.00	5	56 058	96.2	10.39
pr_1002*	259 045	4	**259 045**	3172.8	0.00	5	291 158	7250.3	12.39
Avg.				272.1	0.02			460.2	12.73

Table 10
Computational results for the instances in SET 3 with five extra hotels. Values printed in bold correspond to optimal solutions. Asterisks indicate instances for which the MA+TS approach led to a smaller number of trips than the I2LS approach.

Name_N	TSP	MA+TS				I2LS			
		Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Time (s)	Gap (%)
eil_51*	426	6	**426**	3.5	0.00	7	469	0.0	10.09
berlin_52	7542	6	**7542**	3.6	0.00	6	8200	0.0	8.72
st_70*	675	6	**675**	7.1	0.00	7	777	0.0	15.11
eil_76*	538	6	**538**	9.3	0.00	7	621	0.0	15.42
pr_76*	108 159	6	**108 159**	8.1	0.00	7	123 266	0.1	13.96
kroa_100*	21 282	6	**21 282**	14.5	0.00	7	22 644	0.0	6.39
kroc_100*	20 749	6	**20 749**	13.8	0.00	8	23 394	0.0	12.74
krod_100*	21 294	6	**21 294**	14.7	0.00	7	24 287	0.0	14.05
rd_100*	7910	6	**7910**	13.5	0.00	7	8638	0.0	9.20
eil_101*	629	6	**629**	16.3	0.00	8	727	0.1	15.58
lin_105*	14 379	6	**14 379**	15.4	0.00	7	16 082	0.2	11.84
ch_150*	6528	6	**6528**	40.2	0.00	8	7444	0.4	14.03
tsp_225*	3916	6	**3916**	86.8	0.00	8	4792	5.1	22.36
a_280*	2579	7	2646	193.1	2.59	8	3122	9.8	21.05
pcb_442*	50 778	6	**50 778**	483.2	0.00	8	58 494	97.5	15.19
pr_1002*	259 045	7	259 774	3882.4	0.28	8	298 827	7245.9	15.35
Avg.				300.3	0.18			459.9	13.82

Table 11
Computational results for the instances in SET 3 with 10 extra hotels. Values printed in bold correspond to optimal solutions. Asterisks indicate instances for which the MA+TS approach led to a smaller number of trips than the I2LS approach.

Name_*N*	TSP	MA+TS				I2LS			
		Trips	Cost	Time (s)	Gap (%)	Trips	Cost	Time (s)	Gap (%)
eil_51*	426	10	**426**	3.3	0.00	12	460	0.1	7.98
berlin_52*	7542	8	7864	3.9	–	9	8272	0.1	9.67
st_70	675	10	**675**	6.6	0.00	10	**675**	0.3	0.00
eil_76*	538	11	**538**	9.0	0.00	13	607	0.4	12.82
pr_76*	108 159	11	**108 159**	7.9	0.00	14	121 861	0.5	12.66
kroa_100*	21 282	11	**21 282**	14.1	0.00	12	23 379	0.0	9.85
kroc_100*	20 749	11	**20 749**	13.8	0.00	13	23 337	0.0	12.47
krod_100*	21 294	11	**21 294**	14.1	0.00	13	23 529	1.3	10.49
rd_100*	7910	10	**7910**	14.3	0.00	11	9158	0.9	15.77
eil_101*	629	11	**629**	15.2	0.00	12	671	0.8	6.67
lin_105*	14 379	10	**14 379**	15.3	0.00	11	16 167	1.5	12.43
ch_150*	6528	11	**6528**	35.8	0.00	12	7203	5.4	10.34
tsp_225*	3916	11	**3916**	79.9	0.00	13	4377	24.1	11.77
a_280*	2579	11	2613	134.1	1.31	14	3107	31.5	20.47
pcb_442*	50 778	11	51 774	511.4	1.96	13	56 912	511.0	12.08
pr_1002*	259 045	11	**259 045**	3224.1	0.00	13	289 477	11 831.4	11.74
Avg.				256.4	0.21			775.6	11.08

rc101, which were not solved to optimality, the table's third and fourth columns show the best lower bound reported by the solver. While the I2LS approach produced only one optimal solution, the MA+TS approach matches the exact method for the 11 cases solved to optimality. For instance rc101, the MA+TS approach leads to a solution involving one fewer day trip than the I2LS method. For instance c101, the length of the tour suggested by the MA+TS method is better than that suggested by the I2LS approach, but the number of day trips is the same for the two methods.

Like for the 30-customer cases, the exact method was unable to find optimal solutions for all instances with 40 customers within a time limit of 6 h. However, as can be seen from Table 8, an optimal solution was produced for 10 of the 13 instances. For the same 10 instances, the MA+TS heuristic method found the optimal solution as well. For instances c101, r101 and rc101, for which no optimal solution was found by the exact method, Table 8 reports the best lower bounds found by the exact method. For instances c101 and r101, the gap between the MA+TS solution and the lower bound only amounts to 1.27% and 2.36%, respectively, while, for instance rc101, the gap exceeds 30%. The large gap for the latter instance may be due to a poor lower bound. As a matter of fact, the lower bound is based on a scenario involving three day trips. So small a number of trips may not be feasible for this instance, since neither the MA+TS approach nor the I2LS approach were able to find a feasible solution involving three trips.

The results for SET 3 are shown in Tables 9–11. SET 3 was designed with the purpose of having a known optimal value for the tour length for each instance, which corresponds to its optimal TSP solution. In Column 1, the tables show the name and the size of the instance, while, in Column 2, the optimal length of the TSP solution for each instance is presented. Columns 3–6 contain the solution found by our MA+TS heuristic, the CPU time (in seconds) as well as the gap between the tour length corresponding to the MA+TS solution and the length of the optimal TSP solution. In a similar way, Columns 7–10 show the results obtained with the I2LS heuristic.

The results for the instances in SET 3 indicate that our MA+TS heuristic was able to find solutions with the optimal length for 15 of the 16 instances involving three extra hotels, for 14 of the 16 instances involving five extra hotels, and for 13 of the 16 instances involving 10 extra hotels. This is in contrast with the I2LS approach, which was able to find the optimal length for one instance with 10 extra hotels only.[2] The columns labelled "Gap (%)" in Tables 9–11 show that the MA+TS heuristic consistently outperforms the I2LS approach.

Interestingly, for most of the instances in SET 3, the best solution was found while generating the initial population. This is a result of the way in which this set of instances was generated, i.e., by splitting an optimal solution of the TSP without hotel selection in trips. Because the `C1` construction method works in a

Table 12
Computational results for the instances in SET 4. Asterisks indicate instances for which the MA+TS approach led to a smaller number of trips than the I2LS approach.

Name_*N*	MA+TS			I2LS			Gap (%)
	Trips	Cost	Time (s)	Trips	Cost	Time (s)	
eil_51	6	429	3.8	6	479	0.0	10.44
berlin_52	7	8642	4.2	7	8823	0.0	2.05
st_70*	6	723	8.5	7	745	0.0	2.95
eil_76	6	548	12.4	6	595	0.0	7.90
pr_76*	6	118 061	8.2	7	129 789	0.7	9.04
kroa_100*	6	22 343	19.2	7	22 828	0.0	2.12
kroc_100*	6	20 933	12.8	7	23 744	0.0	11.84
krod_100	6	21 664	17.3	6	24 904	0.0	13.01
rd_100	6	8244	24.2	6	8769	0.0	5.99
eil_101	6	634	22.8	6	693	0.0	8.51
ch_150*	6	6647	54.7	7	7679	0.1	13.44
tsp_225*	6	4571	118.7	7	4819	4.3	5.15
a_280*	6	2646	158.7	7	3123	7.5	15.27
pcb_442*	6	54 339	872.5	7	63 822	72.3	14.86
pr_1002	7	292 690	3423.4	7	330 282	4574.9	11.38
Avg.			317.4			310.6	8.93

[2] A special remark has to be made for instance berlin_52 in Table 11 for which no gap is reported. The reason for this is that the TSPHS solution contains one trip less than the solution with optimal length generated from the optimal TSP solution, containing nine trips. This result shows that — in rare cases — the optimal TSPHS solutions can differ from the optimal TSP solutions.

Table 13
Summary of results.

	SET 1		SET 2		SET 3		SET 4	
	MA+TS	I2LS	MA+TS	I2LS	MA+TS	I2LS	MA+TS	I2LS
# Optimal solutions	–	–	47/47	11/47	–	–	–	–
Best known solutions	16/16	0/16	–	–	42/48	1/48	15/15	0/15
Average gap (%)	0.00	2.49	0.0	2.34	0.14	12.54	0.00	8.93
Average time	108.0	1.9	0.13	–	276.3	565.2	317.4	310.6

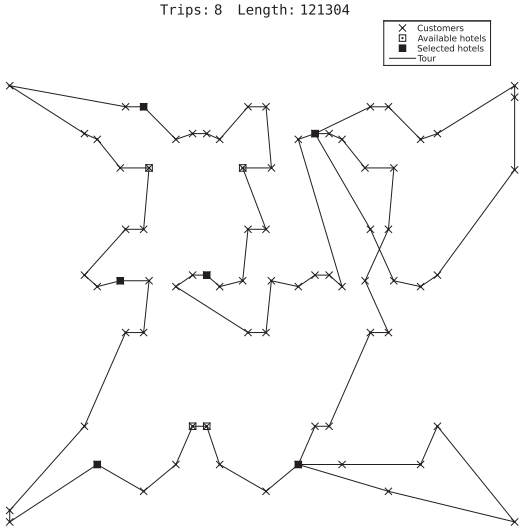

Fig. 3. C1 construction method+tabu search.

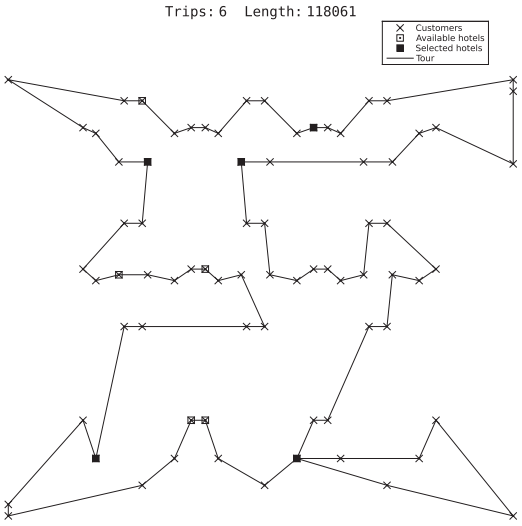

Fig. 4. MA+TS.

very similar way, it is often able to find the optimal sequence of hotels without requiring local search.

Finally, Table 12 contains the results for the benchmark instances in SET 4. This set contains the most difficult instances available. For each of these instances, our heuristic method was able to improve the best known solution. Column 8 presents the percentage improvement in length achieved by our heuristic compared to the best known solution. More importantly, for instances st_70, pr_76, kroa_100, kroc_100, ch_150, tsp_225, a_280 and pcb_442, the MA+TS approach was able to reduce the number of trips by one.

In Table 13, a summary of the results is provided. The table contains information related to the performance of both heuristic approaches (MA+TS and I2LS) for each set of instances. The table contains two columns for each set. Each time, the first column corresponds to the MA+TS heuristic, while the second column corresponds to the I2LS approach. The table's first row shows the number of optimal solutions found by each approach, for those cases where the exact method produced an optimal solution. The second row presents the number of best known solutions found by each heuristic. The third row presents, for each heuristic, the average gap to the best known solution (for sets 1, 3 and 4) or to the optimal solution (for set 2). Finally, the fourth row presents the average time, in seconds, required for each approach. It should be clear that, in terms of the first three performance measures in Table 13, our approach by far outperforms the I2LS approach.

Computing times for SET 1 illustrate the behaviour of both heuristics for most of the instances. In general, the MA+TS is considerably slower than the I2LS approach. However, for SET 3 and SET 4, the I2LS heuristic is slower on average than the MA+TS approach. The large average computing time for the I2LS approach in these sets is, however, entirely due to one instance involving more than 1000 customers, namely instance pr_1002. This is the only instance for which the MA+TS heuristic requires a smaller computing time than the I2LS approach.

6. Conclusions

The TSPHS is a difficult optimisation problem which arises in several practical situations. In this paper, a new heuristic solution method for the TSPHS is presented. The heuristic is a memetic algorithm with a tabu search embedded. The memetic algorithm solves the TSPHS by distinguishing two decision levels: the hotel selection and the routing of customers. This approach clearly outperforms the only existing heuristic in the literature in terms of solution quality. This is demonstrated by the fact that the memetic algorithm was able to find optimal solutions for all instances with a known optimal solution or a known optimal tour length. The newly developed memetic algorithm also outperforms the GRASP+VND approach described in an unpublished working paper by Castro et al. [7].

The hotel selection is a key decision when solving the TSPHS. A poor selection of intermediate hotels has a detrimental impact on the quality of the final solution. This is illustrated by Figs. 3 and 4. Fig. 3 shows a solution for instance pr_76 obtained by the C1 construction method and improved with the tabu search routine including basic hotel selection moves, but without using the memetic algorithm to change the selection of hotels. Fig. 4 shows a solution obtained by the MA+TS approach for the

same instance. In the latter solution, a different selection of hotels makes it possible to visit all customers in six instead of eight trips and using a shorter tour.

We believe that more research on the TSPHS would be useful. First, studying extensions of the problem, including more salesmen, time windows, hotel costs, and vehicle capacities as well as the design of more difficult instances would have added value. Also, it would be useful to seek alternative formulations of the TSPHS problem in order to be able to solve larger problems to optimality with a commercial solver.

Acknowledgement

The first, second and fourth authors gratefully acknowledge financial support of the Fonds voor Wetenschappelijk Onderzoek – Vlaanderen (FWO). This research has also been partially funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office.

Appendix A. Plots of the parametric analysis

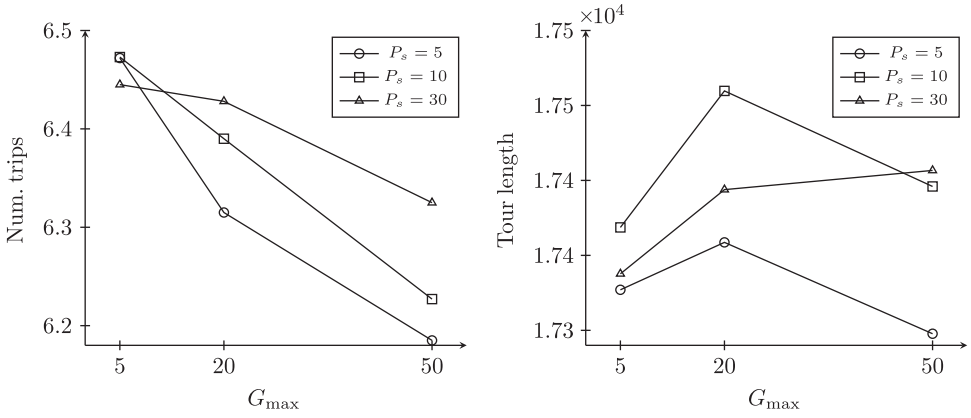

Fig. A1. Interaction $P_s \times G_{\max}$.

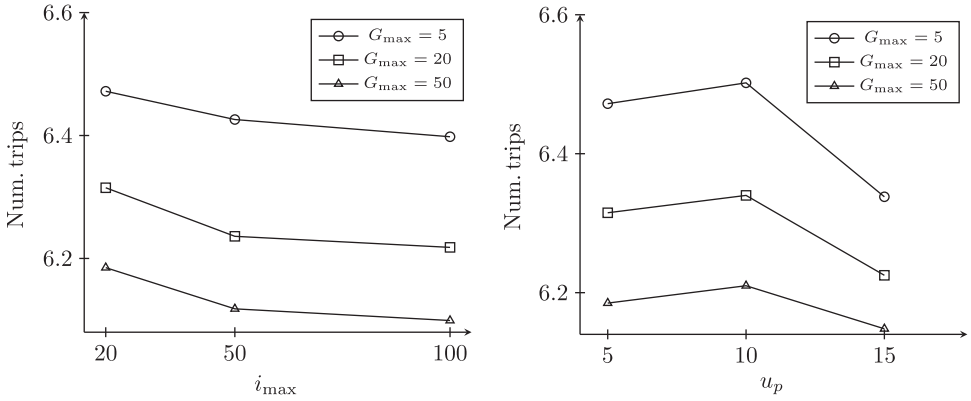

Fig. A2. Interactions $G_{\max} \times i_{\max}$ and $G_{\max} \times u_p$.

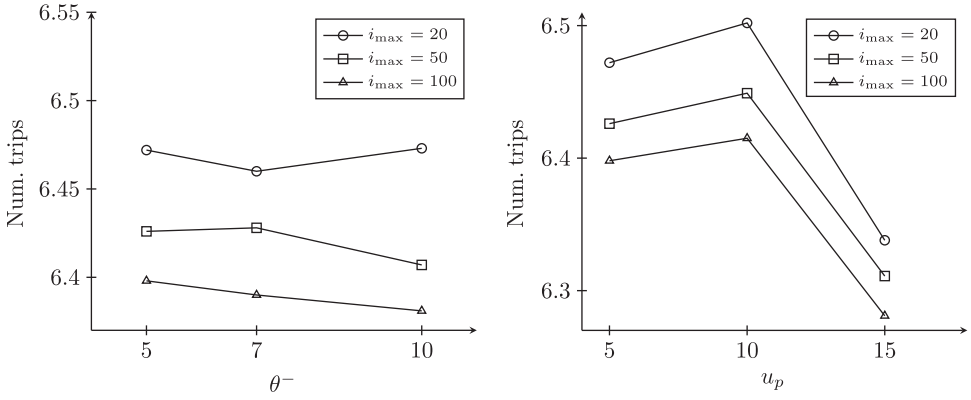

Fig. A3. Interactions $i_{\max} \times \theta^-$ and $i_{\max} \times u_p$.

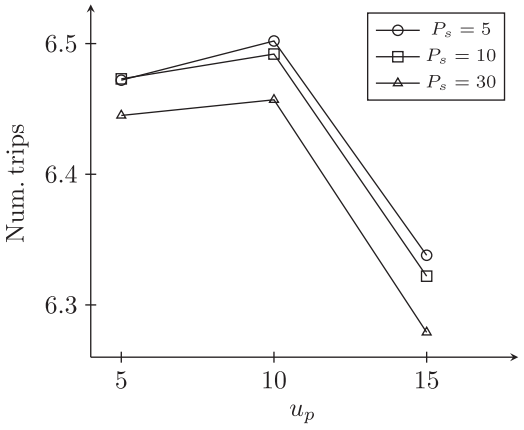

Fig. A4. Interaction $P_s \times u_p$.

References

[1] Angelelli E, Grazia Speranza M. The periodic vehicle routing problem with intermediate facilities. European Journal of Operational Research 2002;137(2):233–47.

[2] Applegate D, Bixby R, Chvátal V, Cook W. Concorde TSP solver; 2006 ⟨http://www.tsp.gatech.edu/concorde⟩.

[3] Archetti C, Speranza MG, Hertz A. A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science 2006;40(1):64–73.

[4] Beasley JE. Route first-cluster second methods for vehicle routing. Omega 1983;11(4):403–8.

[5] Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 2006;34(3):209–19.

[6] Benjamin AM, Beasley JE. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. Computers & Operations Research 2010;37(12):2270–80.

[7] Castro M, Sörensen K, Vansteenwegen P, Goos P. A simple GRASP+VND for the TSPHS. Working Paper 2012/024, Faculteit Toegepaste Economische Wetenschappen. Universiteit Antwerpen, 2012.

[8] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. Wiley; 1979. p. 315–38.

[9] Cordeau JF, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30(2):105–19.

[10] Cordeau JF, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 2001;52(8):928–36.

[11] Crevier B, Cordeau JF, Laporte G. The multi-depot vehicle routing problem with inter-depot routes. European Journal of Operational Research 2007;176(2):756–73.

[12] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Mathematik 1959;1(1):269–71.

[13] Gajpal Y, Abad PL. Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. European Journal of Operational Research 2009;196(1):102–17.

[14] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40(10):1276–90.

[15] Gendreau M, Iori M, Laporte G, Martello S. A tabu search algorithm for a routing and container loading problem. Transportation Science 2006;40(3):342–50.

[16] Ghiani G, Guerriero F, Laporte G, Musmanno R. Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. Journal of Mathematical Modelling and Algorithms 2004;3(3):209–23.

[17] Ghiani G, Improta G, Laporte G. The capacitated arc routing problem with intermediate facilities. Networks 2001;37(3):134–43.

[18] Glover F. Tabu search—Part I. ORSA Journal on Computing 1989;1(3):190–206.

[19] Glover F. Tabu search—Part II. ORSA Journal on Computing 1990;2(1):4–32.

[20] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11(3):305–15.

[21] Johnson DS, McGeoch LA. The traveling salesman problem: a case study in local optimization. In: Aarts EHL, Lenstra JK, editors. Local search in combinatorial optimization. Wiley; 1997. p. 215–310.

[22] Kim BI, Kim S, Sahoo S. Waste collection vehicle routing problem with time windows. Computers & Operations Research 2006;33(12):3624–42.

[23] Laporte G, Gendreau M, Potvin JY, Semet F. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 2000;7(4–5):285–300.

[24] Laporte G, Nobert Y. A cutting planes algorithm for the m-salesmen problem. Journal of the Operational Research Society 1980;31(11):1017–23.

[25] Lin S. Computer solutions of the traveling salesman problem. Bell System Technical Journal 1965;44(10):2245–69.

[26] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling-salesman problem. Operations Research 1973;21(2):498–516.

[27] Moscato P, Berretta R, Cotta C. Memetic algorithms. In: Cochran JJ, Cox LA, Keskinocak P, Kharoufeh JP, Smith JC, editors. Wiley encyclopedia of operations research and management science. Wiley; 2011.

[28] Nagy G, Salhi S. Location-routing: issues, models and methods. European Journal of Operational Research 2007;177(2):649–72.

[29] Polacek M, Doerner KF, Hartl R, Maniezzo V. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. Journal of Heuristics 2008;14(5):405–23.

[30] Polacek M, Hartl RF, Doerner K, Reimann M. A variable neighborhood search for the multi depot vehicle routing problem with time windows. Journal of Heuristics 2004;10(6):613–27.

[31] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 2004;31(12):1985–2002.

[32] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35(2):254–65.

[33] Sörensen K, Sevaux M. MA|PM: memetic algorithms with population management. Computers & Operations Research 2006;33(5):1214–25.

[34] Taillard É. Parallel iterative search methods for vehicle routing problems. Networks 1993;23(8):661–73.

[35] Tarantilis CD, Zachariadis EE, Kiranoudis CT. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS Journal on Computing 2008;20(1):154–68.

[36] Toth P, Vigo D., editors, The vehicle routing problem. Discrete Mathematics and Applications, vol. 9. Society for Industrial and Applied Mathematics; 2002.

[37] Vansteenwegen P, Souffriau W, Sörensen K. The travelling salesperson problem with hotel selection. Journal of the Operational Research Society 2011;63(2):207–17.

[38] Wagner R, Fischer M. The string-to-string correction problem. Journal of the ACM 1974;21(1):168–73.