PERGAMON Computers & Operations Research 26 (1999) 699–714

computers & operations research

Heuristics for the traveling salesman problem with pickup and delivery

Michel Gendreau[a], Gilbert Laporte[a,*], Daniele Vigo[b]

[a]*Centre de recherche sur les transports, Université de Montréal, Succursale Centre-Ville, C.P. 6128 Montreal, Quebec, H3C 3J7 Canada*

[b]*D.E.I.S., Università di Bologna, Italy*

Received February 1998; received in revised form July 1998

Abstract

We consider the Traveling Salesman Problem with Pickup and Delivery (TSPPD), an extension of the well-known Traveling Salesman Problem where each customer to be served is associated with two quantities of goods to be collected and delivered, respectively. A vehicle with given capacity starts at a depot and must visit each customer exactly once. The vehicle capacity must not be exceeded and the total length of the tour must be minimized. We describe new heuristics for TSPPD, the first based on the exact solution of a special case and the second based on tabu search, and we analyze their average performance through extensive computational experiments. © 1999 Elsevier Science Ltd. All rights reserved.

Scope and purpose

In several industries vehicles must routinely visit customers and make at each visit a pickup and a delivery. Clearly this must be accomplished without ever violating the vehicle capacity. A common example arises in beer or soft drinks delivery where full bottles must be delivered and empty bottles must be collected. We study this problem for the case of a single vehicle. Since the problem is extremely hard to solve to optimality, we have developed a number of heuristics and tested them on a large number of instances exhibiting different characteristics.

Keywords: Traveling salesman problem; Pickup and delivery; Heuristics; Tabu search

*Corresponding author. Tel.: 1 514 343 7575; fax: 1 514 343 7121; e-mail: gilbert@crt.umontreal.ca.

1. Introduction

The purpose of this paper is to describe new heuristics for the *Traveling Salesman Problem with Pickup and Delivery* (TSPPD), defined as follows. We are given a complete and undirected graph $G = (V_0, A)$ where $V_0 = \{0\} \cup V$ is the customer set (with $V = \{1, \dots, n\}$). Vertex 0 represents the *depot*, whereas the other vertices $i \in V$ represent the *customers*, each with an associated *delivery quantity*, $d_i \geqslant 0$, and *pickup quantity*, $p_i \geqslant 0$, (with $d_0 = p_0 = 0$). For each arc $(i, j) \in A$, $i, j \in V_0$, a non-negative arc cost, c_{ij}, is also given. A vehicle with *capacity* Q is stationed at the depot. The TSPPD consists of determining a *tour* starting and ending at the depot, serving each customer exactly once, and having minimum length, defined as the sum of the costs of the arcs composing it. The total load of the vehicle along the tour may never exceed the vehicle capacity, Q. Moreover, if a customer requires both delivery and pickup, the two operations must not be serviced separately and the delivery operation is assumed to be performed first.

To ensure the feasibility of the problem we assume that $P = \sum_{i=1}^{n} p_i \leqslant Q$ and $D = \sum_{i=1}^{n} d_i \leqslant Q$. Unless explicitly stated we will also assume that graph G is complete and that the cost matrix satisfies the *triangle inequality*

$$c_{ik} + c_{kj} \geqslant c_{ij} \quad \text{for all } i, j, k \in V_0.$$

This is not a restrictive assumption since it is well known that a strongly connected graph may be always transformed into an equivalent complete graph in which the cost of each arc is defined as the cost of shortest path between its endpoint vertices, computed on the original graph. Note that when the triangular inequality holds, the optimal TSPPD tour is a Hamiltonian tour visiting each customer once.

The TSPPD is NP-Hard in the strong sense since it generalizes the well-known NP-Hard Traveling Salesman Problem (TSP), arising when $p_i = d_i$ for all $i \in V$. The problem finds many practical applications in the design and management of distribution systems, like the transportation of underprivileged children from home to vacation locations described in Mosheiov [9].

We next introduce some additional notation and examine some basic properties of feasible TSPPD solutions. For each customer $i \in V$, define the quantity $\delta_i := p_i - d_i$ which represents the *net* (*pickup*) *demand* of customer i. Observe that δ_i may be negative, meaning that the quantity to be picked up at the associated customer is smaller than the quantity to be delivered. In the following, customers with $\delta_i < 0$ will be also called *delivery customers*, whereas those with non-negative δ_i will be called *pickup customers*. Moreover, observe that in any feasible solution the vehicle leaves the depot with a load D and returns to the depot with a load P.

Consider a subset $S \subseteq V$ of customers visited consecutively and define

$$\Delta_S = \sum_{i \in S} \delta_i,$$

which is the net additional load (possibly negative) associated with the visit of all the customers in S consecutively in any order. Let also L denote the total load on the vehicle before visiting S in a feasible solution. If the order in which the customers in S are visited is not specified the following properties hold.

Property 1. *In any feasible solution we must have* $0 \leqslant L \leqslant Q - \max\{0, \Delta_S\}$.

Proof. The vehicle leaves the depot with a load D, hence along any circuit we have $L \geqslant 0$. If $\Delta_S > 0$ then there must be on the vehicle enough empty space to accommodate the additional net load associated with S, since otherwise, independently from the order in which S is visited, the vehicle will end the visit of the customers in S with a total load exceeding Q. If $\Delta_S \leqslant 0$, then any $L \leqslant Q$ may lead to a feasible solution since the vehicle will end the visit of S possibly with additional empty space on it. □

Property 2. *Given* $S \subseteq V$ *and* L *such that Property* 1 *holds, there always exists a path visiting all customers in* S *which is feasible with respect to the capacity constraint.*

Proof. This is easily seen by considering a path which visits all customers with negative δ_i before visiting those with non-negative δ_i. This path is feasible since $L \leqslant Q - \max\{0, \Delta_S\}$ and since we assumed that $\sum_{i \in S} p_i \leqslant Q$ and $\sum_{i \in S} d_i \leqslant Q$. □

Let us now assume that customers in S are visited in the order $\{i_1, i_2, \ldots, i_{|S|}\}$. Then the following property trivially holds.

Property 3. *In any feasible solution we must have*

$$L \leqslant Q - \max\left\{0, \max\left\{\sum_{j=1}^{h} \delta_{i_j}, h = 1, \ldots, |S|\right\}\right\}.$$

The *worst-case performance ratio* of a heuristic is the smallest value ρ such that, for every instance of an optimization problem, the ratio between the value of the solution found by the heuristic and the optimal solution value is not greater than ρ.

Mosheiov [9] proposed a mathematical model for TSPPD and heuristic algorithms based on the extension of methods for the standard TSP. For one of the proposed algorithms a worst-case performance ratio equal to $1 + \alpha$ was also proved, where α is the worst-case performance ratio of the TSP heuristic used. Anily and Mosheiov [1] described a new heuristic with worst-case performance ratio equal to 2 based on the solution of Shortest Spanning Trees.

The special case of TSPPD, known as Traveling Salesman Problem with Backhauls (TSPB), where each customer is either a pure pickup point or a pure delivery point and where in any feasible solution all delivery points must precede the pickup points has been studied by Gendreau, Hertz and Laporte [4] who presented the extension to TSPB of the GENIUS heuristic for the TSP. Finally, the generalization of TSPPD related to the Vehicle Routing Problem (VRP), where several vehicles are available, has been considered by Halse [8] who proposed a mathematical formulation and a heuristic algorithm based on a Lagrangian relaxation of the problem.

In this paper we present new approximation algorithms for TSPPD. The first is based on the optimal solution of the special case of TSPPD arising when the graph G is a cycle. In particular, we derive a linear time algorithm for the optimal solution of this special case and use it as a base for a heuristic for the general TSPPD. The worst-case performance of the proposed algorithm is studied, and a tight ratio of 3 is derived. However, the results of an extensive computational testing show that this heuristic generally produces a better solution with respect to those previously proposed in the literature. A further improvement is obtained by means of a second heuristic based on the tabu search approach (see [5–7]). The algorithm uses a neighborhood based on exchanges of two arcs and, within a few minutes of computing time, is able to considerably improve the solutions obtained by the previous heuristic.

The remainder of this paper is organized as follows. In Section 1, we briefly review the algorithms for TSPPD proposed by Mosheiov [9] and by Anily and Mosheiov [1]. In Section 2, we describe the linear time algorithm for the TSPPD on a cycle and the resulting heuristic for general TSPPD. The tabu search algorithm is described in Section 4 and the results of the computational testing are analyzed in Section 5.

2. Previous algorithms for TSPPD

The first heuristic solution approaches for TSPPD are due to Mosheiov [9], who proposed and analyzed two TSP-based methods.

Given any TSPPD instance, denote by $v^*(I)$ and $v^*_{\mathrm{TSP}}(I)$ the optimal solution value of the problem and of the associated TSP instance obtained by ignoring the capacity requirement (i.e., by assuming $Q = \infty$), respectively.

Mosheiov first proved that $v^*_{\mathrm{TSP}}(I) \leqslant v^*(I) \leqslant 2v^*_{\mathrm{TSP}}(I)$ for all I and observed that given any tour $T = \{i_1, i_2, \ldots, i_n, i_1\}$ through all the pickup and delivery points, there exists a *starting customer*, i_s, such that the tour $\{0, i_s, i_{s+1}, \ldots, i_n, i_1, \ldots, i_{s-1}, 0\}$ is a feasible TSPPD solution. The starting point i_s may be easily determined in linear time as the customer following the point of maximal capacity violation along the tour T, computed when starting at an arbitrary i_1. The first heuristic directly uses this observation and works as follows: (i) a solution of the standard TSP over all the customers in V is determined, (ii) a starting point i_s for the feasible TSPPD tour is identified, (iii) arc (i_{s-1}, i_s) is removed and is replaced by the arcs $(0, i_s)$ and $(i_{s-1}, 0)$. If in Step (i) the optimal TSP tour is determined, then the resulting algorithm, called PDOT, has a worst-case performance ratio equal to 2. If instead the TSP solution is determined by a heuristic algorithm with worst-case performance ratio equal to α, then the resulting TSPPD algorithm, called PDαT, has a worst-case performance equal to $1 + \alpha$. In the experimental testing of this algorithm Mosheiov used the well-known *cheapest insertion* TSP heuristic for which $\alpha = 2$ (see Rosenkrantz et al. [10]).

The second heuristic proposed by Mosheiov [9] is based on the extension of the cheapest insertion heuristic to TSPPD. The heuristic, called *cheapest feasible insertion* (CFI), first builds a TSP tour over all delivery customers and the depot by means of a cheapest insertion heuristic. The pickup customers are then iteratively inserted into the tour in the best insertion point feasible with respect to the capacity constraint. Although algorithm CFI has no worst-case performance guarantee, it experimentally proved to produce solutions which are often better than those of PDαT.

Anily and Mosheiov [1] proposed a new heuristic for TSPPD based on the computation of shortest spanning trees. They first described a linear time exact algorithm for the solution of the special case of TSPPD arising when the graph is a tree rooted at the depot. This problem is solved by traversing the tree in a depth-first fashion and serving the customers with non-positive δ_i as soon as they are visited, whereas those with positive δ_i are served only when all the customers in the subtree rooted at them are served. In the depth-first traversal the subtrees with non-positive associated total demand are visited before those with positive demand. Note that each edge of the tree is traversed twice. A heuristic for the general TSPPD, hereafter called TREE, is then obtained by determining a shortest spanning tree over the complete graph and determining the visiting order of the customers with the previously described algorithm for TSPPD on a tree. It is easily seen that this visiting order defines a feasible way of applying shortcuts to the tree solution in order to obtain the feasible Hamiltonian solution, and that the worst-case performance of this heuristic is equal to 2. Even if TREE has an appealing worst-case behavior, Anily and Mosheiov experimentally proved that on average it has a worse performance than the algorithm CFI of Mosheiov [9], although it generally requires less time than CFI.

3. A new heuristic for TSPPD

In this section we examine a new heuristic for TSPPD based on the optimal solution of the special case arising when the graph G is a cycle. For this problem, hereafter denoted by $\text{TSPPD}_{\text{cycle}}$, we develop a linear time exact algorithm, which will be later used as a basis for the heuristic algorithms for general TSPPD.

Note that when considering this special case, graph G is no longer complete. Hence a feasible solution to this problem is generally a non-Hamiltonian path which may visit the customers more than once. However, since each customer will be *served* only once, a solution may be defined by the order in which the customers are served.

3.1. The TSPPD on a cycle

Assume that the arc set A of graph G defines a cycle spanning vertex set V_0. Without loss of generality, customers can be renumbered according to the order in which they appear on the cycle. Let $\bar{c}_i = c_{i,i+1}$ for $i = 0, \dots, n-1$, and $\bar{c}_n = c_{n,0}$.

It is obvious that

$$v^*(\text{TSPPD}_{\text{cycle}}) \geqslant C_{\text{cycle}}, \tag{1}$$

where $C_{\text{cycle}} = \sum_{i=0}^{n} \bar{c}_i$ is the total length of the cycle. However, from Property 2 we also have that

$$v^*(\text{TSPPD}_{\text{cycle}}) \leqslant 2C_{\text{cycle}}. \tag{2}$$

In fact, a feasible solution to the problem with cost $2C_{\text{cycle}}$ may be obtained by traversing the cycle twice. The first time only customers with non-positive δ_i are served, while the second time all remaining customers are served. Moreover, it is easy to see that this bound is tight. Consider indeed an instance with n customers, $Q = n$, with a customer, say h, with $p_h = 0$ and $d_h = n$ located on the cycle at a position opposite the depot, and with $n-1$ customers with $p_i = 1$ and $d_i = 0$, located along the cycle so that $\bar{c}_i = 1$, $i = 0, \dots, n$. Here we have $C_{\text{cycle}} = n + 1$. The optimal solution has cost $2n - 2$, and is obtained by first performing a complete traversal of the cycle serving customer h and all the customers $h + 1, h + 2, \dots$ from it to the depot, then serving the remaining customers, $1, \dots, h - 1$, and returning to the depot. For suitably large n the optimal solution value is arbitrarily close to $2C_{\text{cycle}}$.

Hereafter we derive a linear time algorithm for the optimal solution of $\text{TSPPD}_{\text{cycle}}$, obtained as the best of three feasible solutions. In fact the following property holds for the feasible solutions of $\text{TSPPD}_{\text{cycle}}$.

Lemma 1. *Given an instance of $TSPPD_{\text{cycle}}$, in any feasible solution only one of the following two cases holds*:

(a) *one edge is traversed zero times and all the remaining are traversed exactly twice*;
(b) *all edges are traversed either one or three times. In this case the edge $(i, i + 1)$ is traversed three times if $D + \sum_{j \leqslant i} \delta_j > Q$.*

Proof. First observe that if more than one edge is not traversed, the solution is disconnected and hence infeasible. In case (a) one of the edges of A is dropped, hence the cycle is transformed into a line. Then any feasible solution must traverse each of the remaining edges twice in order to serve all customers. In case (b) no edge is dropped, hence all of them must be traversed at least once, visiting the entire cycle either clockwise or counterclockwise. Suppose that the cycle is visited clockwise. We then observe that if $D + \sum_{j \leqslant i} \delta_j > Q$, customer i may not be served immediately by the vehicle (i.e., at its first passage) which must proceed and return to serve i when sufficient space becomes available. This means that in this case edge $(i, i + 1)$ is traversed exactly three times. A similar consideration applies when the cycle is visited counterclockwise. $\square$

From this result it follows that the optimal solution to TSPPD on a cycle can be determined as the best one between those satisfying either case (a or b). In case (a) it is obvious that the minimum cost solution is obtained by determining the edge $(i^*, i^* + 1)$ with the largest cost $\bar{c}_{i^*}$, removing it and then traversing twice each remaining arc. This solution has cost $2C_{\text{cycle}} - \bar{c}_{i^*}$ and may be determined in $O(n)$ time.

Now consider case (b) and assume that the cycle is visited clockwise. Let v_i denote the minimum number of times that edge $(i, i + 1)$ is traversed in any feasible solution. We know from Lemma 1 that if $D + \sum_{j \leqslant i} \delta_j > Q$, then $v_i = 3$, and $v_i = 1$ otherwise. The following algorithm determines a feasible solution which traverses each edge $(i, i + 1)$ exactly v_i times, and is therefore optimal. For each customer i, let σ_i be the index of the first customer following i such that $v_{\sigma_i} = 1$ (observe that $\sigma_i \leqslant n$ for all i).

```
algorithm C
{output: $ord_i$ = index of the customer visited in $i$-th position}
  begin
0. for each $i \leqslant n$ do compute $v_i$;
   $pos := 1$; $i := 1$;
   repeat
1.   while ($v_i = 1$ and $i \leqslant n$) do
       begin
         $ord(pos) := i$; $pos := pos + 1$; $i := i + 1$
       end;
2.   $start := i$;
     while ($v_i = 3$ and $i < n$) do
       begin
         if ($\delta_i \leqslant 0$) then $ord(pos) := i$; $pos := pos + 1$;
         $i := i + 1$
       end;
     $ord(pos) := i$; $pos := pos + 1$;
3.   $i := start$
       while ($v_i = 3$ and $i \leqslant n$) do
         begin
           if ($\delta_i > 0$) then $ord(pos) := i$; $pos := pos + 1$;
           $i := i + 1$
         end;
       $i := i + 1$
     until $i > n$
   end.
```

At Step 0 vectors v is defined, and variables *pos* and i are initialized, denoting the first undefined position in *ord* and the index of the current customer, respectively. The main repeat-until loop of the algorithm includes three different possible movements along the cycle, corresponding to Steps 1–3, which are considered in sequence until the last customer has been served (i.e., it has been included in vector *ord*). At Step 1 customers are served until the first customer i whose demand exceeds the residual vehicle capacity (i.e., such that $v_i = 3$) is encountered. In this case, the customer index is stored in variable *start* and all the subsequent customers with non positive net demand are served until the first customer i such that $v_i = 1$ is reached and served. Then all customers from *start* onwards with positive net demand are served and a new iteration of the repeat-until loop is executed for the possibly remaining customers.

The complexity of algorithm C is $O(n)$ since vectors v and σ may be computed in $O(n)$ time and each edge is considered only twice. Algorithm C may be used directly to solve the case in which the cycle is visited counterclockwise by defining, in linear time, the appropriate instance or by simply modifying it to solve the counterclockwise case explicitly. Therefore, the overall optimal solution may be obtained in $O(n)$ time.

3.2. A heuristic for the general TSPPD

The exact algorithms for $\mathrm{TSPPD}_{\mathrm{cycle}}$ developed in the previous section may be used as a basis for a new heuristic, called H, $\mathrm{TSPPD}_{\mathrm{cycle}}$ for the general TSPPD defined on a complete graph. In the following we describe this algorithm and discuss its worst-case performance for the case in which the graph G is undirected and triangular. Although algorithm H does not have a better worst-case performance with respect to other algorithms proposed in the literature, it is experimentally shown to give better results in general (see Section 5).

Algorithm H first determines a possibly infeasible Hamiltonian tour by heuristically solving the TSP associated with the TSPPD. Then, a feasible solution of the TSPPD on the cycle defined by the TSP solution is derived by means of the exact algorithm of Section 2.1. The final Hamiltonian solution is obtained by applying shortcuts to the solution of the previous step.

The overall complexity of algorithm H depends on that of the algorithm used for obtaining the heuristic TSP solution, since the subsequent steps require linear time. In our computational testing we used a simple "farthest insertion" TSP heuristic which runs in $O(n^2)$ time. Moreover, it may be easily seen that in this case also, there is no need to actually apply the shortcut phase since the resulting Hamiltonian tour is indeed that stored in the vector *ord* returned by the exact algorithm for TSPPD on a cycle.

Now assume that G is undirected, complete and triangular, and that H uses for its first step the well-known TSP heuristic due to Christofides [3] which has a worst-case performance ratio of 3/2.

Theorem 1. *The worst-case performance ratio of Algorithm H is* 3 *and the bound is tight.*

Proof. Given any TSPPD instance I, we know that the optimal solution value $v^*(I)$ is not less than the optimal value of the associated TSP instance $v^*_{\mathrm{TSP}}(I)$. On the other hand, from Eq. (1) we have that the cost of the solution provided by H is at most twice the cost of the heuristic TSP solution which is, in turn, at most 3/2 times $v^*_{\mathrm{TSP}}(I)$. Hence

$$H(I)/v^*(I) \leqslant 3 \quad \text{for all } I. \tag{3}$$

The tightness of this bound directly follows from the tightness of the bound for Christofides heuristic and from that of Eq. (1). $\square$

4. Local search improvements

The solutions obtained with the previous heuristic may be improved by using simple local search routines based on arc exchanges. When dealing with the TSPPD, one has to carefully implement these routines since the solution obtained after an arc exchange is not necessarily feasible with respect to the capacity constraint.

For example, consider the well-known two exchanges. Given a feasible solution, defined by the successor vector σ, a two exchange associated with the node-disjoint arcs (a, σ_a) and (b, σ_b) leads to the solution shown in Fig. 1 containing arcs (a, b) and (σ_a, σ_b) and where the orientation of all the arcs between σ_a and b is changed. Since in the new solution up to a and from σ_b onwards nothing is changed with respect to the original one, the new solution is feasible if and only if the reversed sequence between b and σ_a is such. In particular, let L_a be the load of the vehicle after visiting customer a then the new sequence is feasible if and only if $L_a + \sum_{j=b}^{h} \delta_j \leqslant Q$ for all h of the reversed

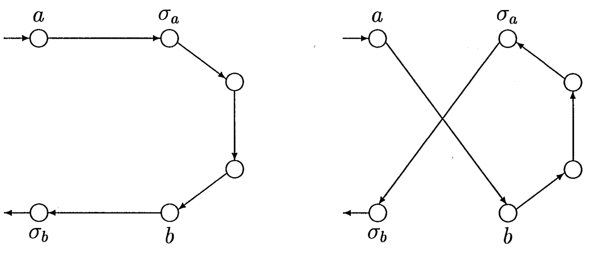

Fig. 1. An example of two exchange.

sequence between b and σ_a. By using parametric techniques, the feasibility check may be implemented to run in constant time, thus the overall complexity of the evaluation of all two-exchanges is $O(n^2)$. Analogous considerations may be applied to more involved exchange procedures.

5. A tabu search algorithm for TSPPD

We have also developed a tabu search (TS) algorithm for TSPPD. This global optimization technique explores the solution space of a problem by moving at each iteration from the current solution to its best neighbor. To avoid cycling some attributes of the current solution are stored and any solution possessing the same attributes is declared forbidden, or tabu, for a number of iterations. Typically the search stops after a given number of iterations without improvement. Tabu search is a generic technique that must be carefully tailored to the problem at hand in order to produce good results. Various enhancements such as diversification, intensification, and randomization have been proposed. For general references on tabu search, see Glover [5, 6] and the recent book by Glover and Laguna [7].

Our implementation is rather simple. It uses the two-exchange neighborhood described in Section 3. Two versions have been implemented and tested. The first, TS1, is a single run algorithm initialized with the solution of our heuristic (H) without the simple descent post-optimization. The second version, TS2, is a multi-start technique in which TS is run four times from different heuristic solutions. We have used the sequence H, TREE, PDαT and CFI. During the course of the search, we allow intermediate infeasible solutions with respect to vehicle capacity by penalizing infeasibilities by a factor of 100 000.

To implement the tabu mechanism, we store the new arcs (a, b) and (σ_a, σ_b) inserted in the solution at each iteration. Any move that removes these arcs for the next iterations is considered tabu. The tabu list is implemented as a matrix which stores, for each arc (i, j), the most recent iteration in which it has been included. This value is set equal to $-\infty$ at the beginning of each restart. This mechanism enables the tabu status of an exchange to be implemented in constant time. The tabu tenure t is initially randomly selected in an interval $[t_{\min}, t_{\max}]$ according to a discrete uniform distribution. Computational experiments have led to the choice of $t_{\min} = \max\{2, n/4\}$ and $t_{\max} = 5n/4$. Every 25 iterations, the value of t is updated as follows: if the best known solution has been improved over the last 25 iterations, then set $t := \min\{t + 2, t_{\max}\}$; otherwise, set $t := \max\{t - 2, t_{\min}\}$.

Restarts, which are initiated from the best known solution, are performed whenever the best known solution has not improved for 200 consecutive iterations. The algorithm terminates when either 300 s of CPU time have elapsed for a given run, or when 500 consecutive non-improving iterations have been performed.

Clearly, other TS implementations are possible, but our choice of strategies and parameters results in an aggressive search capable of producing high-quality solutions within a short computing time. As a rule, TS2 produces slightly better solutions than TS1 and the best of the four runs is generally that initialized with our heuristic H.

6. Computational results

In this section we present the results of an extensive computational testing of the heuristics described in the previous sections. We coded in FORTRAN 77 the algorithms PDαT, CFI, TREE of Section 1, the new algorithm H of Section 2, the local search post-optimization procedure of Section 3 and the two tabu search algorithms TS1 and TS2 of Section 4. The codes were run on a 486/66 MHz personal computer on instances from the literature and on randomly generated test problems.

The first class of test problems we examine is made up by TSPPD instances derived from symmetric capacitated vehicle routing problem (VRP) instances from the literature. As proposed in Halse [8] for the VRP with simultaneous pickups and deliveries, for each VRP instance we obtain a TSPPD instance where the customer set, the depot and the cost matrix are the same as in the VRP one. Let q_i denote the demand of customer i in the VRP instance, the corresponding demands for the TSPPD are defined as follows:

$$d_i := q_i, \qquad p_i := \begin{cases} \lfloor (1-\beta) q_i \rfloor & \text{if } i \text{ is even} \\ \lfloor (1+\beta) q_i \rfloor & \text{if } i \text{ is odd} \end{cases} \qquad i = 1, \ldots, n, \tag{4}$$

where β is a real non-negative parameter smaller than 1. Finally, the vehicle capacity is defined as $Q = \max\{D, P\}$. We have considered 26 symmetric VRP instances (both Euclidean and non-Euclidean) proposed in the literature with n ranging between 6 and 261; for each of them we derived four TSPPD instances, corresponding to $\beta = 0.00, 0.05, 0.10, 0.20$ (where in the case with $\beta = 0.00$ TSPPD becomes a TSP).

Tables 1 and 2 summarize the results obtained with these instances: for each value of β the tables report the number of instances in which the algorithm obtained the best known solution (columns marked "#"), the average over the 26 instances of the percentage ratios of the heuristic solution value with respect to the optimal TSP solution value (columns marked "%"), and of the computing times expressed in PC 486/66 CPU seconds. The line "Average 1" computes the average ratio of the heuristic solution value over the best of two TSP lower bounds: the assignment bound and the 1-SST bound (see [2]).

Table 1 reports the solutions and the computing times of the four basic algorithms TREE, PDαT, CFI, and H without the post-optimization step, whereas Table 2 compares the results of the four basic algorithms with the post-optimization step and the two tabu search algorithms. The computing times indicated for algorithm TS2 are the total for all the four runs, and the reported solutions are the best found in the four runs. The last line in each table reports the average results over all the four different β values.

Table 1
Average results of the four basic algorithms on instances derived from VRP test problems

β	TREE			PDαT			CFI			H		
	#	%	S	#	%	S	#	%	S	#	%	S
0.00	0	132.3	0.03	4	113.8	0.38	3	112.8	0.42	21	107.5	0.02
0.05	1	136.2	0.02	1	121.8	0.37	9	112.9	0.28	18	110.7	0.02
0.10	1	138.9	0.02	2	122.0	0.38	6	114.8	0.28	19	111.1	0.02
0.20	1	139.1	0.03	3	122.7	0.38	7	116.3	0.28	16	113.0	0.01
Average		136.6	0.02		120.1	0.38		114.2	0.31		110.6	0.02
Average 1		158.3	0.02		139.2	0.38		132.8	0.31		128.6	0.02

Table 2
Average results of the four basic algorithms with the post-optimization step and of the tabu search algorithms on instances derived from VRP test problem

β	TREE			PDαT			CFI			H			TS1			TS2		
	#	%	S	#	%	S	#	%	S	#	%	S	#	%	S	#	%	S
0.00	5	104.0	0.70	2	106.8	0.63	2	107.2	0.75	3	105.0	0.32	13	101.1	24.63	26	100.7	99.69
0.05	1	116.3	0.56	1	116.1	0.53	1	110.5	0.38	2	107.9	0.18	10	105.0	16.11	22	103.6	68.20
0.10	1	118.3	0.58	1	116.2	0.51	1	112.4	0.36	2	108.4	0.14	11	105.8	14.44	22	104.5	71.66
0.20	0	120.4	0.57	2	117.5	0.50	2	114.0	0.34	3	110.1	0.13	13	107.2	15.87	23	106.3	69.78
Average		114.7	0.60		114.1	0.54		111.0	0.46		107.8	0.19		104.8	17.76		103.8	77.33
Average 1		133.1	0.60		132.3	0.54		129.1	0.46		125.4	0.19		121.7	17.76		120.5	77.33

We also considered two classes of randomly generated test problems with $n = 25, 50, 75, 100, 150$ and 200. In these classes we uniformly randomly generated the d_i values in the interval [1, 100] and defined p_i according to Eq. (4) with the same β values of the previous class. Instances with uncorrelated values for d_i and p_i, both uniformly random in [1, 100], are also generated and are marked with $\beta = \infty$. For each class and for each pair of n and β values, ten instances are generated and the reported values are the averages over these ten instances. The meaning of the data reported in the tables is the same as in the previous ones with the only difference that here the percentage ratios are computed with respect to the best TSP lower bound provided by the assignment and by the 1-SST relaxations. Since it is well known that on symmetric and Euclidean problems the quality of these lower bounds is generally poor, the reported percentage ratios computed with respect to the actual TSPPD optimal solution may be considerably tighter.

The first class consists of Euclidean problems where for each customer and the depot the coordinates are uniformly randomly generated in the interval [0, 100] and the cost of each arc is defined as the Euclidean distance between its endpoints. The second class consists of symmetric problems where the cost of each arc (i, j) is uniformly randomly generated in the interval [0, 100], then the cost matrix is triangularized by applying the Floyd–Warshall algorithm. Tables 3–6 report the result obtained on these randomly generated problems.

Table 3
Average results of the four basic algorithms on random Euclidean instances

β	n	TREE			PDαT			CFI			H		
		#	%	S	#	%	S	#	%	S	#	%	S
0.00	25	0	156.0	0.00	0	135.1	0.00	1	134.9	0.01	9	125.7	0.00
	50	0	153.4	0.00	0	136.6	0.03	0	136.3	0.03	10	125.0	0.00
	75	0	155.6	0.03	0	136.4	0.11	0	137.3	0.12	10	127.7	0.02
	100	0	156.6	0.04	0	135.2	0.23	1	134.1	0.31	9	129.0	0.02
	150	0	158.1	0.07	0	134.2	0.84	2	133.3	0.97	8	129.2	0.03
	200	0	158.3	0.12	0	135.2	2.18	1	133.2	2.41	9	130.9	0.07
0.05	25	0	163.4	0.00	0	153.0	0.00	3	143.5	0.00	7	134.5	0.00
	50	0	164.3	0.00	0	154.6	0.02	1	135.6	0.03	9	126.9	0.00
	75	0	167.7	0.01	0	150.2	0.11	3	140.4	0.07	7	134.7	0.00
	100	0	166.3	0.03	0	147.0	0.25	3	135.2	0.17	8	131.6	0.03
	150	0	168.0	0.06	0	146.3	0.84	4	133.5	0.61	6	131.6	0.05
	200	0	167.7	0.12	0	146.1	2.19	8	130.5	1.57	2	132.9	0.08
0.10	25	1	163.7	0.00	1	152.3	0.00	1	147.2	0.00	7	136.5	0.00
	50	0	164.6	0.00	1	155.6	0.04	4	142.8	0.02	5	131.8	0.00
	75	0	166.9	0.02	1	150.2	0.10	4	145.5	0.09	5	142.1	0.02
	100	0	166.2	0.04	0	148.0	0.23	5	141.8	0.17	5	138.5	0.03
	150	0	168.2	0.06	1	144.0	0.83	3	137.0	0.62	7	135.9	0.05
	200	0	167.1	0.13	0	147.4	2.18	9	132.0	1.56	1	135.1	0.09
0.20	25	1	163.4	0.00	1	153.9	0.00	1	151.3	0.00	7	140.3	0.00
	50	0	164.5	0.02	0	153.6	0.03	4	152.2	0.06	6	133.0	0.00
	75	0	167.0	0.03	1	151.0	0.10	5	149.2	0.07	5	146.1	0.02
	100	0	166.5	0.04	3	147.8	0.26	4	150.0	0.17	3	142.6	0.02
	150	0	167.8	0.06	2	143.4	0.83	2	141.0	0.62	6	138.6	0.05
	200	0	167.1	0.12	1	145.9	2.17	7	136.2	1.57	2	139.2	0.07
∞	25	0	163.0	0.00	0	160.3	0.00	1	147.8	0.00	9	135.5	0.00
	50	0	164.4	0.00	1	159.5	0.05	1	155.9	0.02	8	142.3	0.00
	75	0	168.2	0.02	1	150.4	0.09	5	143.9	0.07	5	141.8	0.00
	100	0	167.9	0.02	1	146.3	0.23	2	141.7	0.17	7	135.2	0.02
	150	0	168.0	0.07	1	143.6	0.83	4	138.1	0.64	5	142.7	0.05
	200	0	168.8	0.12	2	144.5	2.19	5	143.5	1.55	3	139.2	0.09
Average			164.3	0.04		146.9	0.56		140.8	0.46		135.2	0.03

Results reported in Tables 1–4 indicate that both on instances derived from VRP test problems and on Euclidean instances the relative ranking of the four heuristics form best to worst is H, CFI, PDαT, and TREE. The only exception is found in Table (4) where TREE seems to be better than PDαT when a post-optimization step is used. Heuristic H is not only the best heuristic in terms of solution value, but also the fastest. In all cases, applying the post-optimization phase yields interesting improvements. Results are more dramatic in the case of the worst heuristic (TREE) and less than 5% in the case of the best heuristic (H). The effect on computing time is significant in relative terms, sometimes by one order of magnitude, but for the problem sizes considered, the total time remains equal to only a few seconds. When tabu search is applied on top of the heuristic H and

Table 4
Average results of the four basic algorithms with the post-optimization step, and of the tabu search algorithms on random Euclidean instances

β	n	TREE			PDαT			CFI			H			TS1			TS2		
		#	%	S	#	%	S	#	%	S	#	%	S	#	%	S	#	%	S
0.00	25	2	125.1	0.02	1	127.4	0.01	1	125.9	0.02	2	123.4	0.00	9	120.1	2.25	10	120.1	4.75
	50	0	123.3	0.06	0	130.0	0.06	0	127.8	0.06	1	121.4	0.03	5	117.8	5.03	9	117.1	25.45
	75	0	123.4	0.22	0	126.4	0.21	0	128.0	0.26	0	122.8	0.08	2	118.1	15.34	8	116.6	63.37
	100	0	121.7	0.54	0	128.3	0.42	0	126.9	0.51	0	123.3	0.20	2	117.6	27.64	9	116.5	114.11
	150	0	120.2	1.70	0	127.8	1.47	0	124.9	1.72	0	123.8	0.59	2	116.5	74.78	9	115.8	299.32
	200	0	121.8	4.13	0	126.3	4.02	0	125.4	4.16	0	124.3	1.68	3	117.5	136.11	9	116.7	529.43
0.05	25	1	143.4	0.00	0	147.1	0.00	0	142.1	0.00	5	132.8	0.00	5	132.1	1.14	10	128.9	5.03
	50	0	144.5	0.04	0	148.0	0.04	1	134.0	0.03	4	124.2	0.02	7	122.2	4.58	9	121.5	19.24
	75	0	141.8	0.16	0	141.8	0.16	0	137.6	0.10	3	130.4	0.05	6	129.7	11.81	8	127.5	43.64
	100	0	140.2	0.35	0	140.4	0.34	0	131.1	0.26	0	127.0	0.11	5	124.3	20.73	8	123.4	79.20
	150	0	139.3	1.30	0	139.6	1.19	0	128.4	0.90	0	126.3	0.35	5	123.4	51.96	6	123.0	185.92
	200	0	138.6	3.18	0	138.1	3.09	0	127.0	2.16	0	126.6	0.97	2	124.2	101.11	8	122.9	359.91
0.10	25	1	145.6	0.00	0	148.8	0.00	0	145.5	0.01	6	132.5	0.00	7	132.8	1.07	9	130.0	4.74
	50	0	142.7	0.05	0	147.7	0.04	1	140.9	0.04	5	128.4	0.01	5	127.1	4.65	10	124.4	18.40
	75	1	142.9	0.14	0	142.9	0.15	0	142.3	0.10	2	137.1	0.05	5	134.9	10.49	9	131.1	42.39
	100	0	141.3	0.34	0	143.0	0.33	0	139.4	0.24	1	133.6	0.11	5	131.6	22.19	8	128.4	78.31
	150	0	142.2	1.17	0	138.7	1.14	0	132.5	0.90	0	130.7	0.32	6	128.5	50.27	6	125.4	189.31
	200	0	140.7	2.91	0	139.3	3.01	1	129.0	2.09	0	129.3	0.90	2	127.4	94.10	8	126.0	352.72
0.20	25	1	145.6	0.01	1	150.2	0.00	0	148.8	0.00	3	135.9	0.00	5	133.5	1.15	9	128.9	5.35
	50	0	147.3	0.04	0	148.6	0.03	3	147.4	0.09	5	128.3	0.02	6	127.4	4.42	10	124.6	20.86
	75	0	144.6	0.15	0	144.2	0.15	0	146.6	0.09	1	141.5	0.05	2	140.9	9.95	10	132.3	43.09
	100	0	141.2	0.34	0	143.8	0.33	1	144.5	0.24	1	137.2	0.11	2	134.2	23.35	10	130.2	78.90
	150	0	143.5	1.14	0	136.7	1.21	0	136.9	0.83	1	131.9	0.37	6	129.9	51.85	7	126.3	183.73
	200	0	142.7	2.65	1	138.6	2.96	0	133.1	2.07	0	131.6	1.02	1	130.0	86.03	10	128.2	350.06
∞	25	0	144.6	0.01	0	152.6	0.00	0	141.3	0.00	3	132.2	0.00	5	130.3	1.14	10	126.7	5.03
	50	0	142.7	0.04	0	154.6	0.05	1	152.1	0.02	3	134.9	0.00	5	133.2	4.43	9	129.7	21.19
	75	0	143.5	0.15	0	143.9	0.15	0	142.3	0.10	1	136.0	0.05	6	134.5	10.16	9	129.4	42.34
	100	0	140.3	0.37	0	140.4	0.36	0	137.5	0.24	0	131.3	0.11	3	130.0	20.60	8	127.0	78.12
	150	0	141.9	1.13	0	138.6	1.12	1	136.2	0.80	1	136.3	0.37	2	134.5	45.11	9	127.6	176.82
	200	0	139.9	2.94	0	138.0	2.90	1	137.1	2.15	0	132.0	0.94	4	129.7	94.33	6	127.9	331.08
Average			138.5	0.84		140.4	0.83		136.4	0.67		130.2	0.28		127.8	32.93		125.1	125.06

Table 5
Average results of the four basic algorithms on random symmetric instances

β	n	TREE			PDαT			CFI			H		
		#	%	S	#	%	S	#	%	S	#	%	S
0.00	25	0	162.1	0.00	3	137.6	0.00	3	137.4	0.00	4	141.3	0.00
	50	0	187.8	0.01	3	148.3	0.03	5	148.9	0.05	4	151.4	0.01
	75	0	217.9	0.02	6	150.5	0.10	5	150.5	0.12	0	163.2	0.02
	100	0	226.2	0.03	3	150.0	0.25	7	149.1	0.27	0	161.7	0.02
	150	0	282.6	0.07	7	150.1	0.84	8	148.6	0.93	0	168.0	0.04
	200	0	314.7	0.11	2	139.2	2.03	7	138.0	2.22	2	144.3	0.08
0.05	25	1	161.5	0.00	3	154.2	0.00	4	150.9	0.00	2	151.6	0.00
	50	0	183.3	0.00	0	166.5	0.03	4	153.6	0.00	6	154.1	0.00
	75	0	215.1	0.01	0	164.6	0.10	8	156.3	0.08	2	165.2	0.01
	100	0	227.3	0.04	3	159.1	0.22	6	156.0	0.18	2	165.2	0.01
	150	0	271.0	0.06	3	162.6	0.82	5	160.8	0.61	2	172.3	0.04
	200	0	292.0	0.12	2	150.8	2.03	0	156.1	1.50	8	144.3	0.06
0.10	25	1	160.0	0.00	2	154.2	0.00	6	153.2	0.01	1	155.2	0.01
	50	0	183.3	0.00	0	164.7	0.02	6	155.7	0.03	5	165.6	0.01
	75	0	208.2	0.02	3	164.1	0.11	7	163.7	0.07	0	180.3	0.02
	100	0	219.3	0.04	6	158.3	0.23	2	162.2	0.18	2	174.4	0.02
	150	0	269.3	0.07	5	162.6	0.82	4	165.6	0.61	2	181.1	0.04
	200	0	285.7	0.12	3	150.7	2.03	0	162.2	1.51	8	145.8	0.08
0.20	25	1	158.9	0.00	4	154.4	0.00	3	159.9	0.00	2	155.9	0.00
	50	0	184.2	0.00	1	164.7	0.02	6	158.3	0.03	3	167.8	0.01
	75	0	206.8	0.02	6	164.2	0.10	2	171.9	0.09	2	184.1	0.00
	100	0	220.7	0.02	7	158.3	0.22	1	171.8	0.16	2	176.8	0.03
	150	0	267.9	0.09	6	161.6	0.84	2	170.0	0.62	2	194.5	0.03
	200	0	286.3	0.12	4	150.7	2.04	0	170.4	1.50	6	151.3	0.09
∞	25	1	163.0	0.00	1	150.8	0.02	5	149.6	0.00	3	145.1	0.00
	50	0	184.0	0.00	3	161.8	0.03	4	161.5	0.02	3	163.7	0.00
	75	0	201.0	0.02	6	161.4	0.10	4	167.5	0.06	1	189.4	0.02
	100	0	214.4	0.02	5	165.5	0.22	1	171.3	0.17	4	172.2	0.00
	150	0	276.7	0.06	9	158.7	0.82	0	194.3	0.63	2	181.4	0.05
	200	0	290.1	0.12	8	147.2	2.04	0	174.1	1.47	4	153.9	0.07
Average			224.0	0.04		156.2	0.54		159.6	0.44		164.0	0.03

the post-optimization phase, further gains are made, typically a few percentage points, at the expense this time of a large increase in computing time.

In the case of random symmetric instances, the relative order of the four heuristics, from best to worse in terms of solution quality, is PDαT, CFI, H and TREE if no post-optimization is applied, and PDαT, TREE, H and CFI, but the ranking is rather tight in the latter case. If computing time is taken into account, H still stands out as the best heuristic. The same comment as above can be made about the effect of applying a post-optimization phase or tabu search.

As in several routing problems of this type, the choice of a heuristic depends on whether computing time or solution quality is the determining factor. If a good solution must be determined

Table 6
Average results of the four basic algorithms with the post-optimization step, and of the tabu search algorithms on random symmetric instances

β	n	TREE			PDαT			CFI			H			TS1			TS2		
		#	%	S	#	%	S	#	%	S	#	%	S	#	%	S	#	%	S
0.00	25	4	126.6	0.02	1	128.2	0.00	3	129.5	0.00	2	128.4	0.02	10	120.9	1.18	10	120.9	4.89
	50	0	138.3	0.06	0	137.5	0.05	0	137.7	0.07	0	136.6	0.03	8	130.0	6.63	9	129.8	23.53
	75	0	140.6	0.18	0	139.6	0.15	0	141.6	0.18	0	140.9	0.13	3	129.3	15.26	10	127.9	61.09
	100	0	142.6	0.39	0	140.0	0.38	0	138.5	0.41	0	144.3	0.23	3	129.5	34.40	10	128.6	121.03
	150	0	144.5	1.32	0	141.2	1.08	0	139.0	1.19	0	145.9	0.57	4	128.9	69.91	10	127.6	288.03
	200	0	130.1	3.19	0	125.6	2.62	0	124.5	2.81	0	132.2	0.66	7	121.7	129.68	9	120.8	415.20
0.05	25	0	141.9	0.00	1	146.0	0.00	1	146.6	0.00	0	146.6	0.01	1	138.5	1.36	9	130.0	5.20
	50	0	151.8	0.03	0	157.4	0.05	0	151.5	0.02	0	149.3	0.00	3	144.2	6.33	10	139.1	23.02
	75	0	157.9	0.10	0	154.6	0.13	0	153.9	0.09	0	158.3	0.03	1	150.6	12.68	9	141.0	54.18
	100	0	152.4	0.27	0	152.1	0.26	1	153.9	0.21	0	159.6	0.07	0	150.2	30.12	10	143.3	104.73
	150	0	157.3	0.81	1	154.2	0.91	0	159.6	0.64	0	163.6	0.16	0	154.9	60.34	10	144.3	222.14
	200	3	137.5	1.89	2	141.5	2.21	0	154.9	1.56	0	141.2	0.19	1	140.7	68.89	9	133.4	325.21
0.10	25	1	141.3	0.00	1	147.7	0.01	1	151.8	0.01	0	148.7	0.01	1	137.5	1.25	9	129.0	5.58
	50	0	152.4	0.03	0	156.3	0.03	0	153.6	0.03	0	157.0	0.02	1	152.9	7.14	9	140.9	25.67
	75	0	159.3	0.09	0	155.5	0.12	0	162.2	0.08	0	171.0	0.05	2	160.5	13.75	10	146.3	49.33
	100	0	155.2	0.24	0	152.0	0.27	1	159.2	0.21	0	167.8	0.06	0	160.0	26.15	10	143.7	103.11
	150	0	159.0	0.74	1	155.4	0.91	1	164.0	0.64	0	170.5	0.15	0	161.5	66.54	10	146.4	235.85
	200	3	137.5	1.79	2	142.9	2.20	0	160.2	1.57	0	143.2	0.18	0	143.4	68.63	10	135.2	318.51
0.20	25	1	139.6	0.00	1	147.9	0.01	1	155.6	0.00	0	147.5	0.00	3	135.2	1.50	8	130.3	5.17
	50	0	151.2	0.03	0	156.6	0.03	0	156.2	0.03	0	161.2	0.02	0	154.1	6.14	10	140.8	22.86
	75	0	162.1	0.09	0	156.2	0.12	0	169.0	0.09	1	170.0	0.04	2	159.0	14.59	10	146.5	56.93
	100	0	155.4	0.21	0	151.7	0.28	0	167.4	0.18	0	168.9	0.07	0	161.1	26.42	10	143.2	93.86
	150	0	159.8	0.74	2	154.3	0.91	0	169.0	0.65	0	177.0	0.19	0	165.3	61.52	10	149.6	220.73
	200	1	141.9	1.70	3	142.3	2.19	0	168.0	1.57	1	146.3	0.20	1	145.4	68.85	10	136.5	331.36
∞	25	0	139.8	0.00	0	139.6	0.02	1	148.1	0.00	1	139.0	0.00	3	131.9	1.40	10	126.7	5.35
	50	0	151.3	0.04	0	156.1	0.04	0	158.0	0.02	0	154.6	0.01	1	150.3	5.25	10	140.0	22.23
	75	0	159.9	0.11	0	156.3	0.11	0	163.9	0.09	0	173.4	0.05	0	167.7	11.64	10	146.4	53.91
	100	0	154.2	0.21	0	153.7	0.28	0	168.8	0.18	0	163.5	0.07	1	157.0	31.84	10	143.3	106.36
	150	0	158.4	0.73	0	150.5	0.92	0	186.2	0.70	0	170.3	0.17	0	162.1	65.57	10	144.4	218.11
	200	2	139.7	1.69	2	139.3	2.21	0	171.7	1.54	0	146.7	0.22	0	144.3	82.82	10	133.6	346.05
Average			148.0	0.56		147.7	0.62		155.5	0.49		154.1	0.12		146.3	33.26		137.0	128.97

within a short time, then H with or without post-optimization is recommended. If solution quality is of prime importance and computing time is available, then applying TS2 is indicated.

Finally, for all classes of problems, the quality of the results as measured by the entries in the % column does not seem to be significantly affected by β or by n.

Acknowledgements

Michel Gendreau and Gilbert Laporte were supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP0038861 and OGP0039682. Daniele Vigo was supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica and by Consiglio Nazionale delle Ricerche, Italy. This support is gratefully acknowledged. The authors also thank Giovanni Rinaldi for having provided the optimal solutions of the TSP instances derived from the 26 VRP test problems from the literature. Finally, thanks are due to two anonymous referees for their valuable comments.

References

[1] Anily S, Mosheiov G. The traveling salesman problem with delivery and backhauls. Operations Research Letters 1994;16:11–8.
[2] Balas E, Toth P. Branch and bound methods. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB, editors. The Traveling Salesman Problem. Chichester: Wiley, 1985:361–402.
[3] Christofides N. Worst-case analysis of a new heuristic for the traveling salesman problem. Report 388, GSIA, Carnegie-Mellon University, 1976.
[4] Gendreau M, Hertz A, Laporte G. The travelling salesman problem with backhauls. Computers & Operations Research 1996;23:501–8.
[5] Glover F. Tabu search – Part I. ORSA Journal on Computing 1989;1:190–206.
[6] Glover F. Tabu search – Part II. ORSA Journal on Computing 1990;2:4–32.
[7] Glover F, Laguna M. Tabu Search. Norwell, MA: Kluwer, 1997.
[8] Halse K. Modeling and solving complex vehicle routing problems. PhD Thesis, no. 60, IMSOR, The Technical University of Denmark, 1992.
[9] Mosheiov G. The travelling salesman problem with pick-up and delivery. European Journal of Operational Research 1994;79:299–310.
[10] Rosenkrantz DJ, Stearns RE, Lewis II PM. An analysis of several heuristics for the traveling salesman problem. SIAM Journal on Computing 1997;6:563–81.

Michel Gendreau is professor of Operations Research at the Université de Montréal. He is the Deputy Director of the Centre for Research on Transportation. His main research interests include transportation planning, combinatorial optimization and metaheuristics.

Gilbert Laporte is professor of Operations Research at the École des Hautes Études Commerciales de Montréal and member of the Centre for Research on Transportation and of the GERAD. His research interests include vehicle routing, location and scheduling.

Daniele Vigo is associate professor of Operations Research at the University of Bologna. His main research activity is devoted to the design and analysis of algorithms for combinatorial optimization problems.