PERGAMON

Computers & Operations Research 27 (2000) 905–916

computers & operations research

www.elsevier.com/locate/dsw

A heuristic for the pickup and delivery traveling salesman problem

Jacques Renaud[a,b,*], Fayez F. Boctor[a,c], Jamal Ouenniche[a]

[a]*Network Organization Technology Research Center, Université Laval, Québec, Canada*
[b]*Télé-université, Université du Québec, 2600 boulevard Laurier, Case Postale 10700, Sainte-Foy, Quebec, Canada G1V 4V9*
[c]*Faculté des Sciences de l'Administration, Université Laval, Québec, Canada*

Received 1 December 1997; received in revised form 1 November 1998

Abstract

This paper deals with the pickup and delivery traveling salesman problem. First we show how to adapt some classical traveling salesman heuristics to solve this problem, then we propose a new and efficient composite heuristic. The proposed heuristic is composed of two phases: a solution construction phase including a local optimization component and a deletion and re-insertion improvement phase. To evaluate its performance, the proposed heuristic was compared to the only available heuristic specially designed to solve this problem, to an adaptation of the most efficient heuristic designed to solve the traveling salesman problem with backhaul, to an adaptation of the farthest as well as to an adaptation of the cheapest insertion methods. Each of these heuristics was followed by our deletion and re-insertion procedure which considerably improved their performance. Results based on a new set of test problems show that the proposed heuristic outperforms all these reinforced heuristics.

Scope and purpose

In several physical distribution problems, goods must be picked at an origin and delivered to a destination. Examples include the transportation of handicapped persons, the pickup and delivery of fast courier, of some medical supplies, etc. This problem differs from classical transportation problems because we have to deal with precedence constraints between the customers to be visited. This article describes a powerful heuristic for this difficult problem.

Keywords: Pickup and delivery traveling salesman problem; Heuristics; Improvement procedure

* Corresponding author. Tel.: + 1-418-657-2262; fax: + 1-418-657-2094.
E-mail addresses: jrenaud@teluq.uquebec.ca (J. Renaud), fayez.boctor@osd.ulaval.ca (F.F. Boctor)

0305-0548/00/$ - see front matter
PII: S0305-0548(99)00066-0

1. Introduction

The *Traveling Salesman Problem* (TSP) has received considerable attention over the last two decades. This led to the development of continuously improving optimal solution methods [1,2] capable of handling small problems as well as some efficient heuristics [3–5] to handle larger problems. But, as pointed out by Savelsbergh and Sol [6], much less attention has been devoted to *Traveling Salesman Problems with Precedence Constraints* (TSP-PC) like the *Dial-A-Ride Problem* (DARP), the *Traveling Salesman Problem with Backhaul* (TSPB) or the *Pickup and Delivery Traveling Salesman Problem* (PDTSP).

The Dial-A-Ride Problem is a TSP with precedence relations where a vehicle should transport a number of passengers. Each passenger should be transported from a given location (origin) to a given destination. The DARP is encountered frequently in the transportation of handicapped persons. In this case, as we transport human beings, the objective should no longer be to minimize the traveled distance but to minimize a function of the waiting time of the transported persons or a function measuring the overall client inconvenience [6]. Also, the DARP is a capacitated problem since only a limited number of passengers can be transported by the vehicle. Both the static DARP (where all transportation requests are known in advance) and the dynamic DARP (where some of the requests are transmitted to the vehicle driver on the road) have been considered [7]. Small problems can be solved to optimality by using dynamic programming while heuristic approaches are the only efficient approaches to deal with larger problems. Psaraftis [8] presents a worst-case analysis of the following two-phase heuristic for solving the static single-vehicle DARP. The first phase of the basic version of this heuristic consists of constructing a tour through all vertices by duplicating the minimum spanning tree. Thus, the resulting tour visits some vertices once and other vertices will be visited more than once. In the second phase, we move on the tour clockwise, beginning at the start location, while skipping any vertex that has been visited before or any destination whose origin has not been previously visited, until a feasible solution is obtained. The solution is then improved by performing a sequence of local interchanges.

The Traveling Salesman Problem with Backhaul is also a TSP with precedence relations. Given a set of linehaul customers and a set of backhaul customers, the TSPB consists of determining a least-cost tour such that all linehaul customers are visited before visiting any backhaul customer and each customer is visited exactly once. Gendreau et al. [9] presented some heuristics for the TSPB as well as a lower bound on the optimal tour length. On problems having up to 300 vertices they reported that the solutions obtained by the first version of their Double Cycle heuristic are, on average, within 0.5% of the lower bound.

The Pickup and Delivery Traveling Salesman Problem which will be addressed in this paper is another TSP with precedence relations and can be defined as follow. Let $G = (V, E)$ be a graph where $V = \{1, \ldots, n\}$ is the vertex set and $E = \{(i, j)\colon i \neq j,\ i, j \in V\}$ is the edge set. The vertex set is partitioned into three subsets: $\{1\}$, P, D, where the vertex 1 is the starting point or the depot, P is the set of pickup customers and D is the set of delivery customers. With E is associated a cost matrix $C = (c_{ij})$ representing travel costs, travel distances or travel times. To each pickup customer $i \in P$ is associated one and only one delivery customer, denoted $d(i)$, and to each delivery customer $j \in D$ is associated one and only one pickup customer, denoted $p(j)$. Thus $|P| = |D|$ and n is odd. The problem is to find the least-cost Hamiltonian cycle on G such that each pickup customer is visited before its associated delivery customer. The PDTSP is different from the DARP as the objective of

the PDTSP is simply to minimize the total traveling cost without any capacity restriction. Also, we notice that the PDTSP could be transformed into a TSPB if we add the restriction that all pickup customers should be visited before visiting any delivery customer. Consequently, the TSPB can be seen as a special case of the PDTSP.

Few papers have addressed the PDTSP. Kalantari et al. [10] proposed a branch and bound algorithm to deal with pickup and delivery constraints but this approach can only handle very small instances (less than 40 vertices). Savelsberg [11] investigated the implementation of the k-exchange improvement procedures to solve routing problems with precedence constraints and particularly to solve the PDTSP. He proposed implementations of the 2-exchange, Or-exchange and 3-exchange procedures that allow to reduce their processing time. Healy and Moll [12] proposed an improvement heuristic, called the sacrificing algorithm, and used it to solve the PDTSP. This algorithm can be seen as an augmentation of the 2-opt heuristic, and as thus it produced superior results. However, it was clearly less efficient than the 3-opt heuristic.

In this paper we propose a new heuristic to solve the PDTSP and compare its performance to a number of other heuristics. The paper is organized as follows. In Sections 2 and 3 we show how to extend some TSP and TSPB algorithms to solve the PDTSP. In Section 4 we give the details of the proposed heuristic. The description of our comparative study and the computational results obtained are given in Sections 5 and 6, respectively. Finally, our conclusions follow in Section 7.

2. Extending some traveling salesman heuristics

Several heuristics designed to solve the basic TSP can be extended to handle the PDTSP. Mainly, we have to modify these heuristics in order to make sure that no delivery customer is visited before its associated pickup customer. For example, to extend any insertion heuristic, we have to make sure that delivery customers are inserted only in the tour portion following their corresponding pickup customers. Improvement heuristics based on arc exchanges can also be extended to handle the PDTSP. The simplest way, for such extensions, is to take into consideration exchanges that do not violate precedence constraints only.

In this section we show how we adapt the farthest insertion heuristic; the cheapest insertion algorithms can be adapted in a similar way. Furthermore, we present our adaptation of some TSP improvement heuristics.

2.1. The farthest insertion heuristic

In our comparative study we adapted the farthest insertion heuristic as follows. Notice that this is not the only way to adapt this heuristic.

Step 0: (*Initialization*)

Find $i \in P$ such that c_{1i} is maximum. Set the partial tour $H = \{1, i, 1\}$ and the insertion candidates set $S = P \backslash \{i\} \cup \{d(i)\}$. Recall that $d(i)$ is the delivery vertex associated with i.

Step 1: (*Selection of the farthest vertex among the insertion candidates*)

Find $k \in S$ and $l \in H$ such that: $c_{kl} = \mathrm{Max}_{i \in S}\{\mathrm{Min}_{j \in H} c_{ij}\}$. The vertex k is the one to be inserted.

Step 2: (*Insertion of the selected vertex*)
Find the position to insert k that produces the smallest cost increase. If $k \in P$, then k can be inserted anywhere in the partial tour H, but if $k \in D$ we should insert k in the portion of H following the position of its associate pickup vertex $p(k)$. Let E_k be the set of edges where k can be inserted, then we insert k between s and t where $(s, t) \in E_k$ and

$$c_{sk} + c_{kl} - c_{st} = \underset{(i,j)\in E_k}{\text{Min}} \{c_{ik} + c_{kj} - c_{ij}\},$$

Update H by inserting k between s and t. Set $S = S\backslash\{k\}$ and if $k \in P$ add to S the vertex $d(k)$. If $S = \emptyset$, stop, otherwise go back to step 1.

2.2. k-exchange improvement heuristics

TSP improvement heuristics, based on arc exchange like the 2-opt and the 3-opt proposed by Lin [13], can be adapted to solve the PDTSP by considering only exchanges that respect precedence constraints. While dealing with the TSP, each single exchange takes constant time but for the PDTSP each exchange may take an amount of time proportional to n, the total number of vertices. This is because we have to check for feasibility after each route modification. Thus the complexity of each iteration of the 2-opt heuristic will increases from $O(n^2)$ to $O(n^3)$ and the complexity of each iteration of the 3-opt heuristic from $O(n^3)$ to $O(n^4)$. Several attempts to reduce this complexity are reported in the literature [11,12,14], however, more research is needed to evaluate the performance of these complexity reduction approaches.

To reduce computational time, in our experiment, only exchanges leading to a cost reduction are checked for feasibility. Besides, the feasibility check is done by comparing the position of each delivery customer in the tour with that of the associated pickup customer and we stop as soon as one precedence relation is violated.

The 4-Opt* improvement heuristic [5] was adapted in the same way. Note that since the 4-Opt* relies on a more complex set of moves, reduction procedures like those presented in [14] are not applicable. The results presented in [5] show that the 4-Opt* produces good results for the standard TSP. Compared to the 3-opt heuristic, the 4-Opt* required only 1% of its execution time while producing solutions only 0.95% worse in average.

3. Using TSPB heuristics to solve the PDTSP

The PDTSP can be transformed into a TSPB by considering pickup customers as linehaul customers and delivery customers as backhaul customers (consequently all pickup customers will be visited before visiting any delivery customer) and thus can be solved as a TSPB. Obviously, the TSPB solution satisfies the original PDTSP constraints as each pickup customer will be visited before its associated delivery customer. As mentioned above, the Double Cycle heuristic [9] and particularly its version called Double Cycle 1 (DC1), is a highly efficient TSPB heuristics. A brief description of the general framework of DC1, which will be used in our comparative study, follows.

Step 1: (*Constructing two separate Hamiltonian cycles for P and D*)
Apply a TSP solution algorithm to construct a Hamiltonian cycle through the vertices of the set P and apply it again on the set D. Let E_p and E_d denote, respectively, the set of edges in the tour over P and the set of edges in the tour over D.
Step 2: (*Joining these two cycles and the depot to construct a TSPB solution*)
For each pair of edges $(i, j) \in P$ and $(k, l) \in D$ do the following and retain the best solution. Remove the considered pair of edges and add three new edges to construct a TSPB solution. There are four possible combinations of such edges and the least cost one should be chosen. For example, after removing (i, j) and (k, l) we may add $(1, i)$, (j, k) and $(l, 1)$. The other three edge triplets that could be added are: $\{(1, i), (j, l), (k, 1)\}$, $\{(1, j), (i, k), (l, 1)\}$ or $\{(1, j), (i, l), (k, 1)\}$.

The performance of DC1 heavily depends on the performance of the TSP algorithm used to construct the tours over P and D. In their experiment, the authors used the algorithm called GENIUS [3] with p (a local optimization parameter) set to 3, 4 and 5. In our implementation of DC1, we used the I^3 algorithm [5] which performs just as well as GENIUS with $p = 5$ but requires approximately a quarter of its computing time.

4. The proposed heuristic

The algorithm we propose is composed of two phases. The first phase, called the *Double Insertion heuristic* (DI), inserts each delivery customer simultaneously with its associated pickup customer. The second phase, called the *Deletion and Re-Insertion heuristic*, is an improvement procedure that uses the 4-Opt* improvement heuristic [5].

4.1. The Double Insertion heuristic

Here follows a description of each of the steps of the Double Insertion heuristic which is a tour construction heuristic that includes a local optimization component.
Step 0: (*Initialization*)
Let i be the vertex yielding:

$$Max\{c_{1i} + c_{ij} + c_{j1}\} \quad \text{where } i \in P \text{ and } j = d(i).$$

The initial subtour is the ordered set $H = \{1, i, j, 1\}$. Set $P' = P \backslash \{i\}$.
Step 1: (*Minimum weighted insertion cost*)
If $P' = \emptyset$, stop. Otherwise, for each pair (i, j) where $i \in P'$ and $j = d(i)$, find their combined minimum weighted insertion cost, denoted Δ_i, and the corresponding position in the current subtour H. Let E' denote the set of edges in H, (k, l) be one of these edges and (s, t) be another edge that has a later position in the tour, then the cheapest weighted insertion cost of i and j can be calculated by

$$\Delta_i = \text{Min}\Bigg\{ \underset{(k,l)\in E'}{\text{Min}} \{\alpha c_{ki} + c_{ij} + (2-\alpha)(c_{jl}) - c_{kl}\}, \underset{(k,l),(s,t)\in E'}{\text{Min}} \{\alpha(c_{ki} + c_{il} - c_{kl}) + (2-\alpha)(c_{sj} + c_{jt} - c_{st})\} \Bigg\},$$

where α is a user prespecified weight such that $0 < \alpha < 2$. The first part in this formula gives the weighted cost if both i and j are inserted between any two consecutive vertices k and l. The second part gives the weighted cost if i is inserted between any two consecutive vertices k and l while j is inserted later in the tour between any two consecutive vertices s and t. When α is higher than 1, more weight is given to the distance added by the insertion of the pickup customer. By running the algorithm several times while varying the value α, we may be able to generate different solutions and consequently may be able to reach a better local optimum.

Step 2: (*Insertion*)

Let (i^*, j^*) be the pair of vertices that yields the smallest weighted insertion cost Δ_i. Update the partial tour H by inserting i^* and j^* in the position that produces this minimum cost and set $P' = P' \backslash \{i^*\}$.

Step 3: (*Local optimization*)

Apply the 3-opt improvement heuristic described in Section 2 to the chain composed of i^*, its r predecessors and its r successors. Do the same thing for the chain of $2r + 1$ vertices around j^*. The value of r is a parameter preselected by the user. Go back to step 1.

4.2. The Deletion and Re-Insertion improvement algorithm

This algorithm is composed of the following two steps.

Step 1: (*Improvement*)

Apply the 4-Opt* adaptation to the current PDTSP solution (initially to the solution obtained by the Double Insertion heuristic).

Step 2: (*Deletion and Re-Insertion*)

Consider the pickup customers one by one (in our implementation pickup customers were considered in ascending order of their numbers). Delete the pickup customer and its associated delivery customer from the tour. Find the cheapest re-insertion position for the deleted pair and re-insert it in the position found (see steps 1 and 2 in the Double Insertion algorithm presented above). If, after deleting and re-inserting all the pairs, no improvement is obtained, stop; otherwise go back to step 1.

5. Test problems

As there are no benchmark problems available to evaluate the performance of the proposed heuristic, we generated a set of 108 problem instances divided into 3 subsets (called A, B, and C) of 36 problems each. These problems were derived from 36 TSPLIB problems [15] having up to 441 vertices. All the 108 problems are Euclidean and the distances between the vertices were rounded to the nearest integer. These test problems can be obtained from the authors upon request.

The derived problems are similar to the original TSPLIB problems except that for each problem we designated one of the vertices as a depot, designated some of the vertices as pickup customers and chose for each pickup customer its associated delivery customer. To do so, we applied the following procedure to each of the 36 TSPLIB problems. Designate the first vertex as the depot and repeat the following step as long as there are at least two non-classified vertices (if only one vertex

remains, drop it). Choose a vertex at random and classify it as a pickup customer and choose one of its nearest λ non-classified vertices to be its associated delivery customer.

Three values of λ were used for each TSPLIB problem producing three different PDTSPs. The problems belonging to the subset A were generated by the above procedure with $\lambda = 5$. For the subset B we used $\lambda = 10$ and for C we used λ equal to the number of non-classified vertices. If the original TSPLIB problem has an even number of vertices, say n, then the corresponding PDTSP will have one less vertex, thus $n - 1$ vertices.

6. Computational results

To evaluate its performance, the Double Insertion heuristic (DI) was compared to four other heuristics: the Double Cycle 1 (DC1), the cheapest insertion, the farthest insertion and Psaraftis' algorithm [8]. Each of the 108 test-problems was solved by applying the double insertion heuristic with 30 different combinations of α and r ($\alpha = 0.5, 0.75, 1, 1.25, 1.5$ and $r = 3, 4, 5, 6, 7, 8$) as well as by the other four heuristics (a total of 34 heuristics) followed by the 2-opt, the 3-opt, the 4-Opt* and the Deletion and Re-Insertion (DRI) improvement heuristics. Thus, for each problem we obtained 136 solutions (34 heuristics $\times$ 4 improvement algorithms). For each of these problems, the best of the 136 solutions obtained was used as a reference solution to evaluate the performance of the heuristics tested.

Table 1 gives a summary of the results obtained by the Double Insertion heuristic alone (DI alone) and by the Double Insertion heuristic followed by the Deletion and Re-Insertion phase (DI + DRI). The table gives the average percentage increase over the best of the 136 solutions obtained and the average computing times (in seconds on a Sun Sparc Station 5) for the 108 problems with 30 different combinations of α and r. As expected, computing times increase with r. We notice that, when the Double Insertion heuristic is used alone (DI alone), the average percentage increase above the best solution found decreases with r. But if the Double Insertion heuristic is followed by the Deletion and Re-Insertion heuristic (DI + DRI), no or little improvement was achieved for $r > 6$. The table indicates also that for any value of r, the best results were obtained with $\alpha = 1.25$. Consequently, in the following we retain the parameter combination $\alpha = 1.25$ and $r = 6$ which gave the smallest average percentage increase.

Table 2 compares the results obtained by the proposed heuristic (with $\alpha = 1.25$ and $r = 6$) to those given by the Double Cycle 1 heuristic [9], the cheapest insertion heuristic, the farthest insertion heuristic and Psaraftis' heuristic [8]. The solutions obtained were then improved by using the 2-opt, the 3-opt, the 4-Opt* and the Deletion and Re-Insertion improvement algorithms.

Table 2 shows that the Double Insertion heuristic, whatever the chosen parameter combination, clearly outperforms the other four heuristics. When followed by the Deletion and Re-Insertion heuristic, DI produced an average percentage increase of only 4.48%. But if we use the 3-opt heuristic instead of the Deletion and Re-Insertion heuristic, we obtain an average percentage increase of 3.87%. However, to obtain this 0.61% additional improvement, the average computational time (of the improvement phase alone) increases from 26.8 to 1245 s. Also, if we use the 4-Opt* heuristic instead, we reduce the computational time (again of the improvement phase alone) to an average of less than 4 s but at the expense of increasing the average percentage increase over

Table 1
Average percentage increase over the best solution found for different values of α and r

Value of r	$\alpha = 0.50$				$\alpha = 0.75$				$\alpha = 1.00$				$\alpha = 1.25$				$\alpha = 1.50$			
	DI alone		DI + DRI		DI alone		DI + DRI		DI alone		DI + DRI		DI alone		DI + DRI		DI alone		DI + DRI	
	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time	% increase	Time
3	15.39	57.9	5.88	86.7	15.74	63.8	5.38	95.1	14.90	63.0	5.17	96.0	15.52	64.8	5.05	93.8	15.09	63.3	5.92	93.3
4	14.38	59.2	5.82	90.3	14.26	67.5	5.60	98.9	14.01	64.5	5.03	95.9	13.96	67.6	4.92	95.2	13.98	64.6	5.65	93.6
5	13.48	58.9	5.76	91.0	13.43	67.3	5.33	95.9	13.13	65.7	4.92	95.0	12.69	67.9	4.66	96.5	12.98	64.5	5.34	92.4
6	12.87	61.6	5.61	92.0	12.95	68.5	5.17	98.0	12.61	65.8	4.98	95.9	12.20	69.1	4.48	95.9	12.54	66.2	5.38	93.7
7	12.47	62.5	5.20	92.6	12.27	69.4	5.04	97.8	12.08	67.4	5.20	94.4	11.61	70.1	4.60	97.0	12.14	66.3	5.30	94.5
8	11.61	63.8	5.03	92.5	11.80	70.4	5.14	99.9	11.55	68.6	5.33	95.9	11.39	71.1	4.75	99.6	11.66	67.7	5.20	96.5

Table 2
Average percentage increase over the best solution found

Heuristic	Alone		+ 2-opt		+ 3-opt		+ 4-Opt*		+ Deletion and Re-Insertion	
	Average % increase	Time	Average % increase	Time	Average % increase	Time	Average % increase	Time	Average % increase	Time
Double Insertion ($\alpha = 1.25$, $r = 6$)	12.20	69.1	11.99	69.2	3.87	1314.2	5.64	73.1	4.48	95.9
Double Cycle 1	20.60	1.5	19.56	1.6	5.74	1511.5	7.35	5.1	6.27	29.0
Cheapest Insertion	40.55	4.5	31.02	6.3	8.76	3197.1	12.74	10.3	10.25	36.9
Farthest Insertion	38.55	0.1	33.07	1.0	9.52	2894.9	14.24	6.1	11.15	38.9
Psaraftis	26.27	844.6	26.27	844.7	6.72	3142.7	9.93	850.0	8.51	872.3

Table 3
Average percentage increase over the best solution and average computation time

Problem set	Performance measure	Double Cycle		Farthest Insertion		Psaraftis		Cheapest Insertion		Double Insertion ($\alpha = 1.25$, $r = 6$)			
		Alone	+Deletion/ Re-Insertion	Alone	+Deletion/ Re-Insertion	Alone	+Deletion/ Re-Insertion	Alone	+Deletion/ Re-Insertion	Alone	+3-opt	+4-Opt*	+Deletion/ Re-Insertion
A	Average % increase over the best solution	23.62	8.05	26.28	7.14	31.14	9.56	31.15	8.16	10.83	3.03	5.79	4.58
	Average time (s)	1.54	26.93	0.10	36.23	674.27	705.78	4.30	33.83	70.19	999.69	73.65	96.74
B	Average % increase over the best solution	20.34	6.33	38.43	10.38	27.97	8.96	39.64	10.85	12.39	3.45	6.02	4.60
	Average time (s)	1.53	30.51	0.10	32.56	813.09	840.25	4.40	32.88	65.96	1205.06	70.15	94.94
C	Average % increase over the best solution	17.86	4.44	50.94	15.96	19.71	7.03	50.87	11.77	13.40	5.14	5.13	4.29
	Average time (s)	1.57	29.76	0.11	48.21	1046.50	1071.15	4.80	44.12	71.34	1740.55	75.61	96.27

the best solution found to 5.64%. Given this, the Deletion and Re-Insertion improvement procedure can be viewed as an interesting alternative to the time consuming 3-opt algorithm.

It is interesting to note that, although it was originally designed to solve the TSPB, the Double-Cycle1 heuristic produced better results than the other three heuristics tested here. Finally, comparing the results obtained by the Double Insertion heuristic to those given by the Double Cycle heuristic indicates that important savings may be obtained by visiting pickup and delivery customers alternatively instead of simply visiting all pickup customers before visiting delivery customers.

Table 3 gives the average percentage increase above the best solution and the average computational time for problem subsets A, B, and C. It is easy to see that, because of the method used to generate these three subsets of problems (see Section 5), within the subset A the average distance between delivery customers and their associated pickup customers is smaller than the average distance in subset B which is smaller than the average distance in subset C. Thus, comparing the results given in Table 3 allows us to see the effect of this structural characteristic of the problems on the performance of the proposed heuristic as well as on the performance of the other heuristics tested. For example, it seems that for some of the tested heuristics (the farthest insertion, the cheapest insertion and the Double Insertion heuristic without using any post improvement heuristic), the average percentage deviation from the best solution found increases as the average distance between delivery customers and their associated pickup customers increases. For the remaining heuristics (the Double Cycle and Psaraftis' heuristics), the contrary was observed.

Finally, it is interesting to note that, when applied to the problems of subset C, the Double Insertion heuristic followed by either the Deletion and Re-Insertion algorithm or the 4-Opt* algorithm produced a smaller average percentage increase over the best solution (respectively 4.29 and 5.13%) than the Double Insertion heuristic followed by the 3-opt algorithm (5.14%).

7. Conclusion

This paper presented a new and efficient two-phase heuristic to solve the pickup and delivery traveling salesman problem. In both the first phase, a construction phase called the Double Insertion heuristic, and the second phase, an improvement phase called the Deletion and Re-Insertion heuristic, a pickup customer is inserted or deleted at the same time as the associated delivery customer. To evaluate its performance, the proposed heuristic was compared to 4 other heuristics and our computational results show that it outperforms all the heuristics tested.

These results, based on a set of 108 test problems derived from 36 TSPLIB problems, show that while the performance of some of the tested heuristics deteriorate as the average distance between pickup customers and their associated delivery customers increases, the performance of the proposed heuristic remains on the same level and even becomes slightly better.

Acknowledgements

This work was partially supported by grants OPG0036509 and OGP0172633 from the Canadian Natural Sciences and Engineering Research Council (NSERC). This support is gratefully acknowledged. The authors also thank the referees for their valuable comments and suggestions.

References

[1] Padberg MW, Rinaldi G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problem. SIAM Review 1991;33:60–100.
[2] Grötschel M, Holland O. Solution of large-scale symmetric traveling salesman problems. Mathematical Programming 1991;51:141–202.
[3] Gendreau M, Hertz A, Laporte G. New insertion and post-optimization procedures for the traveling salesman problem. Operations Research 1992;40:1086–94.
[4] Zweig G. An effective tour construction and improvement procedure for the traveling salesman problem. Operations Research 1995;43:1049–57.
[5] Renaud J, Boctor FF, Laporte G. A fast composite heuristic for the symmetric traveling salesman problem. INFORMS Journal on Computing 1996;8:134–43.
[6] Savelsbergh MWP, Sol M. The general pickup and delivery problem. Transportation Science 1995;29:17–29.
[7] Psaraftis H. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 1980;14:130–54.
[8] Psaraftis H. Analysis of an $O(N^2)$ heuristic for the single vehicle many-to-many euclidean dial-a-ride problem. Transportation Research 1983;17B:133–45.
[9] Gendreau M, Hertz A, Laporte G. The traveling salesman problem with backhauls. Computers and Operations Research 1996;23:501–8.
[10] Kalantari B, Hill AV, Arora SR. An algorithm for the traveling salesman problem with pickup and delivery customers. European Journal of Operational Research 1985;22:377–86.
[11] Savelsbergh MWP. An efficient implementation of local search algorithms for constrained routing problems. European Journal of Operational Research 1990;47:75–85.
[12] Healy P, Moll R. A new extension of local search applied to the dial-a-ride problem. European Journal of Operational Research 1995;83:83–104.
[13] Lin S. Computer solutions to the traveling salesman problem. Bell System Technical Journal 1965;44:2245–69.
[14] Psaraftis H. k-interchange procedure for local search in a precedence-constrained routing problem. European Journal of Operational Research 1983;13:391–402.
[15] Reinelt G. TSPLIB – A traveling salesman problem library. ORSA Journal on Computing 1991;3:376–84.

Jacques Renaud is Professor at Télé-université, Quebec city. His research interests include vehicle routing and distribution problems.

Fayez F. Boctor is Professor at the Faculty of Administrative Sciences, Université Laval. His research interests include production and logistics management.

Jamal Ouenniche is an associate member of the Network Organization Technology Research Center. His research interests include production and logistics management.