ELSEVIER

Operations Research Letters 16 (1994) 11–18

operations research letters

The traveling salesman problem with delivery and backhauls

Shoshana Anily[a,*], Gur Mosheiov[b]

[a] *Faculty of Management, Tel-Aviv University, P.O. Box 39010, Ramat Aviv, Tel Aviv 69978, Israel*
[b] *School of Business Administration and the Statistics Department, The Hebrew University, Jerusalem, Israel*

(Received 6 May 1993; revised 1 January 1994)

Abstract

The problem that we consider here deals with a single vehicle of a given capacity that needs to serve N customers: some of the customers require a delivery of stock from the warehouse, whereas others need to deliver stock from their location to the warehouse. The objective is to find a shortest feasible tour visiting all customers, emanating from and ending at the warehouse. We introduce an efficient ($O(N^2)$) heuristic for this problem. The heuristic improves the worst-case bound known in the literature from 2.5 to 2. However, its average performance is shown in our numerical study to be slightly worse than that of a previously published ($O(N^3)$) solution method.

Key words: Traveling salesman problem; Vehicle routing; Heuristic; Worst-case analysis; Computational complexity

1. Introduction

The problem we study here deals with a single period, single warehouse distribution system with two sets of customers dispersed in the plane. The first set, denoted by D, contains "delivery customers" who require delivery of goods from the warehouse, whereas the second set, denoted by B, contains "backhaul customers" who need to deliver goods from their location to the warehouse. (Note that the literature distinguishes between "backhaul" customers and "pickup" customers: a "pickup" customer may deliver goods to any of the other locations including the warehouse. In this research we restrict ourselves to delivery and backhaul customers only.) In order to avoid excess notation we will assume a single commodity problem (however the results can be easily generalized to the multiple commodity case). Each customer i is characterized by its geographic location and its requirement size d_i (b_i) units for delivery (backhaul). The stock is delivered and backhauled by a single vehicle of limited capacity. The distance function between distinct locations is assumed to obey the triangle inequality. The objective is to design a route of minimum total length so that all customers are served: the desired route should emanate from the warehouse, stop at customers for delivery/backhaul purposes and finally terminate at the warehouse without violating the vehicle's capacity restriction along the route. Note that the vehicle is not allowed to visit the warehouse in between its initial and final stops there; also, backhaul stock cannot be used to satisfy delivery requirement. Thus, the resulting tour is a closed tour

* Corresponding author.

SSDI 0167-6377(94)00010-4

on the set of customers and the warehouse that visits each location exactly once. We call this problem the traveling salesman problem with delivery and backhaul (TSPDB).

The multiple vehicle case which we call the vehicle routing problem with delivery and backhaul (VRPDB) and we address in a later paper, has many real life applications. Bodin et al. [1] and Casco et al. [2] describe some applications and review some solution techniques. The problem is faced, for example, by mailing parcel services as UPS: trucks load parcels at the warehouse and on their way they unload parcels at addressees and collect parcels from senders. All parcels collected must be processed at the warehouse before being delivered to the addressees. Another application described by Casco et al. [2], deals with the grocery industry in the USA: this industry has recognized the cost-cutting potential of servicing backhaul points on predominantly delivery routes. As the authors report, the industry has saved more than $160 million a year since 1982 on its distribution costs by allowing vehicles on their delivery routes to collect large volumes of inbound products. Mosheiov [5] reports about a different application where inner-city under privileged children are transported to summer vacations at volunteer families living out of town. The transportation service is made by buses and is confined to a limited number of days when some children start their vacation and others end theirs. In all of these problems, if a heuristic based on cluster-first–route-second is applied then the routing problem related to each of the clusters is a TSPDB type.

There is a vast literature on the traveling salesman problem (TSP) with or without additional constraints, however the problem discussed here has received very little attention in spite of its applicability. It is easy to see that if $D = \emptyset$ or $B = \emptyset$ then the TSPDB reduces to the classical TSP, proving that the problem is NP-hard [4]. As such, exact solution methods become impractical when the problem size increases, giving rise to the need for developing heuristic solution methods whose running time is polynomial in the problem size. In this paper we propose a heuristic for the TSPDB and prove its worst-case performance: a heuristic is said to have a worst-case bound of $1 + \alpha$ if for any data set the heuristic produces a solution whose cost does not exceed $100(1 + \alpha)\%$ the optimal cost. (For example, the best known heuristic in terms of the worst-case bound for the classical TSP, is Christofides' algorithm [3], whose worst-case bound is 1.5). Mosheiov [5] in his paper, considers the TSPDB: he shows that any closed tour on $D \cup B$ contains a certain (initial) point and a certain direction so that starting at that point with a vehicle loaded with the total amount to be delivered and following the tour at that direction, while delivering (loading) stock at delivery (backhaul) customers as required, will not violate the capacity restriction at any point on the tour. Therefore, there exists an arc and a direction on any closed tour on $D \cup B$ so that disconnecting the arc and connecting its two end points to the warehouse will result in a feasible tour. As a consequence any heuristic for the TSP whose worst-case bound is $1 + \alpha$ may be used for constructing the initial tour over $D \cup B$ and by searching the arc that can be disconnected and connecting its two end points to the warehouse will generate a feasible tour for the TSPDB whose worst-case bound is $2 + \alpha$. Since Christofides's heuristic provides the best known worst-case bound (of 1.5) for the TSP, applying it as suggested above will result in a heuristic for the TSPDB whose worst-case bound is 2.5 and complexity of $O(N^4)$. In this paper we propose the 2MST heuristic for the TSPDB which is based on doubling a minimum spanning tree. We prove that the worst-case bound for the 2MST is 2 which is an improvement of Mosheiov's [5] bound. In terms of complexity, 2MST requires $O(N^2)$ operations, which is the size order of the data set. Similar to the TSP, the average performance of various heuristics may be far away from their worst-case bound: e.g., Christofides algorithm, which became a milestone in the theory of the TSP, is rarely used in practice. In our computer study, one version of a heuristic proposed by Mosheiov [5], the cheapest insertion with delivery and backhaul (CIDB), which has a worst-case bound of 3, usually outperforms 2MST.

The paper is organized as follows: in Section 2 we introduce the notation and preliminaries. In Section 3 we propose the heuristic for the TSPDB. We conclude with Section 4 that presents an example

and a short computer study in which we compare our heuristic with the heuristic proposed by Mosheiov [5].

2. Notation and preliminaries

D (B) = the set of delivery (backhaul) customers.
d_i = the demand size of delivery customer i, $i \in D$.
b_i = the amount to be loaded at backhaul customer i, $i \in B$ (d_i and b_i are assumed to be non-negative integers).

Note that if a location i serves both as a delivery and a backhaul customer, the overall effect on the vehicle's load of serving customer i is the difference between its backhaul size and its delivery size. When such a customer is visited then first the vehicle should be unloaded by its delivery size and then should be loaded by its backhaul size. Thus, we can assume without loss of generality that i is a delivery customer if its demand size is greater than or equal to its backhaul size and i is a backhaul customer otherwise. The delivery (backhaul) size should be set to the absolute value of the difference between its delivery and backhaul sizes. Observe that it is not possible to ignore customers for whom the above difference is zero since the location cannot use its own stock to cover its demand and thus must be visited by the vehicle, even though the visit does not affect the vehicle's load.

$N_D = \sum_{i \in D} d_i$ ($N_B = \sum_{i \in B} b_i$) is the total amount to be unloaded (loaded) at delivery (backhaul) customers.
$N = D \cup B$ is the total set of customers. We also use N to denote the total number of customers. Note that for each $i \in N$ either $d_i = 0$ or $b_i = 0$.

We denote the warehouse by 0.

$N^0 = N \cup \{0\}$, the set of customers and the warehouse.

Throughout the paper we assume that the triangle inequality holds, i.e. for any $i, j, k \in N^0$: $d(i,k) \leqslant d(i,j) + d(j,k)$.

q = is the vehicle's capacity.

Since we are not allowing the vehicle to return to the depot in between the beginning and the end of the tour, we require, in order to ensure feasibility, that both N_B and N_D do not exceed the vehicle capacity q.

Let $G(N, A)$ be a graph on the set of nodes N, and the set of arcs $A \subseteq N \times N$. Let $l(G)$ denote the total length of the arcs of G. The graph G may be directed or undirected. In an undirected graph $(i,j) \in N \times N$ represents an arc connecting i and j. In a directed graph we indicate the arc direction by using the symbol $i \rightarrow j$ to denote an arc connecting i to j. An undirected tree is a connected graph that does not contain any cycles. Each rooted tree is a tree with a distinguished node which is called the *root*. The *depth* of a node in a given rooted tree is the number of arcs on the path connecting the root to the node. In a rooted directed tree with all arcs directed away from the root, each node i except the root, is associated with a single node k so that the arc $k \rightarrow i$ is an arc of the tree; node k is said to be the *parent* of i. All nodes j that are connected to i via an arc $i \rightarrow j$ are called the *children* of i. The *leaves* of a tree are all the nodes which do not have any children.

For any given problem, we let V^{OPT} represent its minimum cost. If heuristic H is applied to solve the problem, the cost generated is denoted by V^{H}.

3. The 2MST heuristic

In this section we propose a simple polynomial heuristic for the TSPDB whose worst-case bound is 2. We follow a common procedure where at the first step we solve a relaxation of the problem and in the second step we extend the solution obtained at step 1, into a feasible solution while increasing its cost by a constant factor.

The proposed heuristic, called hereafter 2MST, first constructs a minimal spanning tree (MST) on the set of nodes N^0, which is obviously a relaxation of the TSPDB. In the second step we take two copies of the MST. We will show below that it is possible to transform the two copies of the MST into a feasible tour without additional cost. A similar heuristic for the TSP is well known, however in the TSPDB the feasibility issue in terms of the vehicle's capacity constraint needs to be considered as well. We first present the algorithm, and then explain its various steps.

Algorithm 3.1 (Heuristic 2MST)

Step 0. Let TOUR be an array of length N where TOUR(l) denotes the lth location served by the vehicle after the vehicle leaves the warehouse. Set $l := 1$. Also let $I(v)$ be an indicator variable for $v \in B$ assuming the value 1 if location v was already served and 0 otherwise Set $I(v) := 0$ for all $v \in B$.

Step 1. Construct a minimum spanning tree on N^0 by using the Greedy algorithm. Let MST denote the directed tree obtained, rooted at 0, where all arcs are directed towards the leaves. For each $i \in N$ let $C(i)$ represent the set of children of node i, i.e. $C(i) = \{k: i \to k \in \text{MST}\}$. Let also $p(i)$ represent the parent of node i for $i \in N$, i.e. $p(i) = \{k: k \to i \in \text{MST}\}$.

Step 2. scan the nodes of the MST from the leaves to the root as follows: if node i is a leaf then let $e(i) = -d_i$ in case i is a delivery customer, and otherwise let $e(i) = b_i$. For any node j which is not a leaf let $e(j) = b_j - d_j + \sum_{k:j \to k} e(k)$. ($e(0) = N_B - N_D$.) Start at 0 with a vehicle loaded with N_D units and set $i = 0$.

Step 3. Set $C'(i) = C(i)$;
While $C'(i) = \emptyset$ do begin
if $i = 0$ stop.
if $\{i \in B$ and $I(i) = 0\}$ do begin
load b_i units at i; set TOUR(l) := i;
$l \leftarrow l + 1$; $I(i) := 1$; end;
set $i \leftarrow p(i)$; endwhile;
Select node $v \in C'(i)$ so that $e(v) = \min_{k \in C'(i)} e(k)$. Set $C'(i) \leftarrow C'(i) - \{v\}$.
If $v \in D$ do begin
unload d_v units at v; set TOUR(l) := v;
$l \leftarrow l + 1$; end;
If $\{v \in B$ and $C'(v) = \emptyset\}$ do begin
load b_v units at v; set TOUR(l) := v;
$l \leftarrow l + 1$; $I(v) := 1$; end;
$i \leftarrow v$; repeat Step 3.
end of algorithm.

As explained below, at the end of the algorithm array TOUR consists of exactly N different locations. The scanning procedure of the tree as proposed by the algorithm traverses twice through each of its arcs: firstly, from its tail to its head and secondly, from its head to its tail. We note that each node i is visited exactly $1 + |C(i)|$ times by the algorithm where $C(i)$ is the set of children of node i in the MST. Initially, we set $C'(i) = C(i)$. At each visit at i one element of the set $C'(i)$ is deleted. The last visit at i occurs when $C'(i) = \emptyset$. If i is a delivery customer then it is served at the first visit of the algorithm at i and if it is a backhaul customer it is served only at the last visit of the algorithm at i. Thus each node except 0 is traversed by the algorithm $|C(i)|$ times without performing any work. In the array TOUR we keep by the order of their occurrence only those visits where a loading or unloading is taking place.

In view of the above considerations it is easy to see that the algorithm's complexity is determined by the complexity of generating a minimal spanning tree which is $O(N^2)$.

In the next theorem we prove that starting at the warehouse with a vehicle loaded with N_D units and following the tour $0 \to \text{TOUR}(1) \to \text{TOUR}(2) \to \cdots \to \text{TOUR}(l-1) \to \text{TOUR}(l) \to \cdots \to \text{TOUR}(N) \to 0$ where at each location the vehicle is unloaded or loaded according to the customer's requirement produces a feasible tour, i.e. the total load of the vehicle never exceeds the vehicle's capacity nor it falls below 0.

Theorem 1. *The tour produced by Algorithm* 3.1 *(heuristic* 2MST*) is feasible.*

Proof. We will show that the vehicle's load is feasible in each of the $|C(i)| + 1$ epochs the algorithm traverses through node i, $i \in N^0$. Since the vehicle was initially loaded by $N_D \leqslant q$ units which is the total amount unloaded at the customers, its load can never drop below 0. We will first demonstrate that each time the algorithm traverses through 0 the vehicle's load must be feasible. Observe that $e(i)$ represents the net effect on the vehicle's load if node i and all its successors are served. Thus $\sum_{i \in C(0)} e(i) = N_B - N_D$. Suppose that the children of 0 are indexed in non-decreasing order of their $e(\ )$ values, i.e. $e(1) \leqslant e(2) \leqslant \cdots \leqslant e(|C(0)|)$. Assume, for the sake of contradiction, that the first time infeasibility occurs at 0 is when the service of the subtree rooted at k, $1 \leqslant k \leqslant |C(0)|$ is terminated. Infeasibility means that the load at that visit exceeds q, i.e., $N_D + \sum_{i=1}^{k} e(i) > q$. This implies that $\sum_{i=1}^{k} e(i) > q - N_D \geqslant 0$. Therefore, $e(k)$, $e(k+1), \ldots, e(|C(0)|)$ must be strictly positive

implying that $N_B - N_D = \sum_{i=1}^{|C(0)|} e(i) \geqslant \sum_{i=1}^{k} e(i) > q - N_D$ resulting in $N_B > q$ which contradicts our assumption that the problem is feasible.

Suppose now that infeasibility occurs at a certain node of the MST. It is easy to see that according to the algorithm, for any node i there exists an integer $k(i)$, $1 \leqslant k(i) \leqslant |C(i)| + 1$, such that the vehicle's load during the sequence of visits at node i is non-increasing up to the $k(i)$th visit, and non-decreasing from the $(k(i) + 1)$st visit up to the last visit at i. (Note that $k(i) = 1$ $(k(i) = |C(i)| + 1)$ means that all the $e(\ )$ values of the children of i are non-negative (non-positive).) Therefore the vehicle's load reaches its maximum value in the $|C(i)| + 1$ times the algorithm traverses through node i either at the first entrance to i, i.e. on the arc $p(i) \rightarrow i$, or at the last exit from i, i.e. on the arc $i \rightarrow p(i)$. Suppose node i^* is a node satisfying the following two conditions: (1) infeasibility occurs on one of the arcs of the MST connecting node i^* to an adjacent node; (2) i^* is a node with the minimum depth on the MST among all nodes satisfying condition (1). In view of the above discussion infeasibility occurs either on the arc $i^* \rightarrow p(i^*)$ or on the arc $p(i^*) \rightarrow i^*$ implying that $p(i^*)$, whose depth is smaller by one than the depth of i^*, satisfies condition (1) above. In view of condition (2), i^* must be the root node. However according to our earlier argument at the beginning of the proof, infeasibility cannot occur at the root node, completing the proof. □

In the next theorem we prove that the worst-case bound of the proposed heuristic is 2 and that this bound is tight, i.e. there exists an instance for which the ratio V^{2MST}/V^{OPT} is arbitrarily close to 2.

Theorem 2. (a) *The following inequality holds for any given instance of the problem TSPDB*

$$V^{2MST}/V^{OPT} \leqslant 2.$$

(b) *The worst-case ratio in* (a) *is tight.*

Proof. (a) As mentioned above the heuristic 2MST traverses twice through each of the arcs of MST. Array TOUR with 0 the initial and the final node, which is the solution produced by the heuristic, is a subsequence of the sequence of nodes traversed by the algorithm. According to the triangle inequality $V^{2MST} \leqslant 2l(\text{MST})$. Furthermore, $V^{OPT} \geqslant l(\text{MST})$; this is because any traveling salesman tour on N^0 can be transformed into a tree by erasing one of its edges and the length of the shortest spanning tree is at least as short as the resulting tree, yielding the desired bound.

(b) In order to show that the worst-case ratio of 2 is tight it is sufficient to present a specific instance on which this ratio is achieved. We use a simple modification of the example used by Papadimitriou and Steiglitz [6, pp. 415–416] to show that the corresponding heuristic for the TSP has a worst-case ratio of 2. Suppose we are given n delivery customers and n backhaul customers so that $d_i = b_i = 1$ for $i = 1, \ldots, n$. For simplicity we denote the delivery customers by $d_1, d_2, \ldots, d_n$ and the backhaul customers by $b_1, b_2, \ldots, b_n$. Suppose the customers are located as depicted in Fig. 1(a) with all inner points being backhaul customers and all outer points being delivery customers. Number the customers clockwise such that delivery point i and backhaul point i fall on the same polar through the center of the graph. Suppose also that the location of the warehouse coincides with customer d_1. Let the vehicle's capacity $q = n + 1$. The MST is depicted in Fig. 1(a). Without loss of generality, assume that in the MST the warehouse has a single child and moreover $e(d_1) = 0$, $e(b_1) = 1$, $e(d_i) = -1$ for $i \neq 1$; and $e(b_i) = 0$ for $i \neq 1$. Heuristic 2MST produces the graph depicted in Fig. 1(b) where the optimal solution is depicted in Fig. 1(c). As is shown in Papadimitriou and Steiglitz [6], choosing $R = 1$, $c = 1/n^2$ and n arbitrarily large, the ratio between the lengths of these two graphs can be made arbitrarily close to two. □

It is interesting to observe that the algorithm of Christofides [3, 6] for the TSP does not have a simple analog for the TSPDB. Applying the algorithm of Christofides on the set of nodes N^0 may result in a graph which cannot be transformed into a feasible solution. Consider the following example with $q = 2$, two delivery customers d_1, d_2 and two backhaul customers b_1, b_2 with all demand and backhaul sizes equal one. Suppose the MST is such that the warehouse is connected to the two backhaul customers and each of the backhaul customers

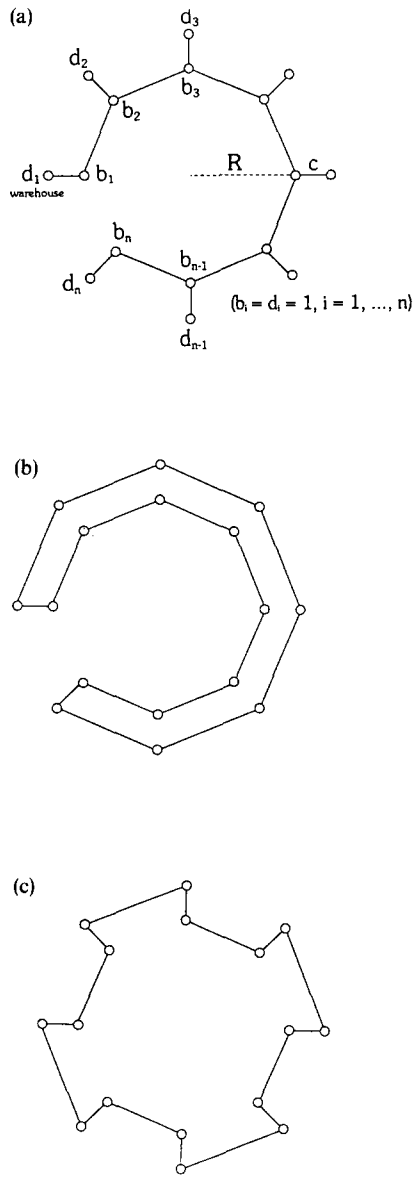

Fig. 1. (a) Minimum spanning tree; (b) tour obtained by heuristic 2MST; (c) optimal tour.

is connected to one delivery customer. According to Christofides algorithm the MST should be completed to an Eulerian graph connecting the two delivery customers one to the other resulting in the cycle $0 \to b_1 \to d_1 \to d_2 \to b_2 \to 0$. As can be seen neither of the cycle's directions is feasible.

4. Numerical test

In this section we compare the 2MST heuristic with the heuristic proposed in Mosheiov [5]. As mentioned in the Introduction, any closed tour on the set N contains a certain arc and a specific direction such that the tour can be made a feasible solution by disconnecting that arc and connecting its two endpoints to the warehouse. Thus, if the original tour is constructed by a TSP heuristic having a worst-case bound of $1 + \alpha$ then the generated solution will be within $100(2 + \alpha)\%$ of the optimal solution. In this numerical test we use the "cheapest insertion" algorithm for the TSP to construct the initial tour. We call this procedure "cheapest insertion with delivery and backhaul" or shortly CIDB. The worst-case bound of the "cheapest insertion" heuristic for the TSP is known [7] to be 2, therefore, the worst-case bound of CIDB for the TSPDB is 3. Mosheiov [5] shows that the average relative error of CIDB is bounded by 10% on randomly generated problems with up to 25 customers. He compares the cost of CIDB with the optimal traveling salesman tour on the set of customers and the warehouse.

Four sets of 10 problems were generated and solved by both heuristics. All programs were coded in PASCAL on an AT-386. In all problem sets the total delivery size equals the total backhaul size and both are equal to the vehicle's capacity. Although, this is not required by any of the heuristics tested, it represents the case where the capacity constraint is the tightest possible. The sets differ in the number of customers N and the range $[1, U]$ from which the delivery/backhaul sizes were generated. After fixing the number of customers N, their locations were generated according to a uniform

Table 1
Average performance of heuristics 2MST vs. CIDB

Set number	N	U	V^{2MST}/V^{CIDB}
1	10	10	1.038
2	30	20	1.080
3	50	20	1.072
4	100	50	1.046

distribution on a square of size 500×500. The warehouse location was set to be the center of the square. All customers, except the last one, were scanned one by one to determine their type (delivery or backhaul) and their requirement size: if the up to date absolute difference between the total delivery and backhaul sizes exceeds U then the next customer type was selected to minimize this difference. Otherwise the current customer was set to a delivery (backhaul) one with probability 0.5. The requirement size of each customer except the last one was uniformly generated on $[1, U]$. The type

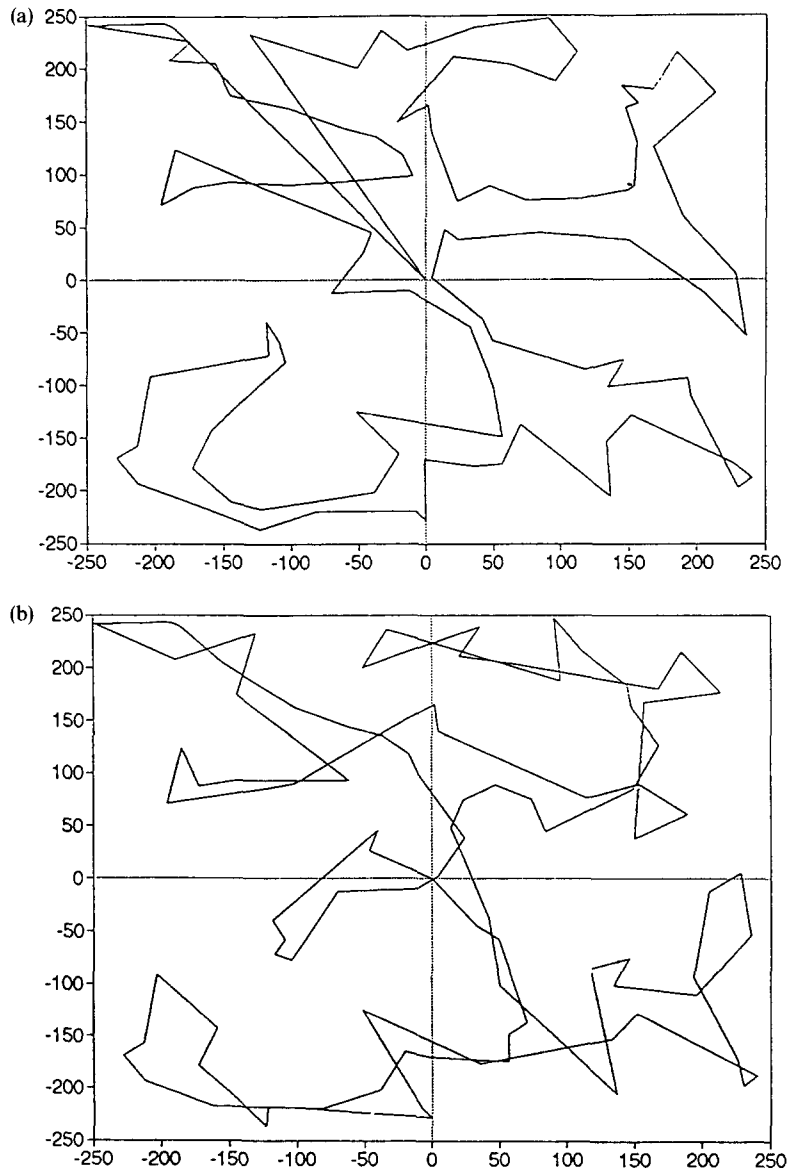

Fig. 2. (a) Tour obtained by CIDB: $N = 100$, $U = 50$, total length = 5108.2; (b) tour obtained by 2MST: $N = 100$, $U = 50$, total length = 5276.8.

and requirement size of the last customer were determined to ensure that $N_B = N_D$.

In all four sets CIDB performed somewhat better than 2MST on average, see Table 1. Note that 2MST is of complexity $O(N^2)$, where CIDB has complexity of $O(N^3)$. Indeed, our computer study indicates that the running time of 2MST is substantially lower in all sets (in set 4, e.g., 2MST requires on average about 20% of the running time of CIDB).

In Figs. 2(a) and (b) we demonstrate the actual tours obtained by each of these two heuristics on a problem from set 4. The tour produced by CIDB is shorter by 1.95% than the one produced by 2MST.

References

[1] L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and scheduling of vehicles and crews – state of the Art" *Comput. Oper. Res.* **10**(2) 63–211 (1983).

[2] D.O. Casco, B.L. Golden and E.A. Wasil, "Vehicle routing with backhauls: Models, algorithms, and case studies", B.L. Golden and A.A. Assad (eds.), *Studies in Management Science and Systems/Vehicle Routing: Methods and Studies*, North Holland, Amsterdam, 1988, 127–147.

[3] N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem", Technical Report, GSIA, Carnegie-Mellon University, 1976.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP Completeness*, Freeman, San Francisco, 1979.

[5] G. Mosheiov, "The traveling salesman problem with pick-up and delivery", *Eur. J. Oper. Res.* **79**(2) (1994).

[6] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[7] D.J. Rosenkrantz, R.E. Stearns and P.M. Lewis II, "An analysis of several heuristics for the traveling salesman problem", *SIAM J. Comput.* **6**, 563–581 (1977).