The Swapping Problem

S. Anily
Faculty of Management
Tel-Aviv University
Tel-Aviv 69978
Israel

R. Hassin
School of Mathematical Sciences
Tel-Aviv University
Tel-Aviv 69978
Israel

Each vertex of a graph initially may contain an object of a known type. A final state, specifying the type of object desired at each vertex, is also given. A single vehicle of unit capacity is available for shipping objects among the vertices. The swapping problem is to compute a shortest route such that a vehicle can accomplish the rearrangement of the objects while following this route. We exhibit several structural properties of shortest routes and develop polynomial approximation algorithms that are variations of a well-known "patching" algorithm for the traveling salesman problem. We prove tight constant performance guarantees for these algorithms and note as a side product that these bounds hold and are tight also for the latter problem.

1. INTRODUCTION

Let $G = (V, E)$ be a (complete undirected) graph with a *length function* $c: E \rightarrow R_+$. Let $S = \{1, \ldots, m\}$ be a set of "object types." An object of type j is called a *j-object*. Each vertex $v \in V$ is associated with two numbers a_v and b_v, where the first denotes the type of the object currently at v while the latter is the desired final type of object at v. If a vertex v is initially empty, we set $a_v = 0$ and say that v contains a 0-object. Similarly, if v is empty in the desired final state, then $b_v = 0$. We assume that for each object type j, the number of vertices initially containing a j-object equals the number of vertices where such an object is desired.

The objects are shipped by a single vehicle of unit capacity starting and terminating its route at a given vertex (the depot). The theorems and algorithms

NETWORKS, Vol. 22 (1992) 419–433

CCC 0028-3045/92/040419-15$04.00

presented below can be adapted, with minor modifications, to the case where the departure and destination points are distinct (see Hoogeveen [6] about possible implications of this assumption).

The set of objects S is partitioned into two subsets: $S = S_d \cup S_n$. The objects in S_d can be temporarily stored at intermediate vertices on the vehicle's route. Thus, the vehicle may unload such an object in order to reload it later. Such an action is called a *drop*. For objects in S_n, no drops are allowed. In many practical situations, one of the sets S_d and S_n may be empty (see, e.g., [1]). The above presentation generalizes the two subcases and also allows unification of some of the results.

A *walk* is a sequence of (not necessarily distinct) edges $W = (v_1, v_2)$, $(v_2, v_3), \ldots, (v_k, v_{k+1})$, where $v_1 = v_{k+1}$. In contrast to the common definition of a cycle, the starting point of a walk, v_1, is a distinguished vertex of the walk (corresponding to the depot). A feasible solution is called a *route*. It consists of a walk and an assignment of exactly one object type to each of the walk's edges. This assignment must be such that the object of the prescribed type is available at v_i while the vehicle starts moving along (v_i, v_{i+1}), and the final state obtained conforms with the desired final state. A walk may use an edge of G more than once. However, when we refer to an edge $(v_i, v_{i+1}) \in W$, we specifically mean the use of this edge as the i-th edge of W. Similarly, each vertex v_i of W corresponds to a vertex of $v \in V$, but we specifically refer to v_i as the visit of v made in between the $i-1$-st and i-th edges of W.

The *length of a route* is the sum of lengths of its edges. The *swapping problem* is to compute an *optimal* route, i.e., a route of minimum length. The *simple swapping problem* is a special case with two types of objects. In this case, $V = U \cup W$, where $|U| = |W|$, $a_v = 1$ $b_v = 2$ for all $v \in U$, and $a_v = 2$ $b_v = 1$ for all $v \in W$. For simplicity, we have assumed that each vertex contains, both prior to the swapping and after it, at most one object. The results of this paper can be easily extended to the more general case by considering duplicates of vertices at zero distance.

A simple example may serve to illustrate that drops may be used to improve the solution.

Example 1.1 *Let* $V = \{1, 2, 3, 4\}$. *Suppose that the vertices lie in the plane as corners of a unit square in the cyclic order* 1, 2, 3, 4. *Let* $a_1 = b_2 = 1$, $a_2 = b_1 = 2$, $a_3 = b_4 = 3$, *and* $a_4 = b_3 = 4$. *Thus, the problem requires two swaps: one between vertices* 1 *and* 2; *the other between* 3 *and* 4. *To form a route, these swaps must be connected. Suppose the depot is at vertex* 1. *A shortest route without drops is* $1 - 2 - 1 - 4 - 3 - 4 - 1$ *and its total length is* 6. *A better solution exists if we allow drops; starting at vertex* 1, *the* 1*-object is shipped to* 4 *and dropped there, then the objects at vertices* 3 *and* 4 *are swapped, the* 1*-object is reloaded and shipped to its destination at vertex* 2, *and the route is completed by shipping the* 2*-object from* 2 *to* 1. *The resulting route is* $1 - 4 - 3 - 4 - 2 - 1$ *has a total length of* $4 + \sqrt{2}$.

We find it helpful to employ the terminology that refers to the underlying

"physical" problem. Thus, we may say that a route "passes" through a vertex or that the vehicle "loads" or "ships" an object, etc.

Throughout this paper, we assume that the length function c obeys the triangle inequality:

Assumption 1.2. *For every triplet of distinct vertices* $u, v, w \in V$,

$$c_{uv} + c_{vw} \geq c_{uw}.$$

Remark 1.3. *Even the simple swapping problem is NP-hard. To verify this observation, consider an instance of the traveling salesman problem (TSP). Replace each vertex* v *by a pair of vertices* v' *and* v'' $a_{v''} = b_{v'} = 1$, $a_{v'} = b_{v''} = 2$ *and* $c_{v',v''} = 0$. *For each pair* u, v, *set the edge lengths* $c_{v',u'} = c_{v',u''} = c_{v'',u'} = c_{v'',u''}$ *equal to* c_{vu}. *Clearly, an optimal route for this swapping problem is also an optimal tour for the TSP instance.*

In this paper, we characterize the structure of optimal solutions for the swapping problem and develop polynomial approximation algorithms with bounded worst-case performance ratio. This ratio is defined as the supremum over all problem instances of the ratio of the approximate solution's length to that of the optimal one. The algorithms are variations of a well-known "patching algorithm" developed by Gilmore and Gomory [5] for the TSP. We prove that the bounds are tight, i.e., they can (asymptotically) be achieved by certain instances. A side product is a proof that the same bounds are also tight for the associated algorithms for the TSP.

A related problem is the stacker crane problem, for which polynomial approximations with bounded performance guarantees were developed by Frederickson et al. [4] (see also [1, 3, 7]. In this special case, there are unique vertices v and u with $a_v = j$ and $b_u = j$, respectively, for each $j = 1, \ldots, m$, and drops are not allowed. Consequently, G contains m directed *special edges* that must be traversed. The algorithm of Frederickson et al. [4] produces a performance guarantee of at most 9/5. In obtaining this bound, it is assumed that the length of the set of edges that are required to be used by any solution (and therefore is constant) is part of the solution's value; different solution strategies are taken depending on whether this constant is "large" or "small." We note that in the generalization considered in the present paper the set of edges where the vehicle is loaded is not fixed but must be computed as part of the solution.

Another related problem is the TSP with pickup and delivery, or the dial-a-ride problem. Also, there, each object-type has a single origin–destination pair, but usually it is assumed that the vehicle's capacity is infinite (see [8] for a comparative study of heuristics). For this variation, a bounded approximation is straightforward: Approximate the shortest tour on the set of all vertices and traverse it twice. The first time, pick all the objects, and the second, drop them at their destination (the heuristic of [9] is in this spirit). As we will see, our problem is considerably different and has both further difficulties and an interesting structure of the solutions that do not exist in the dial-a-ride problem.

2. DROPS AND DEADHEADINGS

The current and desired states can be described by $0-1$ matrices A and B of dimension $|V| \times (m+1)$, where $A_{vj} = 1$ if and only if $a_v = j$, and $B_{vj} = 1$ if and only if $b_v = j$. We assume that $\Sigma_{v \in V} A_{vj} = \Sigma_{v \in V} B_{vj}$ for all $j \in S$, so that the total number of objects of each type is preserved.

Consider a walk with edges $e_i = (v_i, v_{i+1})$ $i = 1, \ldots, k$, $(v_{k+1} = v_1)$. Let $x_{ij} = 1$ if a j-object is assigned to e_i, and $x_{ij} = 0$ otherwise. Clearly, x must satisfy

$$x_{i0} + \sum_{j \in S} x_{ij} = 1 \qquad i = 1, \ldots, k, \tag{2.1}$$

and

$$\sum_{i|v_i = v} x_{ij} - \sum_{i|v_{i+1} = v} x_{ij} = A_{vj} - B_{vj} \qquad j = 1, \ldots, m \qquad v \in V. \tag{2.2}$$

(2.1) requires that each edge of the walk will be assigned an object, possibly a 0-object (i.e., a deadheading), while (2.2) takes care of the conservation of objects at the vertices. The following example demonstrates that these are not sufficient conditions for feasibility.

Example 2.1. *Consider the instance described in Figure* 1 *with a walk,* W, *visiting the vertices in the following order*: (1, 2, 5, 2, 6, 3, 2, 4, 1). *Assign objects to the edges of* W *as indicated by the figure. This assignment satisfies* (2.1) *and* (2.2). *However, edge* (2, 6) *is traversed when the* 3*-object supposed to be transferred by it is not available at vertex* 2.

Summing (2.2) over $j = 1, \ldots, m$, and using (2.1) and $\Sigma_{S \cup \{0\}} A_{vj} = \Sigma_{S \cup \{0\}} B_{vj} = 1$ for all $v \in V$, we obtain

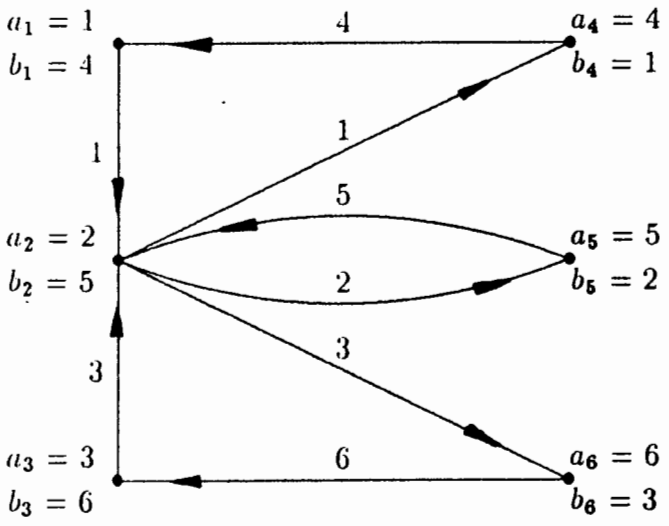

FIG. 1.

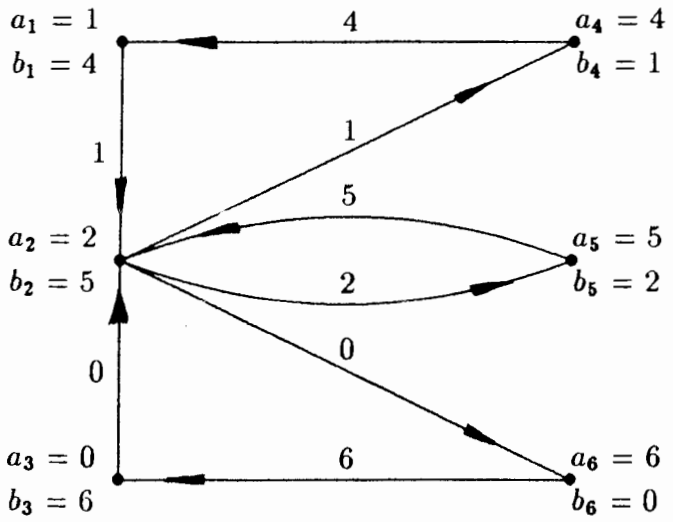

FIG. 2.

$$\sum_{i|v_i=v} x_{i0} - \sum_{i|v_{i+1}=v} x_{i0} = A_{v0} - B_{v0}. \tag{2.3}$$

Hence, (2.2) holds also for 0-objects. However, 0-objects must be distinguished from other object types. This fact is illustrated by the following example:

Example 2.2. *Consider the instance described by Figure* 2. *It resembles the one considered in Example* 2.1, *but* a_3 *and* b_6 *are now* 0. *As a result of this change, edge* (2, 6) *can be traversed before* (3, 2), *and there exists a route conforming with the assignment of object types as indicated in the figure. One such route consists of the following sequence of vertices*: 1, 2, 5, 2, 6, 3, 2, 4, 1. *An alternative sequence is* 1, 2, 6, 3, 2, 5, 2, 4, 1. *Note the drop of the* 1*-object at vertex* 2.

From Assumption 1.2, it follows that in a pure no-drop case there exists an optimal route that does not pass through a vertex v such that $a_v = b_v$. This, however, is not necessarily true when drops are allowed as shown by the following example:

Example 2.3. *Consider the instance described by Figure* 3 *where the square is a unit square in the plane. An optimal route may start at* 1, *drop the* 1*-object at* 5, *continue to* 3 *and* 4, *return to* 5 *to reload the* 1*-object, and then return to* 1 *through* 2. *The total distance traveled is* $2 + 2\sqrt{2}$. *Without dropping at vertex* 5, *a shortest route is* 1 − 2 − 1 − 3 − 4 − 1 *and its length is* $4 + \sqrt{2}$.

We note that vertices v with $a_v = b_v$ used by a route resemble "Steiner points" used in constructing a minimum length connected subgraph that spans a given set of points.

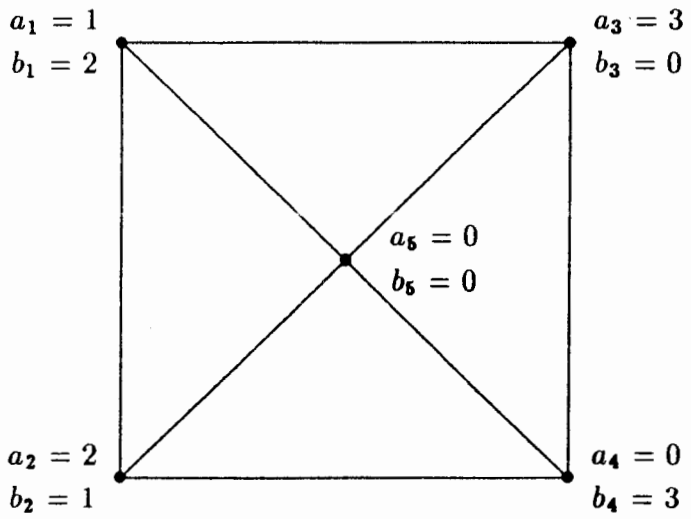

FIG. 3.

Definition 2.4. *A segment of the solution that the vehicle traverses unloaded is called a deadheading.*

By (2.3), if V contains a vertex v with $a_v = 0$, then a route includes a subset of edges corresponding to a path of deadheadings originating at v and ending at a vertex u with $b_u = 0$. Note that this subset does not necessarily correspond to a consecutive sequence of the route's edges as a result of drops. Moreover, the order of the edges in the path may be different from the corresponding order in the route. As an example, consider again Example 2.2. There, a path of deadheadings consists of edges (3, 2) and (2, 6) that are traversed in opposite order by each of the routes described there.

As demonstrated by Example 1.1 for the no-drop case, additional deadheadings are possible in optimal routes, even when no 0-objects exist. From (2.3), these additional deadheadings form a set of cycles. In Theorem 2.9 below, we will show that such cycles of deadheadings are possible only if $S_n \neq \phi$.

When the condition of Assumption 1.2 holds with equality for some triplets of vertices, multiple optimal routes may exist since single edges may be replaced by pairs of edges of the same total length. We show below that optimal routes with minimum number of edges have interesting properties. Therefore, we define:

Definition 2.5. *A minimum cardinality optimal route is an optimal route having the minimum number of edges among all optimal routes.*

By Assumption 1.2, minimum cardinality optimal routes do not include consecutive edges carrying the same object type. This is also true with respect to 0-objects, though it is possible that the first and last edges of every optimal route are deadheadings.

Each of the two routes mentioned in Example 2.2 includes six edges incident with vertex 2. Of these edges, two correspond to the drop of the 1-object, two are deadheadings, and two correspond to the swap between vertices 2 and 5. Theorem 2.7 below shows that this is the maximum number of edges incident with any vertex in a minimum cardinality optimal route. To prove the theorem, we first need the following lemma:

Lemma 2.6. *Consider a minimum cardinality optimal route* $R = (v_1, v_2), \ldots, (v_k, v_{k+1})$. *Suppose that* $v_i = v_j = v_r = v$ *for some* $v \in V$ *and* $1 \leq i < j < r \leq k + 1$. *Moreover, suppose that* $v_l \neq v$ $l = i + 1, \ldots, j - 1$. *Then, there exists an alternative route* $R' = (v'_1, v'_2), \ldots, (v'_k, v'_{k+1})$ *using exactly the same edges and the same assignment of objects to edges as* R, *such that* $v_q = v'_q$ *for* $q = 1, \ldots i$ *and for* $q = r, \ldots, k + 1$, *and* $v'_{i+1} = v_{j+1}$.

Proof. Consider the directed multigraph MG corresponding to the subroute of R from v_i to v_r. MG is Eulerian, i.e., the outdegree and indegree are equal at each vertex. We construct R' as follows: The first edges of R' are $(v_1, v_2), \ldots, (v_{i-1}, v_i)$. Then, we pick (v, v_{j+1}) as the next edge of R'. This is possible since in a minimum cardinality optimal route the object assigned to (v_j, v_{j+1}) is not the one carried into v by (v_{j-1}, v_j). Delete now (v_j, v_{j+1}) from MG. As a result, v and v_{j+1} are the only vertices in which MG has odd degrees. Next, a new edge leaving v_{j+1} in MG is selected and deleted from MG, and the process continues until finally v is reached.

R may define some precedence relations on its edges; specifically, an edge reloading a dropped object cannot be used before the object is unloaded. These relations are imposed on disjoint pairs of edges incident to a common vertex. Therefore, at any time of the selection process, after using an edge of R going into a vertex, there must be at least one edge of MG leaving the current vertex, which can be selected without violating the precedence relations.

As long as MG still contains edges, R' can be augmented as follows: Let (u, w) be the first edge of R that is still in MG. Clearly, u is on R', and a predecessor of u, if R had such one, has already been used by R'. (u, w) can now be appended to R' and the above process be applied till u is reached again. The above process is completed when MG has no edges, and then R' continues exactly as R. ■

Theorem 2.7. *There exists an optimal route* R *in which every vertex* v *satisfies the following: (i)* v *is incident with at most one ingoing edge carrying the* b_v*-object, one outgoing edge carrying the* a_v*-object, two edges associated with a single drop (one entering* v, *the other leaving it), and at most one additional pair of deadheadings (one entering* v, *the other leaving it). (ii) If there is a drop at* v, *then* $v_1 \neq v$. *Moreover in this case, the drop is associated with the first entrance to* v *and the last exit from it, and the edges of* R *associated with* a_v *and* b_v *are used consecutively (among the edges incident with* v).

Proof. Consider a minimum cardinality optimal route R.

Suppose there is a drop at v such that an object is carried into v by an edge e and later carried out by an edge e'. We claim that this is possible only if e and e' are the first incoming and the last outgoing edges incident with v, respectively. Otherwise, if e' is not the last edge on R incident, then by Lemma 2.6, there exits another optimal route in which e and e' are consecutive. This contradicts the minimality of R. If, on the other hand, e' is the last exit from v but e is not the first edge entering v, then v is incident to at least six edges. Therefore, there is at least one pair of incoming and outgoing deadheadings where the outgoing one is not the last exit from v. Again, Lemma 2.6 can be used to obtain an alternative optimal route with two consecutive deadheadings, contradicting the minimality of R. An implication of this observation is that at most one drop at v is possible.

Suppose next that there are two incoming deadheadings into v, e_i and e_j, $i < j$. Each of them is followed by an outgoing edge e_{i+1} and e_{j+1}, respectively. By the minimality of R, neither of e_{i+1} and e_{j+1} is a deadheading. Therefore, one of these outgoing edges carries $a_v \neq 0$, and the other carries an object previously dropped at v. By the previous paragraph, there is at most one outgoing edge reloading a dropped item from v, and this edge is the last exit from v. Moreover, the object was dropped at v by the first edge, e_l $l < i$ entering v. Thus, e_{i+1} carries a_v, e_{j+1} carries the dropped item and is the last exit from v, and e_l is followed by an outgoing deadheading e_{l+1}. By Lemma 2.6, there exists an alternative optimal route R' where e_{i+1} immediately follows e_l. R' has two consecutive deadheadings, contradicting the assumption that R has a minimum number of edges.

To complete the proof, note that if $v = v_1$ and there is a drop at v; then by Lemma 2.6, an alternative optimal route exists in which there are two consecutive edges carrying the dropped item, contradicting the minimality assumption on R.

Corollary 2.8. *There exists an optimal route with no more than* $3|V|$ *edges. If* $S = S_n$, *then there exists an optimal route with no more than* $2|V|$ *edges.*

Proof. From Theorem 2.7, there exists an optimal route with at most six edges incident to each vertex, while if $S = S_n$, there are at most four edges incident to each vertex. Summing over all vertices and noting that each edge is counted twice, we obtain the above bounds.

Theorem 2.9. *If* $S = S_d$, *then there exists an optimal route without cycles of deadheadings (except for one possible cycle containing the depot* v_1 *when* $a_{v_1} = b_{v_1} = 0$).

Proof. Consider a minimum cardinality optimal route R. Let C be a cycle of deadheadings. Let (v_i, v_{i+1}) be the first edge of C used by R.

Case 1. Suppose that $i \neq 1$. Since R is of minimum cardinality, (v_{i-1}, v_i) is not a deadheading. Suppose it carries an r-object, $r > 0$. An alternative optimal route can be constructed by replacing both (v_{i-1}, v_i) and $(v_i,$

v_{i+1}) by an edge (v, u) with $v = v_{i-1}$ and $u = v_{i+1}$. The r-object will be assigned to this new edge and then shipped along the edges of C, with drops at each of its vertices, till it is brought to v_i. The rest of the route is not changed. Clearly, the new route is feasible and not longer than R. Therefore, it is optimal. However, its cardinality is smaller than that of R, contradicting the assumption that R is of minimum cardinality.

Case 2. Suppose that $i = 1$ and that at least one of a_{v_1} and b_{v_1} is not 0. We will prove the case $a_{v_1} = 0$; the other follows by symmetry. Let (v_j, v_{j+1}) be the edge carrying the a_{v_1}-object out of $v_1 = v_j$. By Theorem 2.7, there is no drop at v_1 and no other outgoing edge from v_1 in R. By Lemma 2.6, there is an optimal route R' using the same edge set as R but where (v_1, v_{j+1}) precedes (v_1, v_2). Clearly, R' is also of minimum cardinality. However, Case 1 applies to R' and the same contradiction obtains.

Corollary 2.10. *If $S = S_d$ and $a_v \neq 0$ $\forall v \in V$, then there exists an optimal route with no more than $2|V|$ edges.*

Proof. In view of Theorem 2.9 and (2.3), there exists an optimal route containing no deadheadings. Thus, by Theorem 2.7, each vertex is incident to at most four edges.

3. POLYNOMIAL APPROXIMATIONS

In this section, we describe polynomial approximation algorithms with bounded performance guarantees for the swapping problem. We follow a common procedure for generating such approximations: First, a relaxation of the problem is solved and then it is extended to a feasible solution while increasing its cost by a constant factor. The relaxation consists of a set of assignment problems, and these are then "patched" to yield a route. Thus, the approach resembles that of Gilmore and Gomory [5] for the TSP.

Our approximations use the existence of a bounded approximation for the symmetric TSP. The best known bound is 1.5, obtained by Christofides' algorithm (see e.g., [8]).

Let *OPT* denote the length of an optimal route. We first establish a polynomially computable lower bound on *OPT:*

Algorithm 3.1.

For $j = 0, \ldots, m$:

Set $A_j = \{v \in V | a_v = j, b_v \neq j\}$, $B_j = \{v \in V | b_v = j, a_v \neq j\}$.

[Note that A_j and B_j are disjoint for all j.]

Set the cost of assigning $v \in A_j$ to $u \in B_j$ to c_{vu}, the length of (v, u). [Note, by Assumption 1.2, that this is the length of a shortest $v - u$ path.]

Solve a minimum cost assignment problem on a complete bipartite graph with vertex bipartition (A_j, B_j).

Let E_j denote the set of edges in the optimal solution obtained by Algorithm 3.1 for the j-th assignment problem. Let $c(E_j)$ denote the total cost of E_j.

Theorem 3.2. $\Sigma_j\, c(E_j) \leq OPT$.

Proof. If an object is shipped from its origin v to its destination u via a sequence of drops, then in view of Assumption 1.2, the total length of this part of the route is greater than or equal to c_{vu}. Therefore, we conclude that the part of the route in which the vehicle is loaded with j-objects is at least of length $c(E_j)$. Summation over j yields the claimed inequality. ■

The set of edges $\cup_j E_j$ contains exactly two edges incident with each vertex $v \in V$ in which $a_v \neq b_v$, one from E_{a_v} and one from E_{b_v}. Thus, it consists of a set of disjoint simple cycles without loops.

Algorithm 3.3.

3.3.1. *Apply Algorithm* 3.1 *to compute* $E_j j = 0 \ldots, m$. *Let* $C_1, \ldots, C_k$ *be the cycles composing* $\cup_j E_j$.

3.3.2. Let $U = \{u_1, \ldots, u_k\}$ be a set of arbitrary representative vertices from $C_1, \ldots, C_k$, such that $u_i \in C_i i = 1, \ldots, k$. Suppose the depot is v_1. If $a_{v_1} = b_{v_1}$, then add v_1 to U. else, v_1 must be chosen to represent the cycle containing it. Apply a TSP approximation algorithm to the subgraph induced by U, to obtain a tour T on these vertices.

3.3.3. Construct a route in G by superposing the edges in $\cup_i C_i$ and T.

3.3.4. If possible, improve the solution by replacing consecutive edges carrying the same object type by single edges and applying drops whenever there are cycles of deadheadings (as in the proof of Theorem 2.9).

Theorem 3.4. *Algorithm* 3.3 *produces a feasible route.*

Proof. We must show that the edges of $(\cup_i C_i) \cup T$ can be ordered to form a route. Such an order is the following one: Start at the depot. Follow T by a deadheading to the next representative point. Then, follow the cycle represented by it. After returning to this representative, follow the next edge of T, and so on. Clearly, the process terminates at the depot after all the required swaps were completed. ■

Let APX denote the length of the route produced by Algorithm 3.3.

Theorem 3.5. $APX \leq 2.5\ OPT$.

Proof. Every feasible route must visit the depot and each vertex of $\cup_j (A_j \cup B_j)$ at least once. Therefore, by Assumption 1.2, the optimal solution to the TSP

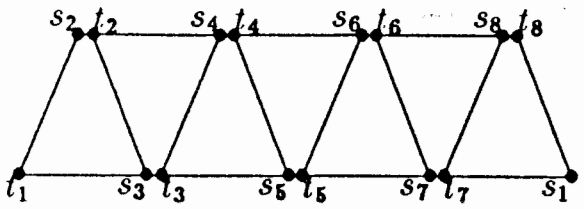

FIG. 4.

defined on the representative vertices constitutes a lower bound on the optimal solution value. Christofides' algorithm generates in Step 3.3.2 a tour of at most 1.5 the length of a shortest tour. Combining this with Theorem 3.2, we obtain the claimed result.

Theorem 3.6. *The ratio APX/OPT can be made arbitrarily close to* 2.5 *even for the simple swapping problem.*

Proof. We prove the theorem by constructing a family of problems where the ratio *APX/OPT* tends to 2.5. The construction is inspired by a similar one by Cornuejols and Nemhauser [2]. Let k be an even positive integer. Consider $n = 2k$ points $s_1, \ldots, s_k, t_1, \ldots, t_k$ located in the plane on the two parallel lines $y = 0$ and $y = \sqrt{3}/2$ as follows: t_1 is located at (0, 0); s_1 at $(k/2, 0)$; s_i at $[(i-1)/2, 0)$ for odd i, $i > 1$; s_i at $[(i-1)/2, \sqrt{3}/2]$ for even i. For $i > 1$, t_i is located very close to the right of s_i. For computational purposes, we assume the distance between s_i and t_i $i > 1$ to be zero. The layout of the points is illustrated in Figure 4. Note that the segments (t_i, s_{i+1}) $i = 1, \ldots, k-1$, are of unit length.

Let G be the (complete, undirected,) graph with $n = 2k$ vertices $s_1, \ldots, s_n$, $t_1, \ldots, t_n$, where the costs c_{uv} represent the Euclidean distance between u and v. We assume $a_v = 1$, $b_v = 2$ for $v = s_1, \ldots, s_k$, and $a_v = 2$, $b_v = 1$ for $v = t_1, \ldots, t_k$. Thus, this is a simple swapping problem. Suppose that s_1 is the depot. The optimal route is $s_1 - t_{k-1} - s_{k-1} \ldots t_1 - s_2 - t_2 - s_4 - t_4 \ldots s_k - t_k - s_1$, whose length is $k + 1$.

A minimum assignment consists of two copies of the edges (s_i, t_i) $i = 1, \ldots, k$, and its length is $2c_{s_1t_1} = k$. It has k disjoint simple cycles. Choosing $s_1, \ldots, s_k$ to represent these cycles, we approximate a shortest tour on these vertices using Christofides' algorithm. Choosing as a minimum spanning tree the path $s_2 - s_3 - s_4 \ldots - s_k - s_1$ (of length $k-1$), and completing it to a tour by matching s_1 with s_2 (by the edge of cost $c_{s_1s_2} > k/2 - 1$), we obtain a tour of length greater than $(3k)/2 - 2$. Superimposing the tour with the minimum assignment yields a route of length $(5k)/2 - 2$ (Fig. 5).

Finally, in Step 3.3.4, we replace paths $s_1 - s_{i+1} - t_{i+1}$ $i = 1, \ldots, k-1$ by edges (s_i, t_{i+1}), and $s_k - s_1 - t_1$ by (s_k, t_1) (Fig. 6). The resulting solution is of length greater than $5(k/2) - 4$. As k tends to infinity, the ratio *APX/OPT* approaches 2.5.

We now discuss a variation of Algorithm 3.3 where instead of arbitrarily selecting representative vertices from $C_1, \ldots, C_k$, as was done by Step 3.3.2, we let the algorithm select a potentially good set of representatives.

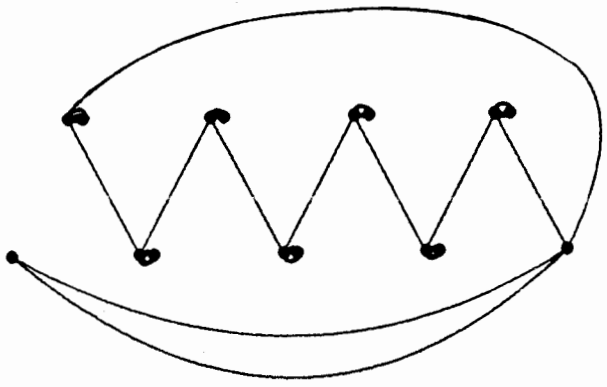

FIG. 5.

The idea is to define an auxiliary graph G' with k vertices each representing one of the cycles $C_1, \ldots, C_k$ produced by 3.3.1. (If $a = b$ for the depot, add an additional vertex representing it.) The edge weights in G' are equal to the length of a shortest edge in G between any pair of vertices in the corresponding cycles. Then, one may proceed by superposing an approximation to the TSP solution in G' and the set of cycles in G. This, however, does not necessarily define a feasible route, since edges that are adjacent in G' do not necessarily correspond to adjacent edges of G. This problem can be amended by performing the matching step of Christofides algorithm on the correct set of odd vertices:

Algorithm 3.7.

3.7.1. *Solve the assignment relaxation* $\cup_j E_j$ *of the problem, as in* 3.3.1. Let $C_1, \ldots, C_k$ be the cycles composing $\cup_j E_j$.

3.7.2. Let G' be an undirected graph with vertex set $V' = \{1, \ldots, k\}$ corresponding to $C_1, \ldots, C_k$, respectively. (Add a vertex corresponding to the depot v_1 if $a_{v_1} = b_{v_1}$). For $i, j \in V'$, let $c'_{ij} = \min \{c_{uv} | u \in C_i, v \in C_j\}$. Compute a spanning tree of minimum weight with respect to the costs

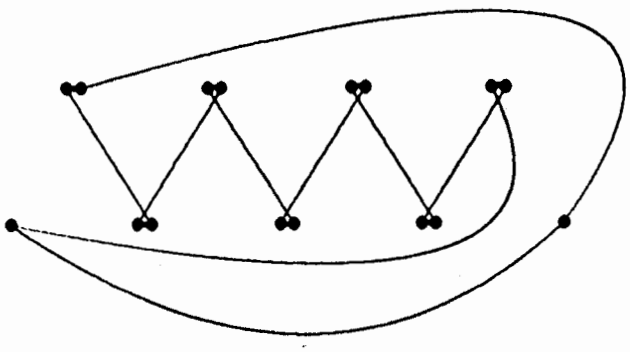

FIG. 6.

c', in G'. Let T denote the set of edges in G associated with this spanning tree.

3.7.3. Consider the subgraph consisting of the edges in $\cup_j E_j$ and T. Let R be the set of vertices with odd degree in it. Let G'' be a complete graph induced by R with edge costs as in G. Compute a minimum cost perfect matching, M, in G''. Superimpose the edges in $\cup_j E_j$, T, and M.

3.7.4. If possible, improve the solution by replacing directed paths by single edges and deadheadings by drops while preserving feasibility.

Theorem 3.8. *Algorithm* 3.7 *produces a feasible route. Let APX*1 *denote the length of this route. Then APX*1 $\leq$ 2.5 *OPT. Moreover, the ratio APX*1/*OPT can be made arbitrarily close to* 2.5 *even for the simple swapping problem.*

Proof. As for Theorems 3.4, 3.5, and 3.6. ■

We note that the performance guarantees of Algorithms 3.3 and 3.7 are identical. However, 3.7 may be more efficient in practice because it does not select the representatives arbitrarily but in a way that minimizes the length of the spanning tree needed to patch the solution.

Remark 3.9. *It is well known that one can obtain an approximation for the TSP with a performance ratio of at most* 2 *by superposing two copies of a minimum spanning tree. This ratio is worse than that of Christofides' algorithm, but it can be computed with lower complexity. In Algorithms* 3.3 *and* 3.7, *the gain in complexity is not significant since the algorithm computes also a minimum cost assignment. However, the resulting performance guarantee, if we use two copies of minimum spanning trees, is easily shown to be bounded by* 3, *and the following example proves that this bound is tight.*

Example 3.10. *Consider again the example used in the proof of Theorem* 3.6 (*see Fig.* 4). *An optimal solution to the assignment relaxation of the problem uses* (*twice*) *each of the* (s_i, t_i) *edges, and its length is* $2C_{s_1t_1} = k$. *A minimum spanning tree consists of the* $k - 1$ *edges of unit length* (t_2, s_3), (t_3, s_4), . . . , (t_{k-1}, s_k), (t_k, s_1). *The edges obtained by superposing two copies of it to the assignment relaxation are shown by Figure* 7. *Finally, we may replace paths* s_i

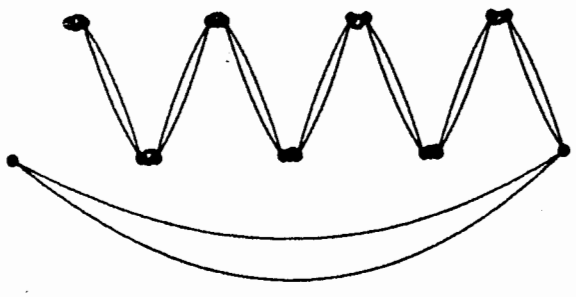

FIG. 7.

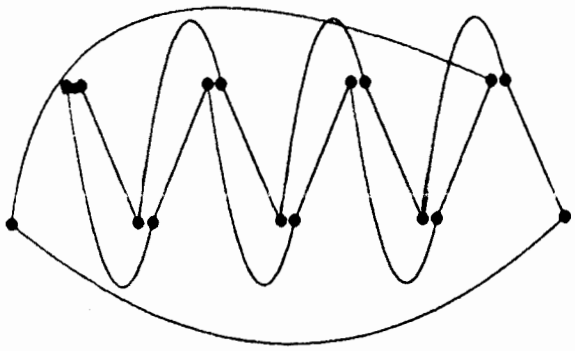

FIG. 8.

$- t_i - s_{i+1} - t_{i+1}$ *by a single edge* $(s_i - t_{i+1})$. *This is done for* $i = 2, \ldots, k-1$, *and the path* $s_k - t_k - s_1 - t_1$ *is replaced by the edge* (s_k, t_1). *The result is a tour of length* $APX = 3k - 4$. *As* k *tends to infinity, the ratio* APX/OPT *approaches* 3. *The final route is illustrated in Figure* 8.

4. FINAL REMARKS

As mentioned in Remark 1.3, the TSP is a special case of the swapping problem. One may ask whether the algorithms discussed above have better worst-case bounds when applied to the TSP. It turns out that the examples shown above demonstrate that the performance guarantee of 2.5 is tight even when applying the corresponding patching approximations on the TSP. The optimal routes for both instances presented in the proof of Theorem 3.6 and Example 3.10 are also the shortest possible tours on the same points. The approximations obtained by the respective algorithms for these instances are tours and they will be obtained also by the corresponding patching algorithms for the TSP.

We made the assumption, represented by Eqs. (2.1), that for each object type the total demand is equal to the total supply. Suppose that this assumption is relaxed, assuming instead that the total supply is at least as large as the total demand. Thus, the desired final states are specified only for a subset of vertices (referred to as "demand vertices"). A bound of 3.5 can be obtained as follows: Solve an assignment problem with inequality constraints for each object type. Duplicate the resulting solutions to obtain a set of cycles. Approximate a shortest tour on the set of demand points (note that every feasible solution must visit these points and therefore the tour is a lower bound on the solution). Combine the tour and cycles to obtain a route, as in Theorem 3.4.

We have mentioned that the phenomenon demonstrated by Example 2.3 resembles that of the well-known Steiner tree problem on a network. The same example can serve to show that drops at points with $a = b = 0$ can reduce the length of an optimal solution also when the problem is defined in the Euclidean plane. It is an interesting research topic checking whether such points can be

characterized in an analogous way to that known for the Steiner problem in the plane.

REFERENCES

[1] M. J. Atallah and S. R. Kosaraju, Minimizing robot arm travel. *SIAM J. Comp.*, **17** (1988) 849–869.
[2] G. Cornuejols and G. L. Nemhauser, Tight bounds for Christofides' traveling salesman heuristic. *Math. Program.* **14** (1978) 116–121.
[3] G. N. Frederickson and D. J. Guan, Ensemble motion planning in trees. *Proc. FOCS* **30** (1989) 66–71.
[4] G. N. Frederickson, M. S. Hecht, and C. E. Kim, Approximation algorithms for some routing problems. *SIAM J. Comput.* **7** (1978) 178–193.
[5] P. C. Gilmore and R. E. Gomory, Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Res.* **12** (1964) 655–679.
[6] J. A. Hoogeveen (1990), Analysis of Christofides' heuristic: Some paths are more difficult than cycles. CWI report BS-R9005, Amsterdam.
[7] D. S. Johnson and C. H. Papadimitriou, Performance guarantees for heuristics. *The Traveling Salesman Problem.* (E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Eds.) Wiley, New York (1985) Chap. 5.
[8] M. Kubo and H. Kasugai, Heuristic algorithms for the single vehicle dial-a-ride problem. *J. Operational Res. Soc. Jpn.* **33** (1990) 354–365.
[9] H. Psarfatis, Analysis of an $O(n^2)$ heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem. *Transportation Sci.* **117B** (1983) 133–145.

Received January 1990
Accepted December 1991