SIAM J. COMPUT.
Vol. 29, No. 1, pp. 327–335


# THE SWAPPING PROBLEM ON A LINE*

SHOSHANA ANILY†, MICHEL GENDREAU‡, AND GILBERT LAPORTE‡

**Abstract.** We consider the problem of optimally swapping objects between $N$ workstations, which we refer to as nodes, located on a line. There are $m$ types of objects, and the set of object-types is denoted by $S = \{1, \ldots, m\}$. Object-type 0 is a dummy type, the *null object*. Each node $v$ contains one unit of a certain object-type $a_v \in S \cup \{0\}$ and requires one unit of object-type $b_v \in S \cup \{0\}$. We assume that the total supply equals the total demand for each of the object-types separately. A vehicle of unit capacity ships the objects so that the requirements of all nodes are satisfied. The set of object-types is partitioned into two sets: objects that may be temporarily dropped at intermediate nodes before reaching their destination and objects that have to be shipped directly from their origin to their destination. The objective is to design a route that starts and ends at the depot and a feasible assignment of object-types to the route's arcs so that the total distance is minimized. We propose an $O(N^2)$ algorithm to compute the optimal solution for this problem.

**Key words.** the swapping problem, transshipment problem

**AMS subject classifications.** 05C12, 05C45, 05C40, 05C85

**PII.** S0097539797323108

## 1. Introduction.

A set $V$ of $N$ workstations, which we refer to as nodes, is located on a line. We are also given a set of $m$ object-types $S = \{1, \ldots, m\}$. Object-type 0 is the null object. Each workstation $v$ contains one unit of a certain object $a_v \in S \cup \{0\}$ and requires one unit of object $b_v \in S \cup \{0\}$. $a_v = 0 (b_v = 0)$ means that no object is currently (required) at node $v$. For each object-type separately, the total supply (i.e., the number of nodes currently containing that object-type) equals the total demand (the number of nodes requiring that object-type). A vehicle of unit capacity that starts and ends at node $v_D \in V$ ships the objects from their initial locations so that the requirements of the workstations are satisfied. The objective is to design a feasible route of minimum length.

The set of object-types is partitioned into two sets $S = S_d \cup S_n$. Objects in $S_d$ may be temporarily dropped at intermediate nodes before reaching their destination, while objects in $S_n$ have to be shipped *directly* from their origin to their destination. One of these two sets may be empty. Clearly, the optimal solution when some of the objects may be dropped cannot be worse than in the case that $S = S_n$.

The general swapping problem where the workstations are the nodes of an undirected complete graph was studied by Anily and Hassin (1992). The authors show that the problem is NP-hard and prove the existence of an optimal solution that satisfies certain structural properties. They design two heuristics whose worst-case bounds are 2.5. The proposed heuristics are based on the composition of the optimal solutions to $m + 1$ matching problems, one for each object-type $i \in S \cup \{0\}$, where nodes $v$ with $a_v = i$ are matched with nodes $w$ with $b_w = i$. This step results in a set of disjoint cycles that are then patched into a single Eulerian tour. Except for Anily and Hassin (1992) and the current paper, the literature confines itself to systems containing one unit of each object-type, and moreover, all objects are in either $S_n$ or

*Received by the editors June 11, 1997; accepted for publication (in revised form) May 29, 1998; published electronically September 22, 1999.
http://www.siam.org/journals/sicomp/29-1/32310.html

†Faculty of Management, Tel-Aviv University, Tel-Aviv 69978, Israel (anily@post.tau.ac.il).
‡École des Hautes Études Commerciales, Montreal, PQ, Canada (michelg@crt.umontreal.ca, gilbert@crt.umontreal.ca).

$S_d$. In the stacker-crane problem analyzed by Frederickson, Hecht, and Kim (1978) (see also Johnson and Papadimitriou (1985)), some directed arcs are given and the objective is to find a directed closed tour of minimum length containing these arcs. The stacker-crane problem is a special case of the swapping problem, where $S = S_n$, and there is one unit of each object-type. The authors propose a polynomial approximation for this problem whose worst-case bound is 9/5. Atallah and Kosaraju (1988) analyze the swapping problem on a line and on a circular track when there exists exactly one unit of each object-type in $S$. They consider the cases of $S = S_n$ (no drops) and $S = S_d$ (with drops). Their motivation to study the problem arises from the movement of a robot arm that is supposed to rearrange $m$ objects among $N$ stations. The robot arm consists of a single link that rotates around a fixed pivot. The link's length is variable since it can be extended in and out. A gripper that can grasp any of the objects is positioned at the end of the link. Minimizing the total distance traveled by the gripper is NP-hard. Instead, the authors focus separately on minimizing the total (a) telescoping motion, which corresponds to moving along a linear track, and (b) angular motion, which corresponds to moving along a circular track. The paper provides low polynomial algorithms for computing the optimal route: for the no-drop case the algorithm runs in $O(m + N\alpha(N))$ ($\alpha()$ is the inverse of Ackerman's function) for the linear track and in $O(m + N \log N)$ for the circular track (this last bound is further tightened in Frederickson (1993)). For the with-drop case the algorithm runs in $O(m + N)$ for both the linear and circular tracks. Frederickson and Guan (1992, 1993) studied the same problem as that of Atallah and Kosaraju (1988) when the objects are located on the vertices of a tree and the vehicle travels along its edges. For the with-drop case they present two algorithms that run in $O(m + Nq)$, where $q \leq \min\{m, N\}$. The no-drop case is shown to be NP-hard, and the authors provide two heuristics that run in low polynomial time with worst-case ratios of 1.5 and 1.25, respectively.

In this paper, we develop an $O(N^2)$-step algorithm for the linear track allowing no-drop and with-drop objects. Section 2 introduces the terminology, and section 3 establishes the necessary structural properties of any optimal solution. Finally, the main algorithm and its analysis is presented in section 4.

**2. Notations and preliminaries.** $V = \{v_1, v_2, \ldots, v_N\}$ is a set of $N$ workstations. Vertex $v_D, 1 \leq D \leq N$, is the depot where the vehicle starts and terminates the tour. The vertices are indexed according to their location on the line from left to right. $S = \{1, \ldots, m\}$ is a set of $m$ object-types. Object-type 0 denotes the null object. Each vertex $v$ is associated with a pair $(a_v, b_v) \in [S \cup \{0\}] \times [S \cup \{0\}]$, where $a_v$ is the object-type currently at $v$ and $b_v$ is the object-type desired at $v$. Without loss of generality (w.l.o.g.) we assume that $a_v \neq b_v$. (If $a_D = b_D = 0$, then an equivalent problem can be defined such that $a_D \neq b_D$ by introducing a new node at the depot's location and a new object-type $m + 1$. Associate the depot with $(0, m + 1)$ and the new vertex with $(m + 1, 0)$.) The set $S$ is partitioned into $S_n$, the set of *no-drop* objects-types, and $S_d$, the set of object-types that can be *dropped* at intermediate vertices. $d(u, v)$ is the distance between vertices $u$ and $v$. $P^i \subseteq V$ are the supply vertices of object-type $i (i = 0, 1, \ldots, m)$, i.e., $P^i = \{v : a_v = i\}$. Let $P^i = \{p_{i1}, p_{i2}, \ldots, p_{ik(i)}\}$. $R^i \subseteq V$ are the vertices that require object-type $i (i = 0, 1, \ldots, m)$, i.e., $R^i = \{v : b_v = i\}$. Let $R^i = \{r_{i1}, r_{i2}, \ldots, r_{ik(i)}\}$. Let $V^i = P^i \cup R^i$, and $k(i) = |P^i| = |R^i|$ is the number of objects of type $i$ in the system. Thus, $\sum_{i=0}^{m} k(i) = N$. The supply (demand) vertices of object-type $i$ are indexed according to their location from left to right.

If $S = S_n$, then once we know which supply vertex in $P^i$ serves each of the demand

vertices in $R^i$ for $i = 1, \ldots, m$, our problem reduces to the respective problem solved by Atallah and Kosaraju (1988).

DEFINITION 2.1. *A* path *is a sequence of directed arcs where the tail of one arc is the head of the preceding arc in the sequence, and all arcs in the sequence are assigned the same object-type* $i, 0 \leq i \leq m$.

DEFINITION 2.2. *If object-type* $i, 1 \leq i \leq m$, *that is initially located at vertex* $p_{ih}$ *is used to supply the requirement of vertex* $r_{ih'}$, *then the path from* $p_{ih}$ *to* $r_{ih'}$ *along which the object is shipped is called a* service path. *(See below for an extension of the definition for object-type* 0.*)*

Assume w.l.o.g. that objects are never dropped in order to pick up objects of the same type. Thus, in any feasible solution and any $v \in V$ with $a_v = i$, there exists a service path of object $i$ initiating at $v$ and ending at vertex $u$, $u \in V$ with $b_u = i$, $1 \leq i \leq m$. If $i \in S_n$, then the arcs of this path appear consecutively in the solution; i.e., the vehicle traverses the service path with no intermediate stops. If $i \in S_d$, then the arcs of this path appear in the solution in the same order as in the path, but not necessarily consecutively, meaning that the vehicle may drop the object and pick it up later.

DEFINITION 2.3. *A segment of the solution that the vehicle traverses empty is called a* deadheading.

According to Anily and Hassin (1992), if $a_v = 0$ for $v \in V$, then any feasible solution includes a path of deadheadings originating at $v$ and ending at some vertex $u$ with $b_u = 0$. Such a path is called a *service path* of the null object. Note that a feasible solution consists of $k(i)$ service paths for each object-type $i$, $i \in S \cup \{0\}$ and possibly some additional deadheadings.

DEFINITION 2.4. *The* end points *of a path starting at* $u$ *and ending at* $v$, $u, v \in V$ *are the vertices* $u$ *and* $v$.

DEFINITION 2.5. *A given arc is said to* cover *all points (not necessarily vertices) on the track in between its tail and its head including its end points. A path covers all points covered by its arcs.*

DEFINITION 2.6. *A path is* left-to-right (right-to-left) *if each of its arcs is directed to the right (left).*

DEFINITION 2.7. *Two paths are* intersecting in opposite directions *if* (a) *one path is left-to-right and other is right-to-left and* (b) *at least one end point of one path is covered by the other path.*

**3. Structural properties of an optimal solution.** In this section we prove some theorems that ensure the existence of an optimal solution that satisfies some specified properties. The theorems hold for all the problem's variants discussed here.

THEOREM 3.1. *There exists an optimal solution that does not contain any pair of service paths for the same object-type* $i, 0 \leq i \leq m$, *that are intersecting in opposite directions.*

*Proof.* The proof is by contradiction. Suppose the theorem is false, i.e., any optimal routing contains service paths of the same object-type that are intersecting in opposite directions. Consider an optimal solution and let $SP_1$ and $SP_2$ be two such service paths for object-type $i$, $0 \leq i \leq m$. Suppose $SP_1$ connects $p_{ij(1)}$ to $r_{ih(1)}$ and $SP_2$ connects $p_{ij(2)}$ to $r_{ih(2)}$. W.l.o.g. let $SP_1$ be a left-to-right path. We show that there exists an alternative feasible solution that follows the same route and the only difference is with respect to the assignment of object $i$ to the arcs of $SP_1$ and $SP_2$. Some arcs on these paths are assigned object $i$ and the others are turned into deadheadings. As a consequence, the two new service paths are no longer intersecting

in opposite directions. For the null object, the new solution is identical to the given one, but we distinguish differently between the service paths of object 0 and the deadheadings.

According to our assumptions, there exists a segment $(u, v)$ that is covered by $SP_1$ and $SP_2$, where $SP_1$ is directed from $u$ to $v$ and $SP_2$ from $v$ to $u$. The alternative solution is defined such that $r_{ih(1)}(r_{ih(2)})$ is served from $p_{ij(2)}(p_{ij(1)})$. It is easy to verify that the two new service paths consist of exactly those parts in $SP_1$ and $SP_2$ that do not cover $(u, v)$ and their directions are consistent with the respective directions of $SP_1$ and $SP_2$. Also, the segment $(u, v)$ on $SP_1$ and $SP_2$ is turned into a deadheading.

In the following, we derive additional properties satisfied by optimal paths.

DEFINITION 3.2. *$V^{\ell(i)} \subseteq V^i$ is* consecutive *if it consists of $V^i$ nodes that appear consecutively on the line.*

DEFINITION 3.3. *A* consecutive partition *of $V^i$ is a partition of $V^i$ into consecutive disjoint subsets.*

DEFINITION 3.4. *A subset of vertices $V^{\ell(i)} \subseteq V^i$ is called* balanced *if it is consecutive and is of even cardinality, where half of its vertices are in $P^i$ and the remaining are in $R^i$.*

DEFINITION 3.5. *A subset is* minimally balanced *if it is balanced and there does not exist any consecutive partition of the subset into two balanced subsets.*

DEFINITION 3.6. *The* consecutive minimally balanced partition (CMBP) *of $V^i$ is the consecutive partition of $V^i$, where each of the subsets in the partition is minimally balanced.*

Note that any feasible solution is associated with $m + 1$ sets of service paths, one set for each object-type. The service paths of object $i$ induce a consecutive balanced partition of $V^i$ defined recursively as follows: (1) the two end points of a service path belong to the same subset in the partition; (2) all vertices of $V^i$ covered by a certain service path belong to the same subset as the end points of the path; and (3) the subsets are minimal. It is easy to see that for a given solution the consecutive balanced partition of $V^i$ induced by the service paths of object $i$ is well defined. Moreover, if the solution does not contain any intersecting in opposite directions service paths for the same object-type, then all service paths within a set of this partition are in the same direction.

THEOREM 3.7. *In any optimal solution that does not contain intersecting in opposite directions service paths of object $i$, the service paths of this object-type* (1) *induce the CMBP of $V^i$ and* (2) *have constant total length.*

*Proof.* Suppose $Z$ is an optimal solution that satisfies the property of the theorem. The service paths of object-type $i$ that are associated with solution $Z$ induce a consecutive balanced partition of $V^i$. We want to show first that the subsets of this partition are *minimally balanced*. Assume w.l.o.g. that the leftmost vertex in $V^i$ is a supply vertex $s_1 \in P^i$. Let $L_{i1} \subseteq V^i$ be the set induced by solution $Z$ that contains $s_1$.

The proof is by induction on the cardinality of $V^i$. The minimum cardinality of $V^i$ is two; i.e., $V^i$ consists of a supply vertex and a demand vertex and the theorem is trivial. According to the inductive hypothesis, the theorem holds for any set of cardinality less than $|V^i|$. Suppose $|V^i| > 2$. We distinguish between two cases: (a) No consecutive partition of $V^i$ into two balanced subsets exists. Thus, $L_{i1} = V^i$; i.e., the partition of $V^i$ induced by the service paths of object-type $i$ in $Z$ is into a single set, which is the CMBP of $V^i$. (b) There exists a consecutive balanced partition of $V^i$ into two subsets: let $V^i = V_{i1} \cup V_{i2}$ be a consecutive partition of $V^i$, where $V_{i1}$ is a minimally balanced subset that contains $s_1$. We will show that $L_{i1} = V_{i1}$ and the

rest will follow by the inductive hypothesis. Suppose that $V_{i1} \subset L_{i1}$ but $V_{i1} \neq L_{i1}$. Since $s_1$ is the leftmost supply vertex of object $i$ on the line, it must serve a demand vertex to its right. As a consequence, all service paths within $L_{i1}$ should be from left to right, otherwise there would be intersecting paths in opposite directions within $L_{i1}$, in contradiction to our assumption about $Z$. Since $V_{i1} \subset L_{i1}$ but $V_{i1} \neq L_{i1}$, there should be in $Z$ a supply vertex $p \in V_{i1}$ that serves a demand vertex $r \in L_{i1} - V_{i1}$. In view of the fact that $V_{i1}$ is balanced, at least one of the demand vertices $r'$ in $V_{i1}$ should be served according to $Z$ by a supply vertex $s' \in L_{i1} - V_{i1}$. This contradicts the assumption that $Z$ does not contain any service paths for object-type $i$ that are intersecting in opposite directions.

In order to prove the second part of the theorem, we will show that the total length of the service paths of object $i$ within any subset of the CMBP of $V^i$ is constant. Suppose w.l.o.g. that we are given a minimally balanced subset of the CMBP of $V^i$ for which all service paths are from left to right. Also suppose that $x$ and $y$ are consecutive vertices in $V^i$, where $x$ is strictly to the left of $y$. Let $lp^i(z)(lr^i(z))$ denote the number of supply (demand) vertices from $V^i$ within the subset that are located to the left or at the location of $z$. Then, the number of service paths of object $i$ that cover the segment connecting $x$ to $y$ is $lp^i(x) - lr^i(x)$, which is strictly positive according to our assumptions. Accordingly, a simple calculation that depends only on the subset gives the total length of service paths within that subset.

**4. The algorithm for the linear track case.** In this section we present an algorithm for finding an optimal policy for linear graphs. As a consequence of the previous section, an optimal solution exists in which the service paths of each object-type $i$, $0 \leq i \leq m$, induce the CMBP of $V^i$. Let $\{V_{i\ell}\}_{\ell=1,\ldots,L(i)}$ denote the CMBP of $V^i$ for $i = 0, \ldots, m$. We now propose an algorithm for computing an optimal routing policy. The first step in the algorithm is to define a directed graph on $V$ that consists of those arcs that must appear in such an optimal solution. Each arc in the graph connects two vertices in $V_{i\ell}$ for some $i = 0, \ldots, m$ and $\ell = 1, \ldots, L(i)$.

DEFINITION 4.1. *A set $V_{i\ell}$ of the CMBP of $V^i$ is called a* left-set (right-set) *if the number of supply vertices is no smaller (no larger) than the number of demand vertices of object-type $i$ in any consecutive subset of $V_{i\ell}$ that contains the leftmost vertex of $V_{i\ell}$.*

The directed graph $G$ on $V$ is defined by the following algorithm, called Basic Graph.

ALGORITHM BASIC GRAPH.

*Step* 0. For $i = 0, \ldots, m$ and $\ell = 1, \ldots, L(i)$ do steps 1 and 2:

*Step* 1. Let $k$ be the number of demand vertices in $V_{i\ell}$. If $V_{i\ell}$ is a left-set (right-set), then index the demand vertices in $V_{i\ell}$ from left to right (right to left) as $\{r_1, r_2, \ldots, r_k\}$. Let $G_{i\ell}$ initially be a graph with no arcs. Add to $G_{i\ell}$ a directed arc from each supply vertex in $P^i \cap V_{i\ell}$ toward the first demand vertex in $R^i \cap V_{i\ell}$ to its right (left), i.e., the first demand vertex it may serve.

*Step* 2. If $|V_{i\ell}| = 2$, then stop (the graph $G_{i\ell}$ contains a single arc). Otherwise, set $j = 1$; while $j < k$ do begin: Count the number of incoming arcs in $G_{i\ell}$ to $r_j$. Let this number be $in(j)$; add $in(j) - 1$ arcs to $G_{i\ell}$ all from $r_j$ towards $r_{j+1}$. endwhile;

*Step* 3. Let graph $G$ be the composition of all subgraphs $G_{i\ell}$ for $i = 0, \ldots, m$ and $\ell = 1, \ldots, L(i)$.

*Remark.* Note that in Step 2, $V_{i\ell}$ is a subset of the CMBP of $V^i$; thus $in(j) > 1$ for $j \leq k - 1$ and $in(k) = 1$.

LEMMA 4.1. *The in-degree equals the out-degree for any of the vertices of graph $G$.*

*Proof.* We have to show that the number of incoming arcs equals the number of outgoing arcs for any vertex $v \in V$. Let $v \in V$, and denote $(a_v, b_v) = (k, j)$ $k$, $j \in S \cup \{0\} k \neq j$. Also suppose that $v \in V_{k\ell} \cap V_{jh}$. The only arcs incident to $v$ are arcs in the subgraphs $G_{k\ell}$ and $G_{jh}$. In $V_{k\ell}$, $v$ is a supply vertex; thus exactly one arc exits from $v$ in $G_{k\ell}$. In $V_{jh}$, $v$ is a demand vertex; thus in $G_{jh}$, the number of outgoing arcs from $v$ is one less than the number of incoming arcs to $v$. Thus overall, $G$ satisfies the lemma.

A directed graph may be partitioned into a collection of equivalence classes such that vertices $w$ and $v$ are in the same class if and only if the graph contains directed paths from $v$ to $w$ and from $w$ to $v$. (See section 5.5 in Aho, Hopcroft, and Ullman (1974).) The subgraph induced by a certain class consists of the vertices in the class as well as all edges in the graph that connect a pair of vertices in the class. These subgraphs are called the *strongly connected components* of the given graph. In light of Lemma 4.1, the union of the strongly connected components of $G$ results in $G$ itself.

We first demonstrate the algorithm when $G$ is strongly connected: Since $G$ is Eulerian, there exists an optimal solution that consists of $G$'s arcs and does not use the drop option. This is observed by noting that if the number of incoming arcs to vertex $v$ is $k$, $k > 1$, then *all* incoming arcs to $v$ carry item $b_v$, where $k - 1$ outgoing arcs carry item $b_v$. Thus, in $k - 1$ of the $k$ entrances to $v$ the vehicle continues with the same item; at one entrance item $b_v$ is unloaded and at one exit item $a_v$ is loaded. Any Euler tour in $G$ produces an optimal solution by starting at the depot. The complexity of finding such a tour is linear in the number of arcs of $G$. A simple calculation demonstrates that the number of arcs in $G$ carrying item $i$ is at most $k(i)(k(i) + 1)/2$. Thus, $G$ contains at most $\sum_{i=0}^{m} k(i)(k(i) + 1)/2$ arcs. Since $\sum_{i=0}^{m} k(i) = N$, the maximum number of arcs in $G$ is of order $O(N^2)$.

We continue with the general case when $G$ is not necessarily strongly connected, i.e., $G$ is the union of a number of strongly connected components where each is an Eulerian subgraph. Up to now, we have not used the fact that the vehicle is empty along arcs associated with the null object and we have not used the drop option. For that sake, we extend the term "connectivity:" It is not necessarily the case that for two different components of $G$ none is reachable from the other, moreover it may well be that each is reachable from the other. An arc in $G$ may cover intermediate vertices, thus it consists of a sequence of *basic arcs*, where a basic arc is defined as a directed arc connecting two consecutive vertices of $V$. We distinguish between three types of arcs in $G$: (1) arcs of object-type $i$, $i \in S_n$; (2) arcs of object-type $i$, $i \in S_d$; and (3) arcs of the null object. Arcs of no-drop objects should be followed continuously from their initial vertex to their terminal vertex; i.e., the respective sequence of basic arcs should occur in the solution consecutively. Arcs of drop-objects should be followed from their initial vertex to their terminal vertex with possible stops at intermediate vertices for drops; i.e., the respective basic arcs should be followed at the order they occur in the sequence, but not necessarily consecutively. The basic arcs of the null object occur in the solution with no restriction on their order.

DEFINITION 4.2. *Let $C_\ell^s$ and $C_k^s$ be different strongly connected components of $G$. $C_k^s$ is* directly reachable *from $C_\ell^s$ if at least one of the following conditions holds:* (1) *There exists an arc of the null object in $G$ that covers vertices $v \in C_\ell^s$ and $w \in C_k^s$ and is directed from $v$ to $w$; or* (2) *there exists an arc in $G$ of object $i$, $i \in S_d$, whose tail is in $C_\ell^s \cap V^i$, and at least one vertex in $C_k^s$ is covered by this arc. $G$'s arcs that allow direct reachability of strongly connected components are called* reachability arcs.

DEFINITION 4.3. *Let $C_1^s, C_2^s, \ldots, C_n^s$ be different strongly connected components of $G$. We say that $C_n^s$ is* reachable *from $C_1^s$ if $C_k^s$ is directly reachable from $C_{k-1}^s$ for $k = 2, \ldots, n$.*

DEFINITION 4.4. *Two different strongly connected components of $G$, $C_1^s$ and $C_2^s$, such that each is reachable from the other are said to be* weakly connected.

For practical purposes, a maximal weakly connected component is a connected component since *starting at any vertex* in the set there exists a closed tour that serves all vertices in the set, possibly by using the drop option. Therefore, we repartition $V$ into rougher equivalence classes by grouping the strongly connected components of $G$ according to the weak connectivity relation. Let $C_1^w, C_2^w, \ldots, C_k^w$ be the weakly connected components of $G$, where the depot is assumed to be in $C_1^w$. In each of these components identify a closed feasible tour that serves all its vertices: This tour is a patching of the tours found in the strongly connected components, where the patching is done along reachability arcs. For that sake, we extend the reachability definition to weakly connected components.

DEFINITION 4.5. *$C_h^w$ is said to be reachable from $C_\ell^w$ if $C_h^w$ contains a strongly connected component that is reachable from a strongly connected component contained in $C_\ell^w$, $\ell \neq h$.*

DEFINITION 4.6. *A weakly connected component of $G$ that is not reachable from any other weakly connected component is said to be an* unreachable component. *Let $C_1^w$ be an unreachable component independently of its reachability from other components as the tour should start at the depot.*

DEFINITION 4.7. *The* reachable set *of the unreachable component $C_\ell^w$ is the set of weakly connected components of $G$ that are reachable from $C_\ell^w$. ($C_\ell^w$ is said to be reachable from itself.) We note that a weakly connected component of $G$ either may be an unreachable component or belongs to at least one reachable set of some unreachable component.*

If $C_k^w$ is reachable from $C_\ell^w$, then there exists a feasible closed tour that *starts* at $C_\ell^w$ and serves all vertices in $C_\ell^w \cup C_k^w$: The vehicle, while either carrying a droppable item loaded at $C_\ell^w$ or while being empty, traverses a reachability arc of $G$ that connects two vertices of $C_\ell^w$ and covers a vertex of $C_k^w$; in the first case, the item may be dropped at such a vertex of $C_k^w$. The vehicle may then serve all vertices of $C_k^w$; in the first case the vehicle then reloads the droppable item. The vehicle then continues toward its destination in $C_\ell^w$. Thus, a closed feasible tour that starts at the unreachable component (the tour starts in $C_1^w$ at the depot) may be identified within each of the reachable sets such that all vertices within the set are served. In order to execute this step, no augmentation of the graph is needed, as it is a patching of the tours that have been found separately in each of the weakly connected components contained within the reachable set, where the patching is done along reachability arcs. Thus, if $C_1^w$ is the only unreachable component, no augmentation of $G$ is required, i.e., there exists an optimal solution with total length identical to the total length of $G$'s arcs. Otherwise, $G$ must be augmented in order to reach each of the unreachable components from $C_1^w$. Each augmentation arc added to $G$ must be traversed twice, once in each direction. We next propose a minimum cost augmentation technique for $G$ whose complexity time is $O(N^2 \log N)$; then we show how the complexity may be reduced to $O(N \log N)$ by using the fact that the vertices are located on a line.

Define a directed graph $T$ on the set of vertices that correspond one-to-one to the unreachable components of $G$. Let node $\ell$ of $T$ represent $C_\ell^w$ (node 1 of $T$ corresponds to $C_1^w$). There exists a directed arc in $T$ from node $\ell$ to node $k$, $k \neq 1$, if and only

if there exists a pair of vertices $v_j$ and $v_h$ in $V$ such that $v_j$ is in the reachable set of $C_\ell^w$ and $v_h$ is in the unreachable component $C_k^w$. The length of this arc, denoted by $\delta(\ell, k)$, is defined as the length of the shortest arc connecting the reachable set of $C_\ell^w$ and unreachable component $C_k^w$. The minimum cost augmentation of $G$ may be found by solving a *minimum directed spanning tree* (MDST) on $T$ rooted at node 1. The MDST was solved independently by Chu and Liu (1965), Edmonds (1967), and Bock (1971). The complexity of their algorithm on a general directed graph with $N$ nodes and $|E|$ edges is $O(\min\{|E| \log N, N^2\})$. The number of nodes of $T$ is the number of unreachable components which is at most $O(N)$. Thus, $T$ contains at most $O(N^2)$ edges. Therefore, the complexity of the MDST algorithm on $T$ is $O(N^2 \log N)$. Below we show that due to the linearity of $G$, we may apply the MDST algorithm on a subgraph of $T$ that contains at most $O(N)$ edges. As a result the complexity of this step will be reduced to $O(N \log N)$.

DEFINITION 4.8. *A vertex $v_j \in V \cap S$, $j > 1$, is called* extreme to the left in $S$ *if and only if $v_{j-1} \notin S$; a vertex $v_j \in V \cap S$, $j < N$, is called* extreme to the right in $S$ *if and only if $v_{j+1} \notin S$. A vertex is called* extreme *if it is either extreme to the left or extreme to the right.*

*Let $v_j$ be an extreme to the left (right) vertex in the reachable set of the unreachable component $C_\ell^w$. Let $v_h \in V$, $h < j (h > j)$, be the rightmost (leftmost) vertex that belongs to some unreachable component $C_k^w$, $k \notin \{1, \ell\}$. If such a vertex $h$ exists, then we say that the extreme vertex $v_j$ and the reachable set of $C_\ell^w$ have a* neighboring unreachable component $C_k^w$. *Each extreme vertex may have at most two neighboring unreachable components (one to its left and one to its right), but a reachable set may have several neighboring unreachable components. Due to the linearity of the track, there exists a minimum cost augmentation of $G$ using only those edges of $T$ that connect reachable sets to* their neighboring *unreachable components. Indeed, it is sufficient to include in $T$ only edges connecting a vertex $\ell$ to a vertex $k$ if and only if $C_k^w$, $k \notin \{1, \ell\}$, is a neighboring unreachable component of the reachable set of $C_\ell^w$. The length of such an edge is given by $\delta(\ell, k) = \min\{d(v, w)$: $v$ is an extreme vertex in the reachable set of $C_\ell^w$ and $w \in C_k^w\}$. According to the new procedure, $T$ now contains a subset of the arcs that were previously contained in $T$. An arc of $T$ that is eliminated is not needed in the augmentation of $G$ as it is too expensive and can be replaced by a cheaper sequence of those arcs that are left in $T$. It is easily verified that any unreachable component $C_k^w$ can be reached from any reachable set $C_\ell^w$, $k \notin \{1, \ell\}$, via those arcs that are left in $T$. This property ensures that $T$ contains a directed spanning tree. The number of directed arcs in $T$ is at most of size $O(N)$, as each extreme vertex may have at most two neighboring unreachable components.*

Let MDST($T$) be the MDST length of $T$, and let $A(T)$ be its corresponding set of arcs on the line. Along the arcs of $A(T)$ the vehicle travels empty twice, once in each direction. There exists an optimal solution for the problem whose length is $L(G) + 2\text{MDST}(T)$, where $L(G)$ is the total length of all arcs in $G$. The complexity of the whole procedure is $O(N^2)$. Below we summarize the complete algorithm.

THE SWAPPING ALGORITHM FOR THE LINEAR TRACK CASE.

*Step* 1. Apply the Basic Graph algorithm on $V$. Let $G$ be the resulting graph, and let $L(G)$ be its total length.

*Step* 2. Identify the strongly connected components of $G$.

*Step* 3. If the system contains the null object or $S_d \neq \emptyset$, then partition the strongly connected components into equivalence classes according to the weak connectivity relation. Let $C_1^w, C_2^w, \ldots, C_k^w$ be the weakly connected components, with the depot

at $C_1^w$. Within each of them find a closed feasible tour that serves all its vertices, using only $G$'s arcs where some of them may be broken into basic arcs. Identify the unreachable components, the reachable sets, and a corresponding set of reachability arcs.

*Step* 4. Define a directed graph $T$ on the set of nodes that corresponds one-to-one to the unreachable components. A directed arc from node $\ell$ to node $k$ in $T$, $k \neq 1$, exists if and only if the unreachable component $C_k^w$ is a neighbor of some extreme vertex in the reachable set of the unreachable component $C_\ell^w$. The distance between these nodes is determined by the minimum distance between an extreme vertex in the reachable set of $C_\ell^w$ and an extreme vertex in its neighboring unreachable component $C_k^w$. Let MDST($T$) be the MDST length on $T$, and let $A(T)$ be the corresponding set of arcs in the linear track.

*Step* 5. Use two copies of $A(T)$'s arcs and reachability arcs that connect weakly connected components within the same reachable set to patch the closed tours associated with the weakly connected components. Along $A(T)$'s arcs the vehicle travels empty. Starting at the depot, follow a feasible closed tour that serves all vertices by traversing $G$'s arcs and twice the arcs of $A(T)$ once in each direction. We conclude the paper with a proof that the above algorithm solves the swapping problem optimally.

THEOREM 4.9. *The swapping algorithm for the linear track case solves the swapping problem on a line optimally.*

*Proof.* Any feasible solution is the union of service paths for each object-type $i \in \{0, \ldots, m\}$ and possibly some deadheadings; see Anily and Hassin (1992). As shown in Theorems 3.1 and 3.7, any feasible solution can be transformed into a feasible solution of the same cost, where the length of service paths of each object-type $i \in \{0, \ldots, m\}$ is separately minimized and possibly some deadheadings. The minimum cost service paths are obtained by algorithm BASIC GRAPH $G$. A minimum cost set of deadheadings is found by applying the MDST procedure on $T$. Two copies of each such deadheading is added to the BASIC GRAPH $G$ in order to preserve the final graph as Eulerian while making it connected.

## REFERENCES


S. ANILY AND R. HASSIN (1992), *The swapping problem*, Networks, 22, pp. 419–433.

A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN (1974), *The Design and Analysis of Computer Algorithm*, Addison–Wesley, Reading, MA.

M. J. ATALLAH AND S. R. KOSARAJU (1988), *Efficient solutions to some transportation problems with applications to minimizing robot arm travel*, SIAM J. Comput., 17, pp. 849–869.

F. BOCK (1971), *An algorithm to construct a minimum directed spanning tree in a directed network*, in Developments in Operations Research, Gordon and Breach, New York, pp. 29–44.

Y. J. CHU AND T. H. LIU (1965), *On the shortest arborescence of a directed graph*, Sci. Sinica, 14, pp. 1396–1400.

J. EDMONDS (1967), *Optimum branchings*, J. Res. Nat. Bur. Standards Sect. B, 71, pp. 233–240.

G. N. FREDERICKSON (1993), *A note on the complexity of a simple transportation problem*, SIAM J. Comput, 22, pp. 57–61.

G. N. FREDERICKSON AND D. J. GUAN (1992), *Preemptive ensemble motion planning on a tree*, SIAM J. Comput., 21, pp. 1130–1152.

G. N. FREDERICKSON AND D. J. GUAN (1993), *Nonpreemptive ensemble motion planning on a tree*, J. Algorithms, 15, pp. 29–60.

G. N. FREDERICKSON, M. S. HECHT, AND C. E. KIM (1978), *Approximation algorithms for some routing problems*, SIAM J. Comput., 7, pp. 178–193.

D. S. JOHNSON AND C. H. PAPADIMITRIOU (1985), *Performance guarantees for heuristics*, in The Traveling Salesman Problem, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., John Wiley, New York, Chapter 5, pp. 145–180.