Semi–preemptive routing on a linear and circular track

Dirk Räbiger Rainer Schrader

Zentrum für Angewandte Informatik, Universität zu Köln
Weyertal 80, 50931 Köln
Germany

Abstract

The problem of routing a robot (or vehicle) between n stations in the plane in order to transport objects is well studied, even if the stations are specially arranged, e.g. on a linear track or circle. The robot may use either *all* or *none* of the stations for reloading. We will generalize these concepts of preemptiveness/nonpreemptiveness and emancipate the robot by letting it choose $k \leq n$ reload–stations. We will show that the problem on the linear and circular track remains polynomial solvable.

Key words: pickup and delivery, dial–a–ride, transportation

1 Introduction

We consider a transportation problem where m heterogeneous objects have to be moved between n stations in the plane. For each object we have a *request* (i, j) indicating that the object is initially located at station i and has to be moved to its destination j. A station can be source and destination for several objects. We assume that every station is source or destination of some request, since otherwise an unused station may be removed.

The transport is done by a robot which can only handle one object at a time. The robot starts at a predefined initial station and moves back and forth along the track to pick up objects, transport them, and drop them. We want to find a motion plan of minimal length so that the robot can move every object from

Email addresses: `raebiger@zpr.uni-koeln.de` (Dirk Räbiger), `schrader@zpr.uni-koeln.de` (Rainer Schrader).

its source to its destination and returns to its initial station afterwards. We focus on the special cases where the stations are arranged on a line or a circle.

Typically, one distinguishes between a *non–preemptive* and a *preemptive* version of the problem. In the first case any object may only be dropped at its destination station once it is picked up. The latter case allows the robot to drop the object at any intermediate node and pick it up later. We will call this action a *reload*. Both cases were solved in [1]. A nice overview of closely related problems is given in [5].

We generalize the concepts of preemptiveness/nonpreemptiveness by allowing the use of up to k reload–stations during the transport for some given $k \leq n$. The reload–stations may be *exogenously* specified in advance and the robot is allowed to use every such given station for reloads. In a different model only the number k is given so that the reload–stations have to be *endogenously* determined such that the total travel length is minimized.

We will describe polynomial time algorithms for both the endogenous (sec. 2) and exogenous (sec. 3) cases. The problem is $\mathcal{NP}$–complete on a tree as a result of the generalization of the non–preemptive case [3]. Thus the proposed generalization does not make the problem harder if we simply distinguish between polynomial time solvable and $\mathcal{NP}$–complete problems.

2 The endogenous routing problem on a line or circle

Let $\mathcal{S} = \{0, \ldots, n-1\}$ be a set of n nodes which are either arranged on a line or on a circle. Between neighboring nodes i and j in this arrangement we have a nonnegative *distance* $l(i,j) = l(j,i)$. If the nodes are arranged on a line we connect the end nodes and put a sufficiently large distance value on this new edge. We may thus treat the line as a special case of the circle. We assume that the nodes are numbered clockwise. We often loosely write $i+1$ instead of $i+1 \mod n$.

The robot starts at station $0 \in \mathcal{S}$, moves the objects and returns to station 0 afterwards. Let $\mathcal{R}$ be the set of m requests, where each request is of the form (i,j). A request will be completed by moving an object from i to j. Each such move defines a path whose length is the sum of distances between adjacent nodes.

A move may be interrupted at a reload node. For each reload station installed we introduce a cost value $\Delta \geq 0$. (Although it is easy to extend the results to $\Delta < 0$ we abandon this case, because any optimal solution will use exactly k reload nodes.) The total length of a transport plan is the length of the travel plus the costs for opening reload stations.

A request (i, j) will be completed by moving the object either clockwise or counter-clockwise along the circle. By a result of Atallah and Kosaraju [1] we may assume that at most one request uses its longer path. So we solve the robot problem by considering $m+1$ restricted problems in which the directions are fixed and at most one specified request $r_0 \in \mathcal{R}$ uses its longer path.

2.1 Graphtheoretic formulation of the problem

From now on we assume that $r_0 \in \mathcal{R}$ and all other directions are fixed. If reloads are forbidden, the transportation problem reduces to a Chinese postman problem where we have to add arcs to the graph $(S, \mathcal{R})$ such that the resulting graph becomes Eulerian. The cost for adding an arc is given by its length. If reloads are allowed, we have a second operation to extend the original graph. We say that an arc (u, v) *crosses* a node $x \in S$ if the associated path contains x. For the second operation we may choose a node x and replace every arc (u, v) which crosses x by two arcs (u, x) and (x, u). The cost for this operation is Δ. If B denotes the set of chosen reload nodes, the extension is valid if $|B| \leq k$.

Every solution to the transportation problem describes an extension of $(S, \mathcal{R})$ to a Eulerian graph with at most k nodes chosen as reload nodes. The arc multiset of the extended graph consists of three types of arcs. For each $r \in \mathcal{R}$ there is path P_r representing r. Let A_R be the corresponding multiset of arcs. In order to make the graph connected we will introduce a set A_C of pairs of antiparallel arcs. Since in $(S, A_R \cup A_C)$ the node balances are the same as in the original graph, there will be a third set A_ψ of arcs needed to balance the graph. Together with A_C, these arcs correspond to "empty" moves of the robot arm without transporting an object.

Atallah and Kosaraju [1] have shown that the creation of the arc sets can be done in two separate steps: we first balance the node degrees and then make the graph strongly connected. Referring to the circle we call the section between the station i and $i+1$ the *interval* i. Atallah and Kosaraju define the *flux* $\phi(i)$ across an interval i as

$$\phi(i) = |\{a \in A \text{ crossing } i \text{ clockwise}\}| - |\{a \in A \text{ crossing } i \text{ counter-clockwise}\}|.$$

Note that $\phi(i)$ also counts arcs crossing i which do neither end in station i nor in station $i + 1$. The $\phi(i)$'s can all be computed in time $O(m)$. Atallah and Kosaraju show that the graph is degree balanced if and only if the flux is constant over i, i.e. $\phi(i) = \phi(j)$ for all $i, j = 0, \ldots, n-1$. Moreover, for a given value ψ, the graph can be balanced at a minimum cost by adding $|\psi - \phi(i)|$

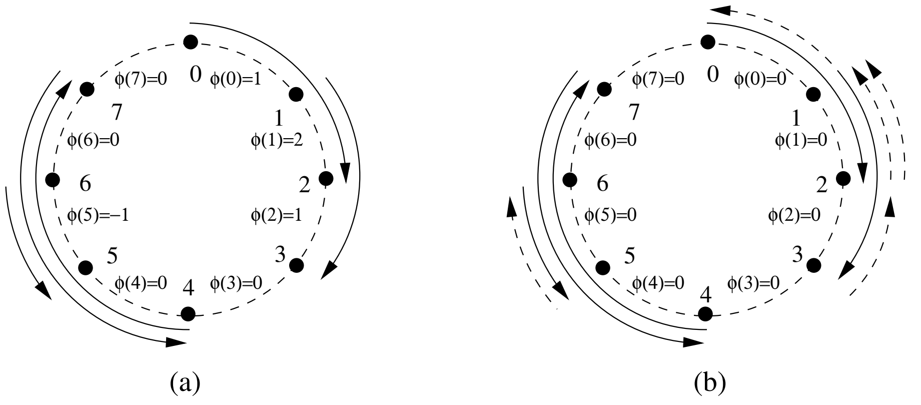

Fig. 1. (a) graph with requests (solid) and flux values (b) same graph with added augmenting arcs (dashed)

many copies of the arc $(i, i+1)$ to A_ψ if $\phi(i) < \psi$ (copies of $(i+1, i)$ if $\phi(i) > \psi$ resp.) for all intervals i. Figure 1 illustrates an example.

It is not yet clear which value we should choose for ψ. The following Lemma reduces the number of choices for ψ to just a few. It allows us to fix a value and enumerate all possible choices.

Lemma 1 (Atallah and Kosaraju [1]) *(1) There exists an optimal augmentation with* $\psi \in [-m-1, m+1]$.

(2) In any optimal transport graph, at most one object is moved to its destination along the major arc.

(3) If an optimal transport graph contains a major arc, then its flux value ψ *equals either* 1 *or* -1.

(4) If the node set is arranged on a line, then $\psi = 0$.

In the following we assume that $r_0 \in \mathcal{R}$ and some value ψ is fixed. In $G = (\mathcal{S}, A_\psi \cup \mathcal{R})$ all nodes are degree balanced. It remains to make the graph strongly connected. For this, we may replace arcs r by paths P_r and add antiparallel arcs from A_C. Since the cost function is additive, we may assume that the antiparallel arcs are all of the form $(i, i+1), (i+1, i)$.

Let K_i denote the strongly connected components of G. For $i \in \mathcal{S}$ let $K(i)$ be the connected component containing i. Since G is balanced the K_i's are the connected components of the underlying undirected graph. In particular, the endpoints of an arc are in the same component.

We construct a directed auxiliary graph $H = (V, E^r \dot\cup E^b)$ with colored arcs and arc weights. The nodes $v_i \in V$ correspond to the strongly connected components K_i where v_0 is the node corresponding to the connected component K_0 containing the start node 0. The arcs are either colored red ($e \in E^r$) or blue ($e \in E^b$). Consider two different nodes $u, v \in V$:

- create a red arc $(u, v) \in E^r$ with cost $c(u, v) = 2l(i, j)$, if there exist nodes

$i, j \in \mathcal{S}$ which are neighbors on the circle but in different connected components $K(i) = K_u, K(j) = K_v$. If there are several candidates choose i, j such that $l(i, j)$ is minimal.

- create a blue arc $(u, v) \in E^b$ with cost $c(u, v) = \Delta$ if there exists a request arc $(i, \ell) \in \mathcal{R}$ with $K(i) = K(\ell) = K_u$ which crosses a node $j \in \mathcal{S}$ with $K(j) = K_v$.

Let $G = (V, E^r \dot{\cup} E^b)$ be a directed multigraph and $v_0 \in V$ be some node. An arborescence $T \subseteq E^r \cup E^b$ is called a (k, v_0)*–arborescence* if it is rooted in $v_0 \in V$ and contains at most k blue arcs. We will use (k, v_0)–arborescences to solve the robot problem. Let $G = (\mathcal{S}, A_\psi \cup \mathcal{R})$ be the balanced graph of the transportation problem, H the auxiliary graph as above and $\gamma := l(A_\psi) + l(A_\mathcal{R})$.

Theorem 2 *Let T be a minimum cost (k, v_0)–arborescence of H. Then an optimal robot schedule has cost $c(T) + \gamma$. It can be constructed from T in time $O(n)$.*

PROOF. We first show that for any (k, v_0)–arborescence T we can construct a transport graph $G_T = (\mathcal{S}, A_\psi \cup A_\mathcal{R} \cup A_C, B)$ with cost $c(T) + \gamma$. Initially we set $G_T = (\mathcal{S}, A_\psi \cup \mathcal{R}, \emptyset)$ with $A_C = \emptyset, A_\mathcal{R} = \mathcal{R}, B = \emptyset$ and cost γ. We start in v_0 and traverse the nodes of T in a depth-first-search manner. Let u be some node and v be a son of u in the search-tree. The arc $(u, v) \in T$ is colored either red or blue.

(1) $(u, v) \in E^r$: by definition there exist nodes $i, j \in \mathcal{S}$ which are neighbors on the circle with $K(i) = K_u$ and $K(j) = K_v$. We add two anti–parallel arcs $(i, j), (j, i)$ to A_C. This preserves the degree balance and increases the cost by $c(u, v)$.

(2) $(u, v) \in E^b$: by definition there exist $i, j, \ell \in \mathcal{S}$ and an arc $(i, \ell) \in A_\mathcal{R}$ with $K(i) = K(\ell) = K_u$ and $K(j) = K_v$. We add j to B and replace (i, ℓ) in $A_\mathcal{R}$ by $(i, j), (j, \ell)$. Again the degree balance is preserved. The cost of this operation is Δ.

In each step both operations melt the connected components K_u and K_v to one connected component. Since T contains a (v_0, v_i)–path for every $v_i \in V$ the extended graph G_T will be strongly connected and thus is Eulerian. Since T uses at most k blue arcs we have $|B| = |T \cap E^b| \leq k$. The cost of G_T is as claimed. T contains at most $\frac{n}{2}$ nodes and we need $O(n)$ time to traverse T.

Now let T be a minimum cost (k, v_0)-arborescence. Suppose G_T is not optimal. Let G_T^* be an optimal transport graph with $l(G_T^*) < l(G_T)$. We construct a (k, v_0)–arborescence T^* with $c(T^*) < c(T)$. G_T^* is of the form $G_T^* = (\mathcal{S}, A_\psi^* \dot{\cup} A_\mathcal{R}^* \dot{\cup} A_C^*, B^*)$ where as before $A_\mathcal{R}^*$ consists of arcs or paths representing requests and $A^*_C \dot\cup A^*_\psi$ represents the empty moves. A^*_C is a set of pairs of antiparallel arcs. We again may assume by the additivity of the length function that these arcs link nodes which are adjacent on the circle. As the arcs in $A^*_{\mathcal{R}} \dot\cup A^*_C$ have no effect on the flux, the remaining arcs in A^*_ψ balance the graph.

Since the balancing operation described by Atallah and Kosaraju [1] has minimal cost, we may assume that $A_\psi = A^*_\psi$. Hence the connected components of the balanced graph $(\mathcal{S}, A^*_\psi \cup \mathcal{R})$ coincide with the connected components $K_i \subseteq \mathcal{S}$ of G. We construct an arborescence $T^* = (V, E^{r*} \dot\cup E^{b*})$ of the auxiliary graph as follows.

We shrink the node sets K_i to supernodes u_i. Consider an Eulertour starting with 0 in the Eulerian graph G^*_T. We follow this tour and whenever we enter a connected component for the first time using an arc (i, j), we add this arc to T^*. Since the endnodes of arcs in $A^*_\psi \cup \mathcal{R}$ are in the same component, an arc in the shrunken graph is either in A^*_ψ or an arc on a path representing a request. If the arc (i, j) is A^*_C and $K(j)$ has not been entered before, we add a red arc $(K(i), K(j))$ and cost $2l(i, j)$ to E^{r*}. In the other case let $(i_0, i_1), (i_1, i_2), \ldots, (i_{\ell-1}, i_\ell)$ be the path representing the request (i_0, i_ℓ). For every component $K(i_j), j = 1, \ldots, l-1$ which has not been reached before we add the blue arc $(K(i_{j-1}), K(i_j))$ to E^{b*} with cost Δ. Since no two such arcs have a common endpoint, we cannot add more than $|B^*| \leq k$ blue arcs to E^{b*}. Since the Eulertour reaches every node in $\mathcal{S}$, T^* will be a (k, v_0)–arborescence. Now

$$\begin{aligned} l(G^*_T) &= l(A^*_\psi) + l(A^*_{\mathcal{R}}) + l(A^*_C) + |B^*|\Delta \\ &= l(A_\psi) + l(A_{\mathcal{R}}) + l(A^*_C) + |B^*|\Delta \\ &< l(G_T) \\ &= l(A_\psi) + l(A_{\mathcal{R}}) + l(A_C) + |B|\Delta, \end{aligned}$$

we have $c(T^*) \leq l(A^*_C) + |B^*|\Delta < l(A_C) + |B|\Delta = c(T)$, in contradiction to T being optimal. □

2.2 (k, v_0)–arborescences: Solving special instances

Unfortunately, the complexity status of computing a minimum cost (k, v_0)–arborescence in general directed graphs seems to be open. For our specially structured auxiliary digraphs we will describe a polynomial algorithm. It is based on dynamic programming by looking at a special case. Let $H = (V, E^r \dot\cup E^b)$ be an auxiliary digraph with the following property:

(H1) every arc $(u, v_0) \in E^b$ is contained in a directed blue circuit

For an undirected multigraph $G = (V, E)$ with $E = E^r \dot{\cup} E^b$ a spanning tree T is a *k–tree* if $|T \cap E^b| \leq k$. Gabow and Tarjan [4] have given an algorithm which, for a given cost function $c : E \to \mathbb{R}$, calculates a minimum cost spanning k–tree $T \subseteq E$ in $O(|E| \log |V| + |V| \log |V|)$ steps or decides that no such tree exists. In general, the cost of a minimum spanning k-tree of the underlying graph will give a lower bound on the cost of a (k, v_0)–arborescence. If the directed graph satisfies (H1) this bound is tight:

Lemma 3 *Let the auxiliary digraph H satisfy (H1) and let $H' = (V, E^{r'} \cup E^{b'})$ be the underlying undirected graph of H. For a minimum cost k–tree T of H' we can construct a (k, v_0)–arborescence A for H with cost $c(A) = c(T)$.*

PROOF. Let T be a minimum cost k–tree for H'. Let A be a set of directed arcs of H chosen as follows: consider some $v \in V$. Let u be the node preceding v on the path from v_0 to v in T. If $\{u, v\}$ is red, then A contains a red arc (u, v). If $\{u, v\}$ is blue and $(u, v) \in E^b$, then A contains the blue arc (u, v). Otherwise A contains the blue arc (v, u) which we have to traverse against its orientation. In this case A fails to be an arborescence.

Let $z(T)$ be the number of blue edges $\{u, v\}$ of T such that H contains both blue arcs (u, v) and (v, u). If A is not an arborescence we will transform T into a minimum spanning tree T' with the same number of blue edges and $z(T') = z(T) + 1$. After at most k such steps A is an arborescence.

Assume that A is not an arborescence. On a path P starting in v_0 let $(v, u) \in A$ be the first arc which is traversed against its orientation. By construction of H there is an arc in K_v crossing a node in K_u. Since $(u, v) \notin E^b$, no arc of K_u crosses a node in K_v. This can only happen if on the circle all nodes $\ell \in \mathcal{S}$ with $K(\ell) = K_u$ lie in a segment defined by two nodes i and j with $K(i) = K(j) = K_v$. We distinguish two cases.

(i) 0 is outside the segment between i and j. Since P connects K_0 to K_u without passing through K_v there must be an arc in $\mathcal{R}$ crossing i or j. Hence, for some node w on P, we have $(w, v) \in E^b$ and also $(v, w) \in E^b$. Then $T' = T \setminus \{u, v\} \cup \{w, v\}$ is a minimum cost k–tree with $z(T') = z(T) + 1$.

(ii) 0 is inside the segment between i and j. Let T_v be the subtree of nodes x in T such that the (v_0, x)-path goes through v. Since 0 is between i and j, also $(v, v_0) \in E^b$. By (H1), there exists a blue directed path from v_0 to v. Let P be a shortest such path. Then for every arc (w, x) on P we also have $(x, w) \in E^b$. Since P connects v_0 to v there exists an arc (w, x) on P with $x \in T_v$ and $w \notin T_v$. Then $T' = T \setminus \{u, v\} \cup \{w, x\}$ a minimum cost k–tree with $z(T') = z(T) + 1$. □

Consider an instance satisfying property (H1). For fixed fixed request r_0 and flux value ψ we construct the auxiliary digraph H and compute a minimum cost k–tree. This tree can be transformed in $O(n^2)$ steps into a minimal (k, v_0)–arborescence using Lemma 3. From this we construct an optimal transport graph by Theorem 2. The running time for this approach is in $O(n^2 + m \log n)$.

2.3 Solving general instances

The algorithm of the previous subsection can only be used for instances that fulfill property (H1). We will now solve general instances by using a dynamic programming approach. The idea for this is to identify a subset $\mathcal{S}'$ of S containing all nodes of K_0, with three properties: (i) $\mathcal{S}'$ is a segment of the circle, (ii) $\mathcal{S}'$ has property (H1) and (iii) there is no arc in $\mathcal{R}$ linking a node in $\mathcal{S}'$ with a node in $S \setminus \mathcal{S}'$. These properties will allow us to decompose the problem in two smaller ones on $\mathcal{S}'$ and on $S \setminus \mathcal{S}'$ and merge the solutions appropriately.

We say that a node v in H *violates* (H1) if there is a directed blue path from v to v_0 but no directed blue path from v_0 to v. We successively delete nodes v which violate (H1) until no such node is left. Observe that the resulting subgraph may depend on the sequence in which we delete nodes. It is true, however, that if v violates (H1) in H and u violates (H1) in $H \setminus v$ then u also violates (H1) in H. This implies, in particular, that $\mathcal{S}'$ will always contain K_0 and hence is nonempty.

Let U be the set of nodes in the resulting subgraph which can be reached from v_0 and $\mathcal{S}' = \{s \in S : K(s) \in U\}$. Observe that $\mathcal{S}'$ is a disjoint union of connected components of G.

Lemma 4 *$\mathcal{S}'$ is a segment of the circle.*

PROOF. Suppose that between two nodes in $\mathcal{S}'$ there is a node $i \notin \mathcal{S}'$. Then there is such a node $i \notin \mathcal{S}'$ which on the circle is a left neighbor of some node in $j \in \mathcal{S}'$. Then, by construction of U, there is no red arc between K_i and K_j. Since U is connected there must be a blue arc linking K_j with the part of $\mathcal{S}'$ which is to the left of i. Since the corresponding arc in $\mathcal{R}$ also crosses i, $K(i)$ is reachable from v_0 in H. Hence $K(i)$ does not violate (H1) and could not have been deleted. □

If $\mathcal{S}' \subsetneq S$ let r be the smallest index such that $r \notin \mathcal{S}'$ and ℓ be the largest index such that $\ell \notin \mathcal{S}'$. Then $\mathcal{S}'$ is the segment between $\ell + 1$ and $r - 1$. Let $\mathcal{R}' \subseteq \mathcal{R}$ be the subset of arcs linking nodes in $\mathcal{S}'$ and H' be the auxiliary digraph.

Lemma 5 *H' satisfies (H1).*

PROOF. Suppose there is a node u in H' with a blue arc (u, v_0). Since u does not violate (H1) in H, there is a directed blue path P from v_0 to u in H. If P no longer exists in H', some node $x \in P$ was removed. Since we only remove nodes violating (H1), there is no path from v_0 to x in H, a contradiction. $\square$

Lemma 6 *There is no arc in $\mathcal{R}$ starting in $\mathcal{S}'$ and crossing ℓ or r.*

PROOF. We only discuss the right border r, the case ℓ being similar. Nodes in S which are linked by an arc $(i, j) \in \mathcal{R}$ are in the same connected component of G. So the arc cannot end in r and hence must cross r. Thus there is a blue arc from $K(i)$ to $K(r)$, in contradiction to the fact that r violates (H1). $\square$

This, in particular, implies that in any spanning arborescence of H there are only three possible ways to reach a node in $\mathcal{S} \setminus \mathcal{S}'$ from some node in $\mathcal{S}'$, namely through a red arc $(\ell + 1, \ell)$, a red arc $(r - 1, r)$ or both. Hence we distinguish three cases (left exit, right exit, both exits). On the other hand, there may exist several ways to link $\mathcal{S} \setminus \mathcal{S}'$ back to $\mathcal{S}'$. This, however, can only be the case if some request arc crosses $\mathcal{S}'$. Moreover, if such a request arc exists, an arc (u, v) with $u \in T \setminus U$ and $v \in U$ necessarily has to be blue.

We model the three different cases by adding copies of the limiting nodes and demand arcs (cf. Figure 2):

$$\begin{aligned}
\mathcal{S}_1^1 &:= \mathcal{S}' \cup \{\ell', \ell, r\} & \mathcal{R}_1^1 &:= \mathcal{R}' \cup \{(\ell', \ell), (\ell, \ell'), (0, r), (r, 0)\} \\
\mathcal{S}_2^1 &:= \mathcal{S}' \cup \{\ell, r, r'\} & \mathcal{R}_2^1 &:= \mathcal{R}' \cup (r, r'), (r', r), (0, \ell), (\ell, 0)\} \\
\mathcal{S}_3^1 &:= \mathcal{S}' \cup \{\ell', \ell, r, r'\} & \mathcal{R}_3^1 &:= \mathcal{R}' \cup \{(\ell', \ell), (\ell, \ell'), (r, r'), (r', r)\}
\end{aligned}$$

Request arcs are added in antiparallel pairs to preserve the flux. The distance between a limiting node and its copy is 0.

Let H_i^1 be the corresponding auxiliary graphs. Note that they all satisfy property (H1).

The corresponding problems on $\mathcal{S} \setminus \mathcal{S}'$ are defined as follows:

$$\begin{aligned}
\mathcal{S}_1^2 &:= \mathcal{S} \setminus (\mathcal{S}' \cup r) \text{ with start node } \ell & \mathcal{R}_1^2 &:= \mathcal{R} \setminus \mathcal{R}' \\
\mathcal{S}_2^2 &:= \mathcal{S} \setminus (\mathcal{S}' \cup \ell) \text{ with start node } r & \mathcal{R}_2^2 &:= \mathcal{R} \setminus \mathcal{R}' \\
\mathcal{S}_3^2 &:= \mathcal{S} \setminus \mathcal{S}' \text{ with start node } r & \mathcal{R}_3^2 &:= \mathcal{R} \setminus \mathcal{R}'
\end{aligned}$$

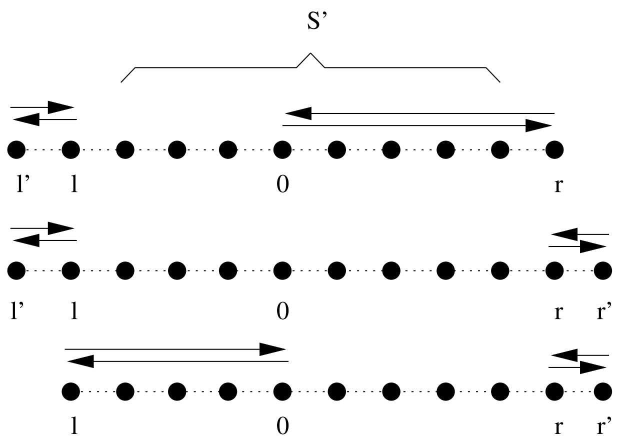

Fig. 2. The graphs corresponding to $\mathcal{S}_i^1, \mathcal{R}_i^1, i = 1,2,3$ without arcs drawn inside $\mathcal{S}_i^1$.

In the third case we identify the nodes ℓ and r. Let H_i^2 be the corresponding auxiliary graphs.

As before we assume that the long arc r_0 and the flux value ψ are fixed. For all $0 \leq p \leq k$ we compute minimum cost (p, v_0)–arborescences $A_i^1(p)$ for $H_i^1, i = 1,2,3$. Since the auxiliary graphs satisfy property (H1) we can apply the algorithm of Section 2.2. We may assume by induction that we can also compute minimum-cost (q, v_0)– arborescences $A_i^2(q)$ for $H_i^2, i = 1,2,3$ for all $0 \leq q \leq k$. Among all arc sets $A_i^1(p) \cup A_i^2(q)$ with $i = 1,2,3$ and $p + q \leq k$ we choose an arc set A with minimum cost.

Theorem 7 *For fixed long arc r_0 and flux value ψ A is a minimum (k, v_0)–arborescence of H.*

PROOF. First observe that A as defined above is a (k, v_0)–arborescence of H. We assume that there is a minimum-cost (k, v_0)– arborescence A^* of H which uses the right exit. The two other cases are shown similarly.

Since A^* uses the right exit, it contains an arc $(r-1, r)$. If all nodes which are reachable from r in A^* are nodes in $H \setminus H_2^1$ then A^* decomposes into two arborescences A_1 of H_2^1 and A_2 of $H \setminus H_2^1$ and the claim follows. If not, by the remarks above, there is a blue arc (u, v) with $u \in H \setminus H_2^1$ and $v \in H_2^1$. We claim that we can replace (u, v) by a blue arc (w, v) with $w \in H_2^1$. This yields an arborescence A' of the same cost and the proof follows by induction.

Consider the problem $\mathcal{S}_2^1$ with demand set $\mathcal{R}_2^1$. If the connected component K_v has a node which lies between ℓ and 0, then the artificial demand $(\ell, 0)$ implies that there is a blue arc (v_0, v) in H_2^1. If all nodes of K_v lie between 0 and r then they have been passed or being crossed on the path from 0 to r. Hence there must exist a request arc in some component K_w crossing the nodes in K_v. Thus (w, v) is a blue arc in H_2^1. □

By Theorem 7 we can reduce the problem to the solution of $3k$ problems satisfying property (H1) and further $3k$ smaller problems. Since the latter problems are defined on three different instances this approach seems to branch and thus to have an exponential running time. We will fix the problem with the help of the following Lemma.

Lemma 8 *Let ℓ and r be as defined above. Then either $K(\ell) = K(r)$ or H has a blue directed circuit containing $K(\ell)$ and $K(r)$.*

PROOF. By definition of ℓ there exists a blue directed $(K(\ell), v_0)$–path P. Then P contains an arc (u, v) with $u \in T \setminus U$ and $v \in U$. By construction, there exists a request $(i, j) \in \mathcal{R}$ starting in $K(i) = K_u$ crossing a node $j \in K(0)$. Since $\mathcal{S}'$ is closed under connected components, (i, j) must cross all nodes of $\mathcal{S}'$. So either $K(r) = K(\ell)$ or (i, j) also crosses ℓ. In the latter case there is a blue directed path from $K(r)$ to $K(\ell)$ and, by symmetry, there is also a path from $K(\ell)$ to $K(r)$. □

Lemma 8 implies that a node u violates (H1) with respect to ℓ if and only if u violates (H1) with respect to r. So when we solve the three different subproblems $(\mathcal{S}_i^2, \mathcal{R}_i^2)$ as before by splitting off segments $\mathcal{S'}_i^2$ satisfying (H1), we can do it in such a way that we generate identical segments. Hence the branching process stops after two steps and returns to a unique subproblem $(\mathcal{S}^2, \mathcal{R}^2) = (\mathcal{S}_1^2 \setminus \mathcal{S'}_1^2, \mathcal{R}_1^2 \setminus \mathcal{R'}_1^2)$.

The algorithm to compute a minimum cost transport plan may be summarized as follows:

(1) enumerate all relevant combinations r_0 and ψ (cf. Lemma 1)
(2) if $\mathcal{S}' = \mathcal{S}$ solve the problem as in Section 2.2
(3) if $\mathcal{S}' \neq \mathcal{S}$
(4) solve the problems $(\mathcal{S}_i^1, \mathcal{R}_i^1, k_1)$ for all $k_1 \leq k$ and $i = 1, 2, 3$ as in Section 2.2
(5) solve the problems $(\mathcal{S'}_i^2, \mathcal{R'}_i^2, k_2)$ for all $k_2 \leq k$ and $i = 1, 2, 3$ as in Section 2.2
(6) solve the problems $(\mathcal{S}^2, \mathcal{R}^2, k_3)$ for all $k_3 \leq k$ recursively
(7) combine the solutions $(\mathcal{S}_i^1, \mathcal{R}_i^1, k_1)$, $(\mathcal{S'}_i^2, \mathcal{R'}_i^2, k_2)$ and $(\mathcal{S}^2, \mathcal{R}^2, k_3)$ for $i = 1, 2, 3$ and $k_1 + k_2 + k_3 \leq k$
(8) select one with minimal cost

Theorem 9 *The transportation problem on a circle with n stations and m requests can be solved in $O(m^2 n^2 \log n)$ steps.*

PROOF. The correctness of the above algorithm follows from the preceding remarks. By Lemma 1 at most one move uses its major arc. If a major arc is used then the flux value is either -1 or 1. If no major arc is used then the flux value is between $-(m+1)$ and $m+1$. So all relevant combinations of major arcs and flux values can be enumerated in $O(m)$ steps. If $\mathcal{S}' = \mathcal{S}$ then the problem satisfies property (H1) and can be solved in $O(n^2 + m \log n)$ steps. If not, we compute arborescences for $6k$ subproblems with property (H1). This process is repeated at most n times. By selecting the best combination, we obtain a minimal (k, v_0)–arborescence. The transformation of this arborescence takes $O(n^2)$ steps, resulting in running time of $O(m^2 n^2 \log n)$ $\square$.

3 Exogenous reload stations

We are now looking at a version of the semi–preemptive routing problem in which all reload nodes are predetermined, i.e. the set B is part of the input. Lemma 1 still holds, and we still only have to care about how to connect the connected components of G. In the endogenous case we could use a reload node in order to connect two components as well, but the consequences were more complicated, because the model implied a multi–criterion objective function.

To solve the exogenous case we construct a directed auxiliary graph H as we did in section 2.1, but we forget about the coloring of the arcs. We are then looking for a minimum cost arborescence rooted in v_0 which can be found in $O(m + n \log n)$ time using a fibonacci heap implementation [2].

Theorem 10 *Let T be an arborescence rooted in v_0 for the uncolored directed auxiliary graph $H = (C, E)$ constructed as above with cost $c(T)$. T can be used to construct a transport graph G_T for (I, ψ, r) with cost $l(G_T) = c(T) + \gamma$ in $O(n)$ time. If T is minimal relative to c then G_T is minimal to l.*

The proof is along the lines of the results in Section 2.3. All possible pairs of (ψ, r) can be enumerated in $O(m)$ time which guides us to an overall running time of $O(m^2 + mn \log n)$.

References

[1] M.J. Atallah and S.R. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, *SIAM Journal Computing.* **17** (1988) 849–869.

[2] H.N. Gabow and Z. Galil and T. Spencer and R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs.

Combinatoria. **6** (1986) 109–122.

[3] G.N. Frederickson and D.J. Guan, Nonpreemptive ensemble motion planning on a tree. *Journal of Algorithms.* **15** (1993) 29–60.

[4] H.N. Gabow and R.E. Tarjan, Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms.* **5** (1984) 80–131.

[5] D.J. Guan, Routing a vehicle of capacity greater than one, *Discrete Applied Mathematics.* **81** (1998) 41–57.