Semi–preemptive routing on a linear and circular track

Dirk Räbiger

Zentrum für Angewandte Informatik Köln, Arbeitsgruppe Faigle/Schrader
Universität zu Köln
Weyertal 80, 50931 Köln

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/non–preemptiveness 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

A robot is given the task of transporting m objects between n stations in the plane. Each (heterogeneous) object is initially located at one of the stations and has to be moved to its destination. The robot is only strong enough to hold one object at a time. A station can be source and destination for several objects. We focus on the special cases where the n stations, given as set $\mathcal{S}$, are arranged on a line or a circle. The distance between neighboring stations s_i, s_j is given by $l(s_i, s_j) \in \mathbb{R}_+$. Every object has a source $s_i \in \mathcal{S}$ and destination $s_j \in \mathcal{S}$ assigned, and we will call this a *request* (s_i, s_j). We will often use *object* as a synonym for request. The set of m requests is given as $\mathcal{R}$. Every station is source or destination of a request, otherwise any unused station will be removed. The robot starts at any predefined station $s_0 \in \mathcal{S}$ and moves back and forth along the track to pick up objects, transport them, and drop them. We want to find the minimal motion to transport every object from the

Email address: raebiger@zaik.de (Dirk Räbiger).

source to its destination and return to the start station afterwards. 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 want to generalize the concepts of preemptiveness/non–preemptiveness and let the robot know a number $k \in \mathbb{N}, k \leq n$ that defines the maximum quantity of *reload–stations* used during the transport. The reload–stations may be *exogenously* given as a subset $B \subseteq \mathcal{S}$ and the robot is allowed to use every station $s \in B$ for reloads. In a different model the k reload–stations have to be *endogenously* determined and the robot has to find out itself which stations are best for reloading in order to minimize the total travel length. Moreover, we introduce a cost value $\Delta \in \mathbb{R}_+$ for every reload station installed. Altough is easy to extend the results for $\Delta < 0$ we abandon this case, because any optimal solution will use exactly k reload nodes. We will give 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 Endogenous reload nodes

2.1 Properties of the line and circle graph

We are given a set of n nodes $\mathcal{S}$ which are either arranged on a line or a circle such that the distance function $l(s_i, s_j) \in \mathbb{R}_+$ is defined on neighboring nodes s_i, s_j. If the nodes are all arranged on a line we can connect the end nodes by using a sufficiently large distance and treat this problem as a special case of the circle case. We will choose the start node s_0 and number all nodes clockwise while we often loosely write s_{i+1} instead of $s_{i+1 \bmod n}$. Referring to the circumference we call the section between the station s_i and s_{i+1} the *interval* i. We are also given a set of m requests $\mathcal{R}$. One can move an object (s_a, s_c) either in *clockwise* or *counter–clockwise* direction along the circumference. Once chosen the direction each request is assigned a unique path along the circumference which uses only neighboring nodes. Now the length of the arc (s_a, s_c) is determined by the sum of the distances along this path. We will say that a request is either moved along its *major* resp. *minor* arc.

We are interested in finding a *transport graph* $G_T = (\mathcal{S}, A, B)$ which is a directed multigraph that contains an Eulerian walk which visits at least all $(s_a, s_c) \in \mathcal{R}$. We partition the arc set A into three subsets $A := A_{\mathcal{R}} \dot\cup A_\psi \dot\cup A_C$ and initially set $A_{\mathcal{R}} = \mathcal{R}$. Within G_T we need to decide for every request in which direction it should be moved. The following result reduces the number of possible combinations to $m + 1$.

Lemma 1 (Atallah, Kosaraju 1988 [1])
In any optimal transport graph, at most one object is moved to its destination along the major arc.

When we are going to determine an optimal transport graph within an algorithm we will iteratively choose one of the requests as being transported via its major arc. For now assume that $r \in \mathcal{R} \cup \{\emptyset\}$ will be this request and that all directions of requests are fixed. (s_a, s_c) *crosses* a node s_b if its unique path contains s_b.

In order to serve all requests we possibly need to use some empty movements while we do not move any object which will be represented by augmenting arcs. We are able to calculate some augmenting arcs which are necessary in every transport graph. The node set of any Eulerian graph must be degree balanced, i.e. $\forall_{s_i \in \mathcal{S}} : \delta^-(s_i) = \delta^+(s_i)$. If any node is unbalanced we will need to add some extra arcs. As the distance function l is additive we can restrict all augmenting arcs to arcs between neighboring nodes. Intuitively speaking we will only use arcs along the circumference. The following definition together with lemma 3 gives us an alternative characterization of the degree balance property on circles.

Definition 2 *The* flow $\phi(i)$ *across an interval* i *is defined as*

$$|\{a \in A \text{ crossing } i \text{ clockwise}\}| - |\{a \in A \text{ crossing } i \text{ counter–clockwise}\}|.$$

Note that $\phi(i)$ also counts arcs crossing i which do neither end in s_i nor s_{i+1}. The $\phi(i)$'s $(i = 0, \ldots, n-1)$ can all be computed in $O(m)$ time. Iff the flow across all intervals equals a value $\psi \in \mathbb{Z}$ then all nodes will be degree balanced:

Lemma 3 (Atallah, Kosaraju 1988 [1])

$$\forall_{s_i \in \mathcal{S}} : \delta^-(s_i) = \delta^+(s_i) \Leftrightarrow \forall_{i=0,\ldots,n-1} : \phi(i) = \psi$$

Given a flow value ψ, we have to add $|\psi - \phi(i)|$ many arcs (s_i, s_{i+1}) (resp. (s_{i+1}, s_i)) to G_T if $\phi(i) < \psi$ (resp. $> \psi$) for all intervals $i = 0, \ldots, n-1$. We will denote this set of augmenting arcs to establish a flow value of ψ by A_ψ. Figure 1 illustrates an example. In $(\mathcal{S}, A_\psi \cup A_{\mathcal{R}})$ all nodes are degree balanced, but not necessarily connected. Thus the graph constructed so far decomposes

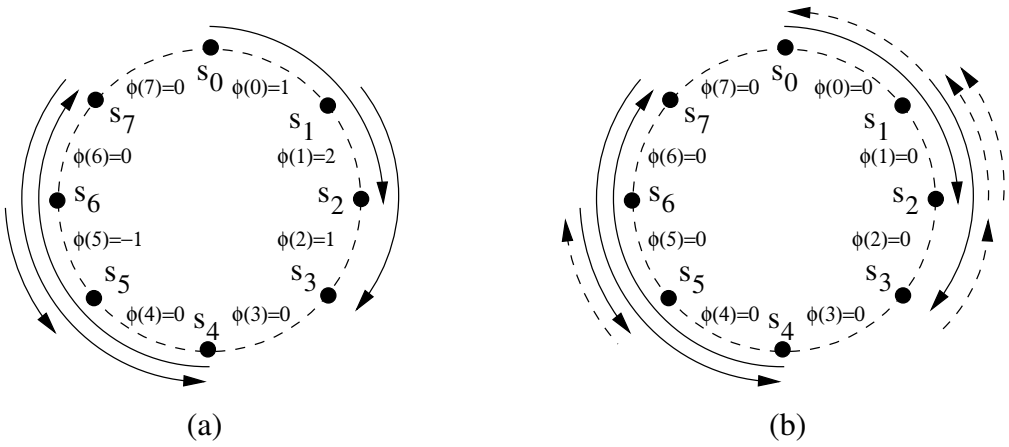

Fig. 1. (a) graph with requests (solid) and flow values (b) same graph with added augmenting arcs (dashed)

into strongly connected components CC_i. Let $I = (\mathcal{S}, l, \Delta, \mathcal{R}, k)$ be the input to our problem. We will often refer to the components CC_i as being defined by (I, ψ, r). The following lemma summarizes which values of ψ we have to take into account and which we will enumerate later.

Lemma 4 (Atallah, Kosaraju 1988 [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 flow value ψ *equals either* 1 *or* -1.
(4) Let the node set be arranged on a line. Then $\psi = 0$.

Hitherto no optimizing has happened yet, because we started constructing G_T by only adding arcs which will be necessary for every graph if were are given the major arc r and ψ. Before we describe the cost function of a transport graph, we consider which alternatives we have in order to make the graph Eulerian. We could add further augmenting arcs between neighboring nodes of different connected components. If we do so then we need to add pairs of anti–parallel arcs $(s_a, s_{a+1}), (s_{a+1}, s_a)$ or otherwise the flow criterion of lemma 3 will be violated. We will denote these additional augmenting arcs by A_C. We are also allowed to determine a set of reload nodes B with $|B| \leq k$. If a request (s_a, s_c) crosses a reload node $s_b \in B$ then we will split the arc (s_a, s_c) and replace it by $(s_a, s_b), (s_b, s_c)$. A request can be split several times. We then refer by $P(s_a, s_c)$ to the path which corresponds to $(s_a, s_c) \in \mathcal{R}$ and only uses nodes of B. By $B(s_a, s_c) \subseteq B$ we denote the set of reload nodes used to split (s_a, s_c). Then let $A_{\mathcal{R}} := \bigcup_{r \in \mathcal{R}} P(r)$. Both operations contribute to make G_T Eulerian. We can now describe the cost of a transport graph by

$c(G_T) = l(A_\psi) + l(A_\mathcal{R}) + l(A_C) + |B|\Delta$. We want to find a minimum cost transport graph. Note that $\gamma := l(A_\psi) + l(A_\mathcal{R})$ is constant for a given (I, ψ, r).

Recall that we currently assume ψ and r to be given. We will construct a directed auxiliary graph $H = (C, E^r \dot\cup E^b)$. For every strongly connected component CC_i defined by (I, ψ, r), create a supernode $v_i \in C$. Let v_0 be the node corresponding to the connected component CC_0 containing s_0. The arcs are either colored red ($e \in E^r$) or blue ($e \in E^b$). In either case an arc is weighted by $c : E^r \cup E^b \to \mathbb{R}$. Starting with $E^r = E^b = \emptyset$, construct H as follows.

- Add a red arc (v_i, v_j) with cost $c(v_i, v_j) = 2l(s_a, s_b)$ to E^r, if there exist nodes $s_a, s_b \in \mathcal{S}$ which are neighbors, but in different connected components $s_a \in CC_i, s_b \in CC_j, i \neq j$. If there are several candidates choose s_a, s_b such that $l(s_a, s_b)$ is minimal.
- Add a blue arc (v_i, v_j) with cost $c(v_i, v_j) = \Delta$ to E^b, if there exists an arc $(s_a, s_c) \in A$ with $s_a, s_c \in CC_i$ which crosses a node $s_b \in \mathcal{S}, s_b \in CC_j, i \neq j$. Note that $(s_a, s_c) \in A$ must correspond to a request $(s_a, s_c) \in \mathcal{R}$, because only these arcs will cross a node.

Definition 5 *Let $G = (V, E^r \dot\cup E^b)$ be a directed multigraph. A (k, r)–*arborescence *is an arborescence $T \subseteq E^r \cup E^b$ rooted in $r \in V$ with $|T \cap E^b| \leq k$.*

Theorem 6 *Let T be a (k, v_0)–arborescence of H constructed as above with cost $c(T)$. T can be used to construct an optimal 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.*

PROOF. Suppose we are given a (k, v_0)–arborescence T of H. We will use T in order to construct a transport graph for our robot problem. We will do so by adding arcs to A_C or splitting arcs of $A_\mathcal{R}$. Start in v_0 and traverse the nodes of T, e.g. by using depth–first–search. Let v_i be the current node and v_j be the current son of v_i. $(v_i, v_j) \in T$ is either colored red or blue.

(1) $(v_i, v_j) \in E^r$: By definition exist neighboring nodes s_a, s_b of different connected components $s_a \in CC_i, s_b \in CC_j$ in G_T. Add two anti–parallel arcs $(s_a, s_b), (s_b, s_a)$ to A in order to maintain degree balancy in both nodes. The cost of adding both arcs to G_T is $c(v_i, v_j)$.
(2) $(v_i, v_j) \in E^b$: By definition exist $s_a, s_c \in CC_i$ with $(s_a, s_c) \in A_\mathcal{R}$ crossing $s_b \in CC_j$. Add s_b to B and replace (s_a, s_c) by $(s_a, s_b), (s_b, s_c)$. The degree balances of all nodes are unchanged. The cost of this operation is Δ.

In T existed a (v_0, v_i)–path for every $v_i \in C$, and after processing T, G_T will contain a (s_0, s_i)–path for every $s_i \in \mathcal{S}$. T uses at most k blue arcs, i.e. $|B| = |T \cap E^b| \leq k$. 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 T. Suppose G_T is not optimal, then let $G_T^* = (\mathcal{S}, A^*, B^*)$ be an optimal transport graph with $l(G_T^*) < l(G_T)$. We will construct a (k, v_0)–arborescence H^* with $c(H^*) < c(T)$. The nodes of any transport graph have to be degree balanced. The minimal augmentation suggested by lemma 3 is unique if we split all augmenting arcs into minimal arcs, and thus we can w.l.o.g. assume that $A_\psi = A_\psi^*$ and $A^* = A_\psi \dot\cup A_\mathcal{R}^* \dot\cup A_C^*$. Let $CC_i \subseteq \mathcal{S}$ be the connected components of $(\mathcal{S}, A_\psi \cup \mathcal{R})$, such that CC_i corresponds to the node v_i of $H = (C, E^r \cup E^b)$. Construct $H^* = (C, E^{r*} \cup E^{b*})$ by shrinking the node sets CC_i of G_T^* into supernodes $v_i \in C$. Let $(s_a, s_b) \in A^*$ be an arc with $s_a \in CC_x, s_b \in CC_y, x \neq y$. Then $(s_a, s_b) \in A_\mathcal{R}^* \dot\cup A_C^*$. If $(s_a, s_b) \in A_C^*$, add a red arc (v_x, v_y) with cost $2l(s_a, s_b)$ to E^{r*}. If $(s_a, s_b) \in A_\mathcal{R}^*$, then there exists a request $(s_c, s_d) \in \mathcal{R}, s_c \in CC_z$, such that $(s_a, s_b) \in P(s_c, s_d)$. Add (v_z, v_y) with cost Δ to E^{b*}. Looking at the variable costs we know that

$$l(G_T^*) < l(G_T) \Leftrightarrow l(A_C^*) + |B^*|\Delta < l(A_C) + |B|\Delta$$

H^* can be assumed to be an arborescence, or otherwise we could remove a pair of anti–parallel arcs from A_C^* and that way reduce the cost of G_T^*. Thus

$$c(H^*) = l(A_C^*) + |B^*|\Delta < l(A_C) + |B|\Delta = c(T)$$

in contradiction to T being optimal. □

Unfortunately we do not know how to calculate a minimum cost (k, v_0)–arborescence. In the following we will first of all look at the undirected version H' of our auxilary graph H and use a known algorithm in order to calculate a minimum cost spanning tree T' using at most k blue edges. After this we will try to direct the edges of T' in order to construct a (k, v_0)–arborescence for H with the same cost. It is obvious that we cannot orient T' in all cases, and thus we will firstly restrict our considerations on instances (I, ψ, r) with a certain property.

2.2 (k, r)–arborescences

Let $H = (C, E^r \dot\cup E^b)$ be a directed multigraph and $c : E^r \dot\cup E^b \to \mathbb{R}$ be a weight function on the arcs. Consider the following properties.

(H1) $\forall_{(v_i,v_j)\in E^r} : c(v_i, v_j) = c(v_j, v_i)$ and $\forall_{(v_i,v_j)\in E^b} : c(v_i, v_j) = \Delta \in \mathbb{R}$
(H2) $(v_i, v_j) \in E^r \Leftrightarrow (v_j, v_i) \in E^r$
(H3) $\forall_{(v_j,v_i)\in E^b} \forall_{(v_0,v_i)\text{–paths } P \text{ in } H - v_j} \exists_{v_x \in P} : (v_x, v_j) \in E^b$

H1 requires the cost function being symmetric on the red arcs and constant on the blue arcs. The second condition demands that the red subgraph (C, E^r)

should be symmetric. The third one is slightly weaker. Whenever there is a blue arc (v_j, v_i) and we examine any directed path from the root to v_i without using v_j there must be a node v_x along this path with $(v_x, v_j) \in E^b$.

Definition 7 *Let $H' = (C, E^{r'} \dot\cup E^{b'})$ be an undirected multigraph. A k–*tree *is a spanning tree $T \subseteq E^{r'} \cup E^{b'}$ with $|T \cap E^{b'}| \leq k$.*

Proposition 8 (Gabow, Tarjan 1984 [4]) *Let $G = (V, E' = E^{r'} \cup E^{b'})$ be an undirected multigraph and $c : E' \to \mathbb{R}$ a cost function. If it exists, a minimum cost k–tree $T \subseteq E'$ can be calculated in $O(|E'| \log |V| + |V| \log |V|)$ time.*

Lemma 9 *Let H fulfill properties H1, H2, and H3, and let $H' = (C, E^{r'} \cup E^{b'})$ be the underlying undirected graph of H. If T' is a minumum cost k–tree of H' then we can construct a (k, v_0)–arborescence T for H with cost $c(T) = c(T')$.*

PROOF. Choose v_0 as root and traverse T' using depth–first–search. Let v_i be the current node and v_j be its son. $(v_i, v_j) \in T'$ can either be colored red or blue. If $(v_i, v_j) \in E^{r'}$ there will always be $(v_i, v_j) \in E^r$ by property H2. If $(v_i, v_j) \in E^{b'}$ and $(v_i, v_j) \in E^b$ exists then we are done, too. Add (v_i, v_j) to T. So suppose $(v_i, v_j) \in E^{b'}$, but $(v_i, v_j) \notin E^b$. Then let P the so far constructed (v_0, v_i)–path on T. By property H3 there exists a node $v_x \in P$ with $(v_x, v_j) \in E^b$. Add this arc to T. $c(T) = c(T')$ by H1. We need $O(m \log n)$ time to construct T from T'. □

2.3 Solving special instances

We will now identify instances of the endogenous semi–preemptive routing problem on a circle whose auxiliary graphs H will satisfy H1 to H3. After we are able to solve these special instances we will present an algorithm for general instances on the circle in section 2.4. Let us at first restrict our considerations on instances (I, ψ, r) with the following property:

(P1) $\nexists (s_a, s_c) \in \mathcal{R} : (s_a, s_c)$ crosses all nodes of CC_0.

Lemma 10 *Let (I, ψ, r) be an instance of the robot transportation problem with fixed ψ, r and H be constructed as described above. If (I, ψ, r) fulfills P1 then H satisfies H1 to H3.*

PROOF. H1 and H2 are provided by construction of H. Let $(v_j, v_i) \in E^b$ be an arbitrary blue arc in H and assume $(v_i, v_j) \notin E^b$. Then let P be any directed (v_0, v_i)–path on $H - v_j$. We will show that P must contain a node

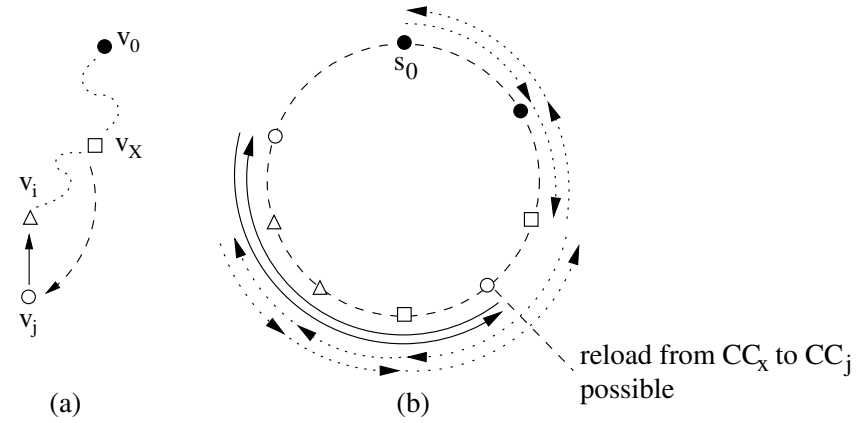

Fig. 2. (a) path P plus the extra arc $(v_x, v_j) \in E^b$, (b) corresponding situation on the circle, using connected components instead of node identifyers.

v_x with $(v_x, v_j) \in E^b$. From the proof of theorem 6 we know that we can use P as a direction how to construct a graph $G_T^P = (\mathcal{S}, A^P, B^P)$ and visit all connected components CC_q with v_q lying on P. In G_T^P no node of CC_j is visited, but all $s' \in CC_i$. $(v_j, v_i) \in E^b$ was constructed, because there exists a request $(s_a, s_c) \in \mathcal{R}$ with $s_a, s_c \in CC_j$ which crosses a node $s_b \in CC_i$. Either s_0 is crossed by (s_a, s_c) as well but then there exists by property P1 a request crossing a node of CC_j and thus an arc $(v_0, v_j) \in E^b$. Or (s_a, s_c) does not cross s_0. Then A^P contains an arc crossing either s_a or s_c, because A^P contains a (s_0, s_b)–path. Let CC_x be the connected component containing the start and end node of this arc. By definition there exists $(v_x, v_j) \in E^b$. Figure 2 illustrates an example. □

Let (I, ψ, r) be given. Using the results above we now can formulate an algorithm `k-RobotLight` to solve instances which fulfill P1. If we combine the results, the algorithm will need $O(m \log n)$ running time.

Algorithm 1 k-RobotLight(I, ψ, r)

Set $A_{\mathcal{R}} := \mathcal{R}$
Calculate A_ψ and connected components CC_i by applying lemma 3
Construct directed auxilary graph $H = (C, E = E^r \cup E^b)$
Construct undirected auxilary graph $H' = (C, E' = E^{r'} \cup E^{b'})$
and $c : E' \to \mathbb{R}_+$.
Calculate minimum cost k–tree T' of (H', c) by applying proposition 8.
Construct (k, v_0)–arborescence T by applying theorems 9 and 10.
Construct transportation graph $G_T = (\mathcal{S}, A_\psi \cup A_{\mathcal{R}} \cup A_C, B)$ by applying theorem 6.

The algorithm of the previous subsection can only be used for instances that fulfill property P1. We will now solve general instances by using a dynamic programming approach. First we identify an area surrounding the start node that we can solve with `k-RobotLight` and allow the robot to use at most $k' = 0, \ldots, k$ reload nodes. We then will determine an area surrounding the already solved section on the circumference. This time we will use at most $k'' = k - k'$ reload nodes, for all $k' = 0, \ldots, k$, and merge the partial solutions.

Let $CC(s)$ be the index i of the connected component $CC_i \subseteq \mathcal{S}$ of $(\mathcal{S}, A_\psi \cup \mathcal{R})$ with $s \in CC_i$. We call $X \subseteq \mathcal{S}$ *consecutive* if for all $s_a, s_c \in X$ exists a walk from s_a to s_c in X by using only neighboring nodes. Let $X \subseteq \mathcal{S}$ be consecutive and $s_0 \in X$. Then we define by $C_X := \{s \in \mathcal{S} | \exists (v_{CC(s)}, v_{CC(t)})$–path in (C, E^b), $\nexists (v_{CC(t)}, v_{CC(s)})$–path in (C, E^b) with $t \in X \subseteq \mathcal{S}\}$ the nodes in $\mathcal{S}$ from which we can reach a node of X just by using reloads, but not the other way round. By $S_X := \max\{X' \subseteq \mathcal{S} \setminus C_X | s_0 \in X', X' \text{ consecutive}\}$ we define the maximal consecutive subset of nodes surrounding s_0 that does not contain a node of C_X. Then let

$$ll_X := \begin{cases} i | s_{i+1} \in S_X, s_i \notin S_X & \text{if } S_X \neq \mathcal{S} \\ 0 & \text{if } S_X = \mathcal{S} \end{cases}$$

$$rl_X := \begin{cases} i | s_{i-1} \in S_X, s_i \notin S_X & \text{if } S_X \neq \mathcal{S} \\ 0 & \text{if } S_X = \mathcal{S} \end{cases}$$

be the *left* resp. *right limit* of S_X on the circumference defined by $X \subseteq \mathcal{S}$. If C_X is empty then S_X will be the whole line and $s_{ll_X} = s_{rl_X} = s_0$.

Initially we will calculate $C_{\{s_0\}}, S_{\{s_0\}}, ll_{\{s_0\}}$, and $rl_{\{s_0\}}$. The nodes $s_i \in S_{\{s_0\}}$ surrounding s_0 can be seen as the part of the circumference on which we can use `k-RobotLight`, i.e. if we restrict the requests to $S_{\{s_0\}}$, such that $\mathcal{R}_{S_{\{s_0\}}} := \mathcal{R} \cap (S_{\{s_0\}} \times S_{\{s_0\}})$ then the modified input $(\mathcal{S}, l, \Delta, \mathcal{R}_{S_{\{s_0\}}}, k, \psi, r)$ will fulfill property P1. In general the definition of S_X implies the following.

Note 1 *If $(\mathcal{S}, l, \Delta, \mathcal{R}_X, k, \psi, r)$ fulfills P1 then $(\mathcal{S}, l, \Delta, \mathcal{R}_{S_X}, k, \psi, r)$ fulfills P1 as well.*

The example shown in figure 3 illustrates that a limiting node $s_{ll_{\{s_0\}}}$ does not necessarily belong to a connected component that contains an arc which crosses the start node. But in H exists a blue (v_i, v_0)–path if $s_{ll_{\{s_0\}}} \in CC_i$ and no blue (v_0, v_i)–path.

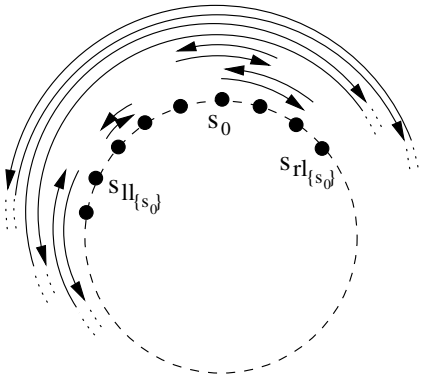

Fig. 3. Graph $(\mathcal{S}, A_\psi \cup \mathcal{R})$ with limits and inner nodes.

Note 2 *Observe that there does not exist a request $(s_i, s_j) \in \mathcal{R}$ with $s_i \in S_X, s_j \in \mathcal{S} \setminus S_X$. (Otherwise there exists a $(v_{CC(s_i)}, v) \in E^b$ for $v \in C_X$.)*

In particular no arc starts in S_X and crosses s_{ll_X} nor s_{rl_X}. Thus the only way to connect the components of S_X with $\mathcal{S} \setminus S_X$ is to use anti–parallel augmenting arcs between a) s_{ll_X} and s_{ll_X+1}, b) s_{rl_X} and s_{rl_X-1}, or c) both.

Lemma 11 *Let $X \subseteq \mathcal{S}$ be consecutive with $s_0 \in X$ and $S_X \neq \mathcal{S}$. Then: $S_{S_{X\cup\{s_{ll_X}\}}} = S_{S_{X\cup\{s_{rl_X}\}}} = S_{S_{X\cup\{s_{ll_X}, s_{rl_X}\}}}$.*

PROOF. By definition $s_{ll_X}, s_{rl_X} \in C_X$, i.e. in (C, E^b) there exists a shortest $(v_{CC(s_{ll_X})}, v_{CC(s_j)})$–path P for some $s_j \in S_X$. Let (v_a, v_b) be the final arc of P. Then there exists a request $(s_i, s_j) \in \mathcal{R}$, starting in $s_i \in CC_a \subseteq \mathcal{S}$ crossing a node $s_j \in CC_b \subseteq S_X$. (s_i, s_j) must cross all nodes of S_X or otherwise exists a blue arc $(v_b, v_a) \in E^b$ in H in contradiction to $s_j \in S_X$. Thus s_{ll_X}, s_{rl_X} are either in the same connected component or (s_i, s_j) crosses s_{rl_X}. In both cases exists a $(v_{CC(s_{ll_X})}, v_{CC(s_{rl_X})})$–path in (C, E^b). We can proof the existance of a $(v_{CC(s_{rl_X})}, v_{CC(s_{ll_X})})$–path in (C, E^b) analogously. Thus there is a $(v_{CC(s_{rl_X})}, v')$–path for any node $v' \in C$ in (C, E^b) iff there is a $(v_{CC(s_{ll_X})}, v')$–path. □

For a given consecutive subset $X \subseteq \mathcal{S}$ with $s_0 \in X$ consider the following inputs which we will abbreviate using the notion for intervals, and which are illustrated in figure 4. Note that if $S_X \neq \mathcal{S}$ these subproblems can be seen as problems on a path.

$(I^{[X)}, k, \psi, r) := (\mathcal{S}', l', \Delta, \mathcal{R}', k, \psi, r)$ with

$$\mathcal{S}' := \quad S_X \cup \{s'_{ll_X}, s_{ll_X}, s_{rl_X}\}$$

$$\mathcal{R}' := \quad (\mathcal{R} \cap (S_X \times S_X)) \cup \{(s'_{ll_X}, s_{ll_X}), (s_{ll_X}, s'_{ll_X}), (s_0, s_{rl_X}), (s_{rl_X}, s_0)\}$$

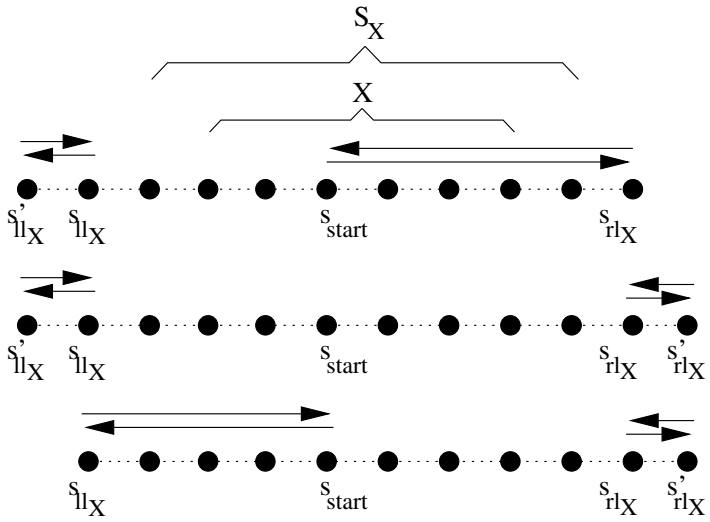

Fig. 4. The graphs corresponding to $(I^{[X)}, k, \psi, r), (I^{[X]}, k, \psi, r)$, and $(I^{(X]}, k, \psi, r)$ without arcs drawn inside S_X.

$(I^{[X]}, k, \psi, r) := (\mathcal{S}', l', \Delta, \mathcal{R}', k, \psi, r)$ with

$$\mathcal{S}' := \quad S_X \cup \{s'_{ll_X}, s_{ll_X}, s_{rl_X}, s'_{rl_X}\}$$

$$\mathcal{R}' := \quad (\mathcal{R} \cap (S_X \times S_X)) \cup \{(s'_{ll_X}, s_{ll_X}), (s_{ll_X}, s'_{ll_X}), (s'_{ll_X}, s_{ll_X}), (s_{ll_X}, s'_{ll_X})\}$$

$(I^{(X]}, k, \psi, r) := (\mathcal{S}', l', \Delta, \mathcal{R}', k, \psi, r)$ with

$$\mathcal{S}' := \quad S_X \cup \{s_{ll_X}, s_{rl_X}, s'_{rl_X}\}$$

$$\mathcal{R}' := \quad (\mathcal{R} \cap (S_X \times S_X)) \cup \{(s'_{rl_X}, s_{rl_X}), (s_{rl_X}, s'_{rl_X}), (s_0, s_{ll_X}), (s_{ll_X}, s_0)\}$$

In all three cases let l' be defined as l restricted to X and with zero distance between an extra node and its original like s'_{ll_X}, s_{ll_X}.

$(I^{[X)}, k, \psi, r)$ assumes that S_X and $\mathcal{S} \setminus S_X$ will be connected by using anti–parallel augmenting arcs at the counter–clockwise end of S_X, $(I^{[X]}, k, \psi, r)$ on both ends, and $(I^{(X]}, k, \psi, r)$ at the clockwise end of S_X. Let $G_{[X)}$ be a transport graph for $(I^{[X)}, k, \psi, r)$. If we remove all arcs of $P(s_0, s_{rl_X})$ and $P(s_{rl_X}, s_0)$ then $G_{[X)}$ will decompose to several connected components while s_0 and s_{ll_X} will remain together in the same connected component. We will iteratively presume that either $G_{[X)}$, $G_{[X]}$, or $G_{(X]}$ can be used as a part of the general optimal solution. Within the next step an extended area X' surrounding the already solved area X will be identified. Both X and X' each have three possible outcomes which we will combine. E.g. let us assume that in the optimal solution for (I, ψ, r) at most k' reload nodes are used inside X and k'' reload nodes inside X'. Let us further assume that this solution connects X with $\mathcal{S} \setminus X$ by using arcs between s_{ll_X}, s_{ll_X+1} and that X' is connected to $\mathcal{S} \setminus X'$ by using arcs between $s_{rl_{X'}}, s_{rl_{X'}-1}$. Figure 5 illustrates this example. Given a transport graph $G^{k'}_{[X)}$ as solution for $(I^{[X)}, k', \psi, r)$ we fix the arcs and

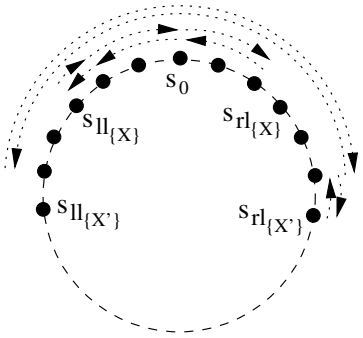

Fig. 5. The idea of the dynamic program. A dotted arc represents a set of arcs.

reload nodes of $G^{k'}_{[X)}$ and construct a new input $(I^{(X']}_{[X)}, k'', \psi, r)$ which is the same as $(I^{(X']}, k'', \psi, r)$ but without any choices inside X.

Algorithm 2 Fix$(G^{k'}_{[X)}, I^{[X')}, \psi, r)$:

$G^{k'}_{[X)} = (\mathcal{S}'_X, A_\psi \cup A\mathcal{R} \cup A_C, B_X)$, $I^{[X')} = (\mathcal{S}'_{X'}, l', \Delta, \mathcal{R}_{\mathcal{X}'})$
For all $e \in A_C$
  Create an artificial request e, add e to $\mathcal{R}_C$
Let $(s_a, s_c) \in \mathcal{R}_{\mathcal{X}'}$ be a request crossing s_0.
Sort $s_{b_{i=1,\ldots,q}} \in B_X$ after the appearance on the unique (s_a, s_c)–path along the circumference using only neighboring nodes.
Let $\mathcal{R}' := (\mathcal{R}_{X'} \setminus \{(s_a, s_c)\}) \cup \{(s_a, s_{b_1}), (s_{b_1}, s_{b_2}), \ldots, (s_{b_q}, s_c)\} \cup \mathcal{R}_C$
Return $(\mathcal{S}', l', \Delta, \mathcal{R}')$

The procedure `Fix` uses the solution $G^{k'}_{[X)}$ of the subproblem $(I^{[X)}, k', \psi, r)$ and fixes the movement of all objects within $S_X \cup \{s_{ll_X}, s_{rl_X}\}$. As we have seen before the problem can be reduced to the question how to augment connected components and where to use reload nodes. Within algorithm `k-Robot` we can guarantee to find a request crossing s_0 as long $X \subsetneq \mathcal{S}$. $G^{k'}_{[X)}$ contains a path from s_0 to s_{ll_X}. After applying the procedure both nodes will always be in the same connected component. We can assume that the s_{b_i}'s are already indexed, so that we do not need to care about sorting time. Thus the running time $O(|X|)$ of `Fix` depends only on the cardinality of X. The algorithms to fix transport graphs of the other types $G^{k'}_{[X]}$ and $G^{k'}_{(X]}$ are constructed analogously.

Theorem 12 *The algorithm* `k-Robot` *finds an optimal solution for a given input* I.

PROOF. By lemma 4 exists an optimal transport graph G^*_T using a flow value $\psi \in [-m-1, m+1]$, and by lemma 1 it moves at most one $r \in \mathcal{R}$ along the major arc. The algorithm enumerates all possible values. In the following we assume that we know the optimal ψ and r. We will prove by induction on the number t of while loop iterations:

Algorithm 3 k-Robot($I = (\mathcal{S}, l, \Delta, \mathcal{R}, k)$)

Set $z := \infty$
For all $r \in \mathcal{R} \cup \{\emptyset\}$
  For all $\psi \in [-m-1, \ldots, m+1]$
    If $r \neq \emptyset$ and $|\psi| \neq 1$
      Next for
    If $S_{\{s_0\}} \neq \mathcal{S}$
      For all $k' = 0, \ldots, k$
        $G^{k'}_{[\{s_0\})} =$ `k-RobotLight(`$I^{[\{s_0\})}, k', \psi, r$`)`
        $G^{k'}_{[\{s_0\}]} =$ `k-RobotLight(`$I^{[\{s_0\}]}, k', \psi, r$`)`
        $G^{k'}_{(\{s_0\}]} =$ `k-RobotLight(`$I^{(\{s_0\}]}, k', \psi, r$`)`
      Set $X = \{s_0\}, X' = S_{X \cup \{sll_X, srl_X\}}$
      While $X \neq \mathcal{S}$
        For all $k' = 0, \ldots, k$
          Construct $I^{[X')}_{[X)}, I^{[X')}_{[X]}, I^{[X')}_{(X]}, I^{[X']}_{[X)}, I^{[X']}_{[X]}, I^{[X']}_{(X]}, I^{(X']}_{[X)}, I^{(X']}_{[X]}, I^{(X']}_{(X]}$
            by using `Fix`.
          For all $k'' = 0, \ldots, k - k'$
            $G^{k''}_{[X')} = \text{argmin}\{l(G') | G' \in \{$
                `k-RobotLight(`$I^{[X')}_{[X)}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{[X')}_{[X]}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{[X')}_{(X]}, k'', \psi, r$`)`$\}\}$
            $G^{k''}_{[X']} = \text{argmin}\{l(G') | G' \in \{$
                `k-RobotLight(`$I^{[X']}_{[X)}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{[X']}_{[X]}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{[X']}_{(X]}, k'', \psi, r$`)`$\}\}$
            $G^{k''}_{(X']} = \text{argmin}\{l(G') | G' \in \{$
                `k-RobotLight(`$I^{(X']}_{[X)}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{(X']}_{[X]}, k'', \psi, r$`)`,
                `k-RobotLight(`$I^{(X']}_{(X]}, k'', \psi, r$`)`$\}\}$
          End for
        End for
        $X = X', X' = S_{X \cup \{sll_X, srl_X\}}$
      End while
      If $l(G^k_{[X]}) < z$
        $G^* = G^k_{[X]}, z = l(G^k_{[X]})$
  End for
End for
Return G^*

$t = 0$: Then $X = S_{\{s_0\}} = \mathcal{S}$, and (I, ψ, r) fulfills property $P1$. $G^k_{[X]}$ is the solution to the problem $(I^{[X]}, k, \psi, r)$ which is (I, ψ, r) with additional requests of length 0.

$t \to t+1$: From iteration t we are given $3(k+1)$ solutions $G^{k'}_{(X]}$, $G^{k'}_{[X]}$, and $G^{k'}_{[X)}$, for $k' = 0, \ldots, k$. Within iteration $t+1$ we will construct $9(k+1)$ problems of the types $I^{[X')}_{[X)}, \ldots, I^{(X']}_{(X]}$ which can be solved by `k-RobotLight` to optimality. We will exemplary assume that we know an optimal solution $G'_T = (\mathcal{S}, A', B')$ for (I, ψ, r) with $(s_{ll_{X'}}, s_{ll_{X'}+1}), (s_{ll_{X'}+1}, s_{ll_{X'}}) \notin A'$, $(s_{rl_{X'}}, s_{rl_{X'}-1})$, $(s_{rl_{X'}-1}, s_{rl_{X'}}) \in A'$, (s_{ll_X}, s_{ll_X+1}), $(s_{ll_X+1}, s_{ll_X}) \in A'$, and (s_{rl_X}, s_{rl_X-1}), $(s_{rl_X-1}, s_{rl_X}) \notin A'$. Let $|B' \cap S_X| \leq k', |B' \cap S_{X'}| \leq k'' + k'$.

Obviously G'_T can be used to construct an optimal transport graph $G^*_T = (S_{X'} \cup \{s_{ll_{X'}}, s_{rl_{X'}}, s'_{rl_{X'}}\}, A^*, B^*)$ for $(I^{(X']}, k''+k', \psi, r)$. Thus we have to show that the minimum cost solution of the problem $(I^{(X']}_{[X)}, k'', \psi, r)$ is as well a minimum cost solution for $(I^{(X']}, k'' + k', \psi, r)$. The other cases are left to the reader. If $X = \mathcal{S}$ then $G^k_{[X]}$ will be the optimal transport graph for $(I^{[X]}, k, \psi, r)$ which is equivalent to (I, ψ, r).

Certainly $G^{k''}_{(X']}$ is a feasible solution for $(I^{(X']}, k'' + k', \psi, r)$. Suppose $l(G^*_T) < l(G^{k''}_{(X']})$. As before we can assume that $l(G^*_T) = l(A^*_\psi) + l(A^*_\mathcal{R}) + l(A^*_C) + |B^*|\Delta$. The first two terms are constant for a given input. Let $Y := S_{X'} \setminus S_X$, then we can further decompose $A^*_C = (A^*_C \cap (S_X \times S_X)) \dot\cup (A^*_C \cap (Y \times Y)) \dot\cup \{(s_{ll_X+1}, s_{ll_X}), (s_{ll_X}, s_{ll_X+1})\}$. We also want to partition the arc set A of $G^{k''}_{(X']} = (S_{X'} \cup \{s_{ll_{X'}}, s_{rl_{X'}}, s'_{rl_{X'}}\}, A, B)$ according to the connected components defined by $(I^{(X']}, k'' + k', \psi, r)$, so let $A_C = A^{S_X}_C \dot\cup A^Y_C \dot\cup \{(s_{ll_X+1}, s_{ll_X}), (s_{ll_X}, s_{ll_X+1})\}$ be its variable portion. Thus:

$$
\begin{gathered}
l(G^*_T) < l(G^{k''}_{(X']}) \Leftrightarrow \\
(A^*_C \cap (S_X \times S_X)) + |B^* \cap S_X|\Delta + l(A^*_C \cap (Y \times Y)) + |B^* \cap Y|\Delta \\
< l(A^{S_X}_C) + k'\Delta + l(A^Y_C) + k''\Delta
\end{gathered}
\tag{1}
$$

Construct $G_X = (S_X \cup \{s'_{ll_X}, s_{ll_X}, s_{rl_X}\}, A_X, B_X)$: Let initially $B_X := B^* \cap S_X$ and $A_X := (A^* \cap (S_X \times S_X))$. Add anti–parallel arcs $(s'_{ll_X}, s_{ll_X}), (s_{ll_X}, s'_{ll_X})$ to A_X. Let $B_{X^r} \subseteq B_X$ be the set of nodes lying in clockwise direction from s_0 within S_X. Index these nodes according to their clockwise appearance. Then add the arcs (s_0, s_{b_1}), $(s_{b_1}, s_{b_2}), \ldots, (s_{b_{|B_{X^r}|}}, s_{rl_X}), (s_{rl_X}, s_0)$. Let P be a path from s_0 to s_{ll_X} in A^*. P exists, otherwise G^*_T would not be connected. For all nodes $s_b \in B_{X^l} := B_X \setminus B_{X^r}$ exists exactly one arc (s_a, s_c) in P crossing s_b. Replace $(s_a, s_c) \in A_X$ by (s_a, s_b) and (s_b, s_c). Now G_X is a transport graph for

$(I^{[X)}, k', \psi, r)$. As $G^{k'}_{[X)}$ is optimal for $(I^{[X)}, k', \psi, r)$, the cost of G_X must be at least $l(G_X) \geq l(G^{k'}_{[X)})$, thus $l(A^*_C \cap (S_X \times S_X)) + |B^* \cap S_X|\Delta \geq l(A^{S_X}_C) + k'\Delta$. Applying this to inequality 1 gives us:

$$l(A^*_C \cap (Y \times Y)) + |B^* \cap Y|\Delta < l(A^Y_C) + k''\Delta \tag{2}$$

Construct $G_Y = (S_{X'} \cup \{s_{ll_{X'}}, s_{rl_{X'}}, s'_{rl_{X'}}\}, A_Y, B_Y)$ by using components of $G^{k'}_{[X)}$ within S_X and components of G^*_T within Y: Let $B_Y := (B^* \cap Y) \cup (B \cap S_X)$, and initially let $A_Y = (A^* \cap (Y \times Y)) \cup (A \cap (S_X \times S_X)) \cup \{(s_{ll_X}, s_{ll_X+1}), (s_{ll_X+1}, s_{ll_X})\}$. Let $(s_a, s_c) \in (A^* \cap (Y \times Y))$ be the arc crossing the nodes of S_X and which was split in procedure `Fix`. Sort all $b_i \in (B \cap S_X)$ according to their appearance on the unique (s_a, s_c)–path along the circumference. Then replace (s_a, s_c) by $(s_a, s_{b_1}), (s_{b_1}, s_{b_2}), \ldots, (s_{b_{|B \cap S_X|}}, s_c)$. Now G_Y is a transport graph for $(I^{(X']}_{[X)}, k'' + k', \psi, r)$ and by applying inequality 2 on its variable costs we follow

$$\begin{aligned} l(G_Y) &= l(A^*_C \cap (Y \times Y)) + |B^* \cap Y|\Delta + l(A_C \cap (S_X \times S_X)) + |B \cap S_X|\Delta \\ &= l(A^*_C \cap (Y \times Y)) + |B^* \cap Y|\Delta + l(A^{S_X}_C) + k'\Delta \\ &< l(A^Y_C) + k''\Delta + l(A^{S_X}_C) + k'\Delta \end{aligned}$$

in contradiction to $G^{k''}_{(X']}$ being optimal for the subproblem defined by $I^{(X']}_{[X)}$.

At the end of each while loop iteration we enlarge X by at least two nodes. We can do this although $G^{k''}_{[X')}$ and $G^{k''}_{(X']}$ do not exactly cover the same set of nodes. Lemma 11 assures that the new limits will be a good choice for both.

If $r = \emptyset$ then $(m + 3)$ many ψ values will be enumerated. If $r \in \mathcal{R}$ then $\psi \in \{-1, 1\}$ will be enumerated (see lemma 4). Thus the two outer for–loops will be passed $m + 3 + 2m \in O(m)$ times. Beside this the algorithm's running time mainly depends on the three nested loops which separately need $O(n)$ time, and on the time `k-RobotLight` needs. We can roughly estimate its worst case time by $O(n^3 m^2 \log n)$. □

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 4 still holds, and we still only have to care about how to connect the connected components defined by (I, ψ, r). 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–criterium 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 13 *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 analog to the one of theorem 6. All possible pairs of (ψ, r) can be enumerated in $O(m)$ time which guides us to a 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.