Semi–preemptive routing on a line

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.

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

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 case when the n stations, given as the set $\mathcal{S}$, are arranged on a line. There are exactly two stations at both ends of the line that have only one neighbor, any other (inner) station has exactly two neighbors. The distance between neighboring stations i, j is given by $l(i, j) \in \mathbb{R}_+$. If two stations are not neighbors their distance will be the sum of the distances over the unique path using only neighboring stations. 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 one terminal station $s_0 \in \mathcal{S}$ of the line and moves back and forth along the line to

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

pick up objects, transport them, and drop them. We want to find the minimal motion to transport every object from the source to its destination. Typically, one distinguishes between a *non–preemptive* and a *preemptive* version of the problem. In the first case any object must 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 studied by ATALLAH,KOSARAJU in [1]. A nice overview of closely related problems is given in [3].

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* the transport is allowed to use. The reload–stations may be *exogenously* given as a subset $B \subseteq \mathcal{S}$ and the robot is allowed to use every node $s \in B$ 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. We will only deal with the more interesting endogenous case in this Extended Abstract. The exogenous case can be easily deducted from it. Moreover, we introduce a cost value $\Delta \in \mathbb{R}$ for every reload station installed.

2 The model and its properties

The goal is to construct a directed multigraph $G_T = (\mathcal{S}, E)$ that the robot can take as routing advice, in the sense that it will move according to the edge set E. If E contains an edge (s_i, s_j) then the robot will move to j when it arrives in i. We cannot use any graph for routing, thus G_T needs certain properties which we will specify soon. The node set $\mathcal{S}$ corresponds to the set of stations. To represent this in G_T, two nodes are *neighboring* if their corresponding stations on the line are neighboring. The distance between nodes corresponds to the stations on the line, and we will use the distance function $l : E \rightarrow \mathbb{R}_+$. We number all nodes continuously according to their appearance on the line, thus we can say the nodes s_i, s_{i+1} are neighboring for all $i = 0, \ldots, n-2$.

A request $r_1 = (s_a, s_c) \in \mathcal{R}$ *crosses* station b if $a < b < c$ or $c < b < a$. Suppose the robot transports object r_1 and b was declared to be a reload station. The robot picks up r_1 at node s_a and starts moving towards s_c. Along the way it will pass s_b. At this point it may drop r_1 and transport any other object, before it returns to s_b and continues the transportation of r_1 towards s_c.

Given a directed multigraph $G = (V, A)$, we denote by $\delta^-(v)$ $(\delta^+(v))$ the number of incoming (outgoing) edges of $v \in V$. We will now define what kind of graphs the robot needs. It is easy to see that all the following properties are necessary and sufficient in order to describe a feasible routing for the robot.

Definition 1 *A* transport graph $G_T = (\mathcal{S}, E)$ *has the following properties:*

(1) For every $(s_i, s_j) \in \mathcal{R}$ *exists a one–to–one sequence* $((s_{x_0}, s_{x_1}), (s_{x_1}, s_{x_2}), \ldots, (s_{x_{p-1}}, s_{x_p}))$ *where* $\forall_{q=0,\ldots,p-1} : (s_{x_q}, s_{x_{q+1}}) \in E$*,* $\forall_{q=1,\ldots,p-1} : s_{x_q} \in B \subseteq \mathcal{S}$*,*

$$\textit{such that} \begin{cases} s_i = s_{x_0} < s_{x_q} < s_{x_{q+1}} < s_{x_p} = s_j & \textit{if } i < j, \textit{ and} \\ s_i = s_{x_0} > s_{x_q} > s_{x_{q+1}} > s_{x_p} = s_j & \textit{if } i > j \end{cases}$$

(2) G_T *uses at most* k *reloads, i.e.* $|B| \leq k$*.*

(3) G_T *is degree balanced, i.e.* $\forall_{s \in \mathcal{S}} : \delta^+(s) = \delta^-(s)$

(4) G_T *is connected*

The *cost* of such a graph G_T is $l(G_T) = \sum_{e \in E} l(e) + |B|\Delta$. Property 1 demands that the movement of every object has to be performed, but it allows to split the single edge (s_i, s_j) into several smaller edges along the unique path from i to j. Note that every request needs its own sequence as the robot can only hold one item at a time. Property 2 permits to split these requests at most k times. Property 3 is known as the *Euler criterium*. Together with the last property it assures that the robot will be able to return to the start node, because every connected component will be strongly connected.

Referring to the line, we call the section between the stations s_i and s_{i+1} the *interval* i, and $l(i) = l(s_i, s_{i+1})$ is the *length of interval* i.

Definition 2 *The* flow $\phi(i)$ *across an interval* i *is defined as*
$\phi(i) = |\{(s_a, s_b) \in E | a \leq i < b\}| - |\{(s_b, s_a) \in E | a \leq i < b\}|$

We know that we do not have to care about the node degrees if and only if we establish a zero flow across all the intervals.

Lemma 3 (Atallah, Kosaraju 1988 [1])
$\forall_{s_i \in \mathcal{S}} : \delta^-(s_i) = \delta^+(s_i) \Leftrightarrow \forall_{i=0,\ldots,n-1} : \phi(i) = 0.$

We now want to construct a minimum cost transport graph G_T. Suppose E initially consists of all request edges $(s_i, s_j) \in \mathcal{R}$. Together with the properties of l, one can deduct from Lemma 3 how to minimally augment G_T in order to degree balance all nodes $s_i \in \mathcal{S}$. We do so by adding augmenting edges to E. For any interval i we have to add $\phi(i)$ edges (s_i, s_{i+1}) (resp. (s_{i+1}, s_i)) to E if $\phi(i) < 0$ (resp. > 0). The cost of such an augmentation is $\gamma := \sum_{i=0}^{n-1} l(i)|\phi(i)|$. Figure 1 illustrates an example. After adding these edges to E all nodes of G_T are degree balanced, but G_T is not necessarily connected.

Note 1 *For a given connected component* $CC \subseteq \mathcal{S}$ *and an arbitrary node* $s_i \in CC$*, all requests of* CC *can be independently transported, such that the robot starts and ends in* s_i*.*

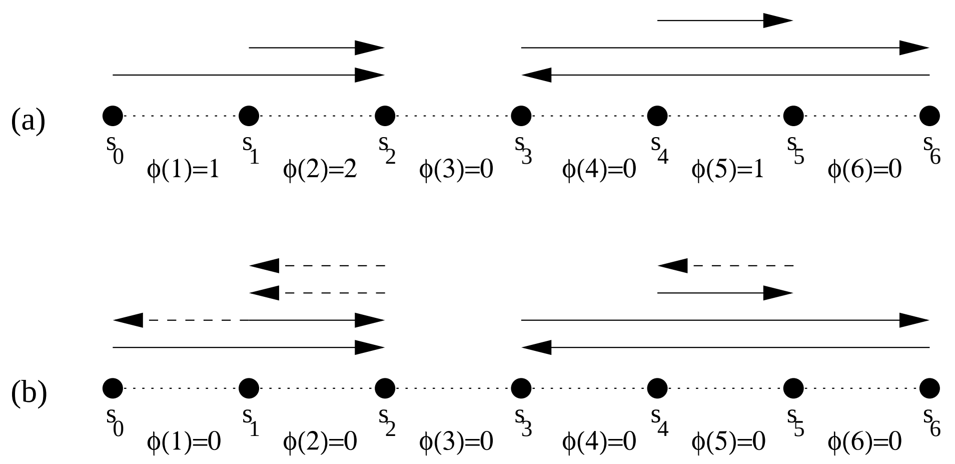

Fig. 1. (a) graph with request edges (solid) and flow values (b) same graph with added augmenting edges (dashed)

In order to transport all objects of every component the robot must be able to switch from one component to another. The robot has two choices to join connected components. First, it could add more augmenting edges between neighboring nodes of different connected components. Doing so, it will always add two anti–parallel edges (s_i, s_{i+1}), (s_{i+1}, s_i), or otherwise the flow criterium of Lemma 3 will be violated. Alternatively, the robot may use one of the k reload stations to switch from one connected component to another and back again. To calculate the optimal solution out of these alternatives we will construct an auxiliary graph H, whose node set will be the connected components of G_T. We use a known algorithm to construct a spanning tree T for H. In one special case we will need to do some local repair on T in order to use it as direction of how to traverse the connected components of G_T.

3 Endogenous reload stations

Construct an undirected auxiliary graph $H = (C, E^r \dot{\cup} E^b)$. For every strongly connected component CC_i of G_T, create a supernode $v_i \in C$. The edges are either colored red ($e \in E^r$) or blue ($e \in E^b$). In either case an edge 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 edge $(v_i, v_j) \in E^r$ with cost $c(v_i, v_j) = l(a, b)$ to H, if there exist nodes $s_a, s_b \in \mathcal{S}$ which are neighbors, but in different connected components $s_a \in CC_i, s_b \in CC_j, i \neq j$ of G_T.
- Add a blue edge $(v_i, v_j) \in E^b$ with cost $c(v_i, v_j) = \Delta$ to H, if there exists an edge $(s_a, s_c) \in \mathcal{R}$ with $s_a, s_c \in CC_i$ which crosses a node $s_b \in \mathcal{S}, s_b \in CC_j, i \neq j$ in G_T.

We know that H contains a spanning tree on the red edges, because every node $s_i \neq s_0$ has a neighbor s_{i-1} "towards" s_0.

Definition 4 *Let $G = (V, E^r \dot{\cup} E^b)$ be an undirected graph. A k–*tree *is a spanning tree $T \subseteq E^r \cup E^b$ with $|T \cap E^b| \leq k$.*

Proposition 5 (Gabow, Tarjan 1984 [2]) *Let $G = (V, E = E^r \cup E^b)$ be an undirected graph and $c : E \to \mathbb{R}$ a cost function. If it exists, a minimal cost k–tree $T \subseteq E$ can be calculated in $O(|E| \log |V| + |V| \log |V|)$ time.*

Theorem 6 *Let T be a minimal k–tree of H with cost $c(T) = \sum_{e \in T \cap E^r} c(e) + |T \cap E^b| \Delta$. T can be used to construct an optimal transport graph G_T^* with cost $l(G_T^*) = 2 \sum_{e \in T \cap E^r} c(e) + \beta \Delta + \gamma$ in $O(n \log n)$ time, where γ is the cost to degree balance the initial graph and*

$$\beta = \begin{cases} |T \cap E^b| & if \Delta \geq 0 \\ k & if \Delta < 0 \end{cases}$$

The idea of using the tree T to construct G_T^* is the following, starting with $G_T^* = G_T$. Let CC_0 be the connected component containing the start node s_0. Choose v_0 as root and traverse T using depth-first search. Suppose v_i is the current node and v_j its son. If $(v_i, v_j) \in E^r$ there exist two neighboring nodes $s \in CC_i$ and $t \in CC_j$. Add both edges $(s,t), (t,s)$ to G_T^*. If $(v_i, v_j) \in E^b$ there is a chance that an edge starting and ending in CC_i crosses a node $t \in CC_j$. In this case split the request edge at t and add t to B. Otherwise there is an edge starting and ending in CC_j that crosses a node $t \in CC_i$. We can exploit the structure of the line to show that there must be another blue edge out of an anchestor of v_j that we can use instead. In case $\Delta < 0, |T \cap E^b| < k$, we can choose any node $t \in \mathcal{S} \setminus B$ and add it to B. Repeating this step until $|T \cap E^b| = k$ will improve the quality. If there is a transport graph D with $l(D) < l(G_T^*)$, we can always construct a k–tree T' with $c(T') < c(T)$.

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 R.E. Tarjan, Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms.* **5** (1984) 80–131.

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