Discrete Applied Mathematics 161 (2013) 454–465

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The uncapacitated swapping problem on a line and on a circle

Shoshana Anily [*], Aharona Pfeffer

Faculty of Management, Tel Aviv University, Tel Aviv 69978, Israel

ARTICLE INFO

Article history:
Received 8 November 2010
Received in revised form 27 June 2012
Accepted 9 July 2012
Available online 30 July 2012

Keywords:
Routing problems: swapping problem
Stacker Crane problem

ABSTRACT

The *uncapacitated swapping problem* is defined by a graph consisting of n vertices, and m object types. Each vertex of the graph is associated with two object types: the one that it currently holds, and the one it demands. Each vertex holds or demands at most one unit of an object. The problem is balanced in the sense that for each object type, its total supply equals its total demand. A vehicle of unlimited capacity is assumed to transport the objects in order to fulfill the requirements of all vertices. The objective is to find a shortest route along which the vehicle can accomplish the rearrangement of the objects, given designated initial and terminal vertices. The uncapacitated swapping problem on a general graph, including a tree graph, is known to be *NP-Hard*. In this paper we show that for the line and circle graphs, the problem is polynomially solvable: we propose an $O(n)$-time algorithm for a line and an $O(n^3)$-time algorithm for a circle.

1. Introduction

The *Swapping Problem* (SP) is defined by a graph and a number of object types, so that each vertex of the graph may initially hold an object of a certain type, and it may request an object of a possibly different type. For each object type, its total supply equals its total demand. Such a problem is said to be *balanced*. The objective of the SP is to find shortest routes along which the vehicle can accomplish the rearrangement of the objects. In this paper we consider the *Uncapacitated SP* (USP) where a single uncapacitated vehicle transports the objects along the edges of the graph in order to fulfill the demands of the vertices. We focus here on two special graphs, a line and a circle.

The SP was first introduced in [3]. The paper considers a complete undirected graph satisfying the triangle inequalities, and a vehicle of a unit capacity. This was the first in a stream of papers that deal with variants of the SP. The variants differ in the following aspects:

- The structure of the underlying graph, e.g., line, circle, tree, or general graph.
- The vehicle's capacity k that can be any positive finite integer, or the *uncapacitated* case where the capacity is not limited.
- The number of vehicles.
- The mobility of the objects: some objects may be *preemptive*, i.e., they can be dropped at some intermediate vertices before reaching their final destination, and others may be *non-preemptive*, meaning that they must be shipped directly to their destination. The objects of a given SP may all be of one type only, or they may be a *mix*, namely, some may be preemptive, while the others are non-preemptive. Note that this distinction is irrelevant in the uncapacitated case.

Applications of the SP arise in inventory repositioning involving a linear and circular movement of a robot arm in a factory or a warehouse (see [4]) in printed circuit board assembly (see [5]) as well as in the operations of automated guided vehicles and of material handling devices used in manufacturing systems; see [13].

[*] Corresponding author.
E-mail addresses: anily@post.tau.ac.il (S. Anily), pfeffer@post.tau.ac.il (A. Pfeffer).

0166-218X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2012.07.002

The SP with a unit capacity vehicle is the most studied version of the problem. Indeed, the unit capacity SP can be viewed as a generalization of the *Stacker-Crane Problem* (SCP); see [12]. In the SCP a single vehicle of a unit capacity carries objects that must be swapped without preemption between specified origin–destination pairs (this is equivalent to having one unit of each object type). On a general graph the SCP is NP-hard, which implies the NP-hardness of the SP. Frederickson et al. [12] provides an $O(|A|^3)$ time heuristic, where A is the set of arcs of the graph, that returns solutions having cost of at most 1.8 times the optimal value, i.e., the worst-case ratio of the proposed algorithm is 1.8. For a general graph with n vertices, Anily and Hassin [3], provides an $O(n^3)$ algorithm having a worst-case ratio of 2.5 for solving the unit capacity SP. Chalasani and Motwani [8], considers a special case of the unit capacity SP with two object types only and no null object, which is also NP-hard, and for this specific variant of the problem a nice algorithm with an improved worst-case ratio of 2 is presented. A branch-and-cut algorithm [6] and heuristics [7] have been developed for the unit capacity SP on general graphs.

Given the NP-hardness of the unit capacity SP on general graphs, researchers have been interested in verifying the complexity of the problem on simpler graph structures. The special case on a line with mixed objects and n vertices is shown to be solvable in $O(n^2)$ by Anily et al. [1]. The complexity on a circle is still unknown. Frederickson and Guan [11] proves that the unit capacity SP on a tree with non-preemptive objects is NP-hard. Anily et al. [2] proves that the preemptive case is also NP-hard. The common practice in papers that prove NP-hardness of such problems is to provide also bounded worst-case polynomial algorithms for solving them. A table that summarizes the complexity results and worst-case bounds for the unit capacity SCP and SP on various graph structures is provided by Anily et al.; see [2].

The *Dial-a-Ride Problem* (DaRP), is a well-known routing problem that deals with the transportation of non-preemptive objects (like voyagers) from specific origins to specific destinations. As such the DaRP usually includes various time constraints. However, a few papers consider the generic form of the DaRP with no time constraints, where the only objective is to minimize the total length of the route(s), and as such the problem can be viewed as a variant of the SP, where similarly to the SCP, there are no multiple units of any object-type. With no time constraints the main difference between the SCP and the DaRP is with respect to the number of vehicles and the vehicles' capacity. The SCP assumes a single vehicle of a unit capacity, where in the DaRP usually there is more than one vehicle, or the vehicles' capacity is greater than 1. For example, Psaraftis [14], considers the DaRP with a number of capacitated vehicles and no time constraints, and de Paepe et al. consider in [9] the uncapacitated single vehicle DaRP with no time constraints.

Solving a SP in contrast to solving a SCP (or a DaRP without time constraints) is more challenging, as the solution route for the SP should also determine the *assignment* or *matching* between the supply vertices and the demand vertices of each object type. This further complication is far of being straightforward to deal with. For example, the SCP on a tree with preemptive objects only, is known to be polynomially solvable (see [10]) while its SP version with possibly multiple units of each object type, is NP-Hard; see [2].

In this paper we consider the USP. The uncapacitated DaRP on a tree is proved to be NP-hard by de Paepe et al. [9], implying the same complexity result for the USP on a tree. In this paper we prove that the USP on a line is polynomially solvable, similarly to the unit capacity version of the problem; see [1]. In addition, we prove in this paper that the USP on a circle is polynomially solvable. Unfortunately, up to date it is still unknown if the unit capacity SP on a circle is also polynomially solvable. It is worth noting that if a SP with a unit capacity is known to be NP-hard on a particular graph structure, then its finite capacity version is also NP-hard. However, such conclusions cannot be made with respect to the uncapacitated case. That means that a-priori, there is no way to know if the USP is simpler or harder than its unit capacity version. Even if both, the unit capacity and the USP, are polynomially solvable, the optimal routes that are designed for these two problems are not alike, as the considerations in constructing the routes are not the same. Having said that, in this paper we show that for a line graph there is a basic property that is shared by both the optimal route of the unit capacity SP presented in [1], and the optimal route for the USP that is presented here: the two routes share the same assignment of supply vertices to the demand vertices, though the routes are obviously very different.

In this paper we consider the USP on linear and circular weighted graphs having n vertices. For both cases we provide polynomial algorithms: (i) for the linear graph, the complexity is linear, i.e., it is $O(n)$; and (ii) for a circular graph, the complexity is cubic, namely $O(n^3)$. The paper is organized as follows: Section 2 contains some general notations and preliminaries for the USP on a line and on a circle. In Section 3, the line case is solved, and in Section 4 the circle case is solved. Section 5 contains some concluding remarks.

2. Problems description, notations and preliminaries

2.1. Line and circle

Let $G = (V, E)$ be a linear or circular undirected graph with n vertices, where $V = \{1, \ldots, n\}$. Let $a, b \in V$ be two pre-specified vertices, where the vehicle starts its route at vertex a, and terminates it at vertex b. The set of edges E in a line contains $n-1$ edges, namely $E = \{(i, i+1) : 1 \le i \le n-1\}$, and in a circle it contains n edges, namely $E = \{(i, i+1) : 1 \le i \le n-1\} \cup \{(n, 1)\}$. We assume, without loss of generality, that in a line the vertices are arranged from left to right, and in a circle the vertices are arranged in a clock-wise direction. Moreover, we assume that the positions of any two consecutive vertices on the graph do not coincide. A distance function $\text{dist} : E \to \Re^+$ maps the edges into the positive real numbers. The distance between the two extreme points of the line is $\nu = \sum_{i=1}^{n-1} \text{dist}(i, i+1)$, and the perimeter of the circle is $\nu = \sum_{i=1}^{n-1} \text{dist}(i, i+1) + \text{dist}(1, n)$. For simplicity sake, we normalize the function dist by dividing it by the constant ν. Let $d(i, j) = \text{dist}(i, j)/\nu$ for any $(i, j) \in E$. We refer to the normalized distance function $d : E \to [0, 1]$ as the length function, implying that the length of the line and the perimeter of the circle are equal to 1 after the normalization.

In addition to the weighted graph, the problem is defined by m *object types*, or simply, *objects*, indexed by $j = 1, \dots, m$, in addition to the *null object* denoted by $j = 0$. Let $S = \{0, 1, \dots, m\}$. Let also $S_{-0} = S \setminus \{0\}$ be the set of tangible objects. As each vertex holds and requires at most one unit of the object, we denote by $s(i) \in S$ the object type currently at vertex i, called the *supply* of i, and $r(i) \in S$ the object type requested by vertex i, called the *demand* of i, for any $i \in V$. We assume, without loss of generality, that the only vertices in V that may be associated with both a supply and a demand of the null object are the vertices a and b. Other such vertices could be removed from the graph. For any object type $j \in S$ let n_j denote the number of units of the object in the graph, thus $\sum_{j \in S} n_j = n$ as each vertex is associated with a certain supply in S. The problem is *balanced* in the sense that for each object in S, its total supply equals its total demand. A feasible route starts with an empty uncapacitated vehicle at vertex a, and terminates at vertex b, after satisfying all the requirements of the vertices by loading and unloading objects along the route. In the solution of the line and the circle, we assume that given a route, the vehicle loads the supply of each vertex at the first visit at the vertex, and unloads the demand at the last visit at the vertex. The objective function of the USP is to find a feasible route of minimum length. The input size for both the line and the circle is $O(n)$ where n is the number of vertices of the graph.

2.2. The line

In this subsection we present a few more definitions and preliminaries for the line: A *walk* (t, h) is a shortest path connecting the *tail* vertex $t \in V$ to the *head* vertex $h \in V$. A walk (t, h) is said to *cross* all the intermediate vertices between t and h, i.e., it crosses all vertices k that satisfy $t < k < h$ or $t > k > h$. A walk (t, h) is also said to *cover* all the edges it consists of. Moreover, given two walks (t_1, h_1) and (t_2, h_2) pointing to the same direction such that $t_1 \le t_2 \le h_2 \le h_1$ or $t_1 \ge t_2 \ge h_2 \ge h_1$, we say that walk (t_1, h_1) *dominates* walk (t_2, h_2). The notion of dominance is essential here because of the absence of capacity restrictions. The definition of the length function d is extended to all pairs of vertices of V, which are not necessarily consecutive. Thus, the length of a walk (t, h) is given by $d(t, h) = \sum_{i=t}^{h-1} d(i, i+1)$ if $t < h$, $d(t, h) = \sum_{i=h}^{t-1} d(i, i+1)$ if $t > h$, and $d(t, t) = 0$.

Anily et al. [1], in their paper on the unit-capacity SP on a line, show that there exists an optimal route where for any two vertices k and ℓ, $k < \ell$, that initially possess a unit of object j, $j \in S_{-0}$, the destination vertex of the object at k is to the left of the destination vertex of the object at vertex ℓ. This property implies a particular optimal assignment of a very simple form between the demand vertices and the supply vertices. As we explain below the optimal assignment is not necessarily unique, implying that also the optimal route is not unique. To put it formally, for the unit capacity SP, Anily et al. [1] defines for each object type $j \in S_{-0}$, a *minimally balanced partition*:

Definition 1. A minimally balanced partition of object $j \in S_{-0}$ is a partition of the set of all j's supply and demand vertices into minimum size consecutive sets so that each set is balanced, i.e., has an equal number of supply and demand vertices of object j.

Thereafter, the paper presents the following optimal assignment: In each set of the minimally balanced partition of object j, number the demand (supply) vertices from left to right, and define:

Definition 2. Given a set of the minimally balanced partition of object $j \in S_{-0}$, let a j-path, or simply a path, be a walk that connects the ith supply vertex to the ith demand vertex in that set.

The paths represent the assignment of the supply to the demand vertices.

Property 1. *There exists an optimal route for the unit capacity SP on a line, where for every path, the object at its tail is used to satisfy the demand of its head.*

Note that all paths of a given set in the minimally balanced partition of object j, follow the same direction, and that each vertex of V that is positioned in between the two extreme vertices of the set, is crossed by at least one of these paths; see Fig. 1. It is possible that a set in the minimally balanced partition for object j is a singleton. In such a case, the supply at the vertex is used to satisfy the demand of the vertex, and the associated path (t, t) is said to be *degenerate*. In fact, any collection of $\sum_{j \in S_{-0}} n_j$ paths that satisfy the property that the tail and the head of each path are in the same set of the minimally balanced partition of a certain object type, is an optimal assignment, and it can be used in constructing an optimal route. The assignment that was pointed out above, has the simplest structure.

Property 1 continues to hold for any capacity, and in particular for the USP. The reasoning behind this claim is based on the analysis in [1] carried-out for the unit-capacity SP. The authors show that any feasible route is transferable into a route of the same total length but with a minimal cumulative distance that the vehicle travels non-empty. This is done by replacing the assignment of supply to demand vertices used by the route, by an optimal assignment of the structure specified above, and replacing some segments of the route that the vehicle traveled non-empty, by empty segments. That means that it is sufficient to consider only routes that use an assignment that satisfies Property 1. Exactly the same arguments hold for any capacity of the vehicle. Thus, we conclude:

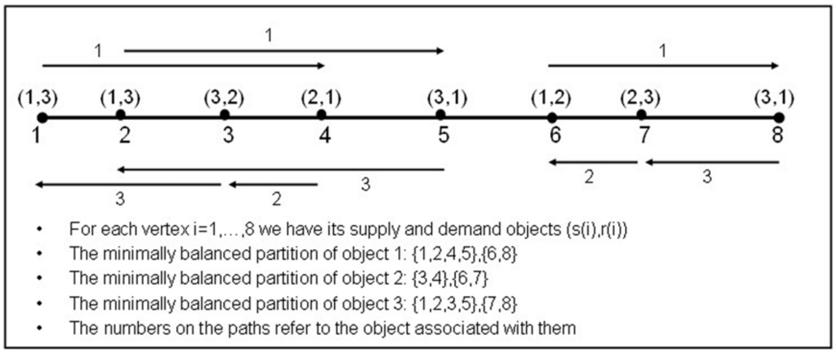
1
1
1
(1,3)
(1,3)
(3,2)
(2,1)
(3,1)
(1,2)
(2,3)
(3,1)
1
2
3
4
5
6
7
8
3
3
2
2
3
For each vertex i=1,…,8 we have its supply and demand objects (s(i),r(i))
The minimally balanced partition of object 1: {1,2,4,5},{6,8}
The minimally balanced partition of object 2: {3,4},{6,7}
The minimally balanced partition of object 3: {1,2,3,5},{7,8}
The numbers on the paths refer to the object associated with them

Fig. 1. The paths of a USP.

Property 2. *There exists an optimal solution for the USP on a line where for every path, the object held by its tail satisfies the demand of its head.*

According to Property 2, the set of paths uniquely defines the destination vertex of all supply vertices that initially hold an object in S_{-0}. Thus, given the tail t of a certain path, its head is well defined, and we denote it by $h(t)$. A given path $P = (t, h(t))$ is said to be a *right path* if $h(t) > t$, otherwise it is a *left path*. In order to distinguish between left and right paths, we denote the tail of a right path by t^+, and the tail of a left path by t^-. We further define $\mathcal{P}$ to be the set that contains all non-degenerate paths for all objects. $\mathcal{P}$ contains at most n paths. By using Property 2, the set $\mathcal{P}$ is an optimal matching between the supply and demand vertices. In addition, let $\mathcal{P}^+$ ($\mathcal{P}^-$) be the subset of right (left) paths, where $\mathcal{P} = \mathcal{P}^- \cup \mathcal{P}^+$. As the vehicle's capacity is not limited, when having two paths where one is dominating the other, (this may occur only if these paths belong to different object types), then the dominated (covered) path can be removed from $\mathcal{P}$, because it can be served while serving the dominating one. Therefore, we assume that all the dominated paths have been removed from $\mathcal{P}$. The next observation will guide us in constructing routes.

Observation 1. *In every feasible solution to the USP on a line, the vehicle must traverse while loaded any edge* $(i, i + 1)$ *from* $i + 1$ *to* i *if the edge is dominated by at least one left path in* $\mathcal{P}^-$*, and any edge* $(i, i + 1)$ *from* i *to* $i + 1$ *if the edge is dominated by at least one right path in* $\mathcal{P}^+$.

As will be seen in Section 3, a special attention is paid to paths that cross either the initial or the terminal vertices. For that sake we need the following definition:

Definition 3. A right (left) path, which crosses vertex a or vertex b (or both) is said to be a right (left) crossing path.

In the sequel we assume without loss of generality that $a < b$ or $a = b$. We note that the case $a > b$ can be solved by reversing the line. If $a \le b$ then all left crossing paths have their tail to the right of a and their head to the left of b. The next property follows immediately from the fact that $\mathcal{P}^+$ and $\mathcal{P}^-$ do not contain any dominated path:

Property 3. *The right crossing paths in* $\mathcal{P}^+$ *that cross both vertex a and vertex b, and the left crossing paths in* $\mathcal{P}^-$ *satisfy the property that the longer the distance of their tail to vertex a is, the shorter is the distance of their head to vertex b.*

In Section 3, a linear-time algorithm for solving the USP on a line, is presented. In addition, simplified algorithms are provided for two special cases (i) $a = b$; (ii) $a = 0$ and $b = 1$.

3. The USP on a line

Let Z^*_{Line} be the optimal cost of the USP on a line. As all the vertices have to be visited by the vehicle, the following lower bound follows immediately:

$$Z^*_{\text{Line}} \ge 2 - d(a, b). \tag{1}$$

The lower bound in (1) is tight. In order to derive an upper bound, we choose the best of two feasible routes, each starts at the initial vertex a, goes to one of the end-points of the line, namely, vertex 1 or vertex n, then the entire line is traversed back and forth, and finally it ends at the terminal vertex b. If the tour starts by going left, the resulting length is $d(1, a) + 2 + d(1, b) = 3 + (d(1, a) - d(b, n))$. If the tour starts by going right, the resulting length is $2 + d(a, n) + d(b, n) = 3 - (d(1, a) - d(b, n))$. The upper bound is the lowest of the two bounds, thus:

$$Z^*_{\text{Line}} \le 3 - |d(1, a) - d(b, n)|. \tag{2}$$

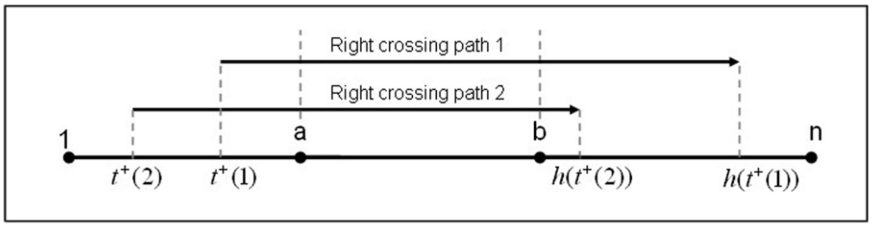

Fig. 2. The right crossing paths.

We distinguish between two kinds of feasible solutions: a solution is said to be a *right (left) solution* if after leaving the initial vertex a for the first time, the vehicle visits vertex n (1) before visiting vertex 1 (n). Accordingly, we define (i) a *right basic route*, which is a concatenation of the walks: (a, n), $(n, 1)$ and $(1, b)$, and whose length is $2 + d(b, a)$; and (ii) a *left basic route*, which is a concatenation of the walks: $(a, 1)$, $(1, n)$ and (n, b), and whose length is $2 - d(a, b)$. Each feasible solution is either a right or a left solution. If the walks that the right (left) basic route consists of are not dominating all paths then the basic route is not sufficient for constructing a feasible solution, and it must be augmented at a minimum cost in order to make it a feasible right (left) solution. The optimal solution is the best between the optimal left and optimal right solutions. In the next two subsections, we propose an algorithm that deals with each case separately, finds the optimal cost for each, and returns the shortest feasible route in complexity $O(n)$. Let Z^+ (Z^-) be the optimal cost of a right (left) solution, implying that $Z^*_{\text{Line}} = \min\{Z^+, Z^-\}$.

3.1. Right solutions

Following the right basic route allows servicing all the left paths that are dominated by the walk $(n, 1)$, as well as the right paths that are dominated by the walk (a, n) and the walk $(1, b)$. The only paths that are not dominated by the right basic route's walks are the right paths in $\mathcal{P}^+$, which cross both a and b. Let C^+ be the set of right crossing paths in $\mathcal{P}^+$ that cross both a and b : $C^+ = \{(t^+, h(t^+)) \in \mathcal{P}^+ : t^+ < a \le b < h(t^+)\}$. Let f^+ be the cardinality of C^+. If $f^+ = 0$, no augmentation is needed, and the right basic route is feasible as is. In this section we show how to find a minimal cost augmentation if $f^+ > 0$. Number the crossing paths $(t^+(k)), h(t^+(k)) \in C^+$ so that $t^+(k) < a \le b < h(t^+(k))$, for $k = 1, \ldots, f^+$ where $t^+(1) > t^+(2) \cdots > t^+(f^+)$. Property 3 implies that $h(t^+(1)) > h(t^+(2)) \cdots > h(t^+(f^+))$. In particular it means that $d(t^+(1), a) < \cdots < d(t^+(f^+), a)$, and $d(h(t^+(1)), b) > \cdots > d(h(t^+(f^+)), b)$, (see Fig. 2).

One possible augmentation consists of visiting all the heads of the crossing paths of C^+ after having completed the traversal of the right basic route, and then drive back to b. This means that at the end of the right basic route the vehicle needs to traverse the walk $(b, h(t^+(1)))$ back and forth. Another possible augmentation is to start the route by first going left from the initial vertex a in order to collect the supplies from all the tails of the crossing paths in C^+, then drive back to a and follow the right basic route. This means that before starting the right basic route the vehicle needs to traverse the walk $(a, t^+(f^+))$ back and forth. But these are only the two extreme augmenting options. In a general augmentation form, the vehicle starts at a by following the walk $(a, t^+(k))$ back and forth for some $k \in \{0, \ldots, f^+\}$ where $t^+(0) = a$, then it follows the right basic route, and finally it follows the walk $(b, h(t^+(k+1)))$ back and forth, where $h(t^+(f^+ + 1)) = b$. It is easy to see that any such augmentation together with the right basic route contain walks that dominate all paths.

The optimal right solution Z^+ and its length are computed as follows: find $\Theta^+ = 2\min_{k=0}^{f^+} d(t^+(k), a) + d(h(t^+(k+1)))$, which is the minimum augmentation cost, and let $k^+ = \arg\min(\Theta^+)$. The optimal right solution consists of the walks $(a, t^+(k^+))$, $(t^+(k^+), n)$, $(n, 1)$, $(1, h(t^+(k^+ + 1)))$, $(h(t^+(k^+ + 1)), b)$, and its length is $Z^+ = 2 + d(b, a) + \Theta^+$. The complexity of the proposed algorithm is $O(n)$.

3.2. Left solutions

Following the left basic route enables servicing all the right paths in $\mathcal{P}^+$ as they are dominated by walk $(1, n)$, and also the left paths in $\mathcal{P}^-$, which are dominated by (i) the walk $(a, 1)$, or (ii) the walk (n, b) of the basic route. However, if there exist left crossing paths, or left paths dominated by the walk (b, a), then the left basic route must be augmented. For this sake define four sets of left paths that are not dominated by the left basic route:

- $C^-_a = \{(t^-, h(t^-)) \in \mathcal{P}^- | h(t^-) < a < t^- \le b\}$
- $C^-_b = \{(t^-, h(t^-)) \in \mathcal{P}^- | a \le h(t^-) < b < t^-\}$
- $C^-_{ab} = \{(t^-, h(t^-)) \in \mathcal{P}^- | h(t^-) < a \le b < t^-\}$
- $B = \{(t^-, h(t^-)) \in \mathcal{P}^- | a \le h(t^-) < t^- \le b\}$.

By Property 3, if $C^-_{ab} \ne \emptyset$ then $B = \emptyset$ as otherwise the paths in B are dominated by each of the paths in C^-_{ab}. Let $C^- = C^-_a \cup C^-_b \cup C^-_{ab}$ be the set of all crossing left paths, and f^-, f_B, f_a, f_B, and f_{ab} be the cardinalities of C^-, B, C^-_a, C^-_b, and C^-_{ab}, respectively. As the number of paths is bounded by n, it must hold that $f^- + f_B \le n$. Number the left crossing

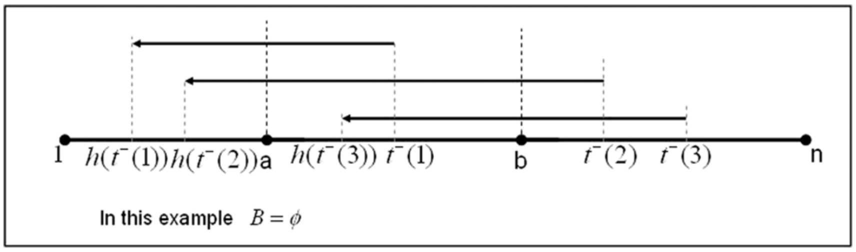

Fig. 3. The left crossing paths in C^-.

paths in C^- by $(t^-(k), h(t^-(k)))$ for $k = 1, \ldots, f^-$, where $a < t^-(1) < t^-(2) < \cdots < t^-(f^-) \le n$. By Property 3, $1 \le h(t^-(1)) < h(t^-(2)) \cdots < h(t^-(f^-)) < b$; see Fig. 3. Because of the fact that no dominated path exists, the first f_a left paths in the sequence $(t^-(k), h(t^-(k)))$, $k = 1, \ldots, f^-$, are members of C_a^-, the next f_{ab} paths are members of C_{ab}^-, and the last f_b paths are members of C_b^-. Let also $t^-(0) = a$, and $h(t^-(f^- + 1)) = b$.

As we are going to see, the algorithm that finds the best left solution is somewhat more involved than the one for finding the best right solution because of the paths in B. If B is empty then the algorithm is a direct modification of the right solution case where the vehicle starts by following the walk $(a, t^-(k))$ back and forth for some $k \in \{0, \ldots, f^-\}$, then it follows the left basic route, and upon its termination it follows the walk $(b, h(t^-(k + 1)))$ back and forth. If $B \neq \emptyset$, then $C_{ab}^- = \emptyset$ and $f_{ab} = 0$. If $C_b^- \neq \emptyset$, then path $(t^-(f_a + 1), h(t^-(f_a + 1)))$, in C^- is a crossing path in C_b^-. If $C_b^- = \emptyset$, then by definition $h(t^-(f_a+1)) = b$. Let $(t_B^-(\ell), h(t_B^-(\ell)))$, for $\ell = 1, \ldots, f_B$, be the paths in B, such that $a < t_B^-(1) < t_B^-(2) < \cdots < t_B^-(f_B) \le b$. By Property 3, $a \le h(t_B^-(1)) < h(t_B^-(2)) \cdots < h(t_B^-(f_B)) < b$. As no dominated path exists, it must hold that $t^-(f_a) < t_B^-(1)$ and $h(t_B^-(f_B)) < h(t^-(f_a + 1))$. In order to service the paths of B, consider the set E_B of edges that are dominated by the paths of B. Namely,

$$E_B = \{(i, i + 1) : a \le i < b, \text{ and } \exists j \in \{1, \ldots, f_B\} \text{ such that } h(t_B^-(j)) \le i < t_B^-(j)\}.$$

The edges of E_B can be concatenated into a minimal number of left walks. For that sake we define the following.

Definition 4. A coverage-B-walk (t, h), $a \le h < t \le b$, is a left walk that satisfies the following conditions:

- Any edge $(i, i + 1)$ satisfying $h \le i < t$ is in the set E_B.
- $h = a$ or $(h - 1, h) \notin E_B$.
- $t = b$ or $(t, t + 1) \notin E_B$.

By definition, the coverage-B-walks are non-overlapping. By scanning consecutively $(t_B^-(\ell), h(t_B^-(\ell)))$, for $\ell = 1, \ldots, f_B$, it is simple to determine the coverage-B-walks in linear time. Let q ($q \le f_B$) be the number of coverage-B-walks, and name them by $W(1), \ldots, W(q)$. Let $t_W(i)$ and $h_W(i)$ be the tail and the head of $W(i)$, respectively, for $i = 1, \ldots, q$, such that $a < t_W(1) < \cdots < t_W(q) \le b$. The removal of dominated paths implies by construction that $t_W(1) > t^-(f_a)$, and $h_W(q) < h(t^-(f_a + 1))$. Thus, if $f_a > 0$, the only coverage-B-walk that can cross vertex $t^-(f_a)$ is $W(1)$, and similarly, if $f_b > 0$, the only coverage-B-walk that can cross vertex $h(t^-(f_a + 1))$ is $W(q)$. In the following we use two indicators, I_a and I_b. If $f_a > 0$ and $W(1)$ crosses $t^-(f_a)$, then $I_a = 1$, otherwise $I_a = 0$. If $f_b > 0$ and $W(q)$ crosses $h(t^-(f_a + 1))$, then $I_b = 1$, otherwise $I_b = 0$. In addition, let

$$\Delta(B) = 2\sum_{i=1}^{q} d(t_W(i), h_W(i)) \tag{3}$$

be twice the total length of the coverage-B-walks. Clearly, any augmentation will cost at least twice the total length of all edges $(i, i + 1)$ where $a \le i < b$ that are dominated by at least one left path as the left basic route does not traverse any of these edges from right to left. Let Γ represent this length, i.e.,

$$\Gamma = \Delta(B) + 2(d(h_W(1), a)I_a + d(t_W(q), b)I_b). \tag{4}$$

Let Θ^- be the optimal augmentation cost of the best left solution. By the above discussion,

$$\Theta^- \ge \Gamma. \tag{5}$$

We next explain how to calculate Θ^- in linear time. We do that by analyzing a few separate cases.

Calculation of the augmentation cost and the structure of an optimal left solution

Case 1. $B = \emptyset$:

Case 1.1 C^- consists of a single connected component: The vehicle starts at a by going right and following the walk $(a, t^-(k))$ back and forth for some $k = 0, 1, \ldots, f^-$, and from there it follows the left basic route until its completion at b. The tour ends by going left following the walk $(b, h(t^-(k + 1)))$ back and forth. The cheapest resulting total augmentation cost under this case is

$$\Theta^- = 2\min\{d(t^-(k), a) + d(h(t^-(k + 1)), b) : 0 \le k \le f^-\}.$$

Case 1.2 C^- consists of two connected components, one connected component is C_a^- and the other is C_b^-, (in such a case $f_{ab} = 0$): The best augmentation in such a case is

$$\Theta^- = 2[d(t^-(f_a), a) + d(h(t^-(f_a + 1)), b)].$$

Case 2. $B \neq \emptyset$: Under this case $f_{ab} = 0$ and $f^- = f_a + f_b$.

Case 2.1 $f^- = 0$, i.e., no left crossing paths exist: The coverage-B-walks are served in loops while traversing the walk $(1, n)$ of the left basic route, that means that for $\ell = 1, \dots, q$, when the vehicles reaches $t_W(\ell)$ it makes there a U-turn and goes back to $h_W(\ell)$, where another U-turn is made to return to $t_W(\ell)$. The total augmentation cost is (see (3) and (4)), and (5)

$$\Theta^- = \Gamma = \Delta(B).$$

Case 2.2 $f^- > 0$, and the vertices $t^-(f_a)$ and $h(t^-(f_a + 1))$ are not crossed by the same coverage-B-walk: The conditions in this case imply that $t^-(f_a) < h(t^-(f_a + 1))$, as otherwise there would either be a single coverage-B-walk that crosses both vertices, or $B = \emptyset$, contradicting our assumptions. If $I_a = 1$ then $W(1)$ is the coverage-B-walk that crosses $t^-(f_a)$. In such a case, let $\tilde{t} = t_W(1)$ and $\tilde{h} = h_W(1)$, then $\tilde{h} < t^-(f_a) < \tilde{t}$. If $I_a = 0$, then let $\tilde{t} = \tilde{h} = t^-(f_a)$. Similarly, if $I_b = 1$ then $W(q)$ is the coverage-B-walk that crosses $h(t^-(f_a + 1))$. In such a case, let $\hat{t} = t_W(q)$ and $\hat{h} = h_W(q)$, implying that $\hat{h} < h(t^-(f_a + 1)) < \hat{t}$. If $I_b = 0$, then let $\hat{t} = \hat{h} = h(t^-(f_a+1))$. The cheapest augmentation cost is achieved when the vehicle starts by going right following the walk $(a, \tilde{t})$ back and forth, and then it starts following left basic route. While traversing the walk $(1, n)$ of the left basic route, each time that it reaches a vertex $t_W(\ell)$ such that $\tilde{t} < t_W(\ell) < \hat{h}$ for $\ell = 1, \dots, q$, it makes there a U-turn to serve all the left paths on the ℓ-th coverage-B-walk, and when reaching the head of the coverage-B-walk, namely $h_W(\ell)$ it makes another U-turn and continues to follow that left basic route. When terminating the left basic route at b, the vehicle continues right and follows the walk $(b, \hat{h})$ back and forth. It is easy to see that the corresponding augmentation cost is equal to the lower bound on the augmentation cost (see (4) and (5)) namely,

$$\Theta^- = \Gamma.$$

Case 2.3 $f_a f_b > 0$ and the two vertices $t^-(f_a)$ and $h(t^-(f_a + 1))$ are crossed by the same coverage-B-walk: Under this case any augmentation costs at least $2d(b, a)$ as the left paths cover the walk connecting b to a.

The removal of dominated paths implies that $q = 1$ and $h_W(1) < \min\{t^-(f_a), h(t^-(f_a + 1))\} \leq \max\{t^-(f_a), h(t^-(f_a + 1))\} < t_W(1)$. Consider first the case that the vehicle starts the tour at a by going right and following the walk $(a, t^-(\ell))$ back and forth, for some $\ell \neq f_a$ and $0 \leq \ell \leq f^-$. Then it follows the left basic route, and upon its completion it follows the walk $(b, h(t^-(\ell + 1)))$ back and forth. Note that such a route ensures servicing of all the paths in B, as either before starting the left basic route ($\ell > f_a$) or after its completion ($\ell < f_a$), the vehicle traverses the walk (b, a). Overall there are f^- such options, and the cost of the best such option is

$$\Theta_1^- = 2\min\{\big(d(t^-(k), a) + d(h(t^-(k + 1)), b)\big) : 0 \leq k \leq f^-,\ k \neq f_a\}. \tag{6}$$

More care should be paid if the vehicle starts the tour by going right to $t^-(f_a)$: Note that here a replication of the form of the tour for $\ell \neq f_a$ may miss the service to paths in B. By our assumptions for this case, at least one such path exists as $t^-(f_a)$ is covered by the single coverage-B-walk. The tail of all paths in B is to the right of $t^-(f_a)$, otherwise they would be dominated by the path $(t^-(f_a), h(t^-(f_a)))$. Thus, we need to consider also the f_B options, where the vehicle starts by going right from a crossing $t^-(f_a)$ up to vertex $t_B^-(k)$, for $k = 1, \dots, f_B$, i.e., the tail of the k-th path in B, makes there a U-turn and after completing the left basic route at b, the vehicle continues left to $h(t_B^-(k + 1))$ if $k < f_B$, and to $h(t^-(f_a + 1))$ if $k = f_B$, where another U-turn is made in order to return to b. Thus, if the vehicle starts by going right and following the walk $(a, t^-(f_a))$, there exist f_B more options for further augmenting the tour with respect to the additional ride to the right, before making the first U-turn in order to start the left basic route. For this sake let $t_B^-(0) = t^-(f_a)$, and $t_B^-(f_B + 1) = t^-(f_a + 1)$. Overall, the best augmentation cost for these $f_B + 1$ options is

$$\Theta_2^- = 2\min\{d(a, t_B^-(j)) + d(b, h(t_B^-(j + 1))) :\ 0 \leq j \leq f_B\}. \tag{7}$$

By using (6) and (7), the best augmentation cost in this case is

$$\Theta^- = \min\{\Theta_1^-, \Theta_2^-\}.$$

The optimal cost of a left solution is therefore equal to $Z^- = (2 - d(a, b)) + \Theta^-$. The optimal tour is obtained by first verifying which of the cases in the description above holds, and then finding the best tour under the relevant case. The best USP solution on a line is $Z^*_{\text{Line}} = \min\{Z^+, Z^-\}$. We conclude as follows:

Theorem 1. *The complexity of computing the cost of the optimal USP solution on a line with n vertices is $O(n)$.*

Recall, that if originally before normalizing the distances, the line had a length non equal to 1, then the optimal cost computed above, should be multiplied by the normalization factor ν.

3.3. *Two special cases*

In this subsection we provide a couple of observations regarding two special cases: $a = b$, and the case where the initial and terminal points are at the two extreme end-points of the line. For both special cases the proposed solution method boils down to simpler versions, though the complexity remains the same.

3.3.1. $d(a, b) = 0$

In this case the crossing paths are the right and left paths that cross vertex a. Both the left and right basic routes are of length of 2. As $B = \emptyset$, the best left solution is calculated similarly to the best right solution (Case 1).

3.3.2. $d(a, b) = 1$

In this case, any right solution requires a traversal of the whole line three times, which is the worst possible; see (2). Considering left solutions, we note that $C^- = \emptyset$, meaning that all the left paths are in B. That means that case 2.1 applies. I.e., the vehicle while traversing the walk $(1, n)$, each time that it reaches the tail of a coverage-B-walk, it makes there a U-turn and rides back to the head of the coverage-B-walk, where it makes again a U-turn in order to continue to the right end-point, namely vertex n. The cost of the optimal solution is therefore $Z^*_{\text{Line}} = Z^- = 1 + \Delta(B)$.

4. The USP on a circle

In this section we assume that $a = 1$. In the following subsection we consider the case where $a = b$, and we call vertex 1 the *depot*, where the vehicle starts and ends the route. For simplicity we name vertex 1 also as vertex $n + 1$. In Section 4.2 we consider the case $a \neq b$.

4.1. $a = b = 1$

Let Z^*_{Circle} denote the optimal cost of the USP on a circle. The value Z^*_{Circle} is bounded from below by the minimum between the cost of a complete encirclement of the circle, i.e., 1, and the cost of covering twice the whole encirclement except of one edge:

$$Z^*_{\text{Circle}} \geq \min\{1; \min\{2(1 - d(i, i + 1)) : i = 1, \ldots, n\}\}.$$

We first propose a simple upper bound that it the minimum length of two feasible walks. Later we strengthen the upper bound by considering the minimum length of n feasible walks. First consider a walk that starts by going clockwise from vertex 1 to n and then counter-clockwise back to 1, and a walk that starts by going counter-clockwise from 1 to 2, and then clockwise back to 1. Thus,

$$Z^*_{\text{Circle}} \leq 2(1 - \max\{d(n, n + 1), d(1, 2)\}). \tag{8}$$

The upper bound (8) can be strengthened as follows: let $k^* = \max\{k : d(1^+, k) \leq d(1^-, k + 1),\ 1 \leq k \leq n\}$, i.e., for all vertices $k \leq k^*$, the length of the circular clockwise arc from 1 to vertex k is no longer than the length of the circular counter-clockwise arc from 1 to vertex $k + 1$. For each k, $2 \leq k \leq k^*$ we consider the following feasible walk: start by going clockwise from vertex 1 to k, make a U-turn at k, and proceed counter-clockwise to vertex $k + 1$, make there a second U-turn and proceed clockwise to vertex k, and make there a final U-turn to finish at vertex 1. The length of such a walk is: $2\big(1 - d(k^+, k + 1) + d(1^+, k)\big)$. For each k, $k^* < k \leq n - 1$, we consider the following feasible walk: start by going counter-clockwise from vertex 1 to vertex $k + 1$, make a U-turn at vertex $k + 1$, and proceed clockwise to vertex k, make there a second U-turn and proceed counter-clockwise to vertex $k + 1$, and make there a final U-turn to finish at vertex 1. The length of such a walk is: $2\big(1 - d(k^+, k + 1) + d(1^-, k + 1)\big)$. Finding the minimum length among the above n proposed walks (including the ones considered in (8) results in the following upper bound:

$$\begin{aligned} Z^*_{\text{Circle}} \leq 2 \min\big\{&\min\{1 - d(k^+, k + 1) \\ &+ d(1^+, k) : k = 1, \ldots, k^*\}; \min\{1 - d(k^+, k + 1) + d(1^-, k + 1) : k = k^* + 1, \ldots, n\}\big\}. \end{aligned} \tag{9}$$

Both lower and upper bounds can be shown to be tight.

Definition 5. A solution to the USP on a circle is said to contain a *positive ring* (*negative ring*) if the vehicle traverses all n edges of the circle in a clockwise (counter-clockwise) direction at least once.

The next property is straightforward:

Property 4. *Any feasible solution to the USP on a circle either contains a ring, or it leaves an edge that is uncovered, while all the other edges are covered in both directions.*

Let Z^{UC} be the optimal solution with an uncovered edge, and Z^R be the optimal solution with a ring. $Z^*_{\text{Circle}} = \min\{Z^{UC}, Z^R\}$. In the next two subsections we describe the solution method of each type of solution. The proof of the next theorem follows from the solution method proposed in the rest of the section:

Theorem 2. *The USP on a circle is polynomially reducible to the USP on a line. That means that Z^*_{Circle} can be obtained by solving a polynomial number of instances of the USP on a line and choosing the best of these solutions.*

4.1.1. An $O(n^2)$ algorithm for computing Z^{UC}

Let $Z^{UC}(i)$, $i = 1, \dots, n$, be the best solution that does not cover edge $(i, i+1)$ in neither direction. As observed above, $Z^{UC}(1) = 2(1-d(1,2))$, and $Z^{UC}(n) = 2(1-d(n,1))$. For $1 < i < n$, computing $Z^{UC}(i)$ boils down to solving an appropriate *USP on a line*, where the initial and terminal points coincide at vertex 1, the left end-point of the line is vertex $i+1$, and the right end-point is vertex i, implying that the length of the line is $\nu(i) = 1 - d(i, i+1)$. The problem can be solved by invoking the method for solving the USP on a line proposed in Section 3 for the special case $a = b$; see Section 3.3.1. Possible savings in calculations can be obtained by using (8) as there is no need to calculate $Z^{UC}(i)$ for $1 < i < n$ if $d(i, i+1) \le \max\{d(1,2), d(n, n+1)\}$. The complexity of calculating $Z^{UC} = \min\{Z^{UC}(i) | i = 1, \dots, n\}$, is $O(n^2)$.

4.1.2. An $O(n^3)$ algorithm for computing Z^R

The problem is solved twice, once for the best solution value containing a positive ring, denoted by Z^{R+}, and once for the best solution value containing a negative ring denoted by Z^{R-}. $Z^R = \min\{Z^{R+}, Z^{R-}\}$. Clearly, if a (positive or negative) ring is feasible then $Z^R = 1$. Otherwise, it is necessary to augment it. We describe how to solve Z^{R+}. The calculation of Z^{R-} follows analogously. As a walk in the circle cannot be defined by just specifying its tail t and its head h, we denote a clockwise traversal from t to h by (t^+, h) and a counter-clockwise traversal from t to h by (t^-, h). A traversal (h^-, t) $((h^+, t))$ that follows a traversal (t^+, h) $((t^-, h))$ is called a *loop*. Recall that $(s(i), r(i)) \in S^2$ denote the supply and the demand (requirement) object of vertex $i \in V$.

Apparently, any optimal solution that contains a positive ring must cover each edge of the circle an odd number of times, where the number of clockwise coverages of each edge is greater by 2 than the number of counter-clockwise coverages of the edge.

Lemma 1. *Suppose there exists an optimal solution for the USP on a circle which contains a positive ring. Then,*

1. *the optimal route covers all edges of the circle an odd number of times, and at least one edge is covered just once. Moreover, the number of clockwise traversals of each edge is greater by one than the number of its counter-clockwise traversals.*
2. *No edge of the circle is traversed more than 3 times by the optimal route.*

Proof. Consider an optimal route that contains a positive ring.

1. The optimal route is a directed close walk on the vertices of the circle. By using the optimality of the solution and the upper bound in (8) it is clear that at least one edge of the circle is traversed only once in a clockwise direction. Suppose that edge $(i, i+1)$ for some $i = 1, \dots, n$ is traversed only once in a clockwise direction. As the walk on the circle is closed, and at least one edge is covered once by a clockwise traversal, any edge of the circle should be traversed clockwise one time more than counter-clockwise, implying, in particular, an odd number of traversals of any edge.
2. We prove that an optimal solution cannot contain more than one counter-clockwise traversal of an edge and more than 2 clockwise traversals of an edge. The proof is by contradiction. Suppose that the optimal route contains a loop (h^-, t), (t^+, h), for $t \ne h$, such that the path (h^-, t) is covered at least twice, and the path (t^+, h) is covered at least three times. The removal of one loop (h^-, t), (t^+, h), does not affect the feasibility of the solution as at least one such loop still exists in addition to a path (t^+, h) along which the vehicle can proceed clockwise from t to h after completing the loop. This contradicts the optimality of the solution. □

We now focus on all feasible solutions satisfying the structure stated in Lemma 1. Clearly, if it is feasible to cover all edges just once, then $Z^R = 1$. Otherwise, some edges must be covered 3 times. Each such solution is associated with a pair of vertices (p, q), $1 \le q < p \le n+1$, so that it consists of three, possibly empty, disjoint subsets of vertices:

1. $PL1(p) = \{\ell : p \le \ell \le n\}$;
2. $PL2(q) = \{\ell : 1 \le \ell \le q\}$;
3. $R(p, q) = \{\ell : q < \ell < p\}$.

In addition, if $p < n+1$ then $s(p) \in S_{-0}$, and if $q > 1$ then $r(q) \in S_{-0}$.

The tour associated with this partition starts by the walk $(1^-, p)$ while picking-up the loads supplied by the vertices in $PL1(p)$; at vertex p the vehicle makes a U-turn and returns to the depot along the walk $(p^+, 1)$. This initial segment of the tour is called the *first pre-loading segment*. If $p = n+1$, then $PL1(n+1) = \emptyset$ and the first pre-loading option is not used. Next, the vehicle follows the walk $(1^+, q)$ while picking-up the supplies of the vertices in $PL2(q)$. This second segment of the tour is called the *second pre-loading segment*. Note that along the two pre-loading segments demands are not satisfied as the vertices in $PL1(p) \cup PL2(q)$ will be visited again at the end of the route. Particularly note that the supply of vertex 1 is picked-up at the beginning of the route, while its demand is supplied at the end of the route. The remaining vertices in $R(p, q)$, are served as follows: from vertex q the vehicle goes clockwise and at each vertex y visited for the first time, the supply is loaded. If object $r(y)$ is available on the vehicle and if the demands of all vertices in $\{q+1, \dots, y-1\}$ have been satisfied then the demand of vertex y is also satisfied. Suppose that vertex j is the minimum indexed vertex in $R(p, q)$ whose

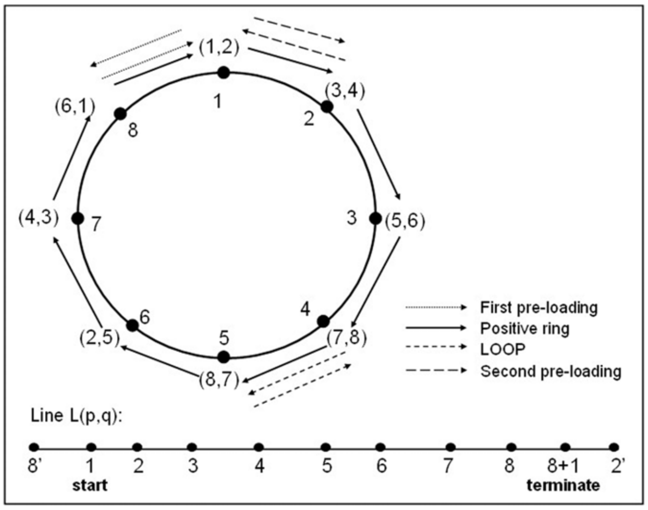

Fig. 4. $(p, q) = (8, 2)$, $PL1(8) = \{8\}$, $PL2(2) = \{1, 2\}$, $R(8, 2) = \{3, 4, 5, 6, 7\}$, $LOOP(8, 2) = \{4, 5\}$.

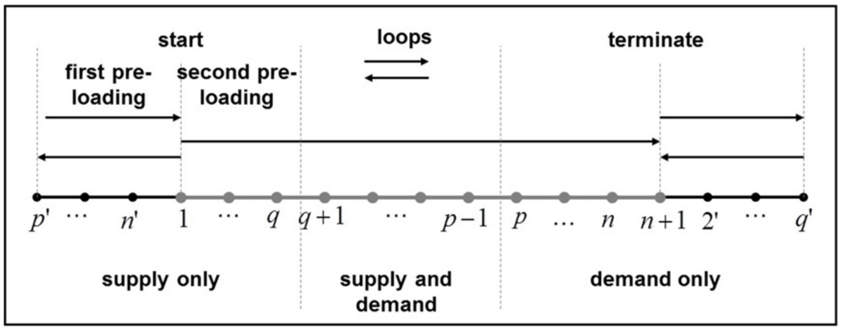

Fig. 5. Line(p, q).

demand could not been served at the first visit because object $r(j)$ is not available on the vehicle. The vehicle then continues clockwise, while loading supplies but without satisfying demands until it reaches a vertex $k, q < j < k < p$, where the current cumulative load on the vehicle, inclusive object $s(k)$, is sufficient to satisfy the demands of all vertices on the walk (j^+, k). In such a case the vehicle follows the loop that consists of the walks (k^-, j) and (j^+, k), while satisfying the demands of the yet unserved vertices. The vehicle then continues clockwise from k to $p - 1$ while making loops as necessary. The set $LOOP(p, q) \subset R(p, q)$ consists of all the vertices covered by such loops. By definition, the sets $PL1(p)$, $PL2(q)$, and $LOOP(p, q)$ are disjoint. When reaching p for the second time, the vehicle has completed a positive ring, thus all the objects have been picked up, and by following the walk (p^+, q) it can satisfy the demands of the vertices in $PL1(p) \cup PL2(q)$. At vertex q the vehicle makes a U-turn and returns to the depot by following the walk $(q^-, 1)$. Fig. 4 demonstrates the possible components of such an augmentation.

The above solution method demonstrates that for given p and q, the problem reduces to solving the USP on a line, denoted by $L(p, q)$, having $n+q+(n-p+1)$ vertices: The left part of $L(p, q)$ is associated with the edges covered by the walk $(1^-, p)$, attached to it at vertex 1 is the middle part of $L(p, q)$ that consists of the edges covered by the positive ring that starts at 1, and thereafter attached to it at 1, is the last part of $L(p, q)$, the part that is associated with the edges covered by the walk $(1^+, q)$. In order to represent the resulting problem as an instance of the USP on a line, we renumber the vertices on $L(p, q)$ from left to right by $p', \dots, n', 1, \dots, q, q+1, \dots, p-1, p, \dots, n, n+1, 2', 3', \dots, q'$. Let $V(p, q)$ denote the set of vertices of $L(p, q)$. Fix the initial vertex $a = 1$, and the terminal vertex is $b = n + 1$. The supply (demand) $s(y)$ $(r(y))$ for vertex y on the circle is assigned to a single vertex on $L(p, q)$ as follows: if $y \in \{p, \dots, n\}$, then $s(y)$ $(r(y))$ is assigned to vertex y' (y) in the left (middle) part of $L(p, q)$. If $y \in \{1, \dots, q\}$ then $s(y)$ $(r(y))$ is assigned to vertex y (y') in the middle (last) part of $L(p, q)$. If $y \in \{q + 1, \dots, p - 1\}$ then $s(y)$ and $r(y)$ are assigned as the supply and demand of vertex y in the middle part of $L(p, q)$. All other supplies and demands of vertices in $V(p, q)$ are set to 0, i.e., the null object. Thus, in the USP on $L(p, q)$ the total supply equals the total demand, for each object in S_{-0}, and it is exactly the same as on the original problem on the circle. See Fig. 5.

The line $L(p, q)$ as constructed here has a special structure as the first $n - p + 1 + q$ left vertices, namely, vertices $\{p', \dots, n'\} \cup \{1, \dots, q\}$ have no demand, and the last $n-p+1+q$ right vertices, namely, vertices $\{p, \dots, n+1\} \cup \{2', \dots, q'\}$

have no supply. When solving the corresponding USP on a line, this special structure implies that right solutions can be ignored, and it is sufficient to compute Z^- for each $L(p, q)$ and keep the best. Furthermore, the special structure of line $L(p, q)$ implies that all the left paths in $\mathcal{P}^-$, are in the set B, denoted here by $B(p, q)$, thus Case 2.1 in Section 3.2 applies. The optimal cost of the USP on line $L(p, q)$ is therefore: $1 + \Delta(B(p, q)) + 2(d(p^+, n + 1) + d(1^+, q))$. Thus,

$$Z^{R+} = \min_{1\le q<p\le n+1} \big(1 + \Delta(B(p, q)) + 2(d(p^+, n + 1) + d(1^+, q))\big). \tag{10}$$

As shown in Section 3, the complexity of the USP on a line is linear in the number of vertices, and as there are $O(n^2)$ different pairs of (p, q), a direct application of the above described algorithm is of complexity $O(n^3)$.

Let denote $\xi = \max\{d(1^+, 2), d(n^+, n + 1)\}$. Considering the upper bound in (8) it is sufficient to restrict our search to pairs of (p, q) for which $d(p^+, n + 1) + \Delta(B(p, q)) + d(1^+, q) < 0.5 - \xi$, which implies that $d(q^+, p) > 0.5 + \xi$. Let $\underline{p}$ and $\overline{q}$ be two vertices on the circle defined such that $d(1^+, \underline{p} - 1) \le 0.5 + \xi < d(1^+, \underline{p})$, and $d(1^+, \overline{q}) \le 0.5 - \xi < d(1^+, \overline{q} + 1)$. Because of the upper bound in (8), there is no need to solve the USP on lines $L(p, q)$ for $p < \underline{p}$ or $q > \overline{q}$. In addition, let $\overline{q}(p)$ for $p \ge \underline{p}$, be the maximum index q such that $d(q^+, p) > 0.5 + \xi$. Clearly, $\overline{q}(p) \le \overline{q}(p + 1) \le \cdots \le \overline{q}(n + 1) = \overline{q}$. Thus, for any $p \ge \underline{p}$, we only need to consider $L(p, q)$ for $1 \le q \le \overline{q}(p)$. The calculation of $\underline{p}$, $\overline{q}$, and the sequence $\overline{q}(p)$ for $p \ge \underline{p}$, takes linear time. Unfortunately, these insights do not reduce the magnitude of the complexity, which is $O(n^3)$.

4.2. The USP on a circle with $b \neq 1$

In this subsection we consider USP on a circle where the initial and terminal vertices do not coincide. This case is also polynomially reducible to the USP on a line, and Property 4 continues to be valid here as well. The optimal solution with an uncovered edge, namely Z^{UC} is computed in the same way as described in Section 4.1.1, while for each line constructed we use the general algorithm for the USP on a line where $b \neq 1$. In computing Z^R we need again to consider both cases where the optimal solution contains a positive ring, i.e., Z^{R+}, or a negative ring Z^{R-}. The two cases are analogous, thus we describe just the first, i.e., the calculation of Z^{R+}.

In the case $b = 1$, an optimal solution that contains a ring has the property that each edge of the circle is covered either once or 3 times; see Lemma 1. However, if $b \neq 1$ all edges of the circle are covered an odd number of times by the solution, except of one of the following two walks (i) $(1^+, b)$, or (ii) $(1^-, b)$, that is covered an even number of times by the solution. An adaptation of Lemma 1 to the case $b \neq 1$ reveals that an optimal solution that contains a ring covers each edge of the circle 1,2,3 or 4 times. If there exists an optimal USP solution that contains a positive ring, and the respective optimal walk terminates at b by a traversal of edge $(b - 1, b)$ (edge $(b, b + 1)$) then we let Z_1^{R+} (Z_2^{R+}) be the cost of this solution.

Thus, the optimal solution that contains a positive ring is obtained by computing $Z^{R+} = \min\{Z_1^{R+}, Z_2^{R+}\}$. We shortly describe the algorithm for each of these two types of solutions:

- Computation of Z_1^{R+}: The tour starts by possibly going in a counter-clockwise direction along $(1^-, p)$, $b \le p \le n + 1$, while picking up the loads of the vertices. At vertex p the vehicle makes a U-turn and follows the walks (p^+, b) and then (b^+, q) (i.e., vertex b is an intermediate vertex on the walk (p^+, q)) where $r(q) \in S_{-0}$. These two consecutive walks are followed while loading the supplies of the vertices but without satisfying their demands. From q the vehicle continues clockwise along the walk (q^+, p) while satisfying the demands of the vertices $\{(q + 1)^+, \ldots, p - 1\}$ by possibly using loops. The tour ends by following the walk (p^+, q) and then (q^-, b) while satisfying all the remaining demands and finally ending at b. The cost of this tour is

  $$Z_1^{R+} = 1 + d(1^+, b) + 2d(p^+, 1) + 2d(b^+, q) + \Delta(B(p, q)).$$

  If this proposed walk is optimal then it must satisfy $Z_1^{R+} \le 2 + d(1^+, b)$ as the expression in the right hand side of the inequality represents the cost of a feasible tour that contains a positive ring. Thus p and q must satisfy $2(d(p^+, 1) + d(b^+, q)) + \Delta(B(p, q)) \le 1$, implying that p and q should be chosen such that $d(p^+, 1) + d(b^+, q) \le 0.5$.
- Computation of Z_2^{R+}: The tour starts by possibly going in a counter-clockwise direction along $(1^-, p)$, $p \le n + 1$, while picking up the loads of the vertices. If $p < n + 1$ then $s(p) \in S_{-0}$. Note that the walk $(1^-, p)$ may cross vertex b. Indeed, in the sequel we distinguish between the case that $p \le b$ and $b < p$. At vertex p the vehicle makes a U-turn and follows the walk (p^+, q) where $r(q) \in S_{-0}$, while picking up the loads at all of these vertices that have not been visited yet but without satisfying their demands. Vertex q may be an intermediate vertex on the walk $(p^+, 1)$ or vertex 1 may be an intermediate vertex on the walk (p^+, q). The vehicle then follows the walk (q^+, r), $r = \min\{p, b\}$, while the vertices in $((q + 1)^+, r - 1)$ are served by loops if such loops exist. Here note that if q is a vertex on the walk $(p^+, 1)$ then when starting to serve the vertices along $((q + 1)^+, r - 1)$ by loops, the vehicle is already loaded by the supplies of all the vertices on the walk $(p^+, 1)$. If $r = p$, then when reaching vertex r, all vertices of the circle have already been visited, and the only vertices that are still waiting for their demand objects are along the walk (p^+, q). In this case the vehicle follows the walk (p^+, q) while satisfying the demands of all the vertices, and then the tour ends at vertex b by following (q^-, b). However, if $r = b$, then when reaching b for the first time, there may still be vertices along (b^+, p) that have not been visited yet, implying that when reaching b for the first time, there may be vertices on the walk $((q + 1)^+, b - 1)$ that are waiting for their demand objects to be served by a loop. This last loop is different as at any case, after reaching b for the

first time, the vehicle continues from b following the walks (b^+, q) while picking up the supplies of the vertices on the walk $(b^+, p-1)$ and then satisfying the demands along the walk (q^-, b). When returning to b, if there are still vertices along $((q+1)^+, b-1)$ that are waiting to be served by a loop, the vehicle makes this loop that starts and ends at b.
In order to present the two cases for the computation of Z_2^{R+} we use the following notation: if vertex 1 is on the walk (p^+, q) then let $q' = q$, and otherwise, i.e., if vertex q is on the walk $(p^+, 1)$, then $q' = 1$. (iii) $\Delta(B(p,q))\,|_{q'}^{r}$ denotes the length of the loops along (q^+, r) if the vehicle starts this segment while loaded with the supplies of all the vertices along the walk (p^+, q').

1. Case I: Vertex b is on the walk $(p^+, 1)$. In this case
$$Z_2^{R+} = 1 + d(1^-, p) + d(p^+, q) + d(q^-, b) + \Delta(B(p,q))\,|_{q'}^{p}\,.$$
Using the upper bound $Z_2^{R+} \le 2 + d(1^-, b)$ we obtain that p and q should be chosen such that $d(p^+, q) \le 0.5$.
2. Case II: Vertex p is on the walk $(b^+, 1)$. After pre-loading the supplies along the walks $(1^-, p)$ and (p^+, q), the vehicle is loaded with the supplies of the vertices along (p^+, q'). The vehicle continues from vertex q clockwise, while satisfying demands by loops, if possible, until reaching vertex b. The tour ends by continuing clockwise to q, making there a U-turn and getting back to b. If necessary, the vehicle continues from b in a counter-clockwise in order to make a loop that serves demands of vertices that have not yet been served. The cost in this case is
$$Z_2^{R+} = 1 + d(1^-, p) + \Delta(B(p,q))\,|_{q'}^{b} + d(b^+, p) + 2d(p^+, q).$$
Using the upper bound $Z_2^{R+} \le 2 + d(1^-, b)$ we obtain that p and q should be chosen such that $d(p^+, q) \le 0.5$.

For any given pair (p, q) and any of the two types of solutions, there exists a respective line $L(p, q)$ whose optimal USP solution solves our problem. It remains to find the best combination of a pair (p, q) for each type of solution and pick-up the best of the two. The complexity is again $O(n^3)$.

5. Concluding remarks

The USP on general graphs is known to be NP-hard (see [9]) that prove the NP-hardness of the USP on a tree. In this paper, we investigate the USP on a line and on a circle. In both cases we propose polynomial-time algorithms to solve them to optimality. For the line, the algorithm is linear in the number of vertices, the best complexity that could be expected. The USP on a circle is solved by re-invoking the solution method for the USP on a line on a polynomial number of instances. Similarly to the USP, also the unit capacity SP is known to be polynomial on a line (see [1]) and NP-hard on general graphs. In particular, the unit capacity SP on a tree, for both the preemptive and non-preemptive cases, are NP-hard like the USP on a tree; see [2,11]. However, it is still an open question whether the unit-capacity SP on a circle is polynomially solvable like its uncapacitated version on a circle, or whether there is a further inherent complexity in the problem that makes it NP-hard. We do hope that this research will encourage researchers in the field to investigate this open problem.

Acknowledgments

The research of the first author was partially funded by the Israel Science Foundation, grant no. 401/08 and by the Israeli Institute for Business Research. The authors are grateful to the reviewers of the paper for their valuable comments that greatly helped them in improving the presentation.

References

[1] S. Anily, M. Gendreau, G. Laporte, The swapping problem on a line, SIAM Journal on Computing 29 (1999) 327–335.
[2] S. Anily, M. Gendreau, G. Laporte, The preemptive swapping problem on a tree, Networks 58 (2011) 83–94.
[3] S. Anily, R. Hassin, The swapping problem, Networks 22 (1992) 419–433.
[4] M.J. Attalah, S.R. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, SIAM Journal on Computing 17 (1988) 849–869.
[5] M.O. Ball, M.J. Magazine, Sequencing of insertions in printed circuit board assembly, Operations Research 36 (1988) 192–201.
[6] C. Bordenave, M. Gendreau, G. laporte, A branch-and-cut algorithm for the non-preemptive swapping problem, Naval Research Logistics 56 (2009) 478–486.
[7] C. Bordenave, M. Gendreau, G. laporte, Heuristics for the mixed swapping problem, Computers & Operations Research 37 (2010) 108–114.
[8] P. Chalasani, R. Motwani, Approximating capacitated routing and delivery problems, SIAM Journal on Computing 28 (1999) 2133–2149.
[9] W. de Paepe, J.K. Lenstra, J. Sgall, R. Sitters, L. Stougie, Computer-aided complexity classification of dial-a-ride problems, INFORMS Journal on Computing 16 (2004) 120–132.
[10] G.N. Frederickson, D.J. Guan, Preemptive ensemble motion planning on a tree, SIAM Journal on Computing 22 (1992) 1130–1152.
[11] G.N. Frederickson, D.J. Guan, Non-preemptive ensemble motion planning on a tree, Journal of Algorithms 15 (1993) 29–60.
[12] G.N. Frederickson, M.S. Hecht, C.E. Kim, Approximation algorithms for some routing problems, SIAM Journal on Computing 7 (1978) 178–193.
[13] N. Katoh, T. Yano, An approximation algorithm for the pickup and delivery vehicle on tress, Discrete Applied Mathematics 154 (2006) 2335–2349.
[14] H. Psaraftis, Analysis of an $O(n^2)$ heuristic for the single vehicle many-to-many euclidean dial-a-ride problem, Transportation Research Part B 17 (1983) 133–145.