Les Cahiers du GERAD ISSN: 0711–2440

The Preemptive Swapping Problem on a Tree

S. Anily, M. Gendreau,
G. Laporte

G–2005–69

September 2005
Revised: August 2006

The Preemptive Swapping Problem on a Tree

Shoshana Anily

Faculty of Management
Tel-Aviv University
Tel-Aviv 69978, Israel
anily@post.tau.ac.il

Michel Gendreau

Centre de recherche sur les transports
Université de Montréal
C.P. 6128, Succursale Centre-ville
Montréal (Québec) Canada, H3C 3J7
michelg@crt.umontreal.ca

Gilbert Laporte

GERAD, CRT and Canada Research Chair in Distribution Management
HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada, H3T 2A7
gilbert@crt.umontreal.ca

September 2005
Revised: August 2006

Les Cahiers du GERAD

G–2005–69

Abstract

This paper considers the swapping problem on a tree. In this problem at most one object of some type is available at each vertex, and each vertex also requests at most one object of a given type. The total demand and the total supply of each object type are identical. The problem is to determine a minimum cost routing plan starting and ending at a prespecified vertex which is the depot, for a single vehicle of unit capacity and m object types, so that all vertex requests are satisfied. We consider the preemptive mode in which objects may be temporarily dropped along the way. It is shown that this problem is NP-hard. A heuristic with a worst-case performance ratio of 1.5 is developed. Finally, it is shown that the case where $m = 1$ is polynomially solvable.

Key Words: Swapping Problem, Stacker Crane Problem, Transshipment.

Résumé

On considère le problème d'échanges d'objets sur un arbre. Dans ce problème, au plus un objet d'un type donné est disponible à chaque sommet de l'arbre et chaque sommet requiert au plus un objet d'un certain type. La demande totale et la disponibilité totale de chaque objet sont identiques. Le problème consiste à déterminer une tournée de coût minimum commençant et se terminant à un sommet donné, appelé le dépôt, pour un véhicule de capacité unitaire et m types d'objet, de façon à satisfaire toutes les demandes. On considère le cas où les objets peuvent être temporairement transbordés à des sommets intermédiaires avant d'atteindre leur destination. On démontre que ce problème est NP-difficile. On développe aussi une heuristique avec ratio de performance de pire cas égal à 1.5. Finalement, on démontre que le cas ou $m = 1$ se résout en temps polynomial.

Mots clés : problème d'échanges, problème de la grue, transbordements.

Acknowledgments: The authors would like to thank several anonymous referees for their useful comments, which helped improve the presentation of the paper. This work was partly supported by the Israel Institute of Business and by the Canadian Natural Sciences and Engineering Research Council under grants. OPG0038816 and OPG0039682. This research was carried out while the first author visited the Canada Research Chair in Distribution Management and the Center for Research in Transportation in Montreal.

1 Introduction

The *Swapping Problem* (SP) introduced by Anily and Hassin (1992) is defined as follows. Let $G = (V, E)$ be a connected undirected graph where $V = \{1, ..., n\}$ is a vertex set, vertex 1 represents a *depot*, and $E \subseteq \{[v, w] : v, w \in V, v < w\}$ is an edge set. Each edge $e \in E$ has a non-negative cost or length $c_e \geq 0$. Let also $c(v, w)$ be the length of a shortest path between vertices v and w. Let $O = \{0, ..., m\}$ be a set of object types. Object types $1, \ldots, m$ are real objects whereas object of type 0, also called the *null object*, is a dummy object introduced to simplify the analysis. With each vertex v is associated a pair (a_v, b_v), where $a_v, b_v \in O$; a_v represents an object type supplied by v, while b_v represents an object type required by v. If v has no supply or no demand, this is represented by the null object $a_v = 0$ or $b_v = 0$. It is assumed that the total demand of each object type is equal to its total supply. In the SP vehicles of finite capacity are used to swap objects between vertices in such a way that each vertex receives its required object. Vehicles start and end their trip empty at the depot and must perform all swapping operations while minimizing the total distance traveled.

In this definition of the SP, it is implicitly assumed that at most one object is supplied or required by any vertex. This assumption is not restrictive as long as the number of objects supplied or required by each vertex is finite, since the vertices can be replicated to accommodate the case of multiple objects. As in Anily and Hassin (1992), we assume that there is a single vehicle of unit capacity. In such problems the objects can be *preemptive*, *non-preemptive* or *mixed*. In the preemptive case all objects can be dropped at intermediate vertices while in the non-preemptive case, all objects remain in the vehicle between their origin and their destination. The mixed case allows some of the objects to be preemptive while the others are non-preemptive, as was done in Anily and Hassin (1992) and in Anily, Gendreau and Laporte (1999). Here we assume that the preemptive mode applies.

Applications of the SP arise in the optimization of robot arm movements (Atallah and Kosaraju, 1988) and in printed circuit board assembly (Ball and Magazine, 1988). The SP generalizes the classical *Stacker Crane Problem* (SCP) (Frederickson, 1978) in which objects must be swapped between *specified* origin-destination pairs without preemption. The SCP is a special case of the SP in which each object type is supplied and required by only one vertex. Since the SCP is NP-hard, this is also the case for the SP. Known complexity results on the SCP and the SP for various graph structures are summarized in Table 1.

The purpose of this article is to investigate the preemptive SP on a tree graph. In Section 2 we introduce some notation and preliminary results. We then show in Section 3 that the problem is NP-hard. In Section 4 we develop a polynomial time heuristic with a worst-case performance ratio of 1.5. Finally, in Section 5, we show that the single-type case ($m = 1$) can be solved in polynomial time.

Table 1: Complexity results for the SCP and the SP

	Stacker crane problem			**Swapping problem**		
Graph structure	**Non-Preemptive**	**Preemptive**	**Mixed**	**Non-Preemptive**	**Preemptive**	**Mixed**
General	NP-hard[1]	NP-hard[2]	NP-hard	NP-hard[3]	NP-hard[3]	NP-hard[3]
Tree	NP-hard[4]	Polynomial[5]	NP-hard	NP-hard[4]	NP-hard[6]	NP-hard
Line	Polynomial[7]	Polynomial[8]	Polynomial[9]	Polynomial[9]	Polynomial[9]	Polynomial[9]
Circle	Polynomial[10]	Polynomial[10]	?	?	?	?

[1] Frederickson, Hecht and Kim (1978) provide a heuristic with a worst-case performance ratio of 9/5.

[2] By reduction from the Traveling Salesman Problem.

[3] Anily and Hassin (1992) provide a heuristic with a worst-case performance ratio of 5/2. For $m = 2$, Chalasani and Motwani (1999) improve this ratio to 2.

[4] Frederickson and Guan (1993) provide several heuristics with bounded worst-case performance ratios.

[5] Frederickson and Guan (1992).

[6] This article, Section 3. A heuristic with a worst-case performance ratio of 3/2 is provided in Section 4.The special case where $m = 1$ is polynomial, see Section 5 of this article.

[7] Atallah and Kosaraju (1988), Ball and Magazine (1988).

[8] Atallah and Kosaraju (1988).

[9] Anily, Gendreau and Laporte (1999).

[10] Atallah and Kosaraju (1988).

2 Notation and preliminary results

The SP considered here is defined on a tree $T = (V, E)$ rooted at vertex 1. Everywhere in the paper, except for the proof of Theorem 2, we assume that the depot is located at the root. We denote by OPT the cost of an optimal SP solution, and by $T_v = (V_v, E_v)$ the subtree rooted at v. If v is a leaf of T, then $T_v = (\{v\}, \emptyset)$. A vertex v with $a_v = b_v = 0$ is called a *transshipment vertex*. As in Frederickson and Guan (1992, 1993), we assume without loss of generality that all transhipment vertices (except the root) have a degree at least equal to 3. As an SP solution consists of a sequence of arcs, each associated with a certain product type, we will also consider the set of $2|E|$ arcs, denoted by A, connecting two adjacent vertices of T. For each edge $e = [u, v] \in E$ both, the arc (u, v) directed from u to v, and the arc (v, u) directed from v to u, are in A. In addition, we define a set of $(m + 1)|A|$ *loaded arcs*, denoted by $\bar{A}$, which associates an object type to each arc in A. Thus, $\bar{A} = \{(u, v)^i : (u, v) \in A, i \in O\}$. A *service path* of object $i \in O$ is a sequence of loaded arcs $(u_\ell, u_{\ell+1})^i \in \bar{A}$, $\ell = 1, \ldots, L$, which is traversed while the vehicle is loaded by object i, where the initial vertex on the path supplies object i, i.e., $a_{u_1} = i$, and the ending vertex on the path demands object i, i.e., $b_{u_{L+1}} = i$. A *service cycle of the null object* is a sequence of loaded arcs $(u_\ell, u_{\ell+1})^0 \in \bar{A}$, for $\ell = 1, \ldots, L$, which is traversed while the vehicle is empty and $u_1 = u_{L+1}$.

A feasible solution to the SP consists of a set of service paths, as well as service cycles of the null object. The loaded arcs of a service path of object $i \neq 0$ do not necessarily occur consecutively in the solution because of the preemption option, but their order is preserved. In particular, the first loaded arc on the service path departs from a supply vertex of object i and the last loaded arc enters a demand vertex of object i. Suppose that V contains n_i vertices whose supply is i but whose demand is different from i. Assuming that a vehicle never unloads an object at a vertex in order to immediately load the same object, any feasible solution contains exactly n_i service paths of object i. Each vertex v for which $i = a_v \neq b_v$ is a starting point for such a service path, and each vertex v for which $i = b_v \neq a_v$ is the end point of such a service path. The set of loaded arcs carrying the null object in a feasible solution forms service paths and service cycles of the null object. However, the loaded arcs of the null service paths are not necessarily traversed consecutively in a feasible solution, or in the order defined by the service path, as any vertex can be assumed to hold or require the null object.

As in Anily, Gendreau and Laporte (1999) we construct the directed *multi-type multi-graph* B, which we call the *basic graph.* “Multi-type” refers to the fact that an object type is associated with each arc, referred to as a loaded arc; “multi-graph” means that several copies of the same loaded arc (origin, destination, object type) can exist. More specifically, the loaded arcs of the basic graph can be partitioned according to their object type, i.e., $B = (V, \bar{A}^0 \cup \cdots \cup \bar{A}^m)$. We construct B as follows: First initialize the sets of loaded arcs $\bar{A}^0 := \bar{A}^1 := ... =: \bar{A}^m = \emptyset$. Consider in turn every vertex $v \in V$, the subtree T_v and the predecessor (father) vertex $p(v)$ of v. For every $i \in O$, compute $\Delta_v^i = |\{w \in V_v : a_w = i\}| - |\{w \in V_v : b_w = i\}|$. The quantity Δ_v^i represents the net supply of object i in T_v. If Δ_v^i is positive, then Δ_v^i copies of the loaded arc $(v, p(v))^i$ are added to $\bar{A}^i$. If Δ_v^i is negative, then $|\Delta_v^i|$ copies of loaded arc $(p(v), v)^i$ are added to $\bar{A}^i$. $|\Delta_v^i|$, for the case $\Delta_v^i > 0$ (resp. $\Delta_v^i < 0$), represents the minimum number of times the loaded arc $(v, p(v))^i$ (resp. $(p(v), v)^i$) will be traversed in an optimal SP solution. This construction process ensures the existence of a service path between the origin of every object and at least one of its destinations. The total length of the loaded arcs of B is denoted by $cost(B)$, and thus $cost(B) \leq OPT$. Constructing B can be achieved in $O(n)$ time, starting from the leaves of T and gradually moving toward the root. The basic graph is fully directed and balanced, i.e., the in-degree of each vertex is equal to its out-degree, since for every subtree T_v we must have $\sum\limits_{i \in O} \Delta_v^i = 0$. The loaded arcs of the basic graph B form a union of service paths, where each service path of object $i \in O$ starts at a vertex which is the origin of i and ends at a vertex that demands i. However, B is not necessarily connected, but when it is connected, it is then strongly connected since it is balanced. Even if B is strongly connected, it may not be possible to obtain a feasible SP solution by using its loaded arcs. Consider for example the tree depicted in Figure 1, where the label of v is (a_v, b_v), and the corresponding basic graph where the label of each loaded arc is the object type carried on this arc. No SP solution using the loaded arcs of B exists given that the vehicle must start and end its trip empty at vertex 1.

A *feasible path* in the basic graph B consists of a sequence $(u_\ell, u_{\ell+1})^{i_\ell} \in \bar{A}$, $\ell = 1, \ldots L$, $i_\ell \in O$ of loaded arcs of B, which a unit capacity vehicle can follow assuming that the

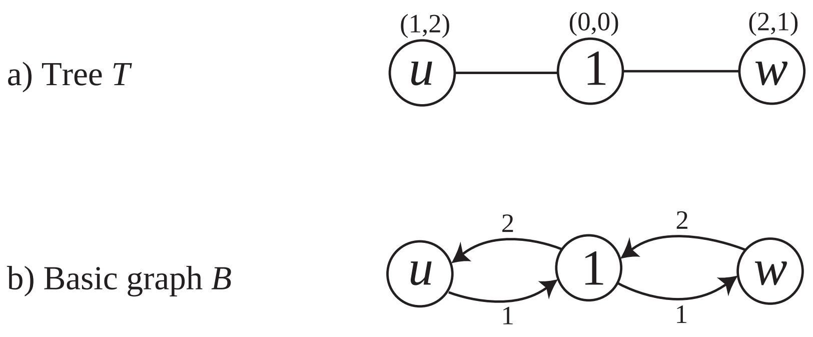

Figure 1: Instance for which B is connected but does not contain a feasible SP solution

vehicle starts empty at the first vertex u_1. This means that the necessary object on each loaded arc is available when the vehicle reaches the tail of the loaded arc.

We should also note that a vertex v with $a_v = b_v \in O - \{0\}$ cannot be replaced by a transshipment vertex, as the supply at vertex v is not necessarily used for covering the demand of the vertex. To see this, consider the example in Figure 1 where $(a_1, b_1) = (1, 1)$. This change does not effect the basic graph B. It is easily seen that a feasible SP solution now exists assuming that the vehicle starts empty at vertex 1. However, without loss of generality, the leaves of the tree can be assumed to supply a different object type than the one they demand.

The example of Figure 1 demonstrates that a stronger property than strong connectivity is required to guarantee a feasible SP solution. To this end we recall the property of *reachability* introduced in Anily, Gendreau and Laporte (1999). Vertex w is said to be reachable from vertex u if there exists a feasible path from u to w in the basic graph B. For example, in Figure 1, vertices 1 and w are reachable from u, vertices 1 and u are reachable from w but neither u nor w is reachable from 1. Thus reachability is not symmetric but is transitive. Two vertices are *mutually reachable* if they are reachable from one another. It may sometimes be necessary to drop an object at an intermediate point along a path in order to allow reachability. For example, in the graph depicted in Figure 2, vertex u is reachable from vertex v assuming object 1 can be dropped at vertex 1. However, v is not reachable from u since object 2 cannot be made available at vertex 1.

Since mutual reachability is symmetric and transitive and each vertex is auto-reachable, the mutual reachability relation induces a partition of V into equivalence classes called *fully connected components*. The vertices of the fully connected components are not necessarily contiguous. For example, in the SP depicted in Figure 1, there are two fully connected components: the first one consists of vertices u and w, which are not contiguous vertices, and the second one consists of vertex 1. However, if instead of being a transshipment point vertex 1 had $a_1 = b_1 = 1$, then the basic graph B would consist of a single fully connected component. We say that a basic graph is *fully reachable* if it consists of a single fully

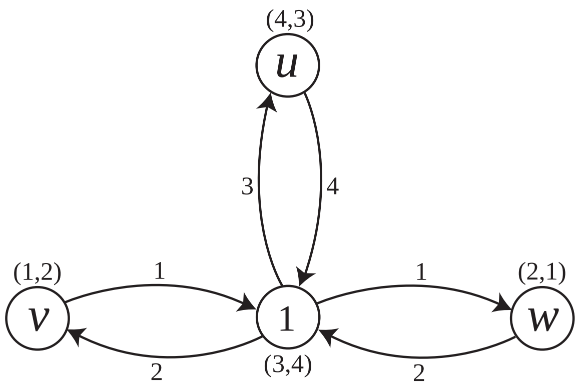

Figure 2: Strongly connected graph in which not all vertices are mutually reachable

connected component, i.e., all vertices are reachable from all vertices. In the next lemma we prove that a fully connected component must be balanced, i.e., in each component the total demand for object $i \in O$ equals the total supply of object i. In particular, this lemma implies that a fully connected component which is a singleton, must necessarily consist of a vertex v with $a_v = b_v$. However, not every vertex v with $a_v = b_v$ defines a singleton in B.

Lemma 2.1 *Any fully connected component of B is balanced.*

Proof. Suppose by contradiction that there exists a fully connected component C_0 of B which is not balanced. This assumption implies that the number of fully connected components of B is greater than one. Moreover, it means that within C_0 there exists an object type $i_1 \in O$ having a total demand greater than its total supply, and an object type $i_2 \in O$, $i_2 \neq i_1$, having a total demand smaller than its total supply. As a result, there exists a fully connected component C_{-1} supplying i_1 to C_0, i.e., C_0 is reachable from C_{-1} along a service path of i_1. And there also exists a fully connected component C_1 to which a unit of i_2 is shipped from C_0, and thus C_1 is reachable from C_0 along a service path of i_2. Clearly, $C_{-1} \neq C_1$ as otherwise C_0 and C_1 would have been mutually reachable. As a unit exits (enters) C_{-1} (C_1), a unit must enter (exits) C_{-1} (C_1), proving that there exist two fully connected components C_{-2} and C_2 such that C_{-1} is reachable from C_{-2} and C_2 is reachable from C_1, and C_{-2}, C_{-1}, C_0, C_1 and C_2 are all disjoint components. Repeating this argument again and again yields a contradiction to the the fact that the number of fully connected components must be finite (no more than the number of vertices in V). $\square$

Theorem 1 *An SP solution exists on the basic graph B if and only if this graph is fully reachable.*

Proof. If B is not fully reachable, then no SP solution exists since either the depot cannot be reached from at least one vertex, or at least one vertex cannot be reached from the depot. If B is fully reachable, then an SP solution can be identified by suitably modifying Hierholzer's (1873) *end-pairing* algorithm for the *Chinese Postman Problem* on an undirected Eulerian graph.

Step 1. Starting at an arbitrary vertex $v \in V$, follow a service path of object of type a_v in the basic graph B emanating from v, until the object reaches its first destination at u. Iteratively apply this process starting from u until v is eventually reached. Go to Step 3.

Step 2. Construct a second circuit starting from a vertex $w \in V$ of the first circuit whose supply has not yet been delivered (such a vertex necessarily exists since B is fully reachable and not all vertices have yet been served). Merge the two circuits $(v, \ldots, w_1, w, w_2, \ldots, v)$ and $(w, w_3, \ldots, w_4, w)$ into a single circuit $(v, \ldots, w_1, w, w_3, \ldots, w_4, w, w_2, \ldots, v)$. When reaching w for the first time on the merged circuit, drop the object loaded on the vehicle and replace it by the object of type a_w. When reaching w for the second time the object dropped at w is loaded on the empty vehicle.

Step 3. If the circuit contains all the loaded arcs of B, stop. Otherwise, go to Step 2. $\square$

The balance of B, and of all fully connected components of B, as proved in Lemma 2.1, and the mutual reachability within each fully connected component ensure that if B consists of a number of fully connected components, then the loaded arcs of B generate a set of cycles, one for each component. Each such cycle induces a feasible closed tour for a unit capacity vehicle servicing the component's demands by using its supplies. Two cycles associated with different components can be disjoint or reachable but not mutually reachable, as otherwise they would be merged into one fully connected component. If B is not fully reachable, it may be augmented into a fully reachable graph by adding new loaded arcs to it. As will be seen in Section 3, the problem of determining a least cost augmentation of B into a fully reachable graph is NP-hard.

3 The preemptive swapping problem on a tree is NP-hard

In this section we prove that the preemptive SP on a tree is NP-hard. To this end we show that the problem is at least as hard as the Steiner tree problem on a bipartite graph, which is defined as follows (see Garey and Johnson (1979), pages 208-209). Consider a bipartite graph $G = (\tilde{V}, \tilde{E})$ with bipartition $\{R, \tilde{V} - R\}$ of $\tilde{V}$, an integer weight c_e^G for each edge $e \in \tilde{E}$, and a positive integer number β. The problem is to determine whether there exists a subtree of G that spans at least the vertices of R such that the total weight of the edges in the subtree does not exceed β. Frederickson and Guan (1993) use a similar version of the Steiner tree problem where all weights are equal to prove that the non-preemptive SC problem on a tree is NP-hard.

Theorem 2 *The preemptive SP on a tree is NP-hard.*

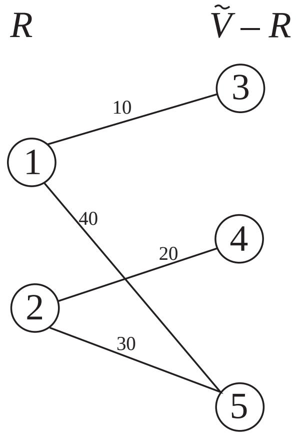

Figure 3: Graph $G = (\tilde{V}, \tilde{E})$ for the Steiner tree problem on a bipartite graph

Proof. Consider a bipartite graph $G = (\tilde{V}, \tilde{E})$ with a bipartition $\{R, \tilde{V} - R\}$ of $\tilde{V}$. Let $|\tilde{V}| = \vartheta$, $|\tilde{E}| = \varepsilon$, and $|R| = \rho$. We number the vertices of R by $1, \dots, \rho$, and the vertices of $\tilde{V} - R$ by $\rho+1, \dots, \vartheta$. The length of edge $[v, u] \in \tilde{E}$ is denoted by $c^G(v, u)$. We also let Γ_v be the set of vertices adjacent to vertex v, i.e., the neighborhood of vertex v. Denote by k_v the cardinality of Γ_v, i.e., $|\Gamma_v| = k_v$, thus $\Sigma_{v=1}^{\rho} k_v = \varepsilon$ and $\Sigma_{u=\rho+1}^{\vartheta} k_u = \varepsilon$. For any vertex $v \in R$, let $\varphi(v, i)$ be the i^{th} closest vertex to v among the k_v vertices in Γ_v (where ties are broken arbitrarily). Thus, for a given vertex $v \in R$, $\varphi(v, \cdot)$ is a one-to-one mapping of $\{1, \dots, k_v\}$ into Γ_v such that $c^G(v, \varphi(v, 1)) \leq c^G(v, \varphi(v, 2)) \leq ... \leq c^G(v, \varphi(v, k_v))$. Figure 3 depicts a graph $G = (\tilde{V}, \tilde{E})$ with $\rho = 2$, $\vartheta = 5$, $\varepsilon = 4$, $c^G(1, 3) = 10$, $c^G(1, 5) = 40$, $c^G(2, 4) = 20$, $c^G(2, 5) = 30$, $\varphi(1, 1) = 3$, $\varphi(1, 2) = 5$, $\varphi(2, 1) = 4$, $\varphi(2, 2) = 5$, $\Gamma_1 = \{3, 5\}$, $\Gamma_2 = \{4, 5\}$, $k_1 = 2$, $k_2 = 2$, $\Gamma_3 = \{1\}$, $\Gamma_4 = \{2\}$, $\Gamma_5 = \{1, 2\}$, $k_3 = 1$, $k_4 = 1$, $k_5 = 2$.

We will show that if there exists a polynomial algorithm for solving the preemptive SP on a tree then we could have used it to determine a minimum Steiner tree problem on $G = (\tilde{V}, \tilde{E})$ spanning the vertices of R. To this end we construct the following reduction.

We first define an SP tree $T = (U, E^*)$ consisting of $m = \vartheta - \rho + \varepsilon$ different objects, in addition to the null object, and $3\varepsilon + 2\rho + 1$ vertices. Each vertex $u \in \tilde{V} - R$ induces $1 + k_u$ different objects, named u^* and u_v for $v \in \Gamma_u$. We let the function $c^T : E^* \rightarrow \Re^+$ represent the length of the edges of T.

We next describe the construction of T. The tree T is rooted at vertex $\ell(0, 0)$ (which does not serve as the depot) and is connected to ρ children named $\ell(1, 0), \dots, \ell(\rho, 0)$. The root and its children are assumed to be transshipment points, i.e., $a_{\ell(v,0)} = b_{\ell(v,0)} = 0$ for $v \in \{0, \dots, \rho\}$. Each of the ρ children of the root serves as the root of a subtree $T_{\ell(v,0)}$ for $v \in \{1, \dots, \rho\}$. Let $c^T(\ell(0, 0), \ell(v, 0)) = c^G(v, \varphi(v, 1))$ for $v \in \{1, \dots, \rho\}$, i.e., the distance between the root of T, namely vertex $\ell(0, 0)$, and its child $\ell(v, 0)$, is equal to the minimum distance between v and a vertex in Γ_v in graph G. We next describe the construction of $T_{\ell(v,0)}$ for some $v \in \{1,\ldots,\rho\}$. Vertex $\ell(v,0)$ is a parent of two children, the right-hand side child $r(v,1)$ and the left-hand side child $\ell(v,1)$. The right-hand side child of $\ell(v,0)$, namely vertex $r(v,1)$, is a leaf associated with objects $(a,b) = (\varphi(v,1)^*, \varphi(v,1)_v)$. We let $c^T(\ell(v,0), r(v,1)) = M$ for some $M \in \Re^+$ to be specified later. If $|\Gamma_v| > 1$, then the left-hand side child of $\ell(v,0)$, namely $\ell(v,1)$, is a transshipment vertex, and $c^T(\ell(v,0), \ell(v,1)) = c^G(v, \varphi(v,2)) - c^G(v, \varphi(v,1))$, which is non-negative by definition of the function $\varphi(v,\cdot)$. Continuing the construction of T, we let $\ell(v,1)$ to be a parent of the two children $r(v,2)$ and $\ell(v,2)$. The right-hand side child of $\ell(v,1)$, namely vertex $r(v,2)$, is a leaf associated with objects $(a,b) = (\varphi(v,2)^*, \varphi(v,2)_v)$. We let $c^T(\ell(v,1), r(v,2)) = M$. If $|\Gamma_v| > 2$ the left-hand side child of $\ell(v,1)$, namely vertex $\ell(v,2)$, is a transshipment vertex. We let $c^T(\ell(v,1), \ell(v,2)) = c^G(v, \varphi(v,3)) - c^G(v, \varphi(v,2))$ which is again non-negative. This construction process continues for $k_v - 1$ generations, where the total distance on T between the root of the tree, namely vertex $\ell(0,0)$ and vertex $\ell(v, k_v - 1)$, is $c^G(v, \varphi(v, k_v))$, which is the distance in graph G between vertex v and the vertex farthest from v and adjacent to it. We complete the construction of the subtree as follows. The right-hand side child of vertex $\ell(v, k_v - 1)$, namely vertex $r(v, k_v)$, is a leaf associated with objects $(a,b) = (\varphi(v,k_v)^*, \varphi(v,k_v)_v)$. We let $c^T(\ell(v, k_v - 1), r(v, k_v)) = M$. The left-hand side child of $\ell(v, k_v - 1)$, namely vertex $\ell(v, k_v)$, is associated with objects $(a,b) = (\varphi(v,1)_v, \varphi(v,1)^*)$. We let $c^T(\ell(v, k_v - 1), \ell(v, k_v)) = M$. Vertex $\ell(v, k_v)$ has two children. The right-hand side child $r(v, k_v + 1)$ is a leaf associated with objects $(a,b) = (\varphi(v,1)^*, \varphi(v,1)_v)$. The left-hand side child of $\ell(v, k_v)$, namely vertex $\ell(v, k_v + 1)$, is the root of a subtree which is a path consisting of exactly k_v vertices named $\ell(v, k_v + 1), \ldots, \ell(v, 2k_v)$. The last vertex on the path, i.e., vertex $\ell(v, 2k_v)$, is associated with objects $(a,b) = (\varphi(v,1)_v, \varphi(v,1)^*)$. For $i = 1, \ldots, k_v - 1$, vertex $\ell(v, k_v + i)$ is associated with objects $(a,b) = (\varphi(v, i+1)_v, \varphi(v, i+1)^*)$. The length of all edges of the subtree, which is rooted at vertex $\ell(v, k_v)$, can be assumed to be 0. Figure 4 depicts the tree $T = (U, E^*)$, corresponding to the example of Figure 3, with $m = \vartheta - \rho + \varepsilon = 5 - 2 + 4 = 7$, $|U| = 3\varepsilon + 2\rho + 1 = 17$, and the object set $O = \{0, 3^*, 3_1, 4^*, 4_2, 5^*, 5_1, 5_2\}$.

We note that by this construction each subtree $T_{\ell(v,0)}$ consists of $3k_v + 2$ vertices, where exactly k_v of them are transshipment points. The subtrees are balanced because the total supply of each object type is equal its total demand. More precisely, each of the objects $\varphi(v,i)^*$ and $\varphi(v,i)_v$ for $i = 2, \ldots, k_v$ is the supply and the demand of exactly one vertex in the subtree. Each of the objects $\varphi(v,1)^*$ and $\varphi(v,1)_v$ is the supply and demand of two vertices in the subtree. We also note that object type $\varphi(v,i)_v$ for $i = 1, \ldots, k_v$ are not used by any subtree $T_{\ell(v',0)}$, $v' = 1, \ldots, \rho$ other than $T_{\ell(v,0)}$. On the other hand, objects $\varphi(v,i)^*$ for $i = 1, \ldots, k_v$ are used by all subtrees $T_{\ell(v',0)}$ with $\varphi(v,i)^* \in \Gamma_{v'}$ and $v' = 1, \ldots, \rho$. In other words, object u^* for $u \in \tilde{V} - R$ is used by all subtrees $T_{\ell(v',0)}$ for which $v' \in \Gamma_u$.

We denote the basic graph associated with T by $B(T)$. It consists of exactly ρ fully connected components which are not singletons. Each fully connected component C_v for $v = 1, \ldots, \rho$ consists of all vertices of the subtree $T_{\ell(v,0)}$, which are not transshipment points, namely vertices $r(v,i)$ for $i = 1, \ldots, k_v + 1$, and vertices $\ell(v, k_v + i)$ for $i = 0, \ldots, k_v$.

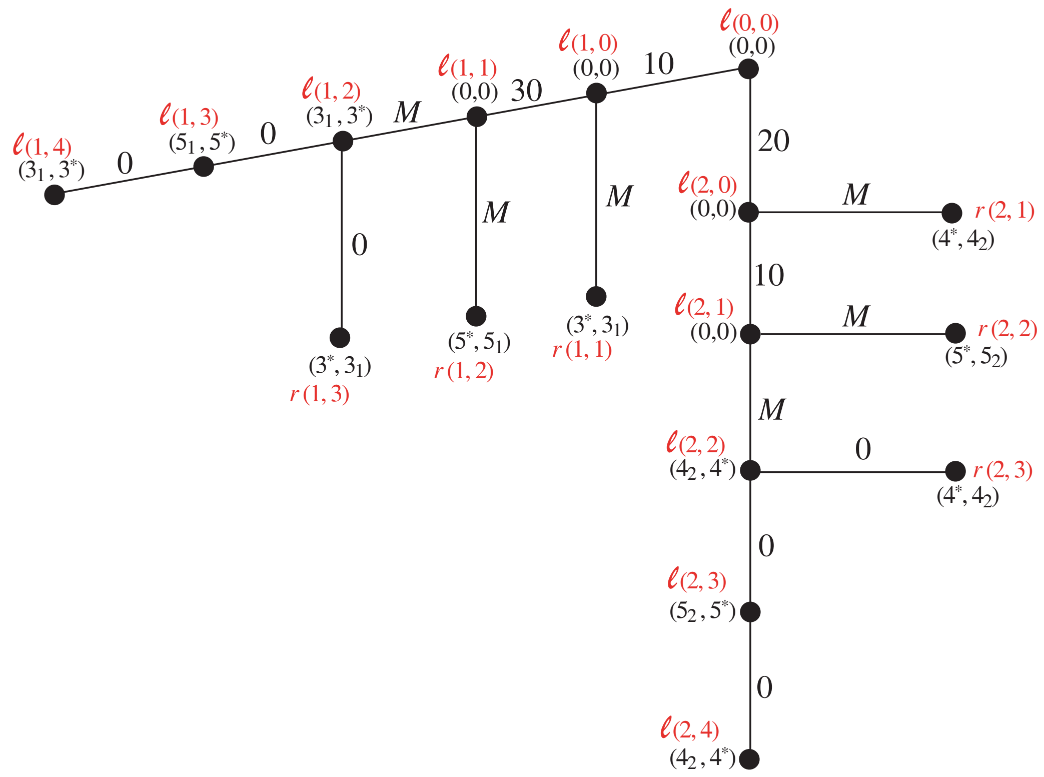

Figure 4: Swapping problem on a tree $T = (U, E^*)$ corresponding to the Steiner tree problem of Figure 3

Each of the transshipment vertices of $T_{\ell(v,0)}$ serves as a fully connected component which is a singleton. In particular, each singleton in $T_{\ell(v,0)}$ is reachable from the component C_v. The graph $B(T)$ is disconnected since the root of T is not incident to any loaded arc of $B(T)$, i.e., vertex $\ell(0,0)$ is not reachable from any of the other fully connected components. Thus graph $B(T)$ must be augmented by adding two opposite direction loaded arcs of the form $(\ell(0,0), \ell(v,0))^{i_v}$ and $(\ell(v,0), \ell(0,0))^{i_v}$ for $v = 1, \ldots, \rho$ and $i_v \in O$. The total cost of these new loaded arcs is $2\Sigma_{v=1}^{\rho} c^G(v, \varphi(v,1))$. Independently of the location of the depot, any feasible solution for the SP on T must be of a length at least equal to $cost(B(T)) + 2\Sigma_{v=1}^{\rho} c^G(v, \varphi(v,1))$. In particular, the total length of the loaded arcs of $B(T)$ whose cost is M, which we call in the sequel *large loaded arcs*, is $\Sigma_{v=1}^{\rho} 4Mk_v = 4M\varepsilon$, as each of the edges of T, $[\ell(v, i-1), r(v,i)]$ for $i = 1, \ldots, k_v$ is associated with two loaded arcs in $B(T)$, and the edge $[\ell(v, k_v - 1), \ell(v, k_v)]$ of T is covered by $2k_v$ loaded arcs in $B(T)$. We choose M to be a very large number, say $M > 2\varepsilon cost(G) = 2\varepsilon \sum_{(v,u)\in\tilde{E}} c^G(v,u)$, in order to guarantee that the optimal SP solution includes the least possible number of copies of large loaded arcs. In $B(T)$ all the large loaded arcs are associated with a real object in $O - \{0\}$. Considering Figure 4, we can see that if the vehicle while empty enters vertex

$\ell(v, 0)$ for $v = 1, \ldots, \rho$, for the first time, coming from $\ell(0, 0)$, then in order to continue servicing component C_v from there, the vehicle must travel along two opposite-direction large loaded arcs of the null object, this is in addition to the loaded arcs of $B(T)$. The first empty such travel along the large loaded arc is needed in order to first reach a vertex in C_v from which an object can be loaded. The cost of the travels along large loaded arcs of the null object can be avoided only if vertex $\ell(v, 0)$ is reached from $\ell(0, 0)$ while the vehicle is already loaded by an object that is needed in C_v. As will be shown below, if there exists a Steiner Tree on the bipartite graph $G = (\tilde{V}, \tilde{E})$ spanning the vertices of R, then any optimal SP solution for T includes only the large loaded arcs that are members of $B(T)$, i.e., exactly 4ε large loaded arcs which are all travelled while the vehicle is loaded by a real object, meaning that objects in one component are used to cover the demand in other components.

In order to complete the definition of the SP on T, we assume here that the depot is located at vertex $r(1, 1)$, which, by definition, is not a transshipment vertex (any other choice of the depot's location at a non-transshipment vertex of T would do as well). Any feasible solution to the preemptive SP on T consists of $B(T)$'s loaded arcs plus two opposite direction service paths connecting $\ell(0, 0)$ with the fully connected components C_v for $v = 1, \ldots, \rho$ in order to make them mutually reachable. As explained above, in a solution in which the fully connected components are autonomous, i.e., each satisfies its own demand, the vehicle traverses the two-opposite direction loaded arcs, $(\ell(0, 0), \ell(v, 0))^0$ and $(\ell(v, 0), \ell(0, 0))^0$ for $v = 1, \ldots, \rho$, as well as the two-opposite direction large loaded arcs $(\ell(v, 0), r(v, 1))^0$ and $(r(v, 1), \ell(v, 0))^0$ in each C_v for $v \neq 1$, in order to reach the first non-transshipment vertex in that component. Thus the total cost of such a solution is $cost(B(T)) + 2\Sigma_{v=1}^{\rho} c^G(v, \varphi(v, 1)) + 2(\rho - 1)M$. We show below that exploiting the possibility of using objects of type u^*, $u \in \{\rho + 1, \ldots, \vartheta\}$, supplied in one fully connected component to cover the demand of another component, may generate shorter SP solutions in which the vehicle does not make empty trips along large loaded arcs.

Consider now a certain fully connected component C_v for $v \in \{2, \ldots, \rho\}$. The only way to serve C_v without paying travels along a large loaded arc which is associated with the null object, is by entering C_v while carrying an object u^* originating from another fully connected component, say $C_{v'}$ for $v' \in \{1, \ldots, \rho\} - \{v\}$ and $u \in \Gamma_v \bigcap \Gamma_{v'}$. To this end suppose that $\varphi(v', i) = u$ and $\varphi(v, j) = u$. The transfer of object u^* from component $C_{v'}$ to component C_v is achieved as follows: suppose that the vehicle loads object u^* at vertex $r(v', i)$, when there is still demand for this object in C_v. The vehicle then follows the loaded arc of $B(T)$, $(r(v', i), \ell(v', i - 1))^{u^*}$, and from vertex $\ell(v', i - 1)$ the vehicle, while loaded with object u^*, exits $C_{v'}$ by continuing to the root, and from there to the fully connected component C_v along the path to vertex $\ell(v, j - 1)$.From $\ell(v, j - 1)$ the vehicle continues along the path that connects $\ell(v, j - 1)$ to the vertex $\ell(v, k_v + j - 1)$ in C_v demanding this object. In order to balance the shipment of the unit of object u^* to C_v, another unit of object u^* must eventually be carried out from C_v in order to cover the demand in some other component. We note that this solution necessitates, in addition to the loaded arcs of $B(T)$, two opposite direction loaded paths associated with object u^*

that connect vertex $\ell(v', i-1)$ to vertex $\ell(v, j-1)$. The cost of these two loaded paths is $2c^T\big(\ell(v', i-1), \ell(0,0)\big) + 2c^T\big(\ell(0,0), \ell(v, j-1)\big) = 2c^G(v', \varphi(v', i)) + 2c^G(v, \varphi(v, j)) = 2c^G(v', u) + 2c^G(v, u)$, i.e., the cost of such a transfer boils down to twice the cost of two edges $[v', u]$ and $[v, u]$ in the bipartite graph G. In other words, it is twice the cost of connecting vertices $v', v \in R$ through vertex $u \in \tilde{V} - R$ while constructing a Steiner tree on G spanning R. If there exists a Steiner Tree on the bipartite graph $G = (\tilde{V}, \tilde{E})$ spanning the vertices of R, then an optimal SP solution for T does not contain any extra large loaded arcs beyond the 4ε such arcs that are part of $B(T)$. In such a case, all fully connected components C_v, $v \in R$, must be connected to each other by transferring objects of type u^*, $u \in \tilde{V} - R$ as explained above. Any feasible solution to the SP on T that does not contain more than 4ε large loaded arcs induces a Steiner tree spanning R on the bipartite graph $G = (\tilde{V}, \tilde{E})$. The cost of the SP solution is the sum of $cost(B(T))$ and twice the cost of the associated Steiner tree on $G(\tilde{V}, \tilde{E})$ used to connect the components. Thus, an algorithm that finds the optimal SP solution on T would also find an optimal Steiner tree spanning R in the bipartite graph $G = (\tilde{V}, \tilde{E})$, proving that the SP is at least as hard as the Steiner tree problem on a bipartite graph, and meaning that the preemptive SP on a tree is NP-hard. $\square$

4 A 1.5-approximation algorithm for the preemptive SP on a tree

A lower bound on the optimal solution for the preemptive SP on a tree T can easily be found. Consider the basic graph B associated with the problem. This graph is balanced but not necessarily connected. In the first step, we augment B by identifying the edges of T not covered by any loaded arc of B. For each such edge we add to B two opposite direction loaded arcs, each associated with the null object. Let B' be the resulting directed multi-type, multi-graph which is clearly balanced and connected. As explained in Section 2, this augmentation does not guarantee the existence of a feasible SP solution on B', but $cost(B') \leq OPT$. We will show below that graph B' can be further augmented to obtain a feasible SP solution by adding loaded arcs of the null object, so that the cost of the heuristic solution is at most $1.5OPT$.

Graph B' consists of a number of fully connected components $C_1, \ldots, C_r$. Recall that the depot is located at the root of T, which we assume to be in component C_1. If $r = 1$, then by Theorem 1 the loaded arcs of B' induce a feasible and optimal SP solution. A fully connected component of B' is said to be *unreachable* if none of its vertices is reachable from a vertex in any other fully connected component by following the service paths induced by B''s loaded arcs. Component C_1, where the depot is located, is considered to be reachable (from the depot). In the next lemma we prove that if $r \geq 2$, then there must exist at least one fully connected component of B' that is unreachable.

Lemma 4.1 *If B' contains at least two fully connected components then at least one of the components is unreachable.*

Proof. Suppose by contradiction that none of the fully connected components of B' is unreachable, i.e, each component (except possibly for the component that contains the depot) can be reached from some other component. Choose any fully connected component that does not contain the depot to start with. Call it $C_{\alpha(1)}$. Since $C_{\alpha(1)}$ is reachable, there exists another component $C_{\alpha(2)}$ from which $C_{\alpha(1)}$ can be reached. Clearly $C_{\alpha(2)}$ cannot be reached from $C_{\alpha(1)}$ as otherwise the two components could be merged. Similarly, $C_{\alpha(2)}$ can be reached from some other component $C_{\alpha(3)}$, $\alpha(3) \neq \alpha(1)$. Component $C_{\alpha(3)}$ cannot be reached from neither $C_{\alpha(1)}$ or $C_{\alpha(2)}$, otherwise two components could be merged, contradicting the partition of B' into fully connected components. This procedure can be followed for a finite number of steps since B' consists of a finite number of components. At the end of the procedure we obtain a sequence of mutually disjoint fully connected components, where $C_{\alpha(q-1)}$ is reachable from $C_{\alpha(q)}$ for $q = 2, \ldots, Q$. Thus, component $C_{\alpha(Q)}$ is unreachable, contradicting our assumption. □

In the algorithm we propose, we will successively add pairs of opposite-direction loaded arcs of the null object until we obtain a partition of V into a fully connected component which contains the depot, and possibly some singleton fully connected components. More specifically, as long as the multi-graph on hand is partitioned to more than one fully connected component which are not singletons, we start a new iteration (a singleton can serve itself). At each iteration we identify the set of all unreachable vertices, which belong to non-singleton fully connected components in the multi-graph at hand, for which all their predecessors (in T) are reachable. Among those unreachable vertices, we find the one which is closest to its predecessor in T in terms of the cost function $c : E \rightarrow \Re^+$. We update the multi-graph at hand by adding to it two opposite-direction loaded arcs of the null object connecting this vertex to its predecessor vertex (in T), and we restart a new iteration. This procedure obviously preserves the balance of the graph and is finite. We present now the *Augmentation Algorithm* for the SP on $T = (V, E)$. The algorithm returns a multi-directed graph H in which each loaded arc is associated with one of the objects in O.

Step 1. Input: $T = (V, E)$, the cost c_e for each $e \in E$ and the associated graph B'. Output: multi-type, directed multi-graph H, and its cost $Cost(H)$. $\tilde{B} \leftarrow B'$. Go to Step 2.

Step 2. Find the partition of $\tilde{B}$ into fully connected components $\{C_1, \ldots, C_r, C_{r+1} \ldots C_{r'}\}$, where $C_{r+1} \ldots C_{r'}$ are singletons. If $r = 1$, return $H \leftarrow \tilde{B}$; and $cost(H)$; stop. Otherwise, let $C'_1, \ldots, C'_K$, $K \geq 1$, be the unreachable components of $\tilde{B}$ which are not singletons. Go to Step 3.

Step 3. For $\ell = 1, \ldots, K$ define $S_\ell \subset V$ to be the set of vertices of the unreachable component C'_ℓ. $S \leftarrow \cup_{\ell=1}^{K} S_\ell$; $\tilde{S} \leftarrow \emptyset$. For each $v \in S$ do begin: if none of the predecessors of v in T is a vertex in S, then $\tilde{S} \leftarrow \tilde{S} \cup \{v\}$; end.

Step 4. For each vertex $v \in \tilde{S}$ do begin: $\kappa_v \leftarrow c(p(v), v)$; Let $v* \in \text{argmin}\{\kappa_v : v \in \tilde{S}\}$, where ties are broken arbitrarily. Go to Step 5.

Step 5. Augment the graph $\tilde{B}$ by adding to it the loaded arcs $(v*, p(v*))^0$ and $(p(v*), v*)^0$. Go to Step 2.

We first prove that the resulting graph H induces a feasible SP solution.

Theorem 3 *There exists a feasible SP solution on H.*

Proof. First note that graph H is connected and balanced since graph B' had this property and during the augmentation the balance is preserved. In the graph H there are no unreachable components, which are not singletons. A feasible SP solution on graph H can be obtained by the modification of the Hierholzer's *end-pairing* algorithm for the *Chinese Postman Problem* used in the proof of Theorem 1. □

In order to analyze the worst-case performance ratio of the proposed heuristic we need a further characterization of the intermediate graphs $\tilde{B}$ obtained during the application of the algorithm.

Lemma 4.2 *Consider a tree $T = (V, E)$ and a corresponding balanced and connected graph $\tilde{B}$ obtained in the application of the Augmentation Algorithm. Suppose that C' is an unreachable component of $\tilde{B}$. Consider a vertex $w \in C'$ and the subtree $T_w = (V_w, E_w)$. If in the subtree $T_w = (V_w, E_w)$, the set of vertices V_w contains a vertex u belonging to an unreachable fully connected component of $\tilde{B}$, say $\tilde{C}$, $\tilde{C} \neq C'$, then $\tilde{C} \subset V_w$.*

Proof. Suppose that $\tilde{C}$ is not a subset of V_w. Then, there exists a vertex $u_1 \in \tilde{C} \cap V_w$, but $\tilde{C}$ has at least one vertex u_2 that is not a member of V_w. By definition of a fully connected component, there must exists in $\tilde{B}$ a service path that starts at u_1 and ends at vertex u_2. By the property of a tree, this unique path must pass through vertex w. This service path makes vertex w reachable from $\tilde{C}$, violating the assumption that w belongs to an unreachable fully connected component of $\tilde{B}$. □

Consider any unreachable component of $\tilde{B}$. In Lemma 4.3 we prove that the root of the minimal subtree containing this component is not a member of the component.

Lemma 4.3 *Consider an unreachable component C' of graph $\tilde{B}$ obtained in the Augmentation Algorithm. Let $T_w = (V_w, E_w)$ be the minimal subtree for which $C' \subseteq V_w$. Then $w \notin C'$.*

Proof. Suppose by contradiction that $w \in C'$. The fact that $T_w = (V_w, E_w)$ is the minimal subtree for which $C' \subseteq V_w$ implies that $p(w) \notin C'$. By Lemma 4.2 all non-reachable components of $\tilde{B}$ having a vertex in V_w are contained in V_w. Suppose that $u \in V_w$, and u belongs to a reachable component of $\tilde{B}$, which we denote by C^*. If $C^* \not\subset V_w$ then there exists a service path in $\tilde{B}$ connecting vertex u to a vertex $v \notin V_w$ through vertex w, contradicting the fact that C' is a non-reachable component. Thus, $C^* \subset V_w$. Therefore, V_w is equal to the union of some fully connected components, which means that the basic graph B is disconnected along the edge $(p(w), w)$ of T. This implies that in the augmentation of B to B' two opposite loaded arcs of the null object connecting the vertices w and $p(w)$ are added. Thus, in graph $\tilde{B}$, the vertices w and $p(w)$ belong to the same fully connected component, contradicting our assumption that $C' \subset V_w$. $\square$

In Theorem 4 we prove that the worst-case performance ratio of the heuristic is 1.5.

Theorem 4 *$cost(H) \leq 1.5OPT$ and this worst-case bound is tight.*

Proof. First note that while implementing the *Augmentation Algorithm*, an edge of T is augmented at most once since in future iterations both end vertices of the edge are reachable in $\tilde{B}$ from vertex 1, where the depot is located. Let θ be the number of iterations run by the Augmentation Algorithm. In order to prove that the worst-case performance ratio of the proposed heuristic is 1.5, we will prove the following statements:

1. In each iteration, say iteration $j = 1, \ldots, \theta$, we identify an edge of T, denoted by $e_j \in E$, that is augmented, and we also identify a set of edges $D_j \subset E$ that are never augmented by the algorithm. In particular, $e_j \notin \bigcup_{i=1}^{\theta} D_i$.
2. $c_{e_j} \leq cost(D_j)$.
3. $e_j \neq e_i$ and $D_j \bigcap D_i = \emptyset$ for $i \neq j$.
4. $\sum_{i=1}^{\theta}\{2c_{e_i} + 2cost(D_i)\} \leq cost(B')$.

We now provide the proofs.

1. We will first show that for a given graph $\tilde{B}$ the set $\tilde{S}$ contains at least two vertices from each (non-singleton) unreachable fully connected component C'_ℓ of $\tilde{B}$, for which $\tilde{S} \cap S_\ell \neq \emptyset$. Suppose by contradiction that for the multi-graph $\tilde{B}$, the set $\tilde{S}$ contains only one vertex v from S_ℓ. As C'_ℓ is not a singleton, the set S_ℓ contains at least two vertices. According to our assumption, none of the vertices in $S_\ell - \{v\}$ is a vertex in $\tilde{S}$, thus either there exists a vertex in $S_\ell - \{v\}$ which is a descendant of a vertex $w \in \tilde{S} - S_\ell$, $w \neq v$, or all vertices of $S_\ell - \{v\}$ are descendants of v. We will show that both cases yield a contradiction.
   First suppose that there exists a vertex in $S_\ell - \{v\}$ which is a descendant of a vertex $w \in \tilde{S} - S_\ell$, $w \neq v$. According to Lemma 4.2, $S_\ell \subset V_w$, where V_w is the set of vertices of the subtree T_w of T. This is in contradiction with the fact that $v \in S_\ell$ and $v \notin V_w$, as $v, w \in \tilde{S}$. Suppose now that all vertices of $S_\ell - \{v\}$ are descendants of v. Thus, the minimal subtree of T containing C'_ℓ is rooted at v. Recall that C'_ℓ is unreachable, and v is a vertex of C'_ℓ contradicting Lemma 4.3.
   Let vertex $v*$ be selected according to Step 4 of the *Augmentation Algorithm* when applied on $\tilde{B}$ in iteration j. Suppose that $v* \in C'_\ell$, where C'_ℓ is an unreachable fully connected component of $\tilde{B}$ (which is not a singleton). Thus, the set $\tilde{S}$ associated with $\tilde{B}$ must contain at least one more vertex from C'_ℓ. Let $\tilde{S}_\ell = \tilde{S} \bigcap S_\ell$. By definition of $v*$, $\kappa_{v*} \leq \kappa_v$ for any $v \in \tilde{S}$. Note that after augmenting $\tilde{B}$ along the edge $[p(v*), v*]$ at a cost of $2\kappa_{v*}$, we have made all vertices in S_ℓ, and in particular all vertices in $\tilde{S}_\ell$,

reachable from vertex 1 where the depot is located. This means that the algorithm in future iterations will never augment the graph $\tilde{B}$ along the edges $[p(v), v]$ for $v \in S_\ell$. Let $e_j = [p(v*), v*]$ and $D_j = \bigcup_{v \in S_\ell - \{v*\}} \{[p(v), v]\} \supseteq \bigcup_{v \in \tilde{S}_\ell - \{v*\}} \{[p(v), v]\}$.

2. It follows from the selection procedure of $v*$ in Step 4 of the *Augmentation Algorithm* that $c_{e_j} \leq cost \left(\bigcup_{v \in \tilde{S}_\ell - \{v*\}} \{[p(v), v]\} \right) \leq cost(D_j)$.
3. This observation follows directly from the fact that once a vertex is made reachable from the root by adding the null object loaded arcs in a certain augmentation iteration, then there is no need to augment along the same edge again. Also, if in some iteration, vertex $v*$ is made reachable by augmenting along the edge $[p(v*), v*]$, then all vertices that belonged to the same unreachable component as $v*$ become reachable from the root, and thus these vertices will never again be members $\tilde{S}$.
4. By construction, the multi-type multi-graph B' is connected and balanced, thus each edge of E is covered in B' at least twice. The Augmentation Algorithm augments B' along the edges $\bigcup_{j=1}^{\theta} \{e_j\}$, and it does not augment along the edges in $\bigcup_{i=1}^{\theta} D_i$. The statement follows directly from the facts that these 2θ sets ($\{e_j\}$, and D_j) for $1 \leq j \leq L$ are disjoint, and $\bigcup_{j=1}^{\theta} (\{e_j\} \bigcup D_j) \subset E$.

Recall that $cost(B')$ is a lower bound on the optimal SP cost. By the above statements it follows that the total augmentation cost, namely $\sum_{j=1}^{\theta} 2c_{e_j} \leq 0.5cost(B')$, which proves that the worst-case performance ratio of the algorithm is 1.5, i.e., $cost(H) \leq 1.5OPT$.

In order to complete the proof we present an example (Figure 5) where the bound 1.5 is tight. Part (a) of the figure demonstrates the SP, where the depot is located at the root of the tree, namely at vertex 1. The numbers on the edges denote their cost. This graph is disconnected and is therefore augmented into B' which contains, in addition to the services paths of B, also the two loaded arcs $(1,3)^0$ and $(3,1)^0$, see part (c) of the figure. No feasible SP solution exists on B' as none of the objects 1 or 2 is available at vertex 3 when the vehicle enters there the first time. One can readily check that $cost(B') = 4M + 4\xi$.

The fully connected components of B' are C_1 containing the root and vertices 2 and 3, and the unreachable component C_2 containing both vertices 4 and 5. The first iteration of the *Augmentation Algorithm* generates the set $\tilde{S} = \{4, 5\}$, where both vertices 4 and 5 belong to the same unreachable component. In this case $\kappa_4 = \kappa_5 = M$, and therefore the algorithm chooses arbitrarily one of these vertices, say vertex 4 as $v*$. Graph B' is augmented by adding to it the two loaded arcs $(3,4)^0$ and $(4,3)^0$. The resulting multi-type multi-graph is the heuristic solution H that the algorithm generates. According to this solution, the vehicle loads object 1 at vertex 1 and carries it to vertex 2, where it unloads it and loads object 2, which is carried to vertex 1. From there the vehicle travels empty to vertex 4 through vertex 3 and swaps objects 2 and 1 between vertices 4 and 5. Finally the vehicle travels empty from vertex 4 to the root. Clearly, $cost(H) = 6M + 4\xi$.

The optimal SP solution to the above example is to start at the depot by loading object 1, and carry it directly to vertex 4, from there the vehicle loads object 2 which is

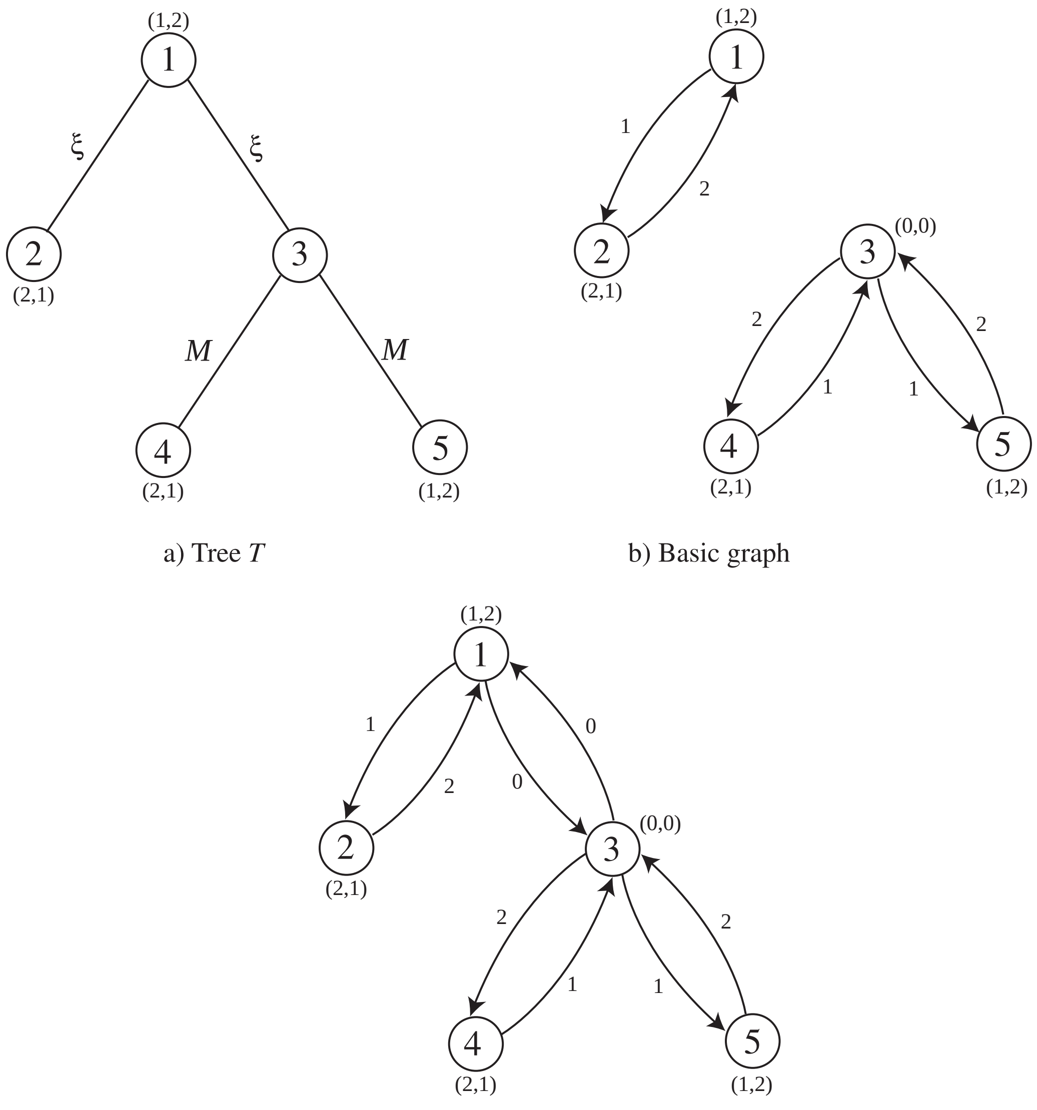

Figure 5: Example showing that the worst-case bound of the heuristic is tight

carried to vertex 5. At vertex 5 object 1 is loaded and carried to vertex 2. There it is unloaded and object 2 is loaded to be carried to the root. Thus, $OPT = 4M + 4\xi$, implying that $\lim_{M\to\infty} cost(H)/OPT = 1.5$. □

5 The case $m = 1$ is polynomial

In this section we consider the special case where the set of items consists of a single type object and the null object, i.e., $O = \{0, 1\}$. We will show that there exists an optimal routing of the vehicle that only uses the loaded arcs of the graph B'. The vertices of the tree are of four types: a vertex is either (i) a supply vertex associated with the pair $(a, b) = (1, 0)$; or (ii) a demand vertex associated with the pair $(a, b) = (0, 1)$; or (iii) a transshipment vertex associated with the pair $(a, b) = (0, 0)$; or (iv) a vertex associated with the pair $(a, b) = (1, 1)$. Clearly, in the case of one product type there is no need to preempt. Thus, there exists an optimal solution in which a vertex v which is associated with $(a_v, b_v) = (1, 1)$ will always serve itself by using its own supply.

Recall from Section 2 that *all* the loaded arcs in the basic graph B, which are directed from vertex u to vertex v, are associated with a an object $i \in \{0, 1\}$, where all the loaded arcs directed from vertex v to vertex u are associated with the other object $1 - i$. Since the basic graph B is balanced, the two sets of loaded arcs have the same cardinality. If B is not connected, the graph is augmented by adding two opposite direction loaded arcs of the null object between any two adjacent vertices of T which are not covered by any loaded arc of B. This results in graph B'. We show in Theorem 5 that graph B' consists of a single reachable fully connected component implying that $OPT = Cost(B')$.

Theorem 5 *For the case $m = 1$, graph B' consists of a single fully connected component.*

Proof. We show that none of the fully connected components of graph B' is unreachable. Suppose by contradiction that there were at least one unreachable fully connected component C. Let the subtree T_w be the minimal subtree that contains C. According to Lemma 4.3, $w \notin C$. By definition, there must exist a service path of object 1 starting at a vertex $v_1 \in C$, passing through vertex w, and ending at a vertex $v_2 \in C$. Since B is balanced, there must also exist a service path of the null object starting at v_2, passing through w and ending at v_1. This makes v_1 reachable from w, contradicting our assumption that C is unreachable. □

References

Anily, S. and R. Hassin. 1992. The Swapping Problem. *Networks* 22:419–433.

Anily, S., M. Gendreau and G. Laporte. 1999. The Swapping Problem on a Line. *SIAM Journal on Computing* 29:327–335.

Arkin, E., R. Hassin and L. Klein. 1997. Restricted Delivery Problems on a Network. *Networks* 29:205–216.

Atallah M.J. and S.R. Kosaraju. 1988. Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel. *SIAM Journal on Computing* 17:849–869.

Ball, M.O. and M.J. Magazine. 1988. Sequencing of Insertions in Printed Circuit Board Assembly. *Operations Research* 36:192–201.

Chalasani, P. and R. Motwani. 1999. Approximating Capacitated Routing and Delivery Problems. *SIAM Journal on Computing* 28:2133–2149.

Frederickson, G.N. and D.J. Guan. 1992. Preemptive Ensemble Motion Planning on a Tree. *SIAM Journal on Computing* 21:1130–1152.

Frederickson, G.N. and D.J. Guan. 1993. Nonpreemptive Ensemble Motion Planning on a Tree. *Journal of Algorithms* 15:29–60.

Frederickson, G.N., M.S. Hecht and C.E. Kim. 1978. Approximation Algorithms for Some Routing Problems. *SIAM Journal on Computing* 7:178–193.

Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

Hierholzer, C. 1873. Über die Möglichkeit einen Linienzug ohne Wiederholung und ohne Unterbrechning zu unifaren. *Mathematische Annalen* VI:30–32.