Semi-Preemptive Routing on Trees

Sven O. Krumke\textsuperscript{a}  Dirk Räbiger\textsuperscript{b}  Rainer Schrader\textsuperscript{b}

\textsuperscript{a}Department of Mathematics, University of Kaiserslautern
Paul-Ehrlich-Str. 14, 67653 Kaiserslautern
Germany

\textsuperscript{b}Zentrum für Angewandte Informatik, Universität zu Köln
Weyertal 80, 50931 Köln
Germany

Abstract

We study a variant of the pickup-and-delivery problem (PDP) in which the objects that have to be transported can be reloaded at most \(d\) times, for a given \(d \in \mathbb{N}\). This problem is known to be polynomially solvable on paths or cycles and NP-complete on trees. We present a \((4/3 + \varepsilon)\)-approximation algorithm if the underlying graph is a tree. By using a result of Charikar et al. (1998), this can be extended to a \(O(\log n \log \log n)\)-approximation for general graphs.

Key words: pickup and delivery, dial-a-ride, transportation, approximation, colored arborescences, stacker crane

1 Introduction

Let \(G = (V, E)\) be an undirected graph on \(n\) nodes with nonnegative edge lengths \(l: E \to \mathbb{R}_+\). For the \textit{semi-preemptive pickup-and-delivery problem} (SPDP) we are given a set of \(m\) objects. Each object corresponds to a transportation request, i.e., a pair \((v_i, v_j)\) of nodes from \(V\) so that the object has to be moved from its initial location \(v_i\) to its destination node \(v_j\). Transportation is done by a vehicle which can handle only one object at a time. The vehicle starts at a predefined start node \(v_0 \in V\) and moves along the edges of the graph \(G\), serves the set of all requests \(R\) and returns to \(v_0\). On its way, the

\textit{Email addresses: krumke@mathematik.uni-kl.de} (Sven O. Krumke),\n\textit{raebiger@zpr.uni-koeln.de} (Dirk Räbiger), \textit{schrader@zpr.uni-koeln.de} (Rainer Schrader).

Preprint submitted to Discrete Applied Mathematics 24 October 2005
vehicle may use up to \(d \leq |V|\) intermediate nodes as reload nodes where it may drop the currently carried object and resume its transportation later. The objective is to minimize the total length of the tour. Hence, we are looking for a (constrained) minimum-length closed walk in \(G\) which contains a \((v_i, v_j)\)-path for every request \((v_i, v_j) \in \mathcal{R}\).

It is easy to see that the famous travelling salesman problem is a special case of the Spdp, and thus Spdp is NP-complete in general. If the underlying graph is a path or a cycle, the problem was shown to be polynomial time solvable for \(d = n\) (the so-called preemptive case) and for \(d = 0\) (the non-preemptive version) by Atallah and Kosaraju (1988) and for arbitrary \(d\) by Räbiger (2004).

In this paper we study the Spdp in the more general case when the underlying graph is a tree. While the preemptive version in this case is known to remain polynomially solvable (cf. Frederickson and Guan (1992)), the non-preemptive variant was shown to be NP-hard in Frederickson and Guan (1993) for trees in general, and even on caterpillars in Hauptmeier et al. (2001). Frederickson and Guan (1993) also give an approximation algorithm for the non-preemptive Spdp on trees with a performance ratio of 1.21363. Recall that a polynomial-time algorithm \(A\) is said to be a \(\rho\)-approximation algorithm for a minimization problem \(P\), if for every problem instance \(I\) of \(P\) with optimal solution value \(\text{OPT}(I)\) the solution of value \(A(I)\) returned by the algorithm satisfies \(A(I) \leq \rho \cdot \text{OPT}(I)\).

Since the non-preemptive version is NP-hard on trees, Spdp is NP-hard on trees as well. We present an approximation algorithm with approximation ratio of \(4/3 + \varepsilon\) for any given \(\varepsilon > 0\). Using results about the approximation of arbitrary metric spaces by tree metrics Charikar et al. (1998) this result implies an \(O((4/3 + \varepsilon) \log n \log \log n)\)-approximation for arbitrary graphs.

This paper is organized as follows. In Section 2 we study the problem of finding a minimum-weight arborescence in a graph with colored arcs subject to the constraint that the arborescence may use only a limited number of blue arcs. This problem turns out to be related intimately to the Spdp as we show in Section 3, which contains our approximation algorithm on trees. In Section 4 we briefly sketch how our approximation algorithm can be extended to general graphs with a polylogarithmic increase in performance guarantee. Section 5 contains a short conclusion.

### 2 Colored arborescences

Let \(G = (V, E)\) be a directed (multi-) graph and \(r \in V\) be a fixed node. An \(r\)-arborescence of \(G\) is a spanning arborescence with root \(r\), that is, a subgraph
(V, A) whose arcs form a spanning tree of G and every node different from r is the head of exactly one arc in A. We now consider the situation where the arc set of \( G = (V, E) \) is partitioned into a subset \( E^r \) of red arcs and a subset \( E^b \) of blue arcs, that is, \( E = E^r \cup E^b \) and \( E^r \cap E^b = \emptyset \). For a given number \( d \in \mathbb{N} \) any \( r \)-arborescence of \( G \) which contains at most \( d \) blue arcs will be referred to as \((d, r)\)-arborescence.

Given a non-negative weight function \( w: E^r \cup E^b \rightarrow \mathbb{N} \) and \( d \in \mathbb{N} \), we consider the problem of finding a minimum-weight \((d, r)\)-arborescence.

To the best of our knowledge, the complexity status of the \((d, r)\)-arborescence problem remains unsolved: it is neither known to be polynomial time solvable nor known to be NP-hard. In view of the approximation of the SpDP, it turns out that it suffices to have an efficient approximation algorithm for the \((d, r)\)-arborescence problem.

Recall that a family \( \{A_\varepsilon\}_\varepsilon \) of approximation algorithms for a minimization problem \( \Pi \), is called a fully polynomial approximation scheme or FPTAS, if algorithm \( A_\varepsilon \) is a \((1 + \varepsilon)\)-approximation algorithm for \( \Pi \) and its running time is polynomial in the size of the input and \( 1/\varepsilon \).

**Theorem 1** There exists an FPTAS for the \((d, r)\)-arborescence problem.

**Proof.** The statement follows from a more general result by Papadimitriou and Yannakakis (2000) on the existence of approximation schemes. For this, let \( \Pi \) be a \( \{0, 1\}\)-minimization problem, that is, a minimization problem where for each instance \( x \) of \( \Pi \) the set of feasible solutions \( F(x) \) is a subset of \( \{0, 1\}^n \).

Given two linear functions \( w_1, w_2 \) and an accuracy requirement \( \varepsilon > 0 \), let \( P_\varepsilon(x) \subseteq F(x) \) be a subset of the solutions \( F(x) \) with the following property: for every \( s' \in F(x) \) there exists a a solution \( s \in P_\varepsilon(x) \) such that \( w_i(s) \leq (1 + \varepsilon)w_i(s') \) for \( i = 1, 2 \) \( (P_\varepsilon(x) \) is an approximate pareto-set for the bicriteria problem of minimizing simultaneously \( w_1 \) and \( w_2 \) over \( F(x) \)).

Papadimitriou and Yannakakis (2000) prove that \( P_\varepsilon(x) \) can be calculated in time polynomial in the encoding length of \( x \) and \( 1/\varepsilon \), provided that there exists a pseudopolynomial algorithm to solve the following decision problem: given a linear weight function \( w \), and a number \( C \in \mathbb{N} \), does there exist a feasible solution of \( \Pi \) with cost exactly \( C \)? In particular, this result implies that the cardinality of the set \( P_\varepsilon(x) \) can be assumed to be polynomially bounded in \( |x| \) and \( 1/\varepsilon \). For the \( r \)-arborescence problem, such a pseudopolynomial algorithm for the related decision problem is described by Barahona and Pulleyblank (1987).

Consider an instance \( x \) of the \((d, r)\)-arborescence problem with corresponding weight function \( w: E^r \cup E^b \rightarrow \mathbb{N} \). Let \( w_1 := w \) and \( w_2 \) be the incidence vector
of the blue arcs. Given \( \varepsilon > 0 \), we use the approach of Papadimitriou and Yannakakis (2000) to construct the set \( P_{\varepsilon'}(x) \) for \( \varepsilon' := \min\{\varepsilon, \frac{1}{n}\} \) containing a small number of “fairly good” solutions in polynomial time.

Let \( T^* \) be a minimum weight \((d, r)\)-arborescence for \( x \) and \( s^* \) the corresponding incidence vector of arcs. By definition of \( P_{\varepsilon}(x) \), there exists a solution \( s \in P_{\varepsilon}(x) \) with \( w_i(s) \leq w_i(s^*)(1 + \varepsilon) \) for \( i = 1, 2 \). Since \( w_2(s) \leq (1 + \varepsilon')w_2(s^*) = (1 + \varepsilon')d < d + 1 \) we can conclude that \( s \) uses at most \( d \) blue arcs. Also \( w_1(s) \leq (1 + \varepsilon)w_1(s^*) \) and, thus, \( s \) is a \((d, r)\)-arborescence which \( \varepsilon \)-approximates the optimal solution. Since the size of \( P_{\varepsilon}(x) \) is polynomially bounded by \( |x| \) and \( 1/\varepsilon \), we can enumerate the elements of \( P_{\varepsilon}(x) \) to find such an \( s \).

\[ \square \]

3 Approximation on trees

In this section we present an approximation algorithm for SPDP on trees with a performance of \( 4/3 + \varepsilon \) for any given \( \varepsilon > 0 \). We will actually consider a slightly more general problem by allowing an additional cost of \( \Delta \) for each reload operation.

The input consists of a tree \( G = (V, E) \), a distance function \( l : E \to \mathbb{N} \) on the edge set, a set of requests \( R \), a start node \( v_0 \in V \), a limit \( d \leq |V| \) for the number of reloads, and a cost \( \Delta \in \mathbb{N} \) for every time we reload. The goal is to find a minimum-cost tour to transport the objects by a vehicle which travels along the edges of the tree \( G \). The vehicle can carry at most one object at a time, and it starts and ends at the designated start node \( v_0 \). After picking up an object which has to be moved from \( v_i \) to \( v_j \) this object may be dropped at an intermediate node \( v \) and picked up later again. Each such “reload” operation involves a cost of \( \Delta \) and we refer to \( v \) as a reload node.

Note that we can assume without loss of generality that each vertex of degree one or two in \( G \) is the source or destination of at least one request, since we can remove useless leaves and replace an unused node of degree two together with its two incident edges by a new edge.

We create a directed graph \( B = (V, R) \) which contains an arc for every request. The length \( l(u, v) \) of an arc is the length of the unique path from \( u \) to \( v \) in \( G \). Let \( l(R) \) be the sum of all arc lengths in \( B \).

As was shown by Frederickson and Guan (1993), we may assume without loss of generality that in \( B \) the indegree of every node is equal to its outdegree (this is called the transition to a balanced instance in Frederickson and Guan (1993)). This follows from the fact that \( G \) is a tree and any transportation starting and ending at \( v_0 \) must traverse each edge \( e = [u, v] \) of \( G \) the same
number of times from \( u \) to \( v \) as from \( v \) to \( u \). Let \( X \) be the connected component of \( G - e \) containing \( u \) and \( Y \) be the component of \( G - e \) containing \( v \) and denote by \( \phi(X, Y) := |\{(x, y) \in R : x \in X \land y \in Y \}| \) the number of requests with source in \( X \) and destination in \( Y \). Any transportation \( W \) which serves all requests in \( R \) must traverse edge \([u, v]\) from \( u \) to \( v \) at least \( b(u, v) \) times, where

\[
b(u, v) := \begin{cases} 
1 & \text{if } \phi(X, Y) = \phi(Y, X) = 0 \\
\phi(Y, X) - \phi(X, Y) & \text{if } \phi(Y, X) > \phi(X, Y) \\
0 & \text{otherwise.}
\end{cases}
\]

Thus, adding \( b(u, v) \) “balancing requests” from \( u \) to \( v \) will not affect the optimum solution. We refer to Frederickson and Guan (1993) for further details.

Thus, in the sequel we will assume without loss of generality that the given instance of SPDP is already balanced. As a consequence, every strongly connected component of \( B \) is a Eulerian subgraph and each weakly connected component is also strongly connected. The component containing the start node \( v_0 \) and a component containing at least two nodes (and thus request arcs) are called non-trivial. The others which consist of isolated nodes are termed trivial. We call two components neighbors if they contain nodes which are adjacent in \( G \).

It turns out that there is a close connection between SPDP and a rooted Steiner arborescence problem which helps in designing our approximation algorithm. Let \( D = (W, A) \) be a directed graph, \( r \in W \) a root node, and \( S \subseteq W \) a set of terminals. Then \( T \subseteq A \) is an \( r \)-Steiner arborescence if there exists a directed \((r, v)\)-path for every \( v \in S \) in \( T \). The undirected version of this problem is the classical Steiner tree problem. Since the undirected variant can be modeled as a directed problem, the Steiner arborescence problem is NP-complete. Now, let the arcs of \( D \) be colored either red or blue, i.e. \( D = (W, A^r \cup A^b) \). We then require in addition that no more than \( d \in \mathbb{N} \) blue arcs should be used and call this problem \((d, r)\)-Steiner arborescence.

We can relate the \((d, r)\)-Steiner arborescence problem to SPDP as follows. We construct a second directed auxiliary graph \( D = (C, E^r \cup E^b) \) whose nodes correspond to the connected components of \( B \) and whose arcs are colored either red or blue, together with a weight function \( w : E^r \cup E^b \rightarrow \mathbb{N} \) on the arc set. We allow parallel arcs if they are colored differently. Let \( SC_0 \) be the node of \( D \) corresponding to the component of \( B \) which contains the start node \( v_0 \).

- If the connected components \( SC_i \neq SC_j \subseteq V \) of \( B \) are neighbors, we add red arcs \((SC_i, SC_j), (SC_j, SC_i)\) to \( E^r \). Let \( w(SC_i, SC_j) = w(SC_j, SC_i) \) be twice the minimum distance of nodes \( u \in SC_i \) and \( v \in SC_j \) in \( G \) (see Figure 1(a)).
- Let \((u, v) \in R\) be a request with \( u, v \in SC_i \). If \( z \in SC_j \) \((i \neq j)\) lies on the
unique \((u, v)\)-path in \(G\), we add a blue arc \((SC_i, SC_j)\) to \(E^b\) with weight \(w(SC_i, SC_j) = \Delta\) (see Figure 1(b)) .

Fig. 1. Construction of the instance of the \((d, r)\)-Steiner arborescence problem.

According to their corresponding components in \(B\) we distinguish between trivial and non-trivial nodes of \(H\). There are two simple observations:

**Fact 2**  
(i) \((u, v) \in E^r\) if and only if \((v, u) \in E^r\) and \(w(u, v) = w(v, u)\)  
(ii) \((u, v) \in E^b\) implies that \(u\) is non-trivial.

The following theorem gives the precise relation between the SPDP and the \((d, r)\)-Steiner arborescence problem.

**Theorem 3** Given an instance \(I\) of the SPDP and a \((d, SC_0)\)-Steiner arborescence \(T\) in \(D\) with terminal set consisting of all non-trivial nodes of \(D\) we can construct in polynomial time a solution of \(I\) with cost at most \(l(R) + W\). Conversely, each solution of \(I\) with cost \(l(R) + W\) implies a \((d, SC_0)\)-Steiner arborescence with weight \(w(T) \leq W\).

**PROOF.** The basic idea behind the proof is that each red arc of an \((d, SC_0)\)-Steiner arborescence \(T\) corresponds to a direct carrying move of an object, that is, a move where the object is transported from its source to its destination without intermediate dropping. On the other hand, each blue arc of \(T\) relates to a reload operation.

We first show how to convert a Steiner arborescence \(T\) into a solution for the SPDP. To this end, we iteratively modify the graph \(B = (V, R)\) by using the arcs of \(T\).

Consider a red arc \((SC_i, SC_j)\) of \(T\). By construction of \(H\), there exist two nodes \(u \in SC_i\) and \(v \in SC_j\) whose distance in \(G\) is \(\frac{1}{2}w(SC_i, SC_j)\). We add two anti-parallel arcs \((u, v), (v, u)\) to \(B\) with cost \(l(u, v) + l(v, u) = w(SC_i, SC_j)\) (see Figure 2(a)).
For every blue arc \((SC_i, SC_j)\) of \(T\) there exists a request \((u, v)\) with both its source and destination node in \(SC_i\) crossing a node \(z\) of the component \(SC_j\). This means that by transporting the object from \(u\) to \(v\) and using only edges of the underlying tree \(G\), we will inevitably pass \(z\). We replace \((u, v)\) in \(B\) by two arcs \((u, z)\) and \((z, v)\) with costs \(w(u, z) = \Delta\) and \(w(z, v) = w(u, v)\) (see Figure 2(a)). The node \(z\) will be designated to be a reload node. When we repeat this process, we may assume that every blue arc corresponds to a path from \(u\) to \(v\). Every arc of this path represents a path in the tree. One of the arcs \((x, y)\) traverses \(z\). We split \((x, y)\) into two arcs \((x, z)\) and \((z, y)\) and assign the costs of \((u, v)\) to the second arc and \(\Delta\) to the first.

Let \(B'\) be the graph obtained from \(B\) by repeatedly applying these two operations to every arc of \(T\) and finally replacing each blue arc \((u, v)\) by a directed path corresponding to the unique shortest path from \(u\) to \(v\) in \(G\). Since each replacement of an arc of \(T\) merges two connected components of \(B\) into one, the nodes of all non-trivial components of \(B\) will be contained in a single component of \(B'\). Both operations do not change the degree balance of any node, thus the resulting super-component will be a Eulerian subgraph which contains the start node \(v_0\). Moreover, since \(T\) has at most \(d\) blue arcs, we use no more than \(d\) reload nodes.

It is straightforward to see that a Eulerian tour in \(B'\) starting and ending at \(v_0\)
gives a valid transportation. As indicated initially, each blue arc gives a reload operation and the total cost of the Euler tour equals the cost of all arcs in $B$ which in turn is exactly $l(\mathcal{R}) + w(T)$.

Now consider conversely a feasible transportation $Q$ for $I$ with cost $l(\mathcal{R}) + W$ and consider again the graph $B = (V, \mathcal{R})$ whose arc set is formed by the set of requests. Initialize $A_\mathcal{R} := \emptyset$, $A_e := \emptyset$ and $A_r := \emptyset$. We follow the movement of the vehicle from its initial position $v_0$ along $Q$ back again to $v_0$.

For each empty move (that is, without carrying an object) of the vehicle along an edge $[u, v]$ of $G$ from $u$ to $v$ we add a directed arc $(u, v)$ to $A_e$. Observe that, since the tour is closed and we assumed the instance to be balanced, it follows that the edge $[u, v]$ is traversed empty by the vehicle in both directions the same number of times. Thus, $A_e$ will consist of pairs of antiparallel arcs.

If a request from $u$ to $v$ is transported directly without intermediate reload operation from $u$ to $v$, we add a directed arc from $u$ to $v$ to $A_\mathcal{R}$. Finally, if an object is transported from its source $u$ to its destination $v$ with reload occurring at $z_1, \ldots, z_p$, we add the arcs $(u, z_1), (z_1, z_2), \ldots, (z_p, v)$ to $A_r$.

The transportation $Q$ implies a Eulerian tour in the graph $(V, A_\mathcal{R} \cup A_e \cup A_r)$. We contract each connected component of $B$ into a single node. Clearly, each nontrivial component of $H$ is reachable from $SC_0$, thus there exists an arborescence rooted at $SC_0$ which by construction is a $(d, SC_0)$-Steiner arborescence of $H$ with weight $W$ as desired. □

Theorem 3 implies that the cost of an optimal transportation is $l(\mathcal{R}) + w(T^*)$, where $T^*$ is an optimal $(d, SC_0)$-Steiner arborescence. Any approximation of the Steiner arborescence problem can be used to approximate $SPDP$. Unfortunately, it is in general not possible to approximate Steiner arborescences within a constant factor (cf. Feige (1998)). On the other hand, the auxiliary graph $H$ has a very special structure that we can exploit. For this, we further simplify the auxiliary graph. Let $C' \subseteq C$ be the subset of nodes which are trivial. Now construct a directed and colored graph $H' = (C \setminus C', E')$ which contains only the non–trivial nodes $V \setminus V'$ of $H$. For every ordered pair of nodes $(SC_i, SC_j) \in C \setminus C'$ find the shortest $(SC_i, SC_j)$-path in $H$ using at most one blue arc. This can be done by a simple modification of a shortest-path algorithm. If the $(SC_i, SC_j)$-path uses only red arcs, color the arc $(SC_i, SC_j)$ in $H'$ red and blue otherwise. Define a weight function $w : E' \to \mathbb{N}$ on all arcs of $E'$ by assigning the distance of a shortest path.

**Theorem 4** Let $T$ be a minimum weight $(d, SC_0)$-arborescence of $H'$, and $T^*$ a be minimum weight $(d, SC_0)$-Steiner arborescence of $H$. Then we have that $w(T) \leq 2w(T^*)$.  

8
PROOF. Given a \((d, SC_0)\)-Steiner arborescence \(T^*\) of \(H\), we will prove the existence of a \((d, SC_0)\)-arborescence in \(H'\) with cost at most \(2w(T^*)\).

Enumerate the nodes of \(T^*\) by depth-first-search so that every node appears as often as its degree in \(T^*\). Let \(P = (SC_0, \ldots, SC_p)\) be the sequence of nodes in the order of their appearance. Any consecutive pair of nodes in \(P\) describes an arc in \(T^*\). Furthermore, every arc \((SC_i, SC_j)\) of \(T^*\) appears twice, once in its orientation \((\ldots, SC_i, SC_j, \ldots)\) and then against its orientation \((\ldots, SC_j, SC_i, \ldots)\). Thus, the costs of all arcs described by \(P\) are \(2w(T^*)\).

Let \(Q = (SC_0, \ldots, SC_q)\) be the subsequence of \(P\) induced by the non-trivial nodes.

We now construct a \((d, SC_0)\)-arborescence \(T\) for \(H'\). We do so by marking the nodes of \(Q\) and adding arcs to \(T\). A node \(SC_i\) will be marked if \(T\) contains a \((SC_0, SC_i)\)-path. The process terminates when all nodes of \(Q\) are marked. Initially mark \(SC_0\). Let \(SC_j\) be the first unmarked node and \(SC_i\) be its predecessor in \(Q\). By definition of \(Q\), \(SC_i\) and \(SC_j\) are non-trivial. Let \(P^{i,j} = (c_i, u_1, \ldots, u_p, c_j)\) be the (unique) subsequence of \(P\), corresponding to the pair \(SC_i, SC_j\). Then all \(u_i\)'s are trivial. \(P^{i,j}\) describes a sequence of arcs, possibly against their orientation. By \(w(P^{i,j})\) we denote the sum of weights of these arcs.

We will show that

(i) \(P^{i,j}\) describes a directed \((SC_i, SC_j)\)-path in \(T^*\) and thus in \(H\) and

(ii) \(P^{i,j}\) contains at most one blue arc.

(i) \(SC_i\) is marked, thus in \(T\) there exists a directed \((SC_0, SC_i)\)-path. By induction, arcs in \(T\) correspond to directed paths in \(T^*\). So in \(T^*\) there exists a directed \((SC_0, SC_i)\)-path, which additionally visits some trivial nodes. Either \(SC_i\) is the root or in \(T\) there exists an arc with head \(SC_i\) which is not induced by \(P^{i,j}\), since \(SC_i\) is already marked. In both cases \((SC_i, u_1)\) is used in its orientation, or else \(T^*\) contains two arcs with the same head node (but \(T^*\) is an arborescence). In the DFS, starting at \(SC_i\), all following arcs have to be used in their orientation. The orientation can only be reversed if we visit a leaf node. Only non-trivial nodes can be leaves, and the first non-trivial node after \(SC_i\) in \(P^{i,j}\) is \(c_j\). Thus \(P^{i,j}\) describes a directed \((SC_i, SC_j)\)-path.

(ii) In particular \((u_p, SC_j)\) is visited in its orientation. By recalling fact 2 we know that the tail-nodes of all arcs used in \(P^{i,j}\) are non-trivial. Hence, only two arcs can be colored blue: \((SC_i, u_1)\) and \((SC_j, u_p)\). This proves (ii).

Thus in \(H'\) there exists an arc \((SC_i, SC_j)\) with cost \(w(SC_i, SC_j) \leq w(P^{i,j})\) which we add to \(T\). As there was a directed \((SC_0, SC_i)\)-path in \(T\), there now is a directed \((SC_0, SC_j)\)-path and we mark \(SC_j\).
We never consider the same subsequence $P^{i,j}$ twice. By induction, all nodes of $Q$ are marked and finally $T$ is a $(d, SC_0)$-arborescence with cost

$$w(T) = \sum_{(c_i, c_j) \in T} w(c_i, c_j) \leq \sum_{(c_i, c_j) \in T} w(P^{i,j}) \leq w(P) \leq 2w(T^*)$$

This completes the proof. \qed

We do not know a polynomial time algorithm to solve $(d, r)$-arborescence problem, but recall that we provided a FPTAS in Theorem 1. We will use this result in the following algorithm which assumes that $I$ is an input for SPDP.

**Algorithm 1**

1: **Construct** auxiliary graphs $H$ and $H'$ from $I$.
2: **Calculate** approximately a $(d, SC_0)$-arborescence $T'$ for $H'$. \label{cor:1}
   Corollary 1
3: **Construct** a $(d, SC_0)$-Steiner arborescence $T$ for $H$ out of $T'$. \label{thm:4}
   Theorem 4
4: **Construct** a feasible transportation from $T'$. \label{thm:3} Theorem 3.

The proof of the following result is along the lines of Frederickson and Guan (1993) but generalizes their result.

**Theorem 5** Let $OPT$ denote the length of an optimal transport graph for an instance of $SPDP$ on a tree. Algorithm 1 constructs a transportation with cost at most $\frac{4}{3} + \epsilon$ for any $\epsilon > 0$.

**PROOF.** The degree balanced graph $B$ contains at least two anti-parallel arcs for every edge of the underlying tree $G$. Thus $l(R) \geq l(E)$. Let $T^*$ be a minimal cost $(d, SC_0)$-arborescence for $H'$. A red arc in $T^*$ corresponds to a path between strongly connected components of $D$ in $G$. To connect all components by anti-parallel arcs we do not need more than twice the length of $G$. Using blue arcs in $T^*$ will not increase its cost. Thus, we have $2l(R) \geq 2l(E) \geq w(T^*)$. Let us estimate the ratio between our solution and an optimal transportation. For this, let $S$ be an optimal $(d, SC_0)$-Steiner arborescence for $H$. Corollary 1 expects an accuracy constant $\epsilon'$ which we set to $\epsilon' := \frac{3}{2}\epsilon$. 


This completes the proof. □

4 Approximation on General Graphs

In this section we show how our approximation algorithm for trees from Section 3 can be used to obtain an approximation algorithm for general graphs. Note that for \textsc{Spdp} on general graphs we can assume without loss of generality that each instance satisfies the metric property. In fact, suppose an edge \((u, v)\) of the underlying graph \(G = (V, E)\) violates the triangle inequality. We can simply replace the length of this edge by the length of a shortest \((u, v)\)-path in \(G\) without consequences to feasibility.

Definition 6 A set of metric spaces \(S\) over \(V\) \(\alpha\)-probabilistically approximates a metric space \(M\) over \(V\), if the following conditions are satisfied:

(i) for all \(u, v \in V\) and \(N \in S\), \(d_N(u, v) \geq d_M(u, v)\), and
(ii) there exists a probability distribution \(\mathcal{D}\) over \(S\) such that for every \(u, v \in V\) we have \(E_{\mathcal{D}}[d_N(u, v)] \leq \alpha \cdot d_M(u, v)\).

Bartal (1998) proved that any metric space over a finite set \(V\) can be \(\alpha\)-probabilistically approximated by tree metrics, where \(\alpha = O(\log n \log \log n)\). Charikar et al. (1998) later improved this result, showing that the mentioned approximation can be achieved by a probability distribution on \(O(n \log n)\) trees, where the trees and the distribution can be computed in polynomial time. In particular, this implies that a deterministic approximation of an arbitrary metric by tree metrics is possible (by using exhaustive search for the best of the \(O(n \log n)\) trees).

Suppose that we are given an instance of \textsc{Spdp} on a general graph \(G\) such that the weight function is metric. We (deterministically) \(\alpha\)-approximate the metric space induced by \(G\) by a set \(S\) of \(O(n \log n)\) trees and run the algorithm
from Section 3 for each of the trees. Picking the best of the $O(n \log n)$-solution then results in a $O((4/3 + \varepsilon) \log n \log \log n)$-approximation. Thus, we have the following result:

**Theorem 7** For any $\varepsilon > 0$, there is a $O((4/3 + \varepsilon) \log n \log \log n)$-approximation algorithm for Spdp.

5 Remarks

In a more restrictive variant of Spdp we are given a set $X \subseteq V$ of nodes which we may use as reload nodes. The problem is then to decide which nodes of $X$ to use in an optimal transportation plan. This version is still a generalization of the preemptive ($X = \emptyset$) and non-preemptive PDP ($X = V$) and hence NP-complete. It is easy to see that our algorithm can be adopted to give a $(4/3 + \varepsilon)$-approximation for this version on trees.

References


