Minimum spanning tree with hop restrictions

Refael Hassin and Asaf Levin [1]

Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel.
{hassin,levinas}@post.tau.ac.il

April 10, 2003

[1]Corresponding author: phone +972-3-6407437, fax +972-3-6409357

Abstract

Let $U = (u_{ij})_{i,j=1}^{n}$ be a symmetric requirement matrix. Let $d = (d_{ij})_{i,j=1}^{n}$ be a cost metric. A spanning tree $T = (V, E_T)$ $V = \{1, 2, \ldots, n\}$ is feasible if for every pair of vertices v, w the $v - w$ path in T contains at most u_{vw} edges. We explore the problem of finding a minimum cost feasible spanning tree, when $u_{ij} \in \{1, 2, \infty\}$. We present a polynomial algorithm for the problem when the graph induced by the edges with $u_{ij} < \infty$ is 2-vertex-connected. We also present a polynomial algorithm with bounded performance guarantee for the general case.

Keywords: Minimum spanning tree, Hop-restriction, Approximation algorithm.

1 Introduction

Let $U = (u_{ij})_{i,j=1}^{n}$ be a symmetric *requirement matrix*. Let $d = (d_{ij})_{i,j=1}^{n}$ be a cost metric, i.e., d is symmetric non-negative and satisfies the triangle inequality. A spanning tree $T = (V, E_T)$ $V = \{1, 2, \ldots, n\}$ is *feasible* if for every pair of vertices v, w the $v - w$ path in T contains at most u_{vw} edges. The cost of a spanning tree T is $\Sigma_{(v,w)\in E_T} d_{vw}$.

The METRIC MINIMUM SPANNING TREE WITH HOP RESTRICTIONS PROBLEM (RHMST) is to find a minimum cost feasible spanning tree.

Note that we consider the edge set of a complete graph on n vertices as candidate edges for the spanning tree.

The METRIC MINIMUM k-HOP SPANNING TREE PROBLEM (kHMST) is the special case where there is a vertex *root* such that $u_{root,i} = k \; \forall i$ and otherwise $u_{ij} = \infty$. This problem was addressed by earlier works: Kortsarz and Peleg [10] presented an $O(\log n)$-approximation algorithm for the special case of the kHMST where k is a constant, and an $O(n^{\epsilon})$-approximation algorithm for unbounded k for any fixed $\epsilon > 0$. Gouveia [7] showed empirically that linear programming formulations and multi-commodity flow models give sharp lower bounds to the kHMST optimal solution. Gouveia and Raquejo [9] presented a Lagrangean relaxation to the kHMST. Voss [14] presented a tabu-search heuristic to the kHMST.

The special case of 2HMST was also addressed by earlier works (see [4] and papers cited there). Alfandari and Paschos [1] proved that this problem is SNP-hard, and for the special case that all the distances are 1 or 2, they presented a $\frac{5}{4}$-approximation algorithm. Monnot [13] also considered this case and presented several approximation algorithms including one with differential ratio $\frac{3}{4}$ (the differential ratio measures how the value of an approximate solution is placed in the interval between the worst and the best solution values of an instance).

Gouveia and Janssen [6] considered a generalization of the kHMST problem defined as follows: let $G = (V, E)$, with primary and secondary costs, c_{ij}^1 and c_{ij}^2 respectively, for every edge $(i, j) \in E$ and natural numbers H, w_1 and w_2. A spanning tree is feasible if for every vertex $v \in V \setminus \{root\}$ the unique path between v and *root* contains a weighted number of primary and secondary edges (with weights w_1 and w_2 respectively) of at most H, and the primary edges constitute a connected subset containing *root*. The goal is to find a minimum cost feasible spanning tree where every edge is either primary or secondary. They presented lower bounds derived by a Lagrangean relaxation together with sub-gradient optimization, and they tested the lower bounds empirically.

The METRIC UNCAPACITATED FACILITY LOCATION PROBLEM is defined as follows: given a bipartite graph $G = (F, C, E)$ with an edge cost metric c_{ij} for assigning a vertex $j \in C$ to facility $i \in F$, and a cost f_i for opening a facility at vertex $i \in F$, the goal is to minimize the cost of the selected facilities plus the sum of costs of assigning every vertex to its closest open facility. This problem is NP-hard and a series of constant factor approximation algorithms was published. Currently, the best factor is 1.52 [12]. If there is a polynomial-time algorithm for the METRIC UNCAPACITATED FACILITY LOCATION PROBLEM with approximation ratio smaller than 1.463, then $NP \subseteq DTIME(n^{O(\log \log n)})$ [8].

Note that the 2HMST is a special case of the METRIC UNCAPACITATED FACILITY LOCATION PROBLEM where the opening facility costs with the assignment costs satisfy the triangle inequality ($f_i + c_{ij} \geq f_j$). Therefore, the 1.52 approximation ratio applies to it. We note that the reduction in [8] produces instances of the METRIC UNCAPACITATED FACILITY LOCATION PROBLEM that satisfy these extra conditions. Therefore, the 1.463 lower-bound applies to the 2HMST.

Another related problem is the MINIMUM COST BOUNDED DIAMETER SPANNING TREE PROBLEM (problem [ND4] in [5]) defined as follows: given a graph $G = (V, E)$ and a cost function $c : E \to Z^+$. The goal is to minimize the cost of a spanning tree T of G such that T contains no simple path with more than D edges. This problem is NP-complete even if $D = 4$ and all edge costs are 1 or 2 (see [5]). This problem is a special case of the METRIC MINIMUM SPANNING TREE WITH HOP RESTRICTIONS PROBLEM with $u_{ij} = D \ \forall i, j$.

Given a graph $G = (V, E)$ and a spanning tree $T = (V, E_T)$ of G, denote by $n_G(v, w)$ and $n_T(v, w)$ the number of edges in the shortest $v - w$ path in G and T, respectively. T is an unweighted tree k-spanner if for every $v, w \in V \ n_T(v, w) \leq k n_G(v, w)$. In [2] it is shown that an unweighted tree 2-spanner, if it exists, can be computed in polynomial time. However, in our problem we are interested in forcing the 2-hop constraints only over a subset of the feasible edge set and we are interested in a minimum cost solution.

We are interested in the special case that $u_{ij} \in \{1, 2, \infty\}$ for every $i \neq j$. This case is quite general. In particular it generalizes the 2HMST, and therefore also the METRIC UNCAPACITATED FACILITY LOCATION PROBLEM. We present a polynomial time algorithm for the case that the graph G induced by the edges with $u_{ij} < \infty$ is a Hamiltonian cycle. Next, we extend the algorithm to solve the problem when G is any 2-vertex-connected graph (these results hold also if d does not satisfy the triangle inequality). Finally, we present a constant ratio approximation algorithm for a general graph (when d is a metric).

2 Preliminaries

We will use the terminology "connected" for "vertex-connected", "cycle" for "simple cycle", and "path" for "simple path". We use the notation (u, v) to denote the undirected edge that connects u and v.

A path between u and v is said to be a *$u - v$ path*.

For a pair of vertices u and v, the $u - v$ paths $P_1, P_2, \ldots, P_k$ are said to be *disjoint* if for every $i \neq j \ P_i \cap P_j = \{u, v\}$.

A graph is *2-connected* if for every pair of vertices x, y there are at least two disjoint $x - y$ paths.

A *k-connected component* of G is a maximal subset of vertices S, such that for each pair of vertices $x, y \in S$, there are at least k disjoint $x - y$ paths.

A k-connected component is said to be *non-trivial* if it contains at least two vertices.

A *connected component* of G is a maximal connected subgraph of G.

For a set of vertices $V' \subseteq V$, denote by $G(V') = (V', E(V'))$ the induced subgraph of G over V' where $E(V')$ consists of the edges having both end-vertices in V', and denote by $d(V')$ the metric induced by d over V'.

We use the following observations:

Observation 2.1 *In a 2-connected graph, for every pair of vertices v, w and edge e, there is a $v - w$ path that contains e.*

Observation 2.2 *In a 2-connected graph, for every pair of edges, there is a cycle that contains both edges.*

For the case $u_{ij} \in \{1, 2, \infty\}$ $\forall i, j$, the input to the problem consists of a cost metric d and a graph $G = (V, E)$ where $E = E_1 \cup E_2$, $E_1 = \{(i, j) | u_{ij} = 1\}$, and $E_2 = \{(i, j) | u_{ij} = 2\}$.

Observation 2.3 *Let $T = (V, E_T)$ be a feasible solution to $G = (V, E_1 \cup E_2)$. Let $E_1' = E_T$, and $E_2' = E_2 \setminus E_T$. Then T is a feasible solution to $(V, E_1' \cup E_2')$.*

Lemma 2.4 *Let $T = (V, E_T)$ be a feasible solution, and suppose that $(x, z), (x, y) \in E_T$. Let P be a $y - z$ path in G such that $x \notin P$. Then $(w, x) \in E_T$ for every vertex $w \in P$.*

Proof: Consider the subtrees resulting from T by deleting x. By assumption, y and z belong to distinct subtrees. If the lemma does not hold, then there exist two adjacent vertices, a, b along P such that a and b belong to distinct subtrees and $(a, x) \notin E_T$. Then, the $a - b$ path in T contains at least 3 edges, and this contradicts $(a, b) \in E$. Therefore, $(w, x) \in E_T$ for every $w \in P$. ∎

Lemma 2.5 *Denote by $C_1, C_2, \ldots, C_l$ the non-trivial 2-connected components of G, and denote by $C_{l+1}, C_{l+2}, \ldots, C_{l+s}$ the components induced by the cut edges. Let $T = (V, E_T)$ be a feasible solution, and assume that $T(C_i)$ is not connected. Then there exists $v \in V \setminus C_i$ such that $T(C_i \cup \{v\})$ is a star with root v.*

Proof: If $|C_i| = 2$ the result is trivial. Assume that $|C_i| \geq 3$. If $T(C_i)$ is not connected, then there are $a, b \in C_i$ such that the $a - b$ path in T is a, v, b where $v \in V \setminus C_i$ and $(a, b) \in E$.

Let $w \in C_i$, then because $G(C_i)$ is 2-connected there is an $a - b$ path, P, in $G(C_i)$ such that $w \in P$. $v \notin P$. By Lemma 2.4, $(v, w) \in E_T$, and the claim follows. ∎

The feasibility question is the following: given G is there a feasible spanning tree T. This problem has its own interest. In the following sections we develop a polynomial time algorithm for the case when G is 2-connected. This algorithm finds a minimum cost feasible spanning tree if it exists, and it finds out if there is no such a feasible spanning tree. As we now show, the feasibility algorithm for 2-connected graphs actually solves the feasibility question for all graphs.

Lemma 2.6 *G has a feasible solution if and only if every 2-connected component of G has a feasible solution.*

Proof: Assume that every 2-connected component of G has a feasible solution. Then consider a tree resulted by the union of feasible solutions for every 2-connected component, and addition of the cut-edges of G. Then this tree is a feasible solution.

Assume that G has a feasible solution $T = (V, E_T)$ and assume that C is a 2-connected component of G. Let $T(C)$ be the induced subgraph of T over C. If $T(C)$ is not connected, then by Lemma 2.5 there is a vertex v that is adjacent in T to all the vertices from C. Pick a vertex $r \in C$, and let T_C be a star rooted at r with leaf set $C \setminus \{r\}$. Then T_C is a feasible solution for the induced subgraph of G over C. ∎

For a cycle C in G and distinct vertices $i, j, k, l \in C$, the edges (i, j) and (k, l) are *crossing edges with respect to* C if i, k, j, l appear in C in this order. Otherwise, the edges are *non-crossing edges with respect to* C. Note that by definition the edges $(i, j), (j, k)$ are non-crossing edges with respect to C.

Lemma 2.7 *Let $T = (V, E_T)$ be a feasible solution, and let C be a cycle in G. Then E_T does not contain crossing edges with respect to C.*

Proof: Suppose otherwise, that there exist $i, j, k, l \in C$ such that $(i, j), (k, l) \in E_T$ are crossing edges with respect to C.

- W.l.o.g. the $i - k$ path in T contains neither j nor l. This means that $d_T(l, j) \geq 3$, and therefore, $(l, j) \notin C$. Among the edges $(k, p) \in E_T$ that cross (i, j) with respect to C, we pick l that is the closest to j.
- Denote by T_l the subtree of T that contains l resulted by deleting k.
- Since the $j - k$ path in T consists of the edge (i, j) and the $i - k$ path in T, $j \notin T_l$.
- Denote by P_{lj} the $j - l$ path in C that does not contain k.
- Since $l \in T_l$ and $j \notin T_l$, there are adjacent vertices a, b in P_{lj} such that $a \in T_l$ and $b \notin T_l$.
- By the definition of l, $(b, k) \notin E_T$. Therefore, $d_T(a, b) \geq 3$ and this contradicts $(a, b) \in E$.

∎

Edges $(i_1, j_1), (i_2, j_2)$ and (i_3, j_3) are *non-overlapping with respect to a cycle* C if $i_1, j_1, i_2, j_2, i_3, j_3$ are distinct and appear in C in this order. The edges are *non-overlapping* if there is a cycle C in G such that they are non-overlapping with respect to C.

Lemma 2.8 *Let $T = (V, E_T)$ be a feasible solution. There is no set $S \subseteq E_T$ of non-overlapping edges.*

Proof: Assume that the lemma does not hold, and suppose that for a cycle C in G there is a set $S = \{(v_{i_1}, v_{j_1}), (v_{i_2}, v_{j_2}), (v_{i_3}, v_{j_3})\} \subset E_T$ such that $v_{i_1}, v_{j_1}, v_{i_2}, v_{j_2}, v_{i_3}, v_{j_3}$ appear in C in this order, v_{i_1} is the lowest index vertex in C (number the vertices in C clockwise in such manner),

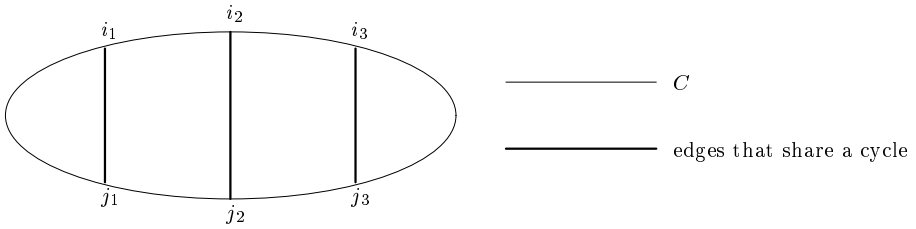

Figure 1: The structure of edges that share a cycle, by Corollary 2.9

and $l(S) = -j_3 + i_3 - j_2 + i_2 - j_1 + i_1$ is minimized (among all the sets of edges in T that satisfy the conditions with respect to C). Since $(v_{j_1}, v_{j_1+1}) \in E$, the $v_{j_1} - v_{j_1+1}$ path in T has at most two edges. $(v_{i_2}, v_{j_2}) \in E_T$, and therefore by Lemma 2.4, $(v_{j_1}, v_{j_1+1}) \notin E_T$. Therefore, the path has exactly two edges. Let v_{j_1}, u, v_{j_1+1} be the $v_{j_1} - v_{j_1+1}$ path in T. If $u \notin C$, then by Lemma 2.4 all the vertices of C are connected to u. This contradicts $(v_{i_1}, v_{j_1}) \in E_T$, and therefore, $u \in C$. To avoid a contradiction with Lemma 2.4, u must be on the path in C between v_{i_1} and v_{j_3} that does not contain v_{j_1}, and $u \neq v_{j_3}$.

If $u \notin \{v_{i_1}, v_{j_3}\}$, then we can replace v_{i_1} by u contradicting the minimality of $l(S)$.

If $u = v_{i_1}$, then we can replace v_{j_1} by v_{j_1+1} contradicting the minimality of $l(S)$ (if $v_{j_1+1} \neq v_{i_2}$) or the existence of such S if $v_{j_1+1} = v_{i_2}$ by Lemma 2.4. ∎

A set $S \subseteq E_T$ of 3 edges *shares a cycle* if there is a cycle C in G that contains all the endpoints of the edges in S.

Corollary 2.9 *If a set $\{(i_1, j_1), (i_2, j_2), (i_3, j_3)\}$ shares a cycle, then $i_1, i_2, i_3, j_3, j_2, j_1$ appear in C in this order (see Figure 1).*

Proof: By Lemmas 2.4 and 2.8. ∎

3 G is a Hamiltonian cycle

In this section we present a polynomial-time algorithm when $G = (V, E)$ is a Hamiltonian cycle, with $E = \{(1,2), (2,3), \ldots, (n-1, n), (n, 1)\}$. This algorithm will be used in the subsequent algorithms. In this section we need not assume that d satisfies the triangle inequality, but only that d is symmetric and non-negative.

Lemma 3.1 *Let $T = (V, E_T)$ be a feasible solution. Then, $E \cap E_T \neq \emptyset$.*

Proof: There may be at most $n-3$ non-crossing edges that are not cycle edges. As $|E_T| = n - 1$, by Lemma 2.7 T must contain cycle edges as well. ∎

Corollary 3.2 *If $(i, j) \in E_T$, then either $(i, j-1) \in E_T$ or $(i+1, j) \in E_T$.*

Proof: Let $S = \{i+1, i+2, \ldots, j-1\}$ (all operations are done *mod* n). Since T is a spanning tree, there is an edge $(u,v) \in E_T$ such that $u \in S$ and $v \in V \setminus S$. If $v \neq i,j$, then the edges (u,v) and (i,j) are crossing and this is impossible by Lemma 2.7. Therefore, w.l.o.g. $v = i$. Since $(i,j),(i,u) \in E_T$, by Lemma 2.4, $(i,j+1) \in E_T$. ∎

We present a polynomial time dynamic programming algorithm (Algorithm DP). Algorithm DP is based on Lemmas 2.4, 2.7, 3.1, and Corollary 3.2. Denote by $F(i,j)$ the minimum cost of a spanning tree over $\{i, i+1, \ldots, j\}$ that contains the edge (i,j), and satisfies the constraints defined by $P_{ij} = \{(i,i+1),(i+1,i+2),\ldots,(j-1,j)\}$ (all operations are done *mod* n). If there is no feasible solution, then Algorithm DP returns ∞.

By Lemma 3.1, we are looking for $Min_i F(i+1,i)$. F satisfies the following conditions: $F(i,i+1) = d_{i,i+1}$ $i = 1,2,\ldots,n$, and for $j \neq i+1$:

$$F(i,j) = d_{ij} + \begin{cases} \sum_{k=i+1}^{j} d_{ik} & \text{if } P_{ij} \cap E_1 = \{(i,i+1)\}, \\ \sum_{k=i}^{j-1} d_{jk} & \text{if } P_{ij} \cap E_1 = \{(j,j-1)\}, \\ d_{ij} + Min\{F(i+1,j), F(i,j-1)\} & \text{if } (i,i+1),(j-1,j) \in E_2, \\ \infty & \text{otherwise.} \end{cases}$$

The correctness of Algorithm DP is based on the fact that given that an edge $(i,j) \in E_T$, by Lemma 2.7, the problem is divided into two subproblems: the first is on P_{ij} and the second is on P_{ji}. By Corollary 3.2, given $(i,j) \in E_T$ then either $(i+1,j) \in E_T$ or $(i,j-1) \in E_T$. If $(i,i+1) \in E_1$ and $(i,j) \in E_T$, then by Lemma 2.4 for every $k \in P_{ij}$, $(i,k) \in E_T$. If $(j,j-1) \in E_1$ and $(i,j) \in E_T$, then by Lemma 2.4 for every $k \in P_{ij}$, $(i,k) \in E_T$. If $(i,i+1),(j,j-1) \notin E_1$, then the algorithm chooses the better possibility between adding $(i,j-1)$ to E_T and adding $(i+1,j)$ to E_T.

4 G is 2-connected

In this section we present a polynomial-time algorithm when G is 2-connected. In this section we need not assume that d satisfies the triangle inequality, but only that d is symmetric and non-negative.

Lemma 4.1 *Let v and w be vertices in a 3-connected component of G, and let T be a feasible solution. Then, the $v-w$ path in T contains at most two edges.*

Proof: Let P_T be the $v-w$ path in T. Let v' be the neighbor of v in P_T, and assume $v' \neq w$ (otherwise the lemma holds). Let $w' \neq v$ be the neighbor of v' in P_T. We claim that $w = w'$. Let P be a path in G between v and w that does not contain as inner vertices v' or w'. There exists such P because v and w belong to a common 3-connected component. Consider the subtrees obtained from T by deleting v'. By assumption, v and w belong to distinct subtrees. Therefore, along P, starting from v, there is the first vertex that belong to the subtree of w. Denote this vertex by w'', and let v'' be the vertex before w'' in P (v'' closer to v than w''). There is a requirement $(v'',w'') \in E$ and v'' and w'' belong to distinct subtrees. Therefore, the $v''-w''$ path in T is exactly $(v'',v'),(v',w'')$. Since w'' and w are in the same subtree, it must be that, $w'' = w' = w$. ∎

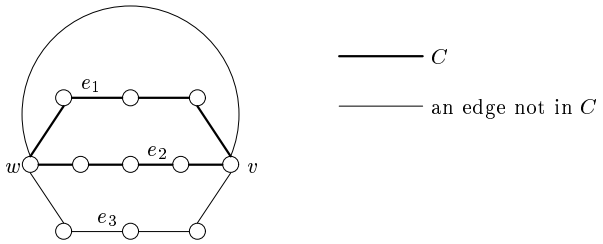

Figure 2: The cycle C in the proof of Lemma 4.5

Remark 4.2 *A consequence of Lemma 4.1 is that if G is 3-connected, then the minimum cost feasible star, if such a feasible star exists, is an optimal solution. If there is no feasible star, then the problem is infeasible.*

Lemma 4.3 *Let T be a feasible solution. Let $S = \{e_1, e_2, e_3\} \subseteq E_1$. Assume that for $v, w \in V$ there are disjoint $v - w$ paths P_1, P_2, P_3 in G, such that for $i = 1, 2, 3$, e_i connects a pair of vertices in P_i. Then $(v, w) \in E_T$.*

Proof: By assumption, there are 3 disjoint $v - w$ paths, P_1, P_2, and P_3. By Lemma 4.1, there is a $v - w$ path in T with at most two edges. If the path contains two edges, assume it is v, u, w. u is in at most one of the 3 disjoint $v - w$ paths. By Lemma 2.4, all the vertices in the other paths are connected to u. This contradicts the fact that $e_1, e_2, e_3 \in E_T$. Therefore, $(v, w) \in E_T$. ∎

Lemma 4.4 *Let T be a feasible solution. Let $S = \{e_1, e_2, e_3\} \subseteq E_1$. If there is no cycle C in G that contains all the vertices of the edges of S, then there exist $(v, w) \in E_T$ and three disjoint $v - w$ paths P_1, P_2, P_3 in G, such that for $i = 1, 2, 3$, e_i connects a pair of vertices in P_i.*

Proof: Since G is 2-connected, there is a cycle C in G that contains e_1 and e_2. Denote by P_1 and P_2 the paths in $C \setminus \{e_1, e_2\}$. By Observation 2.1, for every $w, v \in C$ there exists a $w - v$ path in G that contains e_3. Let P be a minimal path with this property. Thus, $P \cap C = \{v, w\}$.

There is no i such that $v, w \in P_i$ because by assumption e_1, e_2, e_3 do not share a cycle. The claim follows now from Lemma 4.3. ∎

Lemma 4.5 *Let T be a feasible solution. Assume that $|E_1| \geq 3$. Suppose that there are no subset $S = \{e_1, e_2, e_3\} \subseteq E_1$ and cycle C in G such that C contains all the vertices of the edges of S. Then, for every $S = \{e_1, e_2, e_3\} \subseteq E_1$ let v, w, P_1, P_2 and P_3 as in Lemma 4.4. Then, $(v, w) \notin E_1$.*

Proof: Consider the cycle $C = P_1 \cup P_2$ (see Figure 2). C contains all the vertices of $S = \{e_1, e_2, (v, w)\}$. Therefore, by assumption, $S \not\subseteq E_1$, and therefore, $(v, w) \notin E_1$. ∎

Let $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in E$, then (v_3, w_3) *separates* (v_1, w_1) and (v_2, w_2) if $\{v_3, w_3\}$ is a cut-set of G and $\{v_1, w_1\} \setminus \{v_3, w_3\}$ and $\{v_2, w_2\} \setminus \{v_3, w_3\}$ are in distinct components of this cut.

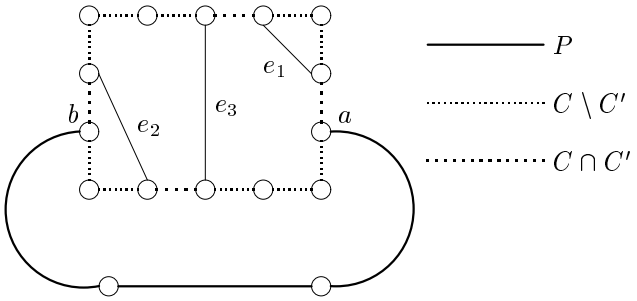

Figure 3: The subgraph $C \cup P \cup \{e_1, e_2, e_3\}$ in the proof of Lemma 4.6

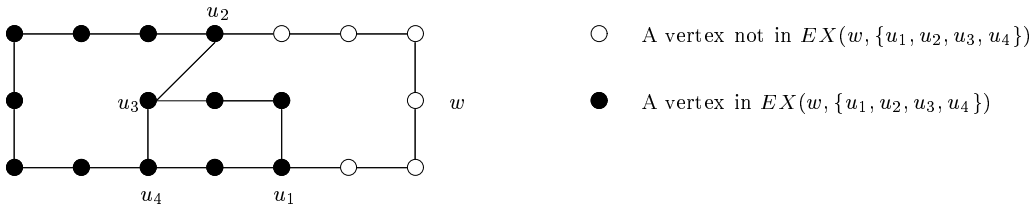

Figure 4: $EX(w, \{u_1, u_2, u_3, u_4\})$

Lemma 4.6 *Assume that G has a feasible solution. Let $S = \{e_1, e_2, e_3\} \subseteq E_1$ be a set of edges whose vertices lie on a common cycle in G. Then, one of the edges, w.l.o.g. $e_3 = (v, w)$, separates $e_1 = (v_1, w_1)$ and $e_2 = (v_2, w_2)$.*

Proof: Let C be a cycle in G such that C contains all the endpoints of S. By Corollary 2.9, the deletion from C of the endpoints of one of the edges, say $e_3 = (v, w)$, results in two paths, say P_1 and P_2, such that $\{v_1, w_1\} \setminus \{v, w\} \subseteq P_1$ and $\{v_2, w_2\} \setminus \{v, w\} \subseteq P_2$. We will show that in $G \setminus \{v, w\}$ there is no path connecting P_1 and P_2.

Assume otherwise, that there are $a \in P_1$ and $b \in P_2$ and an $a - b$ path P (see Figure 3). Consider the subgraph of G defined by $C \cup P \cup \{e_1, e_2, e_3\}$. This subgraph contains a cycle C' such that the set S is non-overlapping with respect to C'. By Lemma 2.8, this is impossible. ∎

For a non-trivial 3-connected component U of G, and a vertex $w \in V$, the *extension $EX(w, U)$ of U with respect to w* is defined as the 2-connected component in $G \setminus \{w\}$ that contains $U \setminus \{w\}$ (see Figure 4).

Lemma 4.7 *Assume that $T = (V, E_T)$ is a feasible solution, and that U is a non-trivial 3-connected component of G. Then, there exists $w \in V$, such that $(w, x) \in E_T$ for every $x \in EX(w, U)$.*

Proof: By Lemmas 2.4 and 4.1. ∎

Consider a nontrivial 3-connected component U of G, and a vertex $w \in V$. Let $C'_1, C'_2, \ldots, C'_l$ be the connected components of $G\,(V \setminus (EX(w, U) \cup \{w\}))$. Let v_i be the unique vertex in $EX(w, U)$

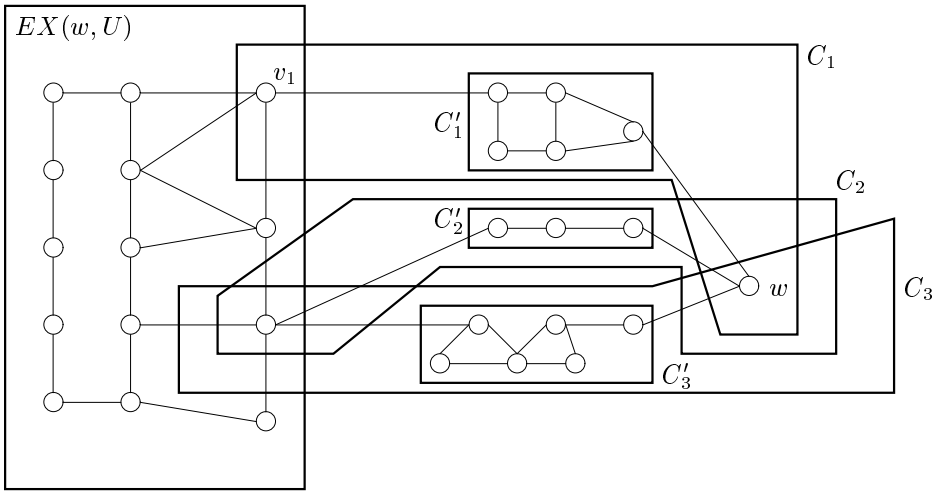

Figure 5: The sets $EX(w,U)$, C_i and C'_i

which is adjacent to a vertex in C'_i (note that $EX(w,U), C'_1, C'_2, \ldots, C'_l, \{w\}$ is a partition of V).

We define a set $SP(w,U)$ of *subproblems* associated with U and w: The set of vertices of subproblem i is:

$$C_i := C'_i \cup \{v_i, w\}$$

(see Figure 5). Let $E'_1 = E_1(C_i) \cup \{(v_i, w)\}$. Let $E'_2 = E(C_i) \setminus E'_1$. A subproblem (C_i, E') is defined as the problem induced by the graph $G' = (C_i, E'_1 \cup E'_2)$ and the metric $d(C_i)$ induced by d over C_i. The set of subproblems defined for a nontrivial 3-connected component U of G and a vertex w is:

$$SP(w,U) := \{(C_i, E')\}.$$

The following lemma shows that the set of subproblems decomposes the original problem.

Lemma 4.8 *Let $T = (V, E_T)$ be a feasible solution. Let U be a non-trivial 3-connected component of $G = (V, E)$, and let $w \in V$ be as in Lemma 4.7. Then, T induces a connected subgraph over every C such that $(C, E') \in SP(w,U)$.*

Proof: Assume that the lemma does not hold for some $(C, E') \in SP(w,U)$. Then, there are $a, b \in C$ such that $(a,b) \in E$, and the a–b path in T is a, z, b such that $z \notin C$. Let $v = EX(w,U) \cap C$. W.l.o.g. $a \neq v, w$ (see Figure 6).

$v \in C$ and $z \notin C$ and therefore, $z \neq v$. There is a cycle H in G that contains w, a, v, z in this order because of the following: since G is 2-connected, there is a $w - v$ path in G that traverses a. Since $\{w, v\}$ is a cut set, this path is in $G(C)$. There is another disjoint $w - v$ path that traverses z. By assumptions, $(a, z), (v, w) \in E_T$. These edges are crossing with respect to H. This is impossible by Lemma 2.7.

■

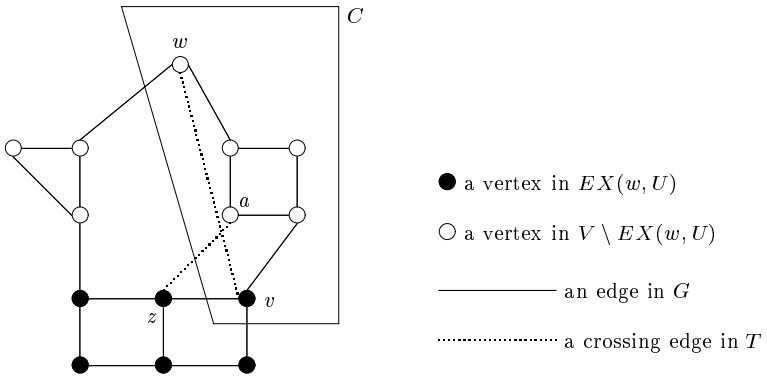

Figure 6: The graph in the proof of Lemma 4.8

Algorithm TVC (see Figure 7) breaks the input problem into subproblems. It continues to decompose the subproblems until each subproblem is defined using a set $E_1^a(C)$ that contains at most two edges, and then the subproblems are solved by recursively calling Algorithm TVC.

Lemma 4.9 *There are at most $O(m^2n)$ subproblems that are solved by recursive calls to Algorithm TVC.*

Proof: Each subproblem can be identified by the set of vertices C and by the set of edges $S = E_1^a(C)$ (with at most two edges). Denote by U the set of end vertices of S. Every edge $(u,v) \in S$ is either originally in E_1 or it corresponds to a cut-set $\{u,v\}$ in the original graph (as in Lemma 4.6). A set of vertices C' that can be used to define a subproblem (C',S) has to satisfy the following: $U \subseteq C'$, and $C' \setminus U$ is a connected component of the graph resulted from G by deleting U. There may be at most $O(n)$ connected components resulted by deleting U. Therefore, there are at most $O(n)$ subproblems that are solved for any S. Therefore, throughout the algorithm at most $O(m^2n)$ subproblems are solved. ∎

Theorem 4.10 *Algorithm TVC returns an optimal solution in strongly-polynomial time.*

Proof: The correctness of the decomposition is based on Lemma 4.8. Further decomposition is based on Lemmas 4.3 and 4.6. By Observation 2.3, when we decide that an edge belongs to T we can add it to E_1 without affecting the feasibility of T.

We prove by induction on the number of recursive calls the algorithm uses, that it returns an optimal solution.

If no such call is used, then by the optimality of Algorithm DP, Algorithm TVC returns an optimal solution.

Consider an optimal solution T. By Lemma 4.7, there is a vertex $w \in V$ such that $(w,x) \in E_T \ \forall x \in EX(w,V')$. We consider the tree T_w built by the algorithm. By Lemmas 4.3, 4.6 and

4.8, the union of the solutions of all these subproblems gives an optimal solution to the original problem. The correctness of the induction follows from an induction assumption.

By Lemma 4.9, the total number of subproblems that are solved throughout the algorithm is at most $O(m^2 n)$. Consider the work involved in a subproblem. Each separation of a subproblem into smaller subproblems takes $O(m)$ time. Each pair of such separations decreases the sum $\sum_{(C,E')\in SP} |C| - 1$ by at least one (applying Lemma 4.3 does not decrease this sum, but the next iteration of the **while** loop chooses the same set C, and then using Lemma 4.6 the sum decreases). Therefore, the total work of the **while** loop is at most $O(mn)$. The work in the **for** loop is dominated by the work in the **while** loop, and it is repeated at most n times. Therefore, the work involved in a subproblem is $O(mn^2)$. Therefore, the total work is $O(m^3 n^3)$. ∎

5 An approximation algorithm when G contains a single cut-vertex

In this section we present an approximation algorithm for the case when G has a single cut-vertex. This algorithm is based on a known approximation algorithm for the METRIC UNCAPACITATED FACILITY LOCATION WITH PENALTIES PROBLEM. It will be used in the next section to present an approximation algorithm for general graphs.

Assume that G is connected, and has a single cut-vertex, r. Denote by $C_1, C_2, \ldots, C_l$ the non-trivial 2-connected components of G, and denote by $N = \{v_1, v_2, \ldots, v_s\} = V \setminus (\cup_{i=1}^{l} C_i)$ the set of neighbors of r which do not belong to any non-trivial 2-connected component. Let $C_{l+i} = \{v_i, r\}$, $i = 1, 2, \ldots, s$.

The METRIC UNCAPACITATED FACILITY LOCATION WITH PENALTIES PROBLEM (UFLP) (see [3] for a 3-approximation algorithm for the problem, and see [11] for an $O(m \log m)$-time 2-approximation algorithm) is defined as follows: given a complete bipartite graph $B = (F, C, E_B)$ with costs f_i for opening a facility at vertex $i \in F$, metric costs $c(i, j)$ for servicing a customer $j \in C$ by an open facility $i \in F$, and costs (penalties) p_j for not servicing customer $j \in C$, the goal is to minimize the total cost of the selected facilities, the penalty costs of the non-served customers, and the servicing costs of every served customer by its closest open facility.

The algorithm in [11] can be extended to the weighted case where each customer $j \in C$ has a weight w_j and the servicing cost is $w_j c(i, j)$ (and the penalty cost is still p_j). This is done using the same method described in [11] for the weighted case of the METRIC UNCAPACITATED FACILITY LOCATION PROBLEM.

We define an instance of UFLP. In this instance, F consists of the vertices $u \in V$. C consists of the components $C_1, \ldots, C_{l+s}$. Opening a facility at a vertex u means that u is directly connected to r in the solution T. The cost of opening the facility is therefore, d_{ru}. Paying the penalty for a component C_j means that C_j induces a connected component in T. The penalty cost is therefore, $TVC(C_j)$, where $TVC(C_j)$ is the solution value of the optimal tree for the problem defined by $(C_j, E_1 \cap G(C_j))$. Servicing a component C_j by an open facility u means that u is connected to every $v \in C_j \setminus \{r\}$. The service cost is therefore, $\sum_{v \in C_j \setminus \{r\}} d_{uv}$.

There are solutions to the UFLP instance which do not correspond to any feasible solution

```
TVC
  input
  A 2-connected graph G = (V, E).
  A symmetric non-negative cost function d.
  A set E_1.
  returns
  A spanning tree T = (V, E_T).
  begin
  if G is a Hamiltonian cycle
    then
        return DP(G, d, E_1).
    end if
  V' := a non-trivial 3-connected component of G.
  A := V.
  for every a ∈ A
        E_a := {(a, x)|x ∈ EX(a, V')}.
        T_a := (V, E_a ∪ E_1).
        SP := SP(a, V').
        while there is (C, E') ∈ SP such that |E'_1| ≥ 3
            SP := SP \ {(C, E')}.
            if E'_1 contains 3 edges e_1, e_2, e_3 that share a cycle
              such that e_3 separates e_1 and e_2
              then
                  E^a_1(C) := E'_1.
                  (v, w) := e_3.
                  elseif E'_1 contains 3 edges e_1, e_2, e_3 such that for some
              v', w' ∈ V the conditions of Lemma 4.3 holds
              then
                  (v, w) := (v', w').
                  E^a_1(C) := E'_1 ∪ {(v, w)}.
              else
                  A := A \ {a}.
                  break while
              end if
            {(V(C^i), E(C^i))}^l_{i=1} := the set of connected components in G(C) \ {v, w}.
            for every i = 1, 2, ..., l
                  V(C^i) := V(C^i) ∪ {v, w}.
                  SP := SP ∪ {(C^i, E^a_1(C^i))}.
            end while
        if a ∈ A
          for every (C, E') ∈ SP
                T' := TVC(G(C), d(C), E'_1).
                if T' infeasible
                  then
                      A := A \ {a}.
                  else
                      T_a := T_a ∪ T'.
                  end if
        end if
  if A = ∅
    return infeasible
  end if
  return Minimum cost spanning tree among {T_a}_{a∈A}.
  end TVC
```

Figure 7: Algorithm TVC for the 2-connected case

of the RHMST. A solution to the UFLP instance corresponds to a graph that satisfies the hop requirements. However, this graph may contain cycles (and even parallel edges), when a facility is placed in a vertex of a component that pays the penalty. However, we use a solution to the UFLP to construct an approximate RHMST solution.

This formulation of the UFLP instance is not metric. To make the instance metric we define the weight of a component C_j to be $|C_j \setminus \{r\}|$, and adjust c appropriately.

We now give the details of the construction.

Define an instance $UFLP(G, d)$ for the UFLP as follows: the graph $B = (F, C, E)$ is defined by $F = V$ and $C = \{C_1, C_2, \ldots, C_{l+s}\}$. Thus, there is a customer for every vertex in N, a customer for every non-trivial 2-connected component, and a facility can be opened in every vertex of V.

We use the notion of C_j both as a component of G and as a vertex in B. The corresponding meaning is clarified by the context.

The weight w_v of a customer $v \in C$ is defined as: $w_v = |C_j| - 1$ if $v = C_j$.

Opening a facility at $u \in F$ costs $f_u = d_{ru}$.

The service cost $c(u, v)$ $u \in F, v \in C$ is defined as follows:

$$c(u,v) = \begin{cases} \frac{1}{|C_j|-1}\Sigma_{v \in C_j \setminus \{r\}} d_{uv} & \text{if } v = C_j \text{ and } E_1(C_j) \subseteq \{(u,x) | x \in C_j\} \\ \infty & \text{otherwise.} \end{cases}$$

The penalty cost p_v $v \in C$ is defined as follows: $p_v = TVC(G(C_j), d(C_j), E_1(C_j))$ if $v = C_j$, where $p_v = \infty$ if $TVC(G(C_j), d(C_j), E_1(C_j))$ is infeasible.

Observation 5.1 *c is a metric.*

Proof: Let $C_i, C_j \in C$ and let $u, v \in V$. We will show that $c(u, C_i) + c(u, C_j) + c(v, C_j) \geq c(v, C_i)$.

Since d is a metric, for every $w \in C_i \setminus \{r\}$ and $x \in C_j \setminus \{r\}$,

$$d_{vw} \leq d_{uw} + d_{ux} + d_{xv}.$$

Averaging over all $x \in C_j \setminus \{r\}$ results

$$d_{vw} \leq d_{uw} + \frac{1}{|C_j| - 1} \sum_{x \in C_j \setminus \{r\}} (d_{ux} + d_{xv}).$$

Averaging over all $w \in C_i \setminus \{r\}$ results

$$\frac{1}{|C_i| - 1} \sum_{w \in C_i \setminus \{r\}} d_{vw} \leq \frac{1}{|C_i| - 1} \sum_{w \in C_i \setminus \{r\}} d_{uw} + \frac{1}{|C_j| - 1} \sum_{x \in C_j \setminus \{r\}} d_{ux} + \frac{1}{|C_j| - 1} \sum_{x \in C_j \setminus \{r\}} d_{xv}.$$

This proves the claim. ∎

Consider a solution returned by an α-approximation algorithm for the UFLP (using [11], $\alpha = 2$). This solution consists of the set of open facilities, F, the set of unserved customers ,P (the customers

that pay the penalties), and a mapping $\phi : C \setminus P \to F$ that maps every served customer to the open facility that serves it.

Algorithm SCV (see Figure 8) approximates the instance $UFLP(G, d)$, and then changes the solution until it becomes a feasible solution to the RHMST. The changes are made in order to provide the following property: if the solution opens a facility at $u \in C_j$, then u also serves C_j. If this property does not hold, then we replace u by a facility x that belongs to the set of vertices that are served by u, such that the resulting cost is minimized.

Let $(\tilde{F}, \tilde{\phi}, \tilde{P})$ be a feasible solution to $UFLP(G, d)$, and let $u \in \tilde{F}$. Denote by $U(u)$ the set of vertices that belong to components that are served by u. Formally, $U(u) = \cup_{j:\tilde{\phi}(C_j)=u} C_j$.

SCV
input
A connected graph $G = (V, E_1 \cup E_2)$ *with a single cut vertex* r.
A cost metric d.
returns
A spanning tree $T = (V, E_T)$.
begin
$E_T := \emptyset$.
$C_1, C_2, \ldots, C_l :=$ *the non-trivial 2-connected components of* G.
$\{v_1, v_2, \ldots, v_s\} := V \setminus (\cup_{i=1}^{l} C_i)$.
for every $i = 1, 2, \ldots, s$
$C_{l+i} := \{v_i, r\}$.
$(F, \phi, P) :=$ *an* α*-approximation solution for* $UFLP(G, d)$.
$[(F', \phi', P') := (F, \phi, P)]$.
if *the cost of* (F, ϕ, P) *equals* ∞
then
return *infeasible.*
end if
for every $C_j \in P$
$T' = (C_j, E_{T'}) := TVC(G(C_j), d(C_j), E_1(C_j))$.
$E_T := E_T \cup E_{T'}$.
for every $u \in F$ $(u \in C_j)$.
if $C_j \in P$ *or* $\phi(C_j) \neq u$
then
$x := argmin_{x \in U(u)}\{d_{rx} + \sum_{v \in U(u)} d_{xv}\}$.
$E_T := E_T \cup \{(r, x)\} \cup \{(v, x) : v \in U(u) \setminus \{x\}\}$.
$[F' := F' \cup \{x\} \setminus \{u\}]$.
$[\phi'(v) := x \ \forall v \in U(u)]$.
else $(\phi(C_j) = u)$
$E_T := E_T \cup \{(r, u)\} \cup \{(u, v) : v \in U(u) \setminus \{u\}\}$.
end if
return $T = (V, E_T)$.
end *SCV*

Figure 8: Algorithm SCV for the single cut-vertex case

Lemma 5.2 *Denote by sol the cost of a feasible solution SOL to $UFLP(G,d)$. Then there exists a solution T to $RHMST(G,d)$ whose cost is at most $2sol$. Moreover, T is found by Algorithm SCV in strongly-polynomial time.*

Proof: Denote by SOL' a modified solution constructed from SOL as follows:

Consider an open facility u. Let $x := argmin_{x \in U(u)}\{d_{rx} + \sum_{v \in U(u)} d_{xv}\}$. $d_{rx} + \sum_{v \in U(u)} d_{xv} \leq d_{ru} + 2\sum_{v \in U(u)} d_{uv}$. To verify this inequality, note that it is satisfied in particular by the vertex in $U(u)$ which is closest to u. Therefore, the replacement of u by x at most doubles the service costs of the solution. Therefore, all these changes produce a modified solution $SOL' = (F', \phi', P)$ defined throughout the algorithm which costs at most $2sol$.

$T = (V, E_T)$ is defined as follows:

$$E_T = \{(r,u) : u \in F'\} \cup \{(u,v) : u \in F',\ v \in U(u)\} \cup_{C_j \in P} TVC(C_j).$$

If SOL' opens a facility at $v \in C_i$, then v also serves C_i. Therefore, T is a feasible solution. The cost of T equals the cost of SOL'.

Therefore, T can be constructed from SOL in strongly-polynomial time, and its cost is at most $2sol$. ∎

Lemma 5.3 *Denote by c_T the cost of a feasible solution $T = (V, E_T)$ to $RHMST(G,d)$. Let $V' \subseteq V$. Then, there exists a solution SOL to $UFLP(G(V'), d(V'))$ whose cost is at most $2c_T$.*

Proof: Define a solution $SOL = (\tilde{F}, \tilde{\phi}, \tilde{P})$ as follows: $\tilde{F} = \{u \in V' : (r,u) \in E_T\}$, $\tilde{\phi}(C_j) = u$ if $u \in \tilde{F}$ and $(u,v) \in E_T\ \forall v \in C_j$, and $\tilde{P} = \{C_j : \text{no such } u \text{ exists for } C_j\}$. SOL is clearly a feasible solution to $UFLP(G(V'), d(V'))$.

The cost of SOL is at most $2c_T$. This is so as each cost in SOL corresponds to a cost in T, and each cost in T is counted at most twice. ∎

Lemma 5.4 *Assume that there exists a feasible solution $T = (V, E_T)$ to $RHMST(G,d)$ with cost c_T. Let $V' \subseteq V$. Then, Algorithm SCV returns a solution T' to $RHMST(G(V'), d(V'))$ whose cost is at most $4\alpha c_T$.*

Proof: The solution obtained by an α-approximation algorithm for $UFLP(G,d)$ costs at most α times the cost of an optimal solution to $UFLP(G,d)$. By Lemma 5.3, this cost is at most $2\alpha c_T$. By Lemma 5.2, the cost of the solution returned by Algorithm SCV is at most $4\alpha c_T$. ∎

Theorem 5.5 *Algorithm SCV is a strongly-polynomial 4α-approximation.*

Proof: To prove feasibility consider an edge $(u,v) \in E$.

- $(u,v) \in E_1$. If $u, v \in C_j$, then the service cost $c(C_j, w) = \infty\ \forall w \neq u, v$. Therefore, either $C_j \in P$, and therefore, by the correctness of Algorithm TVC, T includes as a subtree a feasible tree over C_j, or C_j is served by u or v. In both cases, $(u,v) \in E_T$.

- $(u, v) \in E_2$. If $u, v \in C_j$, then either T induces a feasible solution over C_j, by the correctness of Algorithm TVC, or all the vertices of C_j are connected by a star (the center of the star may be in C_j or outside of C_j).

Therefore, Algorithm SCV returns a feasible solution.

Denote by *opt* the optimal cost of $RHMST(G, d)$. By Lemma 5.4, Algorithm SCV returns a solution whose cost is at most $4\alpha opt$.

Algorithm TVC is strongly-polynomial by Theorem 4.10, there is a strongly-polynomial α-approximation algorithm for UFLP, and the non-trivial 2-connected components of a graph can be computed in a strongly-polynomial time. Therefore, Algorithm SCV takes strongly-polynomial time. ∎

6 An approximation algorithm for general graphs

Algorithm APX (see Figure 9) decomposes the problem into subproblems. Each subproblem corresponds to either a 2-connected component, and in this case it uses TVC to solve it to optimality, or to a subgraph that contains a single cut-vertex, and in this case it uses SCV to approximate it. The algorithm considers the union of all these subtrees, and extends this forest into a spanning tree using edges from a minimum cost spanning tree. The last phase is needed only if G is not connected.

Theorem 6.1 *Algorithm APX is a strongly polynomial* $(16\alpha + 1)$*-approximation.*

Proof: The feasibility of the solution follows from Lemma 2.6 and the correctness of Algorithms TVC and SCV.

Both Algorithms TVC (by Theorem 4.10) and SCV (by Theorem 5.5) are strongly polynomial. The minimum cost extension to a spanning tree in the last step can be done via computation of a minimum cost spanning tree of the graph constructed from G by contracting each connected component of T to a single vertex. Therefore, Algorithm APX takes strongly-polynomial time.

Consider an optimal solution $T_{opt} = (V, E_{opt})$ to the RHMST problem. For every cut-vertex u, let $T(opt)_u = (V(opt)_u, E(opt)_u)$ be a minimal (with respect to inclusion) subtree of T_{opt} that satisfies the requirements of $V(u) = C_u \cup N_u \cup \{u\}$ ($V(opt)_u$ includes $V(u)$ but perhaps some more vertices).

By Lemma 5.4, where the tree $T(opt)_u$ is used as T, $T(u)$ costs at most 4α times the cost of $T(opt)_u$.

We note that every edge $e \in E_{opt}$ participates only in subtrees that correspond to a set $V(u)$ that contains at least one of its endpoints. A cut-vertex (except r) belongs to two such sets, and other vertices belong to one such set. Therefore, e participates in at most four of the spanning trees in the set $\{T(opt)_u\}$.

```
APX
    input
    A graph G = (V, E_1 ∪ E_2).
    A cost metric d.
    returns
    A spanning tree T = (V, E_T).
    begin
    E_T := ∅.
    for every connected component G' = (V', E') of G
            if G' is 2-connected
              then
                    T' = (V', E_T') := TVC(G', d(V'), E_1 ∩ E').
                    if T' is infeasible
                      then
                            return infeasible.
                      end if
                    E_T := E_T ∪ E_T'.
              else
                    r := a cut-vertex in G'.
                    for every cut vertex u of G'
                                C_u := {v ∈ V' : there are two disjoint v − u paths
                                and the r − v paths contain u}.
                                N_u := {v ∈ V' \ C_u : (v, u) ∈ E', and the r − v paths
                                contain u}.
                                U_u := C_u ∪ N_u ∪ {u}.
                                T(u) = (U_u, E_u) := SCV (G(U_u), d(U_u), E_1(U_u)).
                                if T(u) is infeasible
                                  then
                                        return infeasible.
                                  end if
                                E_T := E_T ∪ E_u.
              end if
    return minimum cost extension of T to a spanning tree.
    end RHMST_APX
```

Figure 9: Algorithm APX

Therefore, the cost of T before the extension (using the edges of a minimum spanning tree) is at most 16α times the cost of T_{opt}. Since a minimum cost spanning tree costs at most the cost of T_{opt}, the output tree costs at most $16\alpha + 1$ times the cost of T_{opt}. ∎

References

[1] L. Alfandari and V. T. Paschos, "Approximating minimum spanning tree of depth 2" *Intl. Trans. in Op. Res.*, **6**, 607-622, 1999.

[2] L. Cai and D. G. Corneil, "Tree Spanners", *SIAM J. Disc. Math.*, **8**, 359-387, 1995.

[3] M. Charikar, S. Khuller, D. M. Mount and G. Narasimhan, "Algorithms for facility location problems with outliers", *Proceedings of SODA 2001*, 642-651.

[4] G. Dahl, "The 2-hop spanning tree problem", *Operations Research Letters*, **23**, 21-26, 1998.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[6] L. Gouveia and E. Janssen, "Designing reliable tree networks with two cable technologies", *European Journal of Operational Research*, **105**, 552-568, 1998.

[7] L. Gouveia, "Multicommodity flow models for spanning trees with hop constraints", *European Journal of Operational Research*, **95**, 178-190, 1996.

[8] S. Guha and S. Khuller, "Greedy strikes back: Improved facility location algorithms", *Journal of Algorithms*, **31**, 228-248, 1999.

[9] L. Gouveia and C. Raquejo, "A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem", *European Journal of Operational Research*, **132**, 539-552, 2001.

[10] G. Kortsarz and D. Peleg, "Approximating the weight of shallow Steiner trees," *Discrete Applied Mathematics*, **93**, 265-285, 1999.

[11] M. Mahdian, E. Markakis, A. Saberi and V. V. Vazirani, "A greedy facility location algorithm analyzed using dual fitting", *Proceedings of 5th International Workshop on Randomization and Approximation Techniques in Computer Science, LNCS* **2129**, 127-137, 2001.

[12] M. Mahdian, Y. Ye and J. Zhang, "A 1.52-approximation algorithm for the uncapacitated facility location problem", *Proceedings of APPROX 2002, LNCS* **2462**, 229-242 , 2002.

[13] J. Monnot, "The maximum f-depth spanning tree problem", *Information Processing Letters*, **80**, 179-187, 2001.

[14] S. Voss, "The Steiner tree problem with hop constraints," *Annals of Operations Research*, **86**, 321-345, 1999.