

Optimal and Approximate Station Placement in Networks*

(With Applications to Multicasting and Space Efficient Traversals)

Clemente Galdi¹, Christos Kaklamanis²,
Manuela Montangelo¹, and Pino Persiano¹

¹ Dipartimento di Informatica ed Applicazioni
Università di Salerno, 84081, Baronissi (SA), Italy
{clegal,montange,giuper}@dia.unisa.it

² Computer Technology Institute
Dept. of Computer Engineering and Informatics
University of Patras, 26500 Rio, Greece
kakl@cti.gr

Abstract. In this paper we study the *k-station placement* problem (*k*-SP problem, in short) on graphs. This problem has application to efficient multicasting in circuit-switched networks and to space efficient traversals. We show that the problem is NP-complete even for 3-stage graphs and give an approximation algorithm with logarithmic approximation ratio. Moreover we show that the problem can be solved in polynomial time for trees.

Keywords: Multicasting, approximation algorithms, distributed systems, networks.

1 Introduction

In this paper we introduce and study the *k*-station placement problem on graphs. Consider a communication network modeled by a weighted directed graph $G = (V, E)$ where the length $l(e)$ of an edge e is the cost of sending a message along that edge. Any vertex u of G that needs to communicate with another vertex v does so by first establishing a virtual circuit p from u to v along a path connecting u and v and then by sending the message. The cost of establishing a virtual circuit p is the sum of the lengths of the edges of p . Suppose now, that we are given a distinguished *source vertex* s and a set D of *destination vertices* and that s needs to multicast a message to the vertices of D . One possible way of performing this task would be for s to establish a virtual circuit with each vertex v of D along the shortest path between s and v . This approach has the advantage that transmission is achieved in one step but its cost might be very high: one

* Partially supported by a grant from the Università di Salerno (Italy) and by Progetto cofinanziato MURST 40% Resource Allocation in Communication Networks.

edge might be used by multiple circuits with the effect that its cost would be added with same multiplicity to the total cost. A different approach would be to identify a set S_1 of intermediate stations and assign each vertex of D to a vertex of S_1 . The communication takes places in two steps. First, s establishes a virtual circuit with each of the vertices of S_1 and transfers the message. In the second phase, each vertex of S_1 establishes circuits with its assigned destinations so that the message is finally delivered to the destination vertices. In general, one can have k sets S_1, \dots, S_k of intermediate stations. The communication takes place in k steps: the source vertex establishes a circuit with each of the vertices of S_1 (level 1 stations); vertices of S_i (level i stations), $1 \leq i < k$, establish circuits with the vertices of S_{i+1} (level $i + 1$ stations) and finally vertices of S_k establish circuits with the vertices of D .

There is a clear tradeoff between the number of steps needed to complete the multicasting (that is the number of intermediate stations along a path from s to a destination vertex) and the cost of the transmission: 1-step communication incurs in high cost; having each vertex as an intermediate destination yields minimum cost multicasting but it completely wastes the performance offered by circuit-switched network.

In this paper we study the problem of allocating intermediate stations in a graph so that at most k station are encountered on a path from the source to a destination and the cost of multicasting is minimum.

The k -SP problem has also applications to the problem of traversing an ordered binary tree T . Each vertex of the tree has pointers to the left and right child only and we are provided with a pointer to the root of the tree. The inorder traversal of the tree reaches all the leaves of the tree in time $O(n)$ (here n is the number of vertices of T). However, in the worst case, it needs $\Omega(n)$ registers to store the addresses of the vertices of T for which the traversal has not been completed yet. This is hidden by the recursive approach often used to present the inorder traversal. Alternatively, one might consider the following approach. Each leaf v of a binary tree of depth h is uniquely identified by the binary string $\text{PATH}(v)$ of length at most h that describes the path from the root to v (0 stands for a link to a left child and 1 for a right child). Thus, we have the following very simple algorithm: for each leaf v , start from the root of T and follow the path specified by $\text{PATH}(v)$. As it is easily seen this algorithm does not need any additional register to store addresses of vertices of the tree but, on the other hand, its running time is proportional to the path length of the tree that can be $\Omega(n^2)$. In general, one might ask to perform the fastest traversal of the tree given that only k registers are available. As it will be clear in the sequel, this problem is closely related to the k -SP problem.

The k -Station Placement Problem. We now formally define the k -Station Placement (k -SP) problem.

Definition 1 (The k -SP Problem). *An instance of the k -SP problem consists of a directed graph $G = (V, E)$, a length function ℓ defined over the edges of G , an integer k , a source vertex s and a set of destination vertices D .*

A feasible solution to the k -SP problem (or k -placement) consists of $k+1$ sets of stations S_1, \dots, S_k, S_{k+1} , with $S_{k+1} = D$, and k assignments ϕ_1, \dots, ϕ_k , with ϕ_i mapping vertices of S_{i+1} into vertices of $S_i \cup \{s\}$. For every i , function ϕ_i must satisfy the following property (that we will call strictness): for any $v \in S_{i+1}$, the lightest path from v to $\phi_i(v)$ does not contain any vertex in S_{i+1} other than v .

The cost of a feasible solution $P = (S_1, \dots, S_k, \phi_1, \dots, \phi_k)$ is

$$Cost(P) = \sum_{i=0}^k \sum_{v \in S_{i+1}} w(\phi_i(v), v),$$

where $\phi_0(v) = s$ for all $v \in S_1$ and $w(u, v)$ is the cost of the shortest path from u to v according to ℓ .

The task is to compute a feasible solution of minimum cost.

The *strictness* property guarantees that the shortest path from the source to any destination node contains no more than k stations. If the property does not hold, we have a new problem we call the k -Unrestricted Station Placement problem, or k -USP problem.

Going back to the example of multicasting in a network G , we observe that the minimum-cost k -hop multicasting is obtained by solving the k -SP problem on G . The k -placement gives the k sets S_1, \dots, S_k of intermediate destinations and specifies, by means of the ϕ_i 's, the virtual connections each vertex of S_i has to establish (i.e., $v \in S_i$ has to establish a virtual circuit with each vertex $u \in S_{i+1}$ such that $\phi_i(u) = v$). The cost of the k -placement is the sum of the lengths of the circuits that are established to accomplish multicasting.

Let us now briefly discuss how the k -SP problem can be used to design the fastest traversal of a tree T using at most k registers to store pointers to vertices. Suppose we have the solution to the k -SP problem on T with s equal to the root of T and the set D equal to the set of leaves of T . The traversal proceeds in the following way. From the root, we reach each of the vertices of S_1 . While at $s_1 \in S_1$, we recursively traversal the tree rooted at s_1 using $k - 1$ registers (one register is used to keep a pointer to s_1). It is easy to see that the cost of the k -placement is equal to the time spent to perform the visit.

Missing proofs and other generalizations of the problems can be found into the final version of the paper.

Related Problem. The Steiner tree problem defined as follows shows some similarity to the k -SP problem.

Definition 2. Steiner Tree Problem

INSTANCE: a simple graph $G = (V, E)$, a weight function $w(e) \in N$ for each edge $e \in E$, a target subset $D \subseteq V$ of vertices.

TASK: find a minimum weight subtree of G that covers all vertices in D .

It is well known that this problem is NP-Complete ([ND12] in [4]). The k -SP problem differs from the Steiner tree problem because of different cost functions.

However, in Section 4, we show how an approximate solution to the k -SP can be derived from an approximate solution to a special variation on the Steiner tree problem.

Roadmap. In Section 2, we show that the k -SP problem is NP-complete for any value of k , even if we consider multi-stage directed graphs with all the edges having the same length. In Section 3.1, we present a polynomial-time algorithm k -SP-TREE for the special case of trees and in Section 3.3 we present an algorithm based on dynamic programming for the 1-USP problem on constant degree trees. In Section 4, we give approximation algorithms for the k -USP problem.

2 Hardness Result for Multi-stage Graphs

In this section we first prove that the 1-SP problem is NP-Complete even on 3-stage directed graphs by reducing Set Cover (SC for short) to this problem. Based on this, we also prove that for any k , the k -SP problem is NP-Complete on $(k + 2)$ -stage directed graphs. We further show similar results for undirected multi-stage graphs. The decisional version of k -SP problem is the following:

Definition 3. (Decisional k -Station Placement (k -DSP))

INSTANCE: (G, s, D, ℓ, B) where $G = (V, E)$ is a simple connected graph, $s \in V$ is the source, $D \subseteq V$ is a set of destinations, $\ell : E \rightarrow \mathcal{N}$ is a positive function, representing length of edges, and B is a positive integer.

QUESTION: is there a feasible solution $(S_1, \dots, S_k, \phi_1, \dots, \phi_k)$ to the k -SP problem on G such that $\text{Cost}(S_1, \dots, S_k, \phi_1, \dots, \phi_k) \leq B$?

We now briefly recall the definition of decisional-SC and p -stage graphs and then we show the reduction of decisional-SC to 1-SP problem.

Definition 4. Decisional Set Cover

INSTANCE: (T, C, B) where C is a collection of subsets of a finite set T and B is an integer.

QUESTION: is there a set cover for S (i.e., a subset $C' \subseteq C$ such that every element in S belongs to at least one member of C') of cardinality less or equal to B ?

The problem has been shown to be approximable within $1 + \ln |T|$ in [5] and not approximable within $(1 - \epsilon) \ln |T|$ for any $\epsilon > 0$ unless $\text{NPC} \subseteq \text{DTIME}(n^{\log \log n})$ in [3].

Definition 5 (p -Staged Graphs). A p -stage graph $G = (V, E)$ is a directed graph whose vertices can be partitioned into p sets V_1, \dots, V_p such that for every edge $(u, v) \in E$, $u \in V_i$ and $v \in V_{i+1}$ or vice-versa for some $i = 1, \dots, p - 1$. A weighted p -stage graph is a p -stage graph with weights on edges. A strong p -stage graph is a p -stage graph with edges directed from V_i to V_{i+1} for $i = 1, \dots, p - 1$.

Theorem 1. *1-DSP on weighted strong 3-stage graphs is NP-Complete even if all the edges have the same weight.*

Proof. Obviously, the 1-DSP problem is in NP. We reduced Decisional-SC to 1-DSP. Suppose $I = (T, C, B)$ is an instance of Decisional-SC in which $C = \{C_1, \dots, C_n\}$ and, for $i = 1, \dots, n$, $C_i \subseteq T = \{t_1, \dots, t_m\}$. We construct a quintuple $I' = (G, s, D, \ell, B')$ such that if I' belongs to 1-DSP then I belongs to Decisional-SC, where $G = (V, E)$ is a strong the following 3-stage graph:

$$V = \{s\} \cup \{C_1, \dots, C_n\} \cup \{t_1, \dots, t_m\}$$

$$E = \{(s, C_i) \mid i = 1, \dots, n\} \cup \{(C_i, t_j) \mid t_j \in C_i\}$$

$D = T$ and, w.l.o.g., we assume that for any $e \in E$, $\ell(e) = 1$.

Let (S^*, ϕ^*) be a 1-placement for G such that $Cost(S^*, \phi^*) \leq B + |D|$ for source s and set of destinations $D = \{t_1, \dots, t_m\}$.

Suppose, at first, that $S^* \subseteq C$, then, by definition, S^* is a set cover for C . Moreover, the cardinality of the set cover is less or equal to B :

$$B + |D| \geq Cost(S^*, \phi^*) = \sum_{v \in S^*} w(s, v) + \sum_{d \in D} w(\phi^*(d), d) = \sum_{v \in S^*} 1 + \sum_{d \in D} 1 = |S^*| + |D|.$$

Suppose, now, that $S^* \not\subseteq \{C_1, \dots, C_n\}$. Then, by the *strictness* property, either $S^* = \{s\}$ or S^* contains some vertex in $\{t_1, \dots, t_m\}$. In both cases, we show how to construct, in polynomial time, a new feasible 1-placement (S, ϕ) for G such that $S \subseteq C$ and $Cost(S, \phi) \leq Cost(S^*, \phi^*)$:

1. $S^* = \{s\}$: Notice that $Cost(S^*, \phi^*) = 2|D|$ because, for every $d \in D$, $w(s, d) = 2$. Define a new function $\phi : D \rightarrow \{C_1, \dots, C_n\}$ that associates, to every $d \in D$, a vertex C_i on one path from s to d . Let $S = \{\phi(d) \mid d \in D\}$ and consider the new 1-placement (S, ϕ) ; by construction

$$Cost(S, \phi) = |S| + |D| \leq 2|D| = Cost(S^*, \phi^*).$$

2. There exists a node t such that $t \in S^* \cap \{t_1, \dots, t_m\}$: Also in this case, we construct a new feasible solution by substituting t in S^* with one of its parent in C . As t has no outgoing edges, there is no destination vertex $d \in D$, different from t , such that $\phi^*(d) = t$ and thus the cost of this new placement is less or equal to the cost of (S^*, ϕ^*) . □

We can prove that the problem is NP-complete also for undirected graphs.

Theorem 2. *1-DSP on undirected weighted 3-stage graphs is NP-Complete.*

Moreover, by the known non-approximability results [3] of Set Cover we have the following corollary.

Corollary 1. *1-DSP on weighted 3-stage graphs is not approximable within $(1 - \epsilon) \ln |D|$ for any $\epsilon > 0$, unless $NP \subset DTIME(n^{\log \log n})$, even if all edges have the same weight.*

Lemma 1. *For all $i \geq 1$, i -DSP reduces to $(i + 1)$ -DSP.*

Proof. Let $\mathcal{I} = (G = (V, E), s \in V, D \subseteq V, \ell : E \rightarrow \mathcal{N}, B \in \mathcal{N})$ be an instance of the i -DSP problem. Construct graph $G' = (V \cup \{z_1, z_2\}, E \cup \{(z_1, s), (s, z_2)\})$ and let $\ell' : E' \rightarrow \mathcal{N}$ be the natural extension of ℓ to G' such that $\ell'(z_1, s) = |V| \sum_{v,u \in V} w(v, u)$ and $\ell'(s, z_2) > 0$. Consider the following instance of $(i + 1)$ -DSP problem: $\mathcal{I}' = (G', z_1, D \cup \{z_2\}, \ell', B + \ell'(z_1, s) + \ell'(s, z_2))$.

We now show how to derive a feasible solution $P = (S_1, S_2, \dots, S_i, \phi_1, \dots, \phi_i)$ to instance \mathcal{I} given a feasible solution $P' = (S'_1, S'_2, \dots, S'_{i+1}, \phi'_1, \dots, \phi'_{i+1})$ to instance \mathcal{I}' . By the *strictness* property we deduce that if $s \in S'_1$ then $S'_1 = \{s\}$ and, thus, given P' only the following cases can arise:

1. $S'_1 = \{s\}$: if there is only one station at the first level and this station is exactly s , restricting P' to graph G , we obtain a feasible solution P for \mathcal{I} .

2. $S'_1 \neq \{s\}$: we can construct a new feasible solution $P'' = (S''_1, S''_2, \dots, S''_{i+1}, \phi''_1, \phi''_2, \dots, \phi''_{i+1})$ to \mathcal{I}' , such that $S''_1 = \{s\}$ and, for every $v \in S'_2$, $\phi''_1(v) = s$. P'' is a feasible solution to \mathcal{I}' , and $Cost(P') \geq Cost(P'')$:

$$\begin{aligned}
 Cost(P') &= w(z_1, z_2) + \sum_{v \in S'_1} w(z_1, v) + \sum_{i=1}^k \sum_{v \in S'_{i+1}} w(\phi'_i(v), v) \\
 &= \ell'(z_1, s) + \ell'(s, z_2) + |S'_1| \ell'(z_1, s) + \sum_{v \in S'_1} (s, v) + \sum_{i=1}^k \sum_{v \in S'_{i+1}} w(\phi'_i(v), v) \\
 &\geq \ell'(z_1, s) + \ell'(s, z_2) + |S'_1| \ell'(z_1, s) + \sum_{v \in S'_2} w(\phi'_1(v), v) + \sum_{i=2}^k \sum_{v \in S'_{i+1}} w(\phi'_i(v), v) \quad (1) \\
 &\geq \ell'(z_1, s) + \ell'(s, z_2) + \sum_{v \in S'_2} w(s, v) + \sum_{i=2}^k \sum_{v \in S'_{i+1}} w(\phi'_i(v), v) = Cost(P'') \quad (2)
 \end{aligned}$$

where, to go from (1) to (2), we used the fact that $\ell'(z_1, s) \geq \sum_{v \in S'_2} w(s, \phi'_1(v))$. Restricting P'' to graph G , we have a feasible solution P for \mathcal{I} and

$$B + \ell'(z_1, s) + \ell'(s, z_2) \geq Cost(P'') = \ell'(z_1, s) + \ell'(s, z_2) + Cost(P).$$

□

From Theorem 1 and Lemma 1 we obtain:

Corollary 2. *k -DSP on weighted strong $(k + 2)$ -stage graphs is NP-Complete even if all edges have the same weight.*

From Theorem 2 and Lemma 1 we obtain:

Corollary 3. *The k -DSP problem is NP-Complete on $(k + 2)$ -stage graphs.*

Finally, by Lemma 1, Corollary 1, we have the following non-approximability result:

Theorem 3. *The k -DSP problem is not approximable within $(1 - \epsilon) \ln |D|$ for any $\epsilon > 0$ unless $\text{NP} \subseteq \text{DTIME}(n^{\log \log n})$.*

3 Optimal Placement on Trees

In this section we present polynomial-time algorithms for the k -SP problem and the k -USP on directed trees. We first present an algorithm for the 1-SP problem and, then, extend it to the general case $k > 1$. We then show a simple dynamic-programming algorithm for the 1-USP problem.

W.l.o.g., we can assume that the set D of destinations is the set of leaves in the tree T . Indeed, starting from T we can remove the leaves that are not in D and for any internal vertex $v \in D$, we can add a new leaf d_v , that becomes a new destination, and a new edge (v, d_v) with cost zero, obtaining a new tree T' . It is easy to see that solving the problem on T is equivalent to solve the problem on T' .

In the following, for any vertex v , we denote by $T(v)$ the subtree rooted at v , by $L(v)$ the set of leaves in $T(v)$ and by $p(v)$ the parent of v in T .

3.1 Optimal 1-Station Placement on Directed Trees

We present algorithm 1-SP-TREE for the 1-SP problem on a n -vertex tree T with source s and set of destinations D consisting of all the leaves of T .

The algorithm associates, in $O(n)$ time, a cost $c(u, v)$ to every edge (u, v) of the tree in the following way:

$$c(u, v) \stackrel{\text{def}}{=} w(s, v) + \sum_{d \in L(v)} w(v, d). \tag{3}$$

Referring to the k -hop multicasting example, we can think that $c(u, v)$ corresponds to the cost of multicasting to the vertices of $D \cap T(v)$ by placing one station at node v : the first term is the cost of sending the message from the source to v and the second is the cost of sending messages from v to the vertices of $D \cap T(v)$ without using any other intermediate station.

Next, algorithm 1-SP-TREE constructs a graph G by adding a new vertex t to T and by connecting the vertices of D to t using infinite-cost edges. Then the algorithm computes a minimum cut of G with respect to source s and sink t . Let C be the computed cut; the algorithm outputs placement (S_1, ϕ_1) such that: S_1 consists of all the vertices v such that the edge $(p(v), v)$ belongs to C ; function ϕ_1 assigns to each vertex $v \in D$ its closest ancestor that belongs to S .

Theorem 4. *Algorithm 1-SP-TREE, on input an n -vertex tree T , outputs an optimal solution to the 1-SP problem on T in time $O(M(n))$, where $M(n)$ is the running time of the fastest MIN-CUT algorithm on n vertex graphs.*

Proof. The analysis of the running time is obvious. Correctness follows from two observations: first, by construction, placement (S_1, ϕ_1) output by 1-SP-TREE is a feasible solution. Second, for every 1-placement $P = (S, \phi)$ on T , we can derive a legal cut C' for G such that $Cost(P) = Cost(C')$: cut C' is simply composed by the edges $(p(v), v)$ for every $v \in S$. The proof now simply follows by contradiction. \square

3.2 Optimal k -Station Placement on Trees

In the following, we say that, given a k -placement $P = (S_1, \dots, S_k, \phi_1, \dots, \phi_k)$, the subplacement of P with respect to a vertex $t \in S_i$, denoted by $P|_t$, is the restriction of P to $T(t)$. We start with the following lemma.

Lemma 2. *Let $P = (S_1, \dots, S_k, \phi_1, \dots, \phi_k)$ be an optimal k -placement for a tree T , then for every station $t \in S_1$ the subplacement $P|_t$ is an optimal $(k - 1)$ -placement for the subtree rooted in t .*

Proof. By the way of contradiction, assume that P is an optimal k -placement for a tree T and that there exists one vertex $t \in S_1$ for which $P|_t$ is not an optimal $(k - 1)$ -placement and, thus, there exists a new placement $P'|_t$ with lower cost. We can, thus, construct a new placement P' for T , substituting $P'|_t$ to $P|_t$, such that $Cost(P') < Cost(P)$ contradicting the hypothesis. \square

Algorithm k -SP-TREE works in k phases: the first phase computes optimal solution for the 1-SP problem for $T(v)$, for each vertex v ; phase $j > 1$, computes, for every vertex v , an optimal j -placement for $T(v)$ using the optimal $(j - 1)$ -placements computed at the previous phase. In details:

Phase 1: For every v in T compute an optimal 1-placement $P_1(v)$ for $T(v)$ and its corresponding cost. This is done by running algorithm 1-SP-TREE.

Phase $1 < j < k$: For every node v in T compute an optimal j -placement $P_j(v)$ for $T(v)$ by defining costs for every edge (x, y) in $T(v)$ in the following way: $c(x, y) = w(v, y) + Cost(P_{j-1}(y))$. Compute, then, a MIN-CUT on this subtree (notice that $Cost(P_{j-1}(y))$ has been computed for every y during the previous phase).

Phase k : Compute an optimal k -placement for T defining new costs for every edge (u, v) in T in the following way: $c(u, v) = w(s, v) + Cost(P_{k-1}(v))$. Compute, then, a MIN-CUT on T .

The correctness of algorithm k -SP-TREE follows directly from Lemma 2. Moreover observe that k -SP-TREE has to solve $O(k \cdot n)$ min-cut problem on n vertex graphs and, thus, its running time is $O(k \cdot n \cdot M(n))$.

Theorem 5. *Algorithm k -SP-TREE, on input a weighted tree with n vertices, a distinguished vertex s , a set of destinations D and an integer k , outputs a k -placement of minimum cost in time $O(k \cdot n \cdot M(n))$.*

3.3 Optimal 1-Unrestricted Station Placement on Trees

We will now present algorithm 1-USP-TREE based on dynamic programming to solve the 1-USP problem on a binary directed tree T where the source s is the root of T . For the sake of presentation, we assume the tree to be binary, but the results can be easily extended to constant degree trees. In the following we will say that a destination d is served by a station v if $\phi(d) = v$.

Notice, first, that an optimal 1-placement has an optimal substructure; in fact, with a proof analogous to the one of Lemma 2, we can prove that:

Lemma 3. *Given a tree T rooted in s and an optimal 1-placement OPT for T , let u_1 and u_2 be the children of s . Then, $OPT|_{u_1}$ and $OPT|_{u_2}$ are optimal 1-placements for $T(u_1) \cup \{s\}$ and $T(u_2) \cup \{s\}$, respectively, where the source is s for both placements and $L(u_1)$ and $L(u_2)$ are the destination sets, respectively.*

The proof is analogous to the one of Lemma 2.

We now define the value of an optimal solution recursively in terms of the optimal solutions to subproblems. A subproblem is defined as determining the cost of a placement in a subtree rooted in v placing no more than k stations and serving all the leaves of $T(v)$, except a given number λ .

Given tree T , let $\mathcal{C}(v, k, \lambda)$ be the minimum cost of serving destinations in $L(v)$, using at most k stations and knowing that there are exactly λ leaves of $T(v)$ that are served by some station placed in the up going path from v to s ; *i.e.*, these leaves will not be served by the k stations we will place in $T(v)$. We define $\mathcal{C}(v, k, \lambda)$ in the recursive following way:

If v is a leaf, then $\mathcal{C}(v, k, 1) = 0$, while for $\lambda \neq 1$, we have $\mathcal{C}(v, k, \lambda) = +\infty$.

If v is not a leaf, let u_1 and u_2 be its children. $\mathcal{C}(v, k, \lambda)$ is calculated choosing the cheapest solution between placing or not placing a station in v and looking for the optimal solutions for $T(u_1)$ and $T(u_2)$. This is done according to the following constraints:

1. if we place (respectively not place) a station in v , then we can not place more than $k - 1$ (resp. k) stations in the subtrees;
2. if we place a station in v , then this station will serve $f > 0$ leaves of $T(v)$, $f_1 \leq |L(u_1)|$ of them in $L(u_1)$ and $f_2 \leq |L(u_2)|$ in $L(u_2)$.
3. if $\lambda > 0$, then $\lambda = \lambda_1 + \lambda_2$ leaves are served by a station ancestor of v such that $0 \leq \lambda_1 \leq |L(u_1)|$ are in $T(u_1)$ and $0 \leq \lambda_2 \leq |L(u_2)|$ are in $T(u_2)$.

Now, let $\mathcal{C}_Y(v, k, \lambda)$ (respectively $\mathcal{C}_N(v, k, \lambda)$) be the cost of placing (resp. non placing) a station in v and placing $k - 1$ (resp. k) stations in the subtrees, knowing that $\lambda \geq 0$ leaves are served by an ancestor station. Then,

$$\mathcal{C}(v, k, \lambda) = \begin{cases} \min \{ \mathcal{C}_Y(v, k, \lambda), \mathcal{C}_N(v, k, \lambda) \} & \text{if } \lambda \leq |L(v)| \\ +\infty & \text{otherwise} \end{cases}$$

Both $\mathcal{C}_Y(v, k, \lambda)$ and $\mathcal{C}_N(v, k, \lambda)$ are the sum of two terms: the first counts how many times edges (v, u_1) and (v, u_2) are traversed; *i.e.*, edge (v, u_i) , $i = 1, 2$, is traversed once for every station placed in $T(u_i)$, once for every leaf in $L(u_i)$ served by a station ancestor of v and once for every leaf in $L(u_i)$ served by v .

The second is the recursive call on $T(u_1)$ and $T(u_2)$ with the proper value of the parameters. In details:

1. If $\lambda = 0$ then $\mathcal{C}_N(v, k, 0)$ is equal to:

$$\min_{0 \leq x + y \leq k} \{ \ell(v, u_1) \cdot x + \ell(v, u_2) \cdot y + \mathcal{C}(u_1, x, 0) + \mathcal{C}(u_2, y, 0) \}$$

2. If $\lambda \neq 0$ then $\mathcal{C}_N(v, k, \lambda)$ is equal to:

$$\min_{\substack{0 \leq x + y \leq k, \\ \lambda_1 + \lambda_2 = \lambda, \\ \lambda_i \leq |L(u_i)|, i = 1, 2}} \{ \ell(v, u_1)(x + \lambda_1) + \ell(v, u_2)(y + \lambda_2) + \mathcal{C}(u_1, x, \lambda_1) + \mathcal{C}(u_2, y, \lambda_2) \}$$

In fact, if we do not place a station in v and we do not serve any leaf of an ancestor station, we simply look for the best way to distribute up to k stations in the subtrees. If we serve some leaves of an ancestor station, we have to find the cheapest way to distribute these too.

Before giving the definition of $\mathcal{C}_Y(v, k, \lambda)$, we need the following lemma:

Lemma 4. *Given any optimal solution $OPT = (S, \phi)$ to the 1 – USP problem on binary tree T and a leaf l , there does not exist a vertex $v \in S$, different from $\phi(l)$, that belongs to the unique path going from $\phi(l)$ to l .*

It is important to notice that the previous Lemma does not state that two stations cannot be on the same root-leaf path, but only that, in the optimal solution, leaves are served by the closest station.

As a consequence of the Lemma 4, we define $\mathcal{C}_Y(v, k, \lambda) \neq +\infty$ only when $\lambda = 0$. Thus, if $\lambda = 0$, $\mathcal{C}_Y(v, k, 0)$ is equal to

$$\min_{\substack{0 \leq x + y \leq k - 1, \\ f_1 + f_2 > 0, \\ 0 \leq f_i \leq |L(u_i)|, i = 1, 2}} \{ \ell(v, u_1)(x + f_1) + \ell(v, u_2)(y + f_2) + \mathcal{C}(u_1, x, f_1) + \mathcal{C}(u_2, y, f_2) \}$$

In fact, if we place a station in v we only have up to $k - 1$ station to place in the subtrees and we have to find most convenient set of leaves of $T(v)$ to be served by v .

Finally, the cost of an optimal solution OPT for tree T rooted in s is calculated in the following way:

$$Cost(OPT) = \mathcal{C}(s, n, 0).$$

To recover OPT , it is sufficient to remember the vertices in which we placed the stations and this gives us set S . Function ϕ is easily determinate using Lemma 4: given leaf l , $\phi(l)$ is the first vertex of S we find on the upgoing path from l to the root of the tree.

Theorem 6. *Algorithm 1-USP-TREE on binary trees runs in $O(n^5)$.*

Proof. For every vertex v in T and each of the $O(n^2)$ pairs (x, y) such that $x + y \leq n$ we have to consider the $O(n^2)$ pairs (λ_1, λ_2) such that $\lambda_1 + \lambda_2 \leq n$. and the $O(n^2)$ pairs (f_1, f_2) such that $f_1 + f_2 > 0$ and $f_1, f_2 \leq n$. □

4 Approximation Algorithm on General Graphs

In Section 2 we have shown that the k -SP problems is NP-Complete. This implies the following:

Corollary 4. *The k -DUSP problem is NP-Complete for every k .*

In this section we show an approximation algorithm for this problem on general graphs. The key idea of the algorithm is to reduce the k -USP to the problem of computing a Steiner tree with bounded depth on a graph. Let $K_{|V|}$ be the complete graph over $|V|$ vertices in which the weight of edge (x, y) is the cost of the shortest path from x to y in the graph G . The cost of the $(k+1)$ -depth Steiner minimum tree of $K_{|V|}$ is equal to the cost of the optimal solution of the k -USP problem on G . For the sake of presentation, we will present only the case $k = 1$, but similar arguments can be used for the case $k > 1$.

Lemma 5. *Let T be a Steiner tree rooted at s for graph $K_{|V|}$ with depth at most 2 and target D . There exists a 1-placement P for the graph G with source s on destination set D such that $Cost(P) = Cost(T)$.*

Proof. The placement P is constructed as follows: the set of stations consists of the vertices at level 1 in the tree T . For each vertex in $v \in D$, $\phi_1(v)$ is the parent of v in the tree T . □

Using a dual argument it is possible to prove the following:

Lemma 6. *Let P be a 1-placement for the graph G on destination set D and source s . There exists a Steiner tree T rooted at s with maximum height 2 on the clique $K_{|V|}$ with target D such that $Cost(P) = Cost(T)$.*

Lemma 7. *Let P^* be an optimal 1-placement for a graph G on destination set D and let T^* be a minimum Steiner tree on the complete graph of shortest paths in G . It holds that:*

$$Cost(P^*) = Cost(T^*)$$

Proof. Assume, by contradiction, that $Cost(P^*) < Cost(T^*)$. By Lemma 6, it is possible to construct a new Steiner tree T' on $K_{|V|}$ with destination D such that $Cost(T') = Cost(P^*) < Cost(T^*)$. But this contradicts the hypothesis that T^* was a minimum Steiner tree. A dual argument can be used to prove that if $Cost(P^*) > Cost(T^*)$, then P^* is not an optimal placement.

Since the problem of computing a Minimum Steiner tree is NP-Complete, we use an approximation algorithm for this problem in order to obtain an approximation algorithm for the k -USP. We recall the following results by Kortsarz and Peleg:

Theorem 7. [6] *Let $G = (V, E)$ be a graph and let $D \subseteq V$ be a set of destinations. There is an approximation algorithm for the minimum Steiner tree problem on G with destination D and maximum depth d with approximation ratio $O(\log |D|)$, for any constant d , and $O(|D|^\epsilon)$, for any $\epsilon > 0$, for general d .*

Given the discussion above, the following theorems can be easily proven:

Theorem 8. *For any constant k , there exists approximation algorithm for the k -USP with optimal approximation ratio $O(\log |D|)$.*

Corollary 5. *For any k , and for any $\epsilon > 0$ there exists an $O(|D|^\epsilon)$ approximation algorithm for the k -USP.*

5 Open Problems

The immediate open problem left in our work is the design of better approximation algorithms for general graphs and d . Also, we do not know of any natural class of graphs (other than trees) for which the problem can be solved efficiently.

From a more combinatorial point of view it would be interesting to ask if there exists a function $f(\cdot)$ such that for all trees with n nodes the cost of the best $f(n)$ -SP is $O(n)$. It is obvious that any function $f(n) = \Omega(n)$ will do (just put a station in any vertex). We can show that $f(n) = \log^* n$ works for complete binary trees but could not extend this result to general trees.

Acknowledgments

We would like to thank the anonymous referee for his useful comments used to improve the non-approximability result.

Bibliography

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti Spaccamela and M. Protasi, *Complexity and Approximations - Combinatorial Optimization Problems and their Approximability Properties*, Springer-Verlag, 1999.
2. J. Bar-Ilan and G. Kortsarz and D. Peleg, *Generalized Submodular Cover Problem and Applications*, Proc. 4th Israel Symposium on Theory of Computing and Systems, pp. 110-118, 1996.
3. U. Feige, *A threshold of $\ln n$ for approximating set cover*, Proc. 28th ACM Symposium on Theory of computation, pp. 314-318, 1996.
4. M.R. Garey and D.S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979, New York.
5. D. S. Johnson, *Approximation Algorithms for Combinatorial Problems*, J. Computers and System Sciences, Vol 9, pp. 256-278, 1974.
6. G. Kortsarz and D. Peleg, *Approximating the Weight of Shallow Steiner Tree*, Proc. SODA 97, 1997.
7. J. Plesnık, *The Complexity of Designing a Network with Minimum Diameter*, Networks, Vol 11, pp. 77-85, 1981.