An Approximation Algorithm for the Directed Telephone Multicast Problem *

Michael Elkin [†‡] Guy Kortsarz[§]

Abstract

Consider a network of processors modeled by an n-vertex *directed* graph $G = (V, E)$. Assume that the communication in the network is synchronous, i.e., occurs in discrete "rounds", and in every round every processor is allowed to pick one of its neighbors, and to send him a message. A set of terminals $\mathcal{T} \subseteq V$ of size $|\mathcal{T}| = k$ is given. The *telephone k-multicast* problem requires to compute a schedule with minimal number of rounds that delivers a message from a given single processor, that generates the message, to all the processors of $\mathcal{T}$. The processors of $V \setminus \mathcal{T}$ may be left uninformed.

The telephone multicast is a basic primitive in distributed computing and computer communication theory. In this paper we devise an algorithm that constructs a schedule with $O(\log k \cdot b^* + k^{1/2})$ rounds for the *directed k-multicast* problem, where b^* is the value of the optimum solution. This is the first algorithm with a non-trivial approximation guarantee for this problem.

We show that our algorithm for the directed multicast problem can be used to derive an algorithm with a similar ratio for the *directed Steiner poise* problem, that is, the problem of constructing an arborescence that spans a collection $\mathcal{T}$ of terminals and has the minimum *poise*.

*A preliminary version of this paper was published in ICALP 2003, [EK03p].

[†]Department of Computer Science, Yale University, New Haven, CT, USA, 06520-8285, elkin@cs.yale.edu.

[‡]This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795.

[§]Computer Science department, Rutgers University, Camden, NY, USA. Email: guyk@crab.rutgers.edu

1 Introduction

1.1 The Problem and Previous Research

Consider a network of processors modeled by an n-vertex graph $G = (V, E)$. The input also contains a subset $\mathcal{T} \subseteq V$ of k nodes called *terminals*. Assume that the communication in the network is synchronous, i.e., occurs in discrete "rounds", and in every round every processor is allowed to pick one of its neighbors, and to send him a message. The *telephone k-multicast* problem requires to compute a schedule with minimal number of rounds that delivers a message from a given single processor s, that generates the message, to all the processors of $\mathcal{T}$. The processors of $V \setminus \mathcal{T}$ may be left uninformed. The case $\mathcal{T} = V$ is called the *broadcast* problem.

The telephone multicast and broadcast are basic primitives in distributed computing and computer communication theory, and are used as building blocks for various more complicated tasks in these areas (see, e.g., [HHL88]). The optimization variants of the multicast and broadcast primitives were intensively studied during the last decade. Particularly, a long list of gradually improving approximation algorithms were devised for the *undirected multicast* problem [KP95, R94, BGN+98, EK03].

Devising an approximation algorithm for the *directed multicast* problem is an open question posed by Ravi in his paper [R94]. Recently the authors of the current paper devised a logarithmic ($O(\log n)$) approximation algorithm for the *directed broadcast* problem. However, until now no approximation algorithm that provides a non-trivial (that is, $o(k)$) approximation guarantee for the more general *directed multicast* problem was known.

1.2 Our Results

In this paper we devise an approximation algorithm for the directed multicast problem that given an instance that admits a schedule of length b^* returns an addmisable schedule of length $O(b^* \cdot \log k + \sqrt{k})$. (See Section 1.5 for the formal definition of the notion *addmisable schedule*.) In other words, our algorithm has a *multiplicative approximation guarantee* of $O(\log k)$, and an *additive approximation guarantee* of $O(\sqrt{k})$. Such an approximation guarantee will be henceforth termed as $(O(\log k), O(\sqrt{k}))$ approximation guarantee, with O-notations omitted for convenience.

Note that for the special case of $br^* = \Omega(\sqrt{k})$ (for example, graphs with high diameter) the ratio is $O(\log k)$. It is easy to show that for $k = \Omega(n^\epsilon)$ with $\epsilon > 0$ a (possibly small) constant, and $br^* = \Omega(\sqrt{k})$, the achieved ratio (that is, $O(\log k)$) is the best possible up to a constant factor, unless $P = NP$. The running time of our algorithm is $\tilde{O}(|E||V|)$ (the notation $\tilde{O}$ ignores polylogarithmic factors).

1.3 Additional Results

The postal model: In the more general *postal model* (introduced in [BGN+98]) each vertex v has a delay number $0 \leq \mu(v) \leq 1$. The vertex sending a message is "busy" only at the first μ time units starting from the initial sending time. After μ time units, v is free to send another message. In addition, every edge e has a delay number ℓ_e representing the time required to send the message over e. It is argued in [BGN+98] that this more general model is of a particular practical importance. We extend our algorithm and its analysis to this more general problem, and show that this problem admits the same approximation ratio.

The poise problem:

For an undirected rooted tree T, let $h(T)$ denote its *depth* (the maximum unweighted distance between a vertex and the root), and $maxdeg(T)$ denote its *maximum degree*. The *poise* of T, denoted $poise(T)$, is the sum of its depth and its maximum degree. For an undirected graph $G = (V, E)$, its *poise* is the minimum poise of a spanning tree T of G. A *Steiner poise* of the graph G with respect to a subset $\mathcal{T} \subseteq V$ of vertices is the minimum poise of a spanning tree $T \subseteq E$ that spans $\mathcal{T}$.

The notion of poise was introduced by Ravi [R94]. It was shown there that the problems of computing the poise and the Steiner poise of a graph (henceforth, the *poise* and *Steiner poise* problems) are closely related to the telephone broadcast and multicast problems, respectively. Ravi has also devised an $O(\frac{\log^2 k}{\log\log k})$-approximation algorithm for the Steiner poise problem, which was consequently improved to $O(\log k)$ by Bar-Noy et al. [BGN+98], and further improved by the authors to $O(\frac{\log k}{\log\log k})$ in [EK03].

In our opinion, when studying the directed broadcast and multicast problems, it is natural to study the *poise* and the *Steiner poise* problems as well. Formally, let T be a (directed) (out-)arborescence with vertex set V, that is, a directed acyclic graph, connected in the undirected sense, and such that there exists a (unique) vertex v such that for every other vertex $w \in V \setminus \{v\}$ there exists a (unique) path P in T from v to w. The *depth* of T is the maximum distance $\max\{dist(v, w, T) \mid w \in V \setminus \{v\}\}$ from this special vertex v (called the *root* of the arborescence) to some other vertex. The *poise* of T is the sum of its depth and maximum out-degree. For a digraph G, the poise of G is the minimum poise of a spanning out-arborescence T of G. For a digraph $G = (V, E)$ and a subset $\mathcal{T} \subseteq V$ of terminals, the *Steiner poise* of G with respect to $\mathcal{T}$ is the minimum poise of an out-arborescence $T \subseteq E$ that spans the set $\mathcal{T}$.

A logarithmic approximation algorithm for the *directed poise* problem was devised by the authors in [EK02], but no algorithm that provides a non-trivial approximation guarantee for the *directed Steiner poise* problem was known prior to this paper. In this paper we devise an algorithm that given a digraph $G = (V, E)$ and a subset $\mathcal{T} \subseteq V$ of vertices for which there exists a spanning out-arborescence T of depth h^* and maximum out-degree d^*, constructs a spanning out-arborescence of depth $O(h^*)$ and maximum out-degree $O(\log k \cdot d^* + \sqrt{k})$. Consequently, this algorithm provides a $(\log k, \sqrt{k})$*-approximation* for the *directed Steiner poise* problem.

Related work: Furer and Raghavachari [FR90] devised an $O(\log n)$-approximation algorithm for the problem of constructing a spanning Steiner tree of minimum out-degree for a directed graph, and Krishnan and Raghavachari [KR01] devised an *additive* $O(\log n)$-approximation algorithm for the same problem; the latter algorithm has, however, a super-polynomial running time of $n^{O(\log n)}$.

The minimum outdegree Steiner tree problem: The paper of Furer and Raghavachari does not provide a result for the case of a Steiner (as opposed to spanning) tree. It follows that even under a weaker approximation goal, namely designing a Steiner arborescence minimizing the maximum outdegree (with no bounds on the height) no approximation was known previous to the result presented here.

1.4 Overview of the Algorithm

Let s be the root that multicasts the message. The multicast schedule that we define can be roughly divided into two parts. In the first part of the schedule, the goal is to construct a partition of the vertex set V into two disjoint subsets I and U. The partition has to satisfy certain properties. The set of vertices that got the message will be referred to as *informed* vertices. Throughout, U will contain only uninformed vertices. While I may contain at some stages of the algorithm uninformed vertices, it should be possible to multicast to I from s "with relatively few rounds". This is the intuition behind the chosen letter I (I for informed).

Let $UT = U \cap \mathcal{T}$. The second required property is that no U vertex is allowed to have in its vicinity "many" uninformed terminals (vertices of the set UT). These vicinities are defined with respect to the graph $G(U)$ induced by the vertex set U. These properties will be captured formally by the notion of $\sqrt{k}$ degree-depth partition, that will be defined later on. In Section 2 we describe the algorithm for constructing the partition with the desired properties. The construction is done by a greedy partitioning technique that uses minimum distance arborescences.

The second stage informs UT after I is informed. We now explain how to complete the multicast under the assumption that the I vertices are informed. This stage is also partitioned into two steps. In the first step, the schedule informs a subset $D \subseteq U$ of vertices, that satisfies the condition that every vertex $u \in UT \setminus D$ has a nearby vertex $w(u) \in D$. In order to inform D we reduce this task to a certain generalization of the set-cover problem, that we call *multiple set-cover* problem. After D is informed we use a collection of shortest path arborescences with roots belonging to D to complete the broadcast process informing UT. In the analysis, the properties of $\sqrt{k}$ degree-depth partitions are used.

1.5 Preliminaries

The input of the directed multicast problem consists of a directed graph $G(V, E)$, a source vertex s and a collection $\mathcal{T} \subseteq V$ of terminals. Note that $s \notin \mathcal{T}$, namely, s is not a terminal. Throughout the execution of the broadcast protocol, each vertex $v \in V$ either is informed or uninformed. The vertex set is always disjointedly partitioned into $V = I \cup U$ so that all the U vertices are uninformed. We denote $UT = U \cap \mathcal{T}$. At the beginning of the the execution $I = \{s\}$, $U \leftarrow V \setminus \{s\}$ and $UT \leftarrow \mathcal{T}$.

A *round* is a *directed matching* of a subset $X \subseteq I$ to a subset $Y \subseteq U$. Thus, the matching edges are directed from X to Y. After the round, the vertices of Y become informed, and $I \leftarrow I \cup Y$, $U \leftarrow U \setminus Y$, $UT \leftarrow UT \setminus Y$. A *schedule* is a sequence of rounds. The *length* of a schedule is the number of rounds it contains. A schedule that informs all the terminals of $\mathcal{T}$ is called *admissible.* The objective of the problem is to construct an admissible schedule of minimum length.

Throughout the paper we denote $|\mathcal{T}| = k$, and the optimum value for the instance at hand is denoted by b^*. As the number of informed vertices can at most double at every round, $b^* \geq \lceil \log_2 k \rceil$. In addition, since at every round we can inform at least one additional vertex, we may assume that $b^* \leq n - 1$. We assume that the value of b^* is known and can be used by the algorithm. Indeed, the correct value of b^* can be found by repeated doubling starting from minimum possible value of $\lceil \log_2 k \rceil$.

Given a subgraph G', the distance (minimum number of edges in a shortest path) between u and v is denoted by $dist(u, v, G')$. Given an arborescence T, its depth (the largest distance from a root to a leaf) is denoted by $h(T)$. The set of neighbors of u (vertices at distance 1 from u) is denoted by $N(u)$.

The graph induced by a set of nodes U is denoted by $G(U)$.

We denote by $deg(v)$ the out-degree of a vertex v. (The notation $deg(v)$ is used, and not, say, $deg_{out}(v)$ as we are mainly interested in the out-degree of vertices in this paper). The out-degree of v in a subgraph G' is denoted by $deg(v, G')$.

For a vertex v and positive integer $\ell = 1, 2, \ldots$, the *ℓ-out-neighborhood* in G is the vertex set $\{u \mid dist(v, u, G) \leq \ell\}$.

A leaf in a rooted arborescence is a vertex that has no children.

An (r, t) approximation algorithm for a minimization problem P is an algorithm that given an instance α of P returns a solution of value at most $r \cdot opt(\alpha) + t$, where $opt(\alpha)$ is the value of the optimal solution of the problem P on the instance α.

The busy schedule procedure: One of the tools that are used in our multicast procedure is the well known busy schedule. The busy schedules constitute a class of schedules. In a busy schedule, an informed node u considers its set of neighbors. If there exists an uninformed neighbor of u, then u chooses an *arbitrary* uninformed neighbor and sends it the message. Such a schedule is also called *non lazy* (this terminology is due to [BGN+98]).

Thus, the busy schedule is the simple greedy strategy that essentially makes sure that whenever possible no informed vertex is "idle". Our future claims hold for any schedule that uses some busy schedule.

Finally, we use the following lemma. A very similar lemma is proven in [EK02], and the proof here is provided for completeness.

Lemma 1.1 [EK02] *Let Q be an arborescence rooted at s with leaf set L and depth h. Assume that s knows the message. Then the busy schedule is a multicast scheme to all the vertices in Q with no more than $h + |L|$ rounds.*

Proof: Let ℓ be a leaf, v be the closest to ℓ informed ancestor of ℓ, and u be the next vertex on the path from v to ℓ. We divide all the rounds into two types. Either v sends the message to u, in which case the round is called a round of type one (thus, a round of type one brings the message one edge closer to ℓ), or v can send the message to a sibling w of u, in which case we say that this is a round of type two for ℓ. Let B be the collection of vertices having at least 2 children in T.

There are at most h rounds of type one. The number of possible rounds of type two is at most $\sum_{v \in B(T)}(deg(v) - 1) \leq |L|$. The claim follows. ∎

Troughout the paper, $\log n$ stands for $\log_2 n$.

2 Constructing the partition and informing the vertices of I

2.1 Computing a $\sqrt{k}$ partition

We start with the definition of the $\sqrt{k}$ degree-depth partition. Recall that it is assume that b^* is the minimum number of rounds for the multicast on the input at hand, and is given as a parameter to the algorithm.

Definition 2.1 *A* $\sqrt{k}$ degree-depth partition *(henceforth,* $\sqrt{k}$ partition*) is a partition of* V *into two disjoint and nonempty sets* $V = I \cup U$, $I \cap U = \emptyset$, $U \neq \emptyset$, $s \in I$, *such that in* $G(U)$ *the* b^*-*neighborhood of any node contains at most* $\sqrt{k}$ *terminals in* $U \cap \mathcal{T} = UT$.

Intuition: To understand why a $\sqrt{k}$ partition is useful, assume that we are given a $\sqrt{k}$ partition, the I vertices are all informed and that the remaining goal is to inform UT.

In Section 3.2 it will be shown that a subset D of U that has the property that each terminal $u \in UT$ has a nearby vertex $v \in D$ can be quickly informed (the distance is measured in $G(U)$).

Then, we define a collection of arborescences in $G(U)$ and use it to send the message to the uninformed terminals of UT via this collection. For each vertex $u \in D$, let T_u denote the arborescence rooted in u. The sets of leaves of each multicast arborescence T_u are contained in the set $UT \setminus D$. The $\sqrt{k}$ partition properties are used to guarantee that the trees T_u are relatively shallow, and that each of them has at most $\sqrt{k}$ leaves.

Procedure $Comp - Par$**:** The following procedure starts with $U = V \setminus \{s\}$ and $I = \{s\}$, and moves vertices from U to I until it reaches a $\sqrt{k}$ degree-depth partition (U, I).

We say that $u \in U$ is a $\sqrt{k}$-*bad* vertex if its b^*-out-neighborhood in $G(U)$ contains more than $\sqrt{k}$ terminals.

Input: A graph $G = (V, E)$, a set $\mathcal{T}$ of terminals, and a source $s \in V$.

Output: A $\sqrt{k}$ partition (I, U) of V, a set *Roots*, a collection of trees $\{T_u\}$, T_u rooted at $u \in Roots$.

1. $I \leftarrow \{s\}$; $\quad U \leftarrow V \setminus \{s\}$;
2. $Roots \leftarrow \emptyset$; $\quad$ /* The roots are later used to inform the new vertices of I. */
3. **While** $G(U)$ contains a $\sqrt{k}-$bad vertex u **Do:**
   (a) Let $Cl(u)$ be the $\lfloor \sqrt{k} \rfloor$ uninformed terminals closest to u, in $G(U)$.
   (b) Let T_u be the shortest path arborescence leading from u to $Cl(u)$ in $G(U)$.
   (c) $I \leftarrow I \cup V(T_u)$, $\quad U \leftarrow U \setminus V(T_u)$, $\quad Roots \leftarrow Roots \cup \{u\}$.
4. Output $(I, U, Roots)$

Properties of the algorithm: The following claim is derived directly from the algorithm.

Claim 2.2 *The pair* (I, U) *output by Procedure* $Comp - Par$ *is a* $\sqrt{k}$ *partition.*

Proof: Follows as the loop in Line 3 stops only if there are no $\sqrt{k}-$bad vertices. ∎

Claim 2.3 *Suppose that* $k \geq 4$. *Then the set Roots has cardinality* $|Roots| \leq \sqrt{k} + 2$.

Proof: Every time a root u is added to *Roots* in Line 3c at least $\sqrt{k} - 1$ vertices (the leaves of T_u) are removed from UT. As the initial size of U is $|\mathcal{T}| = k$, the claim follows for $k \geq 4$. ∎

Claim 2.4 *It is possible to use $2b^* + 2\sqrt{k} + 2$ rounds or less in order to inform I. Further, the partition (I, U) is a $\sqrt{k}$ degree-depth partition.*

Proof: We may assume that each vertex in $V \setminus \{s\}$ has distance at most b^* from s, as vertices of larger distance from s cannot be used in an optimum schedule and may be discarded.

First, form a shortest path arborescence leading from s to $Roots$. It follows from Claim 2.3 and the above discussion that such an arborescence has depth at most b^*, and at most $\sqrt{k} + 2$ leaves. By Lemma 1.1, the busy schedule informs $Roots$ in $b^* + \sqrt{k} + 2$ rounds.

Once all the vertices of the set $Roots$ are informed, we can use the collection of arborescences $\{T_u\}$ as defined in Line 3b in Procedure $Comp - Par$. Each vertex u has a arborescence T_u associated with it, having exactly $\lfloor \sqrt{k} \rfloor$ leaves, and each leaf is a terminal. Observe that the arborescences are vertex-disjoint. This follows as in Line 3c of Procedure $Comp - Par$ the vertices of T_u are removed from U and so the next arborescence T_w is computed over a subgraph $G(U)$ not containing $V(T_u)$.

It follows that all the vertices in $Roots$ can broadcast over T_u *in parallel*. By Lemma 1.1, this broadcast requires $b^* + \sqrt{k}$ rounds. In total, the number of rounds is at most $2b^* + 2\sqrt{k} + 2$. Moreover, by Claim 2.2 the current set I of informed vertices output by the algorithm induces a $\sqrt{k}$ degree-depth partition. ∎

3 From a $\sqrt{k}$ degree-depth partition to a complete schedule

3.1 An algorithmic tool: the multiple set-cover problem

Consider the following problem that we call the *multiple set-cover* problem. Let $G(V_1, V_2, E)$ be a bipartite graph. A set $S \subseteq V_1$ is called a *set-cover* of V_2 if $N(S, G) = V_2$, namely, every $v_2 \in V_2$ has at least one neighbor in S. The minimum set-cover problem is to find a set-cover $S \subset V_1$ of V_2 with minimum cardinality.

The multiple set-cover problem is defined as follows.

Input: A bipartite graph $G(V_1, V_2, E)$ with $|V_1| + |V_2| = n$. The set V_1 is partitioned into a disjoint union of sets $V_1 = \bigcup_{j=1}^{d} A_j$.

Output: A set cover $S \subset V_1$ of V_2.

Definition 3.1 *For a cover S, let the* value *of S be $val(S) = \max\{|S \cap A_i|\}_{i=1}^{d}$.*

Optimization goal: Minimize $val(S)$.

Observe that the usual set-cover problem is a special case of the multiple set-cover problem with $d = 1$. This problem is called the set-cover problem with groups constraints in [CK04].

The maximization version of this problem is defined as follows. Given the same input, and cardinality bounds k_i for each A_i, select at most k_i sets from every A_i, and maximize the number of elements covered. This maximization version was studied by Chekuri and Gupta [CK04], and they called it the "maximum coverage with group budgets constraints" (MCG) problem. They devised a 2-approximation algorithm for MCG.

The first approximation ratio for the multiple set-cover problem was proved by us using randomized rounding in the preliminary version of this paper [EK03p], and it was $O(\log(|V_1| + |V_2|)$.

In [CK04], this bound was improved to $\log|V_2|+1$ using the 2-approximation algorithm for MCG as follows. Guess (search for) the optimum cost v^* for the multiple set-cover instance at hand. Create an instance of the maximum coverage with group budgets constraints problem by imposing a uniform budget bound v^* on every A_i. Clearly, there is a feasible solution for the resulting maximum coverage with group constains problem, that covers all the elements. Since [CK04] provide a 2-approximation algorithm for MCG, the solution derived by their algorithm covers at least half of the elements.

Run the algorithm for MCG until all elements are covered. Since each time the solution chooses $O(v^*)$ sets per A_i, the desired $(\log|V_2|+1)$-approximation for the multiple set-cover problem follows.

Theorem 3.2 [CK04] *The multiple set-cover problem admits a polynomial time greedy* $(\log|V_2|+1)$*-approximation algorithm.*

Next, we show how to reduce a special case of the directed multicast problem to the multiple set-cover problem.

3.2 Informing a distance$-b^*$ dominating set D

A *dominating set* D in a directed graph G is a set D so that $D \cup N(D) = V$. A *distance*$-b^*$ *dominating set* in G is a set D that satisfies that for every $v \in V$ there is a vertex $u \in D$ such that $dist(u,v) \leq b^*$. Finally, D is a distance$-b^*$ dominating set with respect to a subset W if the above condition holds for every vertex in W (that is, the vertices of W are on distance at most b^* from D, but the vertices from $V \setminus W$ may be at a larger distance from D).

Let I, U be a disjoint partition of V such that the source s belongs to the set I. Suppose that all the vertices of the set I are already informed. It remains, therefore, to inform the vertices of UT. We relax this problem as follows. Let $P_{I,U}$ be the problem of informing some distance$-b^*$ dominating set D of UT using minimum number of rounds.

We now describe a procedure *Dist_Dom* that uses the solution for the multiple set-cover problem to approximate the $P_{I,U}$ problem.

Procedure ***Dist_Dom*****:** Consider the following bipartite graph $\mathcal{B}_{I,U}(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$. Let

$$\mathcal{V}_1 = \{x_{(v,u)} \mid \quad v \in I,\ u \in U,\ (v,u) \in E\}\ .$$

Intuitively, the set $\mathcal{V}_1$ is the set of edges (v,u) of the original graph G, with v in I, and u in U. Observe that, in a sense, a vertex $u \in U$ may be "duplicated" many times. That is, if the vertex u has d incoming edges, it appear in d different vertices $x_{v,u}$.

Let $A_v = \left\{x_{(v,u)} \mid \quad u \in N_G(v)\right\}$. The disjoint partition of $\mathcal{V}_1$ that is provided as input for the multiple set-cover problem is $\mathcal{V}_1 = \bigcup_{v \in V} A_v$.

The set $\mathcal{V}_2$ is set to $\mathcal{V}_2 \leftarrow UT$ (the set of terminals that are still uninformed).

We now define the edge set $\mathcal{E}$ of the graph $\mathcal{B}_{I,U}(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$. There is an edge between a vertex $x_{(v,u)}$ to a vertex $w \in UT$ if there is a directed path of length at most b^* from v to w in $G(U)$. See Figure 1 for an illustration.

Lemma 3.3 *The multiple set-cover instance* $\mathcal{B}$ *admits a solution* $S^* \subseteq \mathcal{V}_1$ *such that* $val(S^*) \leq b^*$.

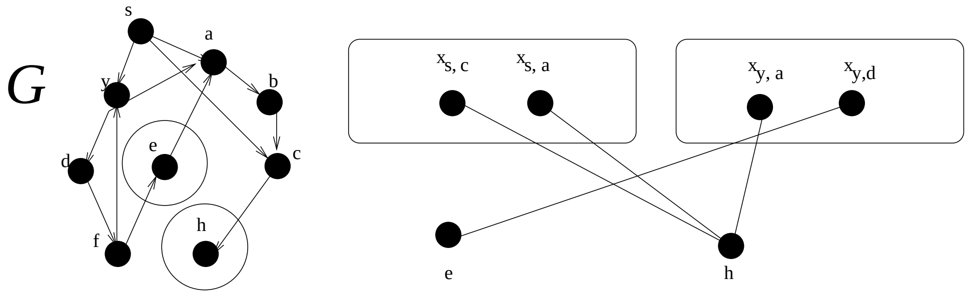

Figure 1: An example of the application of the multiple set-cover problem. The terminals e and h are depicted by cycles around the vertices. It is easy to see that for the graph G, $b^* = 4$. The disjoint partition of the vertex set V into $I \cup U$ that is obtained after one round is $I = \{s, y\}$ and $U = V \setminus I$. This partition induces the bipartite graph $B_{I,U}$ that is depicted on the right side. For example, the vertex $x_{s,a}$ is connected to h as the distance between a and h in the graph $G(U)$ induced by U is $3 \leq b^*$. In contrast, the respective distance of h from d is greater than b^*. Consequently, there is no edge between $x_{y,d}$ and h in the bipartite graph $B_{I,U}$.

Proof: Let T^* be the optimum directed multicast arborescence with respect to the source s and the set $\mathcal{T}$ of terminals. Note that the maximum degree of a vertex in T^* is at most b^*. For every $w \in UT$ let $v = v(w)$ be the last vertex in the path from s to w in T^* that belongs to I. Observe that this vertex is well-defined as the path starts with $s \in I$ and ends in $w \in U$. Let $u = u(w)$ be the next vertex in the path. Observe that $v = par(u)$ (v is the parent of u in T^*) and $u \in U$.

Procedure *Dist_Dom* forms the solution S^* for the instance $\mathcal{B}_{I,U}$ of the multiple set-cover problem in the following way. For every $w \in UT$, add the vertex $x_{(par(u(w)),u(w))}$ into the solution S^*. We claim that S^* is a feasible solution for the instance $\mathcal{B}$ of $P_{I,U}$ with value $val(S^*) \leq b^*$. To see that, first observe that the multicast value of T^* is at most b^*. It follows that $dist(u, w, T^*) \leq b^*$. Next, by definition of $v(w)$, the path from $u(w)$ to w is fully contained in $G(U)$. Hence, by definition, $dist(u, w, G(U)) \leq b^*$ and the edge $(x_{(v,u(w))}, w)$ is in $\mathcal{E}$. Thus, the set S^* is a feasible solution for the instance $\mathcal{B}$ of the multiple set-cover problem.

Next, we show that $val(S^*) \leq b^*$.

Consider $|S^* \cap A_v|$ for some informed vertex $v \in I$. Observe that by construction, $x_{(v,u)} \in S^*$ if and only if u is a child of v in T^*.

It follows that $|S^* \cap A_v| \leq deg(v, T^*)$. Finally, $deg(v, T^*) \leq \max\{deg(z, T^*) \mid z \in V(T^*)\} \leq b^*$. Thus for every $v \in I$,

$$|S^* \cap A_v| \leq deg(v, T^*) \leq b^*,$$

as required. Lemma 3.3 follows directly from the above discussion. ∎

Now, assume that we compute the instance $\mathcal{B}_{I,U}$ of the multiple set-cover instance of I and U as described above. By Theorem 3.2 there exists a polynomial-time $\log k + 1$-approximation algorithm for the multiple set-cover problem. By this theorem and Lemma 3.3 the algorithm returns a solution D for the multiple set-cover problem with value $O(b^* \log k)$.

If $x_{v,u}$ is chosen into the cover, we say that the vertex u is assigned to the vertex v. Observe that a vertex $u \in U$ can be choosen many times in the solution. In other words, it may happen that several vertices $x_{v_1,u}, x_{v_2,u}, \ldots$ are in the solution. In this case it is enough to delete all variables $x_{v_i,u}$ except for one.

Lemma 3.4 $\max_{v \in V_1} |D \cap A_v| = O(\log k) \cdot b^*$.

Proof: Follows directly from the above discussion. ∎

Lemma 3.5 *Procedure Dist_Dom finds in polynomial time schedule of length $O(\log k \cdot b^*)$ rounds for $P_{I,U}$.*

Proof: Since the set cover D has value $val(D) = O(b^* \log k)$, it follows that each vertex $v \in I$ is assigned to $|D \cap A_v| = O(b^* \log k)$ neighbors of v in U. Thus, the busy schedule can inform all the vertices in $D \cap A_v$ in at most $O(b^* \log k)$ rounds. Once all the vertices of D are informed, we observe that the goal of the $P_{I,U}$ problem has been accomplished; Indeed, as the set D covers $\mathcal{V}_2$ in $\mathcal{B}_{I,U}$, it follows from the definition of the edge set $\mathcal{E}$ of $\mathcal{B}_{I,U}$ that each vertex in UT is of distance at most b^* from some vertex in D. ∎

3.3 Procedure $Final$: Using D to send the message to the set UT

We describe Procedure $Final$ that ends the schedule. We start with a $\sqrt{k}$ degree-depth partition (I, U) and elaborate on the following task: how to send the message from I to UT using as small number of rounds as possible. The algorithm has two phases. The first phase applies Procedure *Dist_Dom*. The second phase defines a collection of arborescences in the graph $G(U)$ induced by U and uses this collection in order to inform the terminals of UT.

For simplicity, we are first going to describe a solution for phase 2 that uses a collection of not necessarily disjoint arborescences. Later, we explain a fix that allows to maintain the useful properties of the defined arborescences, while converting the collection of arborescences into a vertex-disjoint collection.

Procedure $Final$

1. Run Procedure *Dist_Dom* to compute the $P_{I,U}$ instance of U and I and ts approximate solution D.
2. **Phase 1:** Every v such that $D \cap A_v \neq \emptyset$ sends the message in an arbitrary order to all the vertices in $\{u \mid \ x_{v,u} \in D \cap A_v\}$ (namely, all the vertices u assigned to v by Procedure *Dist_Dom*).
3. **Assignment Phase:** An assignment of the terminas UT is performed by the following procedure called Procedure $Assign$:

   Procedure $Assign$

   (a) Initialize all the vertices of $UT \setminus D$ as "unassigned".

   (b) **While** there is an unassigned vertex in $UT \setminus D$ **do:**

      i. Let u be a vertex in D so that there exists an unassigned vertex w, $w \in UT \setminus D$ so that $dist(u, w, G(U)) \leq b^*$.

ii. **For all** $z \in UT \setminus D$ so that $dist(u, z, G(U)) \le b^*$, set $\psi(z) = u$ and declare that z is assigned.

4. **Phase 2:** Let u be a vertex that has been assigned to at least one vertex of $UT \setminus D$. Let T_u be the shortest path arborescence leading from $u \in D$ to $W_u = \{w \mid \ u = \psi(w)\}$ in the graph induced by U. Let $\mathcal{J}$ be the collection of all arborescences for all the different vertices u.
/* Observe that we are using a non-disjoint collection of arborescences. The arborescences T_u are all computed over the same graph $G(U)$ and are not edge or vertex disjoint. */
Use the busy schedule to broadcast over $\mathcal{J}$.

The following claim applies.

Claim 3.6 *Phase 1 requires $O(\log k) \cdot b^*$ rounds.*

Proof: By Lemma 3.5 number of rounds for the first phase of the schedule is bounded by: $\max_{v \in V_1} |D \cap A_v| \le O(\log k) \cdot b^*$. ▮

We now consider some of the properties of the collection of trees $\mathcal{J}$ constructed in Phase 2.

Claim 3.7 *The maximum depth of a tree in $\mathcal{J}$ is at most b^*.*

Proof: By the definition of $\mathcal{E}$ and $P_{I,U}$, for every w, $w \in UT$ there exists a path of length b^* from $\psi(w)$ to $w \in UT$. Thus, if $u = \psi(w)$ then by definition, $dist(u, w, G(U)) \le b^*$.

Thus, the arborescences $T_u \in \mathcal{J}$ as defined in phase 2 of the schedule have depth at most b^*. ▮

In addition, by the properties of the $\sqrt{k}$ partition, the number of leaves in T_u is at most $\sqrt{k}$ (as otherwise more than $\sqrt{k}$ of the UT vertices are reachable from u via a path of length at most b^*).

As the arborescences are not vertex disjoint, the broadcasting task on one arborescence may conflict with the broadcasting task on other arborescences (since a single vertex is not allowed to send the message to two neighbors in the same round). It therefore follows that in the way Phase 2 is described now, the busy schedule may use too many rounds to inform the vertices of $\bigcup T_u$. The following change is used to form a vertex-disjoint collection of arborescences out of the jungle $\mathcal{J}$.

Define a level for every vertex in every arborescence in $\mathcal{J}$. The root u of the arborescence T_u is at level 0 and the level of a non-root in T_u is one larger than the level of its parent. For simplicity, we say that every root u has a parent at level -1. For every v appearing in several arborescences, let $par^*(v)$ be the parent of v in one of the arborescences, so that the level of $par^*(v)$ is minimum among the level of all the other parents of v in all other arborescences. Define a collection $\mathcal{F}$ of arborescences by taking the directed graph induced by the edges $(par^*(v), v)$ (the edges going from $par^*(v)$ into v). Note that some vertices of $U \setminus \mathcal{T}$ may become leaves (namely, have no children). Such vertices are discarded. Moreover, we discard every vertex that has no terminals in its sub-arborescence in $\mathcal{F}$.

Claim 3.8 *The above definition changes the collection of arborescences $\mathcal{J}$ into a forest $\mathcal{F} = \{T'_w\}$, namely, into a collection of vertex disjoint arborescences.*

Proof: Observe that a unique parent $par^*(v)$ is defined for each vertex v. The edges $(par^*(v), v)$ cannot induce a directed cycle as the level of $par^*(v)$ is smaller than the level of v (thus if we construct a path following $(par(v), v)$ edges, the level of the end-vertex of the path keeps increasing.) Also, a cycle cannot exist even in the corresponding undirected graph (resulting by ignoring the directions) as this will require a vertex with in-degree 2 in the graph induced by $(par^*(v), v)$. Thus, $\mathcal{F}$ is a collection of arborescences. It also follows that the arborescences are vertex-disjoint. Indeed, the constructed graph is a graph induced by a collection of edges and therefore the forest contains at most one copy of every vertex. ∎

Claim 3.9

1. *Every arborescence T'_w in F is rooted by some $w \in U$ that is also a root in $\mathcal{J}$.*
2. *The depth of every arborescence in $\mathcal{F}$ is at most b^*.*
3. *The number of leaves in every T'_w in $\mathcal{F}$ is at most b^*.*

Proof: We first show by induction on the level of v in $\mathcal{J}$ that the level of v in $\mathcal{F}$ is no larger than the *minimum* level of v in over all arborescences in $\mathcal{J}$. The base case is with $v = u$ for some T_u namely, v is a root of some T_u. In this case, the vertex u has both minimum level 0 in $\mathcal{J}$ and level 0 in $\mathcal{F}$.

For the induction step: if v is not a root in some tree in $\mathcal{J}$, let $par^*(v)$ be the minimum level parent of v over all the arborescences (containing v) in $\mathcal{F}$. Let T_u be the respective arborescence. By the induction hypothesis, the level of $par^*(v)$ did not increase in $\mathcal{F}$ compared to its level in this specific arborescence T_u (in fact the level of $par^*(v)$ in $\mathcal{F}$ is at most the minimum level of $par^*(v)$ in $\mathcal{J}$). Recall, that $par^*(v)$ is the lowest level parent v has in $\mathcal{J}$. Thus, the level of v in $\mathcal{F}$ is at most the minimum level of v in $\mathcal{J}$.

We deduce from this that the depth of each $T'_w \in \mathcal{F}$ is at most b^*; indeed, the depth of T'_w equals the level of the deepest leaf ℓ in T'_w. The level of ℓ in T'_w is at most the minimum level of ℓ in $\mathcal{J}$ which in turn by Claim 3.7 is at most b^*.

In addition, the root of every arborescence must be a root in $\mathcal{J}$ as well; indeed, a vertex that does not appear as a root in $\mathcal{J}$ is assigned (by definition) some parent.

Finally, observe that every arborescence T'_w in $\mathcal{F}$ is still a arborescence in $G(U)$. Thus, such an arborescence can not contain more than b^* leaves, for otherwise more than b^* terminals are of distance b^* or less from w in T'_w. ∎

Claim 3.10 *There exists a procedure that accepts as input a $\sqrt{k}$ degree-depth partition (U, I) and outputs a schedule with $O(\log k \cdot b^*) + 2\sqrt{k}$ rounds that informs UT. Furthermore, this procedure requires polynomial time.*

Proof: By Claim 3.6 we are able to inform a distance b^* dominating set of UT in $O(\log k) \cdot b^*$ rounds. In the second phase, we use busy schedule over the modifies forest $\mathcal{F}$. By the properties of Claims 3.8 and Claim 3.9 and by Lemma 1.1 it follows that the task of informing UT can be completed in $\sqrt{k} + b^*$ rounds. The claim follows. ∎

4 The multicast algorithm and its analysis

In this section the pieces are put together and a schedule with $O(\log k) \cdot b^* + 2 \cdot \sqrt{k}$ rounds is constructed.

4.1 The algorithm

We start with a formal description of Procedure $Multicast$.

Input: A graph $G = (V, E)$, a set $\mathcal{T}$ of terminals and a source $s \in V$.

Output: A schedule with $O(\log k) \cdot b^* + 2 \cdot \sqrt{k}$ rounds.
Procedure $Multicast$

1. Invoke Procedure $Comp - Par$ to construct (I, U, Roots).
2. Use a shortest path arborescence T leading from s to Roots. Form a busy schedule that sends the message from s to Roots.
3. For $u \in \mathit{Roots}$, let T_u be the arborescence as defined in Line 3b in Procedure $Comp - Par$. Every vertex $u \in \mathit{Roots}$ broadcasts over T_u (in parallel) using the busy schedule.
4. Use Procedure *Dist_Dome* to inform a distance$-b^*$ dominating set D of UT.
5. Use Procedure *Final* to inform UT.

It is easy to see that the total running time for computing the schedule is $\tilde{O}(|E| \cdot |V|)$ due to the computation of the $\sqrt{k}-$bad vertices in Line 3 of Procedure $Comp - Par$. In particular the [CK04] algorithm is a simple greedy algorithm whose running time is smaller than $O(|E| \cdot |V|)$. Combining this and Claim 2.4, Claim 3.6 and Claim 3.10, we get:

Theorem 4.1 *The minimum time directed telephone multicast problem admits a* $(\log k, \sqrt{k})$ *ratio* $O(|E| \cdot |V|)$ *time approximation algorithm*

4.2 Applications and extensions

The postal model: Essentially the same algorithm gives the same result for the more general postal model. We need to use weighted shortest paths and not the breadth first search trees. In addition, recall that a vertex v is only busy for $\mu(v)$ time units. Hence v "can handle" ρ/μ messages in ρ time units. Therefore, we use $\sqrt{k}/\mu(v)$ height trees with at most $\sqrt{k}/\mu(v)$ leaves. The arguments that prove the desired approximation guarantee are essentially identical.

The case of large b^* and large k. Consider the special case of the directed multicast problem under the restriction that $k = \Theta(n^\epsilon)$ for some constant $\epsilon > 0$, and $b^* \geq \sqrt{k}/\log n$. The ratio derived is $O(\log n)$. We observe that this ratio is the best possible, up to constants, unless $P = NP$.

To show this, consider a variant of the reduction of [F01] that can be described as follows: the reduction is from the set-cover problem, with $G(V_1, V_2, E)$. The goal is to choose a minimum size subset of V_1 that covers V_2. We may assume that $|V_1| = |V_2|$. The reduction directs the edges from

V_1 to V_2, and adds a source s connected to all of V_1. Without loss of generality, we may assume that the number of vertices assigned in the cover to every $v_1 \in V_1$ is negligible with respect to the size of the minimum set-cover (one way to do it is to take many disjoint copies of G). In addition, set $\mathcal{T} = V_2$. It follows that the only efficient way to inform V_2 is to broadcast the message to a small set-cover of V_2 and from there to V_2.

Thus, the problem reduces to the minimum size set-cover problem. Let $n = |V_1| + |V_2|$. Note that $k = n/2$. We can make $k = \Theta(n^\epsilon)$ for example by adding enough copies of the V_1 vertices (this clearly does not reduce the broadcast time). The size of the instance is increased but this effects only the constants in the approximation ratio.

The directed Steiner poise problem: Essentially the same algorithm that we used for the multicast problem provides a similar approximation ratio for the directed Steiner poise problem. Let h^* and d^* be the depth and degree in the optimal tree.

Claim 4.2 *The directed Steiner poise problem admits an* $(O(\log k), \sqrt{k} + O(1))$ *approximation.*

Proof: We may assume that h^* and d^* are known. We start the construction by constructing the arborescence leading to *Roots* as described in Line 1 in Procedure *Multicast*. This arborescence has maximum out-degree $\sqrt{k}+2$ and maximum depth h^* (we use h^* here and not b^*). We "extend" this arborescence by concatenating to its leaves the arborescences $\{T_u\}$ defined in Line 3a in Procedure $Comp - Par$. This adds b^* to the depth and the maximum out-degree remains $\sqrt{k}+2$. Let I be the set of vertices that appear in the tree constructed so far, and let $U = V \setminus I$. It follows that (I, U) is a $\sqrt{k}$ degree-depth partition.

In an almost identical claim to Claim 3.5 we show that it is possible to extend the arborescence with a new collection of additional vertices D, so that every vertex in $U \setminus D$ is at distance at most h^* from D in $G(U)$. The addition of the new vertices will increase the out-degree in the schedule to $\max\{\sqrt{k}+2, O(\log k) \cdot d^*\}$ and the depth by 1 (see Lemma 3.5). Finally, composing the arborescences of $\mathcal{F}$ as defined in Subsection 3.3 over the current arborescence, does not add to the maximum out-degree, but adds at most h^* to the depth. ▮

Remark: Observe that the depth of the arborescence is at most $O(h^*)$. Thus our algorithm gives a bicriteria arborescence that approximates the depth up to $O(1)$ and the maximum out-degree up to $(O(\log k), \sqrt{k}+2)$.

Discussion and open problems: The main question that remains open is either to devise a logarithmic approximation algorithm for the directed multicast problem or to prove that no such an algorithm exists (unless $P = NP$). To the best of our knowledge, even the seemingly simpler problem of finding a minimum out-degree Steiner arborescence (with no restriction on the depth) is not known to have a logarithmic ratio approximation. In fact, the [F01] reduction implies a $\log n$ lower bound for this Steiner degree minimization problem. This should be contrasted with the minimum degree *Spanning tree* arborescence problem that admits an $O(\log n)$ approximation in polynomial time [FR90] and a $O(\log n)$ *additive* approximation in $n^{O(\log n)}$ time [KR01].

Moreover, even if the radius of G (with respect to the root s) is *constant*, no logarithmic approximation algorithm is known for the directed Steiner degree minimization problem. Observe that the [F01] construction uses a graph of radius 2 which implies a $\log n$ lower bound for this variant. We suspect that the situation is worse even for radius at least 3. Specifically, we suspect that unless $P = NP$, no polynomial time logarithmic ratio approximation algorithm exists for the directed minimum degree Steiner arborescence problem, even if the problem is restricted to the special case of the root s having radius 3.

References

[BGN+98] A. Bar-noy, S. Guha, J. Naor and B. Schieber. Multicasting in Heterogeneous Networks, *SIAM Journal on Comput.* 30(2): 347-358, 2000

[BK94] A. Bar-Noy and S. Kipnis, Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems, *Mathematical System Theory*, Vol. 27, pp. 431–452, 1994.

[CK04] C. Chekuri and A. Kumar, Maximum Coverage Problem with Group Budget Constraints and Applications, *APPROX-RANDOM*, , pages 72-83, 2004.

[EK02] M. Elkin and G. Kortsarz, A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem, In *Proc. of 34th ACM Annual Symp. on Theory of Computing*, pp. 438-447, 2002.

[EK03] M. Elkin and G. Kortsarz, A sublogarithmic approximation algorithm for the undirected telephone broadcast problem: a path out of a jungle. In *Proc. of 14th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 76-85, 2003.

[EK03p] M. Elkin and G. Kortsarz, Approximation Algorithm for the Directed Telephone Multicast Problem. In *Proc. 30th Intern. Colloq. on Automata, Languages and Programming*, ICALP'03, pp.212-223, Eindhoven, Netherlands, June 2003.

[F01] P. Fraigniaud Approximation Algorithms for Minimum-Time Broadcast under the Vertex-Disjoint Paths Mode, In *9th Annual European Symposium on Algorithms (ESA '01)*, LNCS Vol. 2161, pp. 440-451, 2001.

[FR90] M. Furer and B. Raghavachari. An *NC* approximation algorithm for the minimum degree spanning tree problem. *In Proc. of the 28th Annual Allerton Conf. on Communication, Control and Computing*, pp. 274–281, 1990.

[HHL88] S. Hedetniemi, S. Hedetniemi, and A. Liestman. A survey of broadcasting and gossiping in communication networks. *Networks*, 18: 319-349, 1988.

[KP95] G. Kortsarz and D. Peleg. Approximation algorithms for minimum time broadcast, *SIAM journal on discrete methods*, vol. 8, pp. 401-427, 1995.

[KR01] R. Krishnan and B. Raghavachari. The Directed Minimum-Degree Spanning Tree Problem. In *FSTTCS 2001*, 232-243.

[R94] R. Ravi, Rapid rumor ramification: Approximating the minimum broadcast time. In *Proc. of the IEEE Symp. on Foundations of Computer Science (FOCS '94)*, pp. 202-213, 1994.

[S00] C. Schindelhauer, On the Inapproximability of Broadcasting Time, In *The 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'00)*, 2000.