Sublogarithmic Approximation for Telephone Multicast: Path out of Jungle

(Extended abstract)

Michael Elkin *† Guy Kortsarz‡

Abstract

Consider a network of processors modeled by an n-vertex 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 it 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, that generates the message, to all the processors of a given set $T \subseteq V$, $|T| = k$, whereas the processors of $V \setminus T$ may be left uninformed. The case $T = V$ is called *broadcast* problem.

The telephone multicast and broadcast are basic primitives in distributed computing and computer communication theory. Several approximation algorithms with a polylogarithmic ratio were suggested for these problems, and the upper bound on their approximation threshold stands currently on $O(\log k)$ and $O(\log n)$, respectively.

In this paper we devise an $O(\frac{\log k}{\log \log k})$-approximation algorithm for the k-multicast problem, and, consequently, an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the broadcast problem. Even stronger than that, whenever an instance of the k-multicast problem admits a schedule of length br^*, our algorithm guarantees an approximation ratio of $O(\frac{\log k}{\log br^*})$. As br^* is always at least $\log k$, the ratio of $O(\frac{\log k}{\log \log k})$ follows. In addition, whenever $br^* = \Omega(k^\delta)$ for some constant $\delta > 0$, we obtain a *constant* $O(1/\delta)$-approximation ratio for the problem.

Regarding the techniques, some previous papers on the subject used the idea of covering the set T of terminals by a forest, and broadcasting the message through this forest. In the current paper we develop a novel technique of covering the set of terminals by a *jungle*, that is, a collection of trees that are even not necessarily edge-disjoint, which nevertheless possess some other useful properties. The usage of jungles enables to obtain much smaller collections of trees, and this is reflected in the improved approximation ratio.

We also derive results regarding the *directed* and *edge-dependent heterogenous* k-multicast problems.

*School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA, 08540. elkin@ias.edu. This material is based upon work supported by the National Science Foundation under agreement No. DMS-9729992. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†Part of this work was done in Weizmann Institute, Department of Computer Science and Mathematics, Rehovot, Israel.

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

1 Introduction

Consider a network of processors modeled by an n-vertex 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 it 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, that generates the message, to all the processors of a given set $T \subseteq V$, $|T| = k$, whereas the processors of $V \setminus T$ may be left uninformed. The case $T = V$ is called *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 (cf. [5]). The optimization variants of the multicast and broadcast primitives were intensively studied during the last decade [2, 6, 8, 4]. Several approximation algorithms with a polylogarithmic ratio were suggested for these problems [2, 8, 3], and the upper bound on their approximation threshold stands currently on $O(\log k)$ and $O(\log n)$, respectively [2, 3]. The problems are known to be inapproximable within a factor of $3 - \epsilon$ for any $\epsilon > 0$ unless $NP = P$ [3].

In this paper we devise an $O(\frac{\log k}{\log \log k})$-approximation algorithm for the k-multicast problem, and, consequently, an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the broadcast problem. Even stronger than that, whenever an instance of the k-multicast problem admits a schedule of length br^*, our algorithm guarantees an approximation ratio of $O(\frac{\log k}{\log br^*})$. As br^* is always at least $\log k$, the ratio of $O(\frac{\log k}{\log \log k})$ follows. In addition, whenever $br^* = \Omega(k^\delta)$ for some constant $\delta > 0$, we obtain a *constant* $O(1/\delta)$-approximation ratio for the problem.

We believe that the main contribution of this paper is in demonstrating that telephone multicast and broadcast problems both admit *sublogarithmic approximation* algorithms. It is a widely accepted practice (c.f.

[1]) to divide all the approximation problems into four classes. The first class consists of problems that admit constant ratio approximation algorithms, the second class consists of problems with at least logarithmic and at most polylogarithmic approximation thresholds, and the third and the fourth classes consist of problems with even higher approximation thresholds. In view of several different approximation algorithms that achieved logarithmic or slightly higher than logarithmic approximation ratios for the multicast and broadcast problems [8, 2, 3], it was natural to assume that these problems belong to the second class of the aforementioned hierarchy. However, we prove that this is not the case, and that these problems are more likely to have a constant approximation threshold (or, else, there must be a *new class* of problems that are in-between the first and the second classes).

Regarding our techniques, our approximation algorithm has a similar structure to the one of [3]. Both algorithms, given a set of terminal vertices (shortly, terminals) T, compute a new smaller set of vertices T' (not necessarily a subset of T) and recurse on T'. Both algorithms construct collections of trees such that their set of roots is contained in T' and their set of leaves contains T. However, probably the most significant difference is that while in the algorithm of [3] the collection of trees is a *forest*, that is, contains *vertex-disjoint* trees, in our algorithm this collection is not a forest, and, furthermore, the same edge may appear in more than one tree.

The "profit" of allowing this more general combinatorial structure is that the set of terminals T can be covered using much smaller number of trees, yielding a smaller cardinality of T', and, therefore, a smaller depth of the recursion. The latter results, in turn, in a smaller approximation ratio. However, the "price" for this decrease in the number of trees is that broadcast schedules on collections of intersecting trees may a-priori be much longer than the ones on forests. This is because it may happen that a vertex that belongs to more than one tree will have to send the message to *all* its children in *all* the trees it belongs to. As in the telephone model a vertex can inform at most one neighbor in a round, this may result in much longer schedules than in the case of broadcasting over a forest.

We remark that constructing a broadcast schedule for a forest is a simple task, and covering the set of terminals by a forest and broadcasting the message through the forest is a basic technique that appears in some form in all known approximation algorithms for the problem. However, whenever the trees are allowed to intersect, devising a schedule and analyzing its length becomes a highly non-trivial task. The conceptual novelty of this paper is in characterizing a quite general family of collections of trees, that we call *jungles*, and demonstrating that broadcast on a jungle can be performed almost as efficiently as on forests. We stress that jungle is not an arbitrary collection of trees, but rather a collection that has to satisfy some specific properties. One of these properties is that for every pair of vertices that belong to an intersection of the vertex sets of some two trees from the jungle, and satisfy the ancestor-descendent relation in both trees (but the ancestor in one tree may be a descendent in the other), the paths between them in the two trees coincide. Paradoxically, the fact that the same edge may belong to more than one tree turns out to be helpful in our analysis, because sending the message along this edge advances the broadcast in all the trees to which this edge belongs.

We also derive approximation results regarding the *directed k-multicast* problem. While the directed *broadcast* problem is known to admit a logarithmic approximation ratio [3], no approximation algorithm with a non-trivial approximation ratio for the directed k-multicast problem is currently known (the trivial ratio is $O(k/\log k)$). For $0 < \alpha < 1/2$, we devise an algorithm that given an instance of the directed k-multicast problem that admits a schedule of length br^*, returns a schedule of length $O(k^\alpha \cdot br^* + k^{\max\{2\alpha, 1-\alpha\}})$. In particular, it implies an approximation ratio of $O(k^{2/3}/\log k)$ for the problem. Finally, some our results generalize to the *edge-dependent heterogenous* problem [2, 3] (see Section 3.3 for its definition). Most proofs are omitted from this extended abtract.

2 Preliminaries

The input in the telephone multicast problem consists of an undirected and connected graph $G(V, E)$, a source vertex s and a collection $T \subseteq V$ of terminals. Throughout the execution of the broadcast protocol, the vertex set V is splitted into a subset of *informed* vertices I and a subset of *uninformed* vertices U. Let $UT = U \cap T$ be the set of *uninformed terminals*. At the beginning of the the broadcast protocol $I = \{s\}$, $U = V \setminus \{s\}$ and $UT = T$.

A *round* is a matching of a subset $X \subseteq I$ to a subset $Y \subseteq U$. After the round, the vertices of Y become informed. A schedule is a sequence of rounds. The goal is to use the fewest possible number of rounds and inform the entire set of terminals T. Throughout the paper we denote $|T| = k$, and the optimum value for the instance at hand is denoted by br^*.

Given a subgraph G', the distance (minimum number of edges in a simple path) between u and v is denoted by $dist_{G'}(u, v)$. The set of neighbors of u is denoted by

$\Gamma(u)$. The graph induced by a set of nodes U is denoted by $G(U)$. For a tree τ, let $L(\tau)$ be the set of leaves of τ, $B(\tau)$ be the set of vertices with degree at least 3, and $M(\tau)$ the set of all other vertices of τ. For any pair of vertices $u, w \in V(\tau)$, $P^{\tau}_{u,w}$ is the path between u and w in τ. For a rooted tree (τ, r), let $depth(\tau, r)$ be the maximal distance in τ from r to some other vertex of $V(\tau)$. The relation $\widehat{AD}(\tau, r)$ is defined to be the set of pairs (u, w) such that u is an ancestor of w in the rooted tree (τ, r) or vice versa.

3 The algorithm and its Description

We start with the definition of *jungle*.

DEFINITION 3.1. *Given vertex sets $UT \subseteq U \subseteq V$ and an integer d, a collection of rooted trees $J = \{(\tau, r)\}$ is a (U, UT, d)-*jungle *if*

1. *For every pair of trees $(\tau_1, r_1), (\tau_2, r_2) \in J$, $\tau_1 \neq \tau_2$ implies $dist_{G(U)}(r_1, r_2) > d$ and $L(\tau_1) \cap L(\tau_2) = \emptyset$.*
2. *$depth(J) = \max_{\tau \in J} depth(\tau, r) \leq d$.*
3. *For every pair of vertices $u, w \in V$ such that $\{u, w\} \in \widehat{AD}(\tau_1, r_1) \cap \widehat{AD}(\tau_2, r_2)$ for some $(\tau_1, r_1), (\tau_2, r_2) \in J$, $P^{\tau_1}_{u,w} = P^{\tau_2}_{u,w}$.*
4. *For any tree $(\tau, r) \in J$, $V(\tau) \subseteq U$, $L(\tau) \subseteq UT$.*

In [3] *fork triples* were used to ensure that the number of vertices that will serve as terminals for the recursive invocation (i.e., the cardinality of T') is at least by a factor of 2 smaller than the number of original terminals (i.e., the cardinality of T). This reduction in the number of terminals guaranteed a logarithmic approximation ratio for the algorithm that was devised in [3]. To achieve a *sublogarithmic* approximation ratio, we use a somewhat more elaborated notion of *fork tree*. Intuitively, a *fork tree* is a tree that has "many" terminals "close" to the root, which is a terminal by itself. A fork tree τ has at most br^* leaves, all its leaves belong to UT, and there are exactly br^* vertices of UT in τ.

DEFINITION 3.2. *Given an (undirected) graph $G = (V, E)$, two sets of vertices $U, UT \subseteq V$ such that $UT \subseteq U$, and a positive integer br^*, the tree $\tau = (V(\tau), E(\tau))$ rooted in a vertex $z \in V(\tau) \cap UT$ is a (U, UT, br^*)* fork tree *for G if $L(\tau) \subseteq UT$, $V(\tau) \subseteq U$, $E(\tau) \subseteq E(U)$, $depth(\tau) \leq 6br^*$ and $|V(\tau) \cap UT| = br^*$.*

Procedure *Busy_Schedule***:** Throughout the paper this procedure is invoked only on collections of trees (more specifically, either on forests or jungles). In a busy schedule, an informed vertex v picks an arbitrary tree τ such that $v \in V(\tau)$, and there is an uninformed child u of v in τ. Then v sends the message to u. Note that messages are sent *only from parents to children*.

Intuitively, busy schedule is a greedy strategy that tries to guarantee that whenever possible no informed vertex is "idle".

Phase 1: We start with the input graph and construct a *forest* $\mathcal{F} = \{Q_1, Q_2, \ldots\}$ of *vertex-disjoint* fork trees. Initialize $U = V$ and $UT = T$. We identify a fork tree Q_i in $G(U)$ (if any) and remove $V(Q_i)$ from U. Remove the terminals of $T \cap V(Q_i)$ from UT. This loops continues until $G(U)$ contains no fork trees.

We will later argue that broadcast can be performed efficiently on a single given fork tree. Because the trees $\{Q_i\}$ are vertex- (and, thus, edge-) disjoint, the broadcast can be performed *in parallel* over these trees.

When the loop of the first phase terminates, there still could be some uncovered terminals, that is, it may happen that $UT \neq \emptyset$. This happens if some terminals do not appear in any Q_i. Let UT be the terminals that are not contained in $\mathcal{F}$ when the loop is completed. At this stage U contains no (U, UT, br^*)-fork tree. In other words, there is no vertex $u \in UT$, so that at least br^* UT vertices are of distance at most $6br^*$ from u in $G(U)$.

At the end of Phase 1 all the roots of the fork trees are inserted into the set T'.

Phase 2: Recursively inform the collection T' of roots of $\mathcal{F} = \{Q_i\}$, and then use busy schedule to inform $I = V(\mathcal{F})$.

Now, we define two more phases of the algorithm. These phases are used to inform UT.

Constructing the jungle: Before describing Phase 3, we describe a procedure that accepts as input the subset U that is obtained at Phase 1, and outputs a $(U, UT, 2br^*)$-jungle $\mathcal{J}$.

The trees in $\mathcal{J}$ are formed by Procedure *Constr_newtrees*(U, UT, br^*).

Procedure *Constr_newtrees*(U, UT, br^*)**:** The procedure covers UT by the BFS (in $G(U)$) trees τ_i. At the beginning, set $UT' = UT$. The procedure operates as long as $UT' \neq \emptyset$. The resulting collection of trees $\{\tau_i\}$ should be canonical with respect to Def. 3.1.

Let $(e_1, e_2, \ldots, e_m)$ be some arbitrary ordering of the edges of $E(U)$. Assign e_i weight $w(e_i) = 2^i$. The procedure starts by picking an arbitrary terminal $r_1 \in UT'$.

The procedure chooses an *unweighted* shortest path tree τ_1 rooted at r_1 leading to some terminals of UT'. Thus, if $u \in \tau_1$ is a leaf, the path between r_1 and u contains the minimal number of edges. However, among all possible paths with minimum number of edges, the procedure picks the one with the minimum weight.

Thus, the graph the procedure operates on, is the following directed version of the breadth-first-search (henceforth, BFS) layered graph rooted at r_1 in $G(U)$. The directed layered graph $\vec{G}^1(U)$ is obtained from a BFS computation from r_1 in $\vec{G}^1(U)$. We partition the vertices into layers, according to their (unweighted) distance from r_1. Layer L_i are the vertices of (unweighted) distance i from r_1. Edges between two vertices in the same layer, are deleted. Only edges from L_i to L_{i+1} remain, and these edges are directed from the L_i vertices, to L_{i+1} vertices. We also delete all layers i, with $i \geq 2br^* + 1$, namely, the graph $\vec{G}^1(U)$ contains only terminals, of unweighted distance at most $2br^*$ from r_1.

The tree τ_1 is obtained as follows. We compute the *weighted* shortest path tree τ in $\vec{G}^1(U)$, output by the algorithm, when run on the directed layered graph, with the weights of arcs as defined above. In particular, all the leaves are terminals. The tree τ_1 is the underlying tree of the tree that was constructed by this process (the same tree ignoring the orientations of arcs).

To summarize, the algorithm will find a path from r_1 to any leaf-terminal at distance at most $2br^*$, with the following two properties: (1) This path is one of the shortest paths in the unweighted sense, in $G(U)$, between r_1 and ℓ. (2) Among all the shortest paths in the unweighted sense, this path has the smallest weight.

Note that all the terminals at unweighted distance at most $2br^*$ from r_1 in $G(U)$ are included in τ_1, and all the leaves of τ_1 are terminals (all the leaves belong to UT). The tree τ_1 is the first tree into $\mathcal{J}$. All the terminals that belong to τ_1 are deleted from UT'.

Note that unlike the process of extraction of fork trees in Phase 1, the tree τ_1 is *not extracted* from U. All the remaining trees are computed on the same original graph $G(U)$. However, before τ_2 is computed, terminals belonging to $UT \cap V(\tau_1)$ are removed from UT. We compute τ_i, $i \geq 2$ similarly. We pick a new terminal $r_i \in UT'$, and take the underlying tree τ_i of the shortest path tree from r_i, in the directed layered graph of r_i. Again the depth of this graph at most $2br^*$, and the tree connects r_i to all the terminals of UT' that are within unweighted distance at most $2br^*$ from r_i in $G(U)$. All the terminals that belong to τ_i are removed from UT', before computing τ_{i+1}. The procedure continues until UT becomes empty. Let $\mathcal{J} = \{\tau_1, \ldots, \tau_q\}$ denote the collection of all trees that are constructed by the procedure.

Intuition: Suppose that two trees τ_1 and τ_2 have a path from v to u and u is a descendant of v in both trees. The message should be forwarded from v to u both in τ_1 and τ_2. Both trees have its own sets of leaves-terminals that should be to be informed. If the shortest path used would be different in the two trees, the busy schedule will produce a large delay.

One the other hand, in our scheme, the same path between v and u is used in all the trees of the jungle $\mathcal{J}$.

The procedure returns the set $\mathcal{R}$ that contains all the roots of the trees of the jungle $\mathcal{J}$.

Phase 3: Procedure $Constr\_Spiders(I, \mathcal{R}, br^*)$ defined in [3] accepts as input a collection $\mathcal{R}$ of vertices such that no pair of them is closer than $2br^* + 1$ one to another. In our algorithm, the procedure operates on the collection $\mathcal{R}$ of roots of the trees in the jungle $\mathcal{J}$. The procedure broadcasts the message from source s to all the vertices of $\mathcal{R}$. The main two tools that are used in the implementation of this procedure are computations of flow and their relation to constructing spiders (for the definition of this notion, see [2]). The following theorem is proved in [3].

THEOREM 3.1. *[3] Suppose that there exists a multicast schedule of length br^* for the instance (G, T, s) of the problem. Let I and $\mathcal{R}$ be two disjoint vertex sets such that $s \in I$, and for every pair of vertices $r, r' \in \mathcal{R}$, $dist_{G(U)}(r, r') \geq 2br^* + 1$. Then the invocation $Constr\_Spiders(I, \mathcal{R}, br^*)$ returns an $(I, \mathcal{R})$-schedule with $O(br^*)$ rounds. ($(I, \mathcal{R})$-schedule is a schedule that describes how the vertices of $\mathcal{R}$ can be informed given that the vertices of I are informed.)*

Thus, in Phase 3 Procedure $Constr\_Spiders$ is used to send the message from the set I of informed vertices to the set $\mathcal{R}$ of roots of the jungle trees.

Phase 4: The broadcast procedure is completed by informing UT over the jungle edges using the busy schedule. After phase 4 ends, all terminals are informed.

3.1 Algorithm *Multicast*

Input: graph $G = (V, E)$, source $s \in V$, set of terminals $T \subseteq V \setminus \{s\}$, positive integer br^*.
Output: Broadcast schedule from s to T.

1. $T' \leftarrow \emptyset$; /* Set of terminals on which the recursion will be invoked. */
2. $U \leftarrow V \setminus \{s\}$; /* Set of vertices that still can be used for (U, UT, br^*)-fork trees. */
3. $UT \leftarrow T$; /* Set of "uncovered" terminals. */
4. $\mathcal{F} \leftarrow \emptyset$; /* Forest of vertex-disjoint (U, UT, br^*) fork trees. */
5. $I \leftarrow \{s\}$; /* Set of informed vertices. */
6. **While** there exists a (U, UT, br^*)-fork tree in $G(U)$ **do**

   /* Phase 1: Constructing the forest. */

   (a) Compute (U, UT, br^*)-fork tree Q_i;

/* Q_i is a BFS tree rooted at some some $u \in UT$, having depth at most $6br^*$ and containing exactly br^* terminals of UT. */

(b) $\mathcal{F} \leftarrow F \cup \{Q_i\}$; /* Inserting into the forest the new tree Q_i. */

(c) $U \leftarrow U \setminus V(Q_i)$; /* The nodes of T can no longer be used for (U, UT, br^*)-fork trees. */

(d) $I \leftarrow I \cup V(Q_i)$;

7. Recursively inform T'; /* Part of Phase 2: informing the roots of $\mathcal{F}'$. */

8. $Busy\_Schedule(G(F), T')$;

   /* The rest of Phase 2: informing the vertices of $I(F)$, assuming that the vertices of T' are already informed.*/

9. $\mathcal{J} \leftarrow Constr\_newtrees(U, UT, br^*)$;
   /* Construct a jungle. The trees are not (necessarily) vertex-disjoint but satisfy the definition of jungle.*/

10. $\mathcal{R} \leftarrow Roots(\mathcal{J})$; /* $\mathcal{R}$ contains the collection of roots of $\mathcal{J}$. */

11. $Constr\_Spiders(I, \mathcal{R}, br^*)$; /* Phase 3: Inform the roots $\mathcal{R}$ assuming that the vertices of I are already informed. */

12. $Busy\_Schedule(G(J), \mathcal{R})$;
    /* Phase 4: inform all the vertices of $V(J)$ via a busy schedule invoked on the jungle, assuming that the roots of the jungle are already informed. */

3.2 Approximation Ratio We next prove that Algorithm $Multicast$ guarantees an approximation ratio of $O(\log k/\log\log k)$ for the k-multicast problem.

Phases 1 and 2: Consider the forest $\mathcal{F} = \{Q_i\}$ that is constructed on Phase 1 of the algorithm, and its collection of roots T'. The terminals of T' are informed recursively.

FACT 3.1. *For a tree τ, $|L(\tau)| = 2+\sum_{v\in B(\tau)}(deg_\tau(v)-2)$.*

LEMMA 3.1. *Let Q be an undirected tree rooted at s with leaf set L and depth h. Assume that s is informed. Then a busy schedule informs all the vertices in Q in at most than $h + |L|$ rounds.*

LEMMA 3.2. *Whenever T' is informed, the number of rounds that are required to inform $V(\mathcal{F})$ is $O(br^*)$.*

Proof. Note that the trees in $\mathcal{F}$ are (U, UT, br^*)-fork trees. It follows that their depth is at most $6br^*$, and each of them has at most br^* leaves. The lemma follows now from Lemma 3.1. ∎

LEMMA 3.3. *Consider some invocation $Multicast(G, s, T, br^*)$ for some graph $G = (V, E)$, vertex $s \in V$, set of terminals $T \subseteq V$, and integer br^*. Then $|T'| \le |T|/br^*$.*

Proof. Note that every tree $Q_i \in F$ is a (U, UT, br^*)-fork tree, and thus contains exactly br^* terminals. Only one of these terminals, namely, the root of Q_i, enters T'. The lemma follows. ∎

Hence the recursion height is $O(\log k/\log br^*)$.

Properties of jungle $\mathcal{J}$: We next show that the collection of trees $\mathcal{J}$ that is constructed by Algorithm $Multicast$ satisfies the definition of jungle.

The set U obtained at the end of Phase 1 will not be changed throughout the computation of the jungle. The vertices $V(\tau_i)$ are all subsets of the *same* vertex set U, and so a vertex $u \in U$ may appear in several different trees.

Let $\mathcal{R} = Roots(J) = \{r_1, r_2, \ldots\}$.

1. **Leaf property:** Every terminal is a leaf in at most one tree $\tau \in J$.

2. **Distance property:** For any pair of vertices $r, r' \in \mathcal{R}$, $d_{G(U)}(r, r') \ge 2br^* + 1$.

To show that $\mathcal{J}$ satisfies the leaf property suppose that the lowest index of a tree that contains ℓ as a leaf is τ_i. Then, for τ_j, $j > i$, ℓ is not included in τ_j as a leaf, because ℓ no longer belongs to UT'.

Let τ^ℓ denote the unique tree that contains ℓ as a leaf.

The distance property follows directly from the construction.

Suppose that u and v appear in two different trees τ_i and τ_j and share in both trees the ancestor-descendant relation. Then:

LEMMA 3.4. *The paths from u to v in τ_i and in τ_j coincide.*

Proof. Let r_i and r_j be, respectively, the roots of τ_i and τ_j. First assume that in both paths u is an ancestor of v. Observe that subpaths of any shortest unweighted path P are also shortest paths as well. Thus, both paths from u to v in τ_i and τ_j are shortest paths in the unweighted sense.

Thus, the paths from u to v used in both trees must coincide. To see this, recall that the algorithm takes the minimum weight path among all unweighted shortest paths. By the choice of edge weights, there is a unique *minimum weight* path.

The other cases are similar, and their proof is omitted. ▮

Phase 3: Recall that in Phase 3 we assume that all the vertices of I are already informed, and the goal is to inform the vertices of $\mathcal{R}$. The following corollary follows by the distance property and Theorem 3.1.

COROLLARY 3.1. *The number of rounds in Phase 3 is $O(br^*)$.*

Analyzing the number of rounds in Phase 4: In this section we analyze the length of the schedule produced by an invocation of Procedure *Busy_Schedule* on the jungle $\mathcal{J}$. This is the crux of the analysis. Consider the following charging scheme. Let ℓ be a leaf in τ^ℓ (recall that such a tree is unique). Let r_ℓ be the root of τ^ℓ, and P^ℓ be the unique path from r_ℓ to ℓ in τ^ℓ. At any given moment let $cc(\ell)$ (cc stands for "contiguous close") is defined as follows:

1. $cc(\ell)$ is informed, and so are all the vertices of the subpath of P^ℓ between r_ℓ and $cc(\ell)$.
2. Among all the vertices that satisfy the condition 1, $cc(\ell)$ is the vertex x with minimum distance $dist_{P^\ell}(x,\ell)$ in P^ℓ from ℓ.

Intuitively, we look for the largest "contiguous" subpath of P^ℓ so that all the vertices in the path are informed, and the child of $cc(\ell)$ is not informed. Clearly, as the root r_ℓ is always a legal choice, $cc(\ell)$ is well-defined.

Observe that since the trees in the jungle intersect, it may happen that the children of v in τ_i are informed before v (via other trees). This means that the set of informed vertices may not be contiguous in some paths. There may be informed vertices much closer to ℓ than $cc(\ell)$, even on P^ℓ. However, we require that $cc(\ell)$ belongs to P^ℓ, and all the vertices closer to r_ℓ in P^ℓ than $cc(\ell)$ are informed. Therefore, by definition, the child of $cc(\ell)$ that belongs to the path P^ℓ is not informed.

Categorizing types of delay: We next compute an upper bound on the number of rounds that precede to the round in which the leaf ℓ receives the message. Because the schedule is busy, and $cc(\ell)$ has an uninformed child in P^ℓ, there *necessarily exists* a tree τ_j in which $cc(\ell)$ is active. In other words, $cc(\ell)$ sends the message to one of its children in *some* tree τ_j. Let τ_j be the tree so that $cc(\ell) \in \tau_j$ and $cc(\ell)$ sends the message to a child in τ_j. We distinguish four different types of rounds with respect to some fixed leaf ℓ.

1. **Type 1:** In a round of type 1, the vertex $cc(\ell)$ sends the message along P^ℓ over the next edge in the path P^ℓ (recall that P^ℓ is the path in τ^ℓ from r_ℓ to ℓ).
2. **Type 2:** $deg_{\tau_j}(cc(\ell)) \geq 3$, or $cc(\ell) = r_j$ (the root of τ_j) and $deg_{\tau_j}(cc(\ell)) = 2$, ($r_j$ has exactly two children in τ_j.)
3. **Type 3:** $\deg_{\tau_j}(cc(\ell)) = 2$, and $cc(\ell)$ is not the root of τ_j. Observe that in this case $cc(\ell)$ has exactly one child.
4. **Type 4:** $deg_{\tau_j}(cc(\ell)) = 1$.

   Observe that in the case of a round of type 4, $cc(\ell)$ must be the root of τ_j. It can not be a leaf in τ_j, as in this case it may send the message only to its parent in τ_j (contradicting the definitions of the function cc and busy schedule).

We now use the above division for charging delays for ℓ to edges.

Chosen triplets and charged edges: We charge exactly one edge, for every round in which ℓ is still uninformed. The charging is defined as follows. For $cc(\ell)$, let e_ℓ be the next edge in P^ℓ towards ℓ.

1. In a round of type 1, the chosen triplet is $(\tau^\ell, cc(\ell), e_\ell)$. The delay is charged to e_ℓ.
2. In a round of type 2, let e be the edge in τ_j, along which $cc(\ell)$ sends the message (to one of the children of $cc(\ell)$ in τ_j). The chosen triplet is $(\tau_j, cc(\ell), e)$ and the delay is charged to e.
3. In a round of type 3, $cc(\ell)$ has a unique child v in τ_j. Let w be the first descendent of v that has degree different than 2 (it may be that w is a leaf, or otherwise, w has degree at least 3). Let $e = (p(w), w)$ be the edge between the parent $p(w)$ of w in τ_j. The chosen triplet is (τ_j, w, e) and the delay is charged to e.
4. In a round of type 4, $cc(\ell) = r_j$ and the root of τ_j has exactly one child. Let w be the first descendent of $r_j = cc(\ell)$ in τ_j that has degree not equal to 2 (exactly as the case of type 3) and the chosen triplet is $(\tau_j, w, (w, p(w)))$. The edge $(w, p(w))$ is charged for this round.

We next "collect" the charged edges, according to the vertices that are incident to them, and the trees they belong to. We analyze separately the four possible types of rounds:

Type 1: The chosen triplet is (u, e_ℓ, τ^ℓ). In every round of type 1, the message propagates one unit towards ℓ in

P^ℓ. Hence after j rounds of type 1, the distance from ℓ to its closest informed vertex in $G(J)$ is at most $|P^\ell|-j$. Thus, the number of rounds of type 1 is at most $|P^\ell|$. Observe that $|P^\ell|$ is at most $2br^*$, as this is an upper bound the depth of the tree τ^ℓ. Thus, the number of rounds of type 1 is at most $2br^*$.

Type 2: Fix some tree τ_j that appears (at least once) in some chosen triplet of type 2. We first count the total charge for triplets containing τ_j. Consider a tree τ_j and a vertex $u \in V(\tau_j)$. Note that u and τ_j may appear together in a charged triplet more than once. However, every chosen triplet containing τ_j and u, has a different edge $e \in \tau_j$ as the third member of the triplet. Indeed, if $e=(u,v)$ appears in a triplet, then after this round the vertex v is informed and the edge $e=(u,v)$ will be never charged again (as the child vertex in a charged edge must be an uninformed vertex).

To summarize,

Corollary 3.2. *Given a tree τ and a vertex $v \in V(\tau)$, the number of triplets containing both u and τ_j in a chosen triplet of round of type 2 is at most $deg_{\tau_j}(u)$.*

There are at most two rounds with $cc(u)=r_j$ (because $deg_{\tau_j}(r_j)=2$). We now deal with the main case, namely, the chosen triplet is, (u,e,τ_j), and $deg_{\tau_j}(u)\ge 3$. Let $B(\tau_j)$ be the collection of all vertices of degree at least 3 in τ_j. Let $L(\tau_j)$ be the collection of leaves in τ_j.

Lemma 3.5. *The number of rounds of type 2 for ℓ is at most $3br^*+2$.*

Proof. By Corollary 3.2, the number of rounds charged here for triplets containing τ_j is at most $\sum_{u\in B(\tau_j)} deg(u)$. By Fact 3.1,

$$\sum_{u\in B(\tau_j)} deg(u) = 2|B(\tau_j)| \quad + \textstyle\sum_{v\in B(\tau_j)}(deg(u)-2)$$
$$= 2|B(\tau_j)| + |L(\tau_j)| - 2.$$

Again, by Fact 3.1, $|B(\tau_j)| \le |L(\tau_j)|$ and so,

$$\sum_{u\in B(\tau_j)} deg(u) \le 3|L(\tau_j)| \ . \qquad (3.1)$$

For continuation of the proof, we need the following lemma.

Lemma 3.6. *Let τ_j be a tree that belongs to a chosen triplet, and let ℓ' be a leaf in τ_j. Let r_ℓ be the root of τ^ℓ. Then $dist_{G(\mathcal{J})}(\ell', r_\ell) \le 6br^*$.*

We are now ready to complete the proof of the lemma. Note that by the leaf property, the leaves of a tree τ_j and a tree τ_i are pairwise disjoint for $i \ne j$. Let Γ_τ be the set of trees $\tau \in J$ that occur in chosen triples. Stopping condition of Line 6 of Algorithm *Multicast* implies that

$$\sum_{\tau\in\Gamma_\tau} |L(\tau)| \le br^* \ .$$

Now, by inequality (3.1), the total number of rounds of type 2 with respect to ℓ is at most $3br^*+2$ (the $+2$ here accounts for the case of a root with two children). The lemma follows. ∎

Type 3:

Lemma 3.7. *The number of rounds of type 3 is at most $2br^*$.*

Proof. Let $(\tau_j, w, (p(w), w))$ be some chosen triplet in a round of type 3. Whenever a triplet that contains τ_j and w appeared in a round of type 3, this happened due to a vertex $u \in P^\ell$, so that $u = cc(\ell)$. Let u be the first (with respect to the time steps of the algorithm) vertex to cause the appearence of τ_j and w in a chosen triplet of a round of type 3. The vertex u has a unique child v in τ_j (that is, has degree 2 in τ_j) and in this round u sends the message to v. This is because busy schedule chooses a tree τ_j so that u sends the message to one of its children in τ_j, and u has only one child in τ_j. The path from u to w, contains only vertices of degree equal to 2 (observe that it may happen that $w=v$). We call u the *causing vertex* for the triplet $(\tau_j, w, (p(w), w))$.

We next prove that the pair (τ_j, w) will not appear *together* more than once, in a round of type 3. The next causing vertex x of a triplet that contains both τ_j and w, must be an ancestor of w, and thus, an ancestor or descendant of u in τ_j. We will first show that no vertex x in the path from u to w in τ_j (a descendant of u and ancestor of w) can be the causing vertex of the second triplet of type 3 that contains both τ_j and w. Assume that x belongs to the path from u to w in τ_j. As x is the causing vertex, this implies that $x \in P^\ell$. Indeed, any causing vertex must belong to P^ℓ (as only the vertex that serves as $cc(\ell)$ for at least one round may be a causing vertex for the triplet). Since the round of type 3 of x occured after the round of type 3 of u, the vertex x is closer to ℓ in P^ℓ than u. Otherwise, x can never become $cc(\ell)$: it is further away from ℓ in P^ℓ than u, and u served as $cc(\ell)$ already. It follows that u is an ancestor of x, both in τ^ℓ, and in τ_j. Thus, by Lemma 3.4, the paths between u and x in τ^ℓ and τ_j coincide. This yields a contradiction: u could not have caused a round of type 3 with respect to ℓ, when sending the message to v. Indeed, v is the child of u in τ_j, and so it must belong to P^ℓ as well. In other words, the delay when u sends the message to v is a delay of type 1, and not of type 3, and this is a contradiction.

Finally, consider an ancestor x of u in τ_j, causing a second round of type 3 for τ_j and w. Observe that by the same argument, x must be closer in P^ℓ than u (as otherwise, x can never become $cc(\ell)$).

By Lemma 3.4, the paths between x and u in τ_j and in P^ℓ coincides. Observe that, by definition of $cc(\ell)$, when $x = cc(\ell)$, all the vertices between u and x in P^ℓ, including the parent $p_\ell(x)$ of x in P^ℓ, are already informed. This is a contradiction. Indeed, $p_\ell(x)$ is the *child* of x that is informed by x in this round. However, $p_\ell(x)$ is already informed, and in a busy schedule, no vertex is informed twice.

Thus, every pair (τ_j, w) can appear together at most once in a chosen triplet of type 3. Thus, the number of rounds that are of type 3 and contain τ_j, with any vertex w, is at most $|L(\tau_j)| + |B(\tau_j)|$. This follows, as, by definition, w has degree different than 2, and w does not appear in a round of type 3 together with τ_j more than once. The overall charging of rounds of type 3, is bounded by $\sum_j |L(\tau_j)| + |B(\tau_j)| \leq 2br^*$. The last inequality follows exactly as in the analysis of rounds of type 2. ∎

Type 4: The vertex $cc(\ell)$ is the root of τ_j, with a single child. This case is analyzed along the same line as case 3. The total charge is at most $2br^*$.

Summarizing, the number of rounds that are required for the message to reach any leaf ℓ is $O(br^*)$ in a single recursive invocation. As the depth of of the recursion is at most $O(\log k/\log br^*)$, the following theorem is derived.

Theorem 3.2. *Algorithm Multicast is an $O(\log k/\log br^*)-$ratio approximation algorithm for the k-multicast problem.*

Corollary 3.3. *The k-multicast problem admits an $O(\log k/\log\log k)$-approximation ratio.*

3.3 Edge-dependent heterogenous postal model

In [2] the more general *postal model* was introduced. In this model each vertex v has a delay number $0 \leq \rho(v) \leq 1$. The vertex sending a message is "busy" only at the first ρ time units starting from the initial sending time. After ρ 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. Generalizing the problem further, in [3] the edge-dependent heterogenous postal model was introduced. In this model the delay of a vertex v depends on the edge used to send the message. A generalization of our algorithms guarantees a $O(\log k/\log br^*)$-approximation ratio to the edge-dependent heterogenous postal model as well. Note, however, that in this model br^* might be $O(1)$.

We next sketch the proof of this statement. Essentially the same algorithm directly applies to the heterogenous postal model. However, in this more general setting, the problem of constructing a short schedule for informing the roots of the jungle seems not to be reducable to a computation of maximal flow. Instead, the problem of minimizing the makespan of parallel machines arises. The latter problem admits a 2-approximation algorithm [7]. It follows that our algorithm guarantees an approximation ratio that is greater at most by a factor of 2 for the k-multicast problem in the heterogenous postal model than the ratio it guarantees for the original k-multicast problem. In other words, the problem in the general setting admits an approximation ratio of $O(\log k/\log br^*)$.

4 Approximation algorithm for directed multicast problem

In this section we present an algorithm that inform the subset T of terminals in a directed graph, in $(O(\log k) + k^{1/3}) \cdot br^* + O(k^{2/3})$ rounds. This gives an $O(k^{2/3}/\log k)$ approximation ratio.

4.1 An algorithmic tool: the multiple set-cover problem

Consider the following problem called the multiple set-cover problem. Let $G(V_1, V_2, E)$ be a bipartite graph. The set $S \subseteq V_1$ is called a *set-cover* of V_2 if $\Gamma(S, G) = V_2$, 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.
Optimization goal: Minimize $\max\{|S \cap A_i|\}_{i=1}^d$.

Definition 4.1. *Define $val(S) = \max\{|S \cap A_i|\}_{i=1}^d$.*

Using standard randomized rounding techniques [10] an $O(\log k)$ approximation algorithm for the problem is derived. This is the best ratio possible (up to constants) because of the logarithmic lower bound on approximating set-cover (unless $P = NP$) [9].

Corollary 4.1. *There exists a randomized rounding algorithm that produces a feasible solution to the multiple set-cover problem, that has value $O(\log n) \cdot opt$. The construction can be derandomized to give a deterministic $O(\log n)$ approximation ratio.*

4.2 $k^{1/3}$ broadcast partitions Suppose we are given a set of terminals T. A $k^{1/3}$ broadcast partition (or, shortly, a $k^{1/3}$ partition) is a partition of V into two disjoint sets $V = I \cup U$, $I \cap U = \emptyset$, such that in the graph $G(U)$ induced by U no vertex $u \in U$ can reach via a directed path of length at most br^* more than $k^{1/3}$ terminals of $T \cap U$. It is assumed that the source s belongs to I.

4.3 The relation between $k^{1/3}$ partitions and multiple set-cover We explain the way the multiple set-cover problem can be used to give an approximate solution to a special case of the directed multicast problem.

Let I, U be a $k^{1/3}$ broadcast partition of V. Let $P_{I,U}$ be the problem of informing $T \cap U$ under the assumption that I is informed.

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

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

i.e., $\mathcal{V}_1$ corresponds to the collection of pairs (v,u) so that v is in I and u is in U and there is an arc from v to u. Let

$$A_v = \left\{x_{(v,u)} \mid \quad u \in \Gamma_G(v)\right\}.$$

Note that the sets A_v induce a partition of $\mathcal{V}_1$, i.e., $\mathcal{V}_1 = \bigcup_{v \in V} A_v$.

Let $\mathcal{V}_2 = T \cap U$.

We now define the edges between $\mathcal{V}_1$ and $\mathcal{V}_2$. There is an edge between a vertex $x_{(v,u)}$ to a node $w \in T \cap U$ if there is a directed path of length at most br^* from u to w in $G(U)$.

LEMMA 4.1. *The multiple set-cover instance $\mathcal{B}$ admits a solution $S \subseteq \mathcal{V}_1$ such that $val(S) \leq br^*$.*

Let S_R be the solution output by the randomized rounding. Let $S_v = S_R \cap A_v$.

LEMMA 4.2. $\max_{v \in V_1} |S_v| = O(\log k) \cdot br^*$.

LEMMA 4.3. *There is a polynomially computable schedule $O((\log k + k^{1/3}) \cdot br^* + k^{2/3})$ rounds schedule Pro−Bip for $P_{I,U}$.*

Proof. Given a jungle $\mathcal{J}$, the *fair busy schedule* is defined as follows. A vertex u transmits the message in all the trees it belongs to, one after the other, in a round robin fashion. Thus, an informed vertex u goes over all trees T_u it belongs to, one by one, and chooses an uninformed child v in T_u (if any) and transmits the message to v

We now define a schedule in two main phases.

1. **Phase 1 in the schedule**: Every v such that $S_v \neq \emptyset$ sends the message in arbitrary order to all the vertices in $\{u \mid \quad x_{v,u} \in S_v\}$.

2. **An assignment phase:** Let $\mathcal{K} \subseteq U$ be subset of the informed vertices in U. For a vertex $w \in T \cap U$, let u be the vertex minimizing $dist_{G(U)}(u,w)$. Define a function $\psi(w) = u$ (resolving ties in an arbitrary way).

3. **Phase 2 in the schedule:** Let T_u be the shortest path tree leading from u to $W_u = \{w \mid \quad u = \psi(w)\}$. Observe that we are using a jungle again; the T_u trees are not edge or vertex disjoint. The vertex u uses the fair busy schedule to broadcast over all the trees it belongs to.

LEMMA 4.4. *The first phase of the schedule requires $O(\log k) \cdot br^*$ rounds. The second phase of the schedule requires $O(k^{1/3} \cdot br^* + k^{2/3})$ rounds*

Proof. Clearly, by Lemma 4.2 the number of rounds for the first phase of the schedule is at most

$$\max_{v \in V_1} |S_v| \leq O(\log k) \cdot br^*.$$

In addition, by the definition of $\mathcal{E}$, for every $w \in T \cap U$ there exists a path of length $O(br^*)$ from $\psi(w)$ to $w \in T \cap U$. Thus, if $u = \psi(w)$ then by definition $dist_{G(U)}(u,w) \leq br^*$.

Thus, the tree T_u as defined in phase 2 of the schedule has depth at most br^*. In addition, by the properties of the $k^{1/3}$ decomposition, the number of leaves in T_u is at most $k^{1/3}$ (as otherwise more than $k^{1/3}$ of the $T \cap U$ vertices are reachable from u via a path of length at most br^*). In addition, a vertex w can appear in at most $k^{1/3}$ different trees. Indeed, every tree w appears in gives a (different extra) leaf "close" to w, and so the bound on the number of trees w appears in follows from the properties of a $k^{1/3}$ partition.

Suppose first that the trees happen to be disjoint. By Lemma 3.1, u can deliver the message to all of T_u in $O(br^* + k^{1/3})$ rounds. Observe that a vertex w can actually appear in as many as $k^{1/3}$ trees. Hence the number of rounds that are required to broadcast over all the trees T_u is at most $k^{1/3} \cdot O(br^* + k^{1/3})$. The lemma follows. ∎

This completes the proof of Lemma 4.3. ∎

COROLLARY 4.2. *The $\mathcal{P}_{I,U}$ problem admits an schedule with at most $O(\log k + k^{1/3}) \cdot br^* + k^{2/3}$ rounds.*

4.4 Computing a $k^{1/3}$ partition

In the previous subsection we have shown that it is enough for a fast schedule to get to a situation where the graph admits a $k^{1/3}$ (I,U) partition, with all the vertices of I are informed. However, it remains to show how to get to such a situation.

The following algorithm computes a $k^{1/3}$ partition. We say that $u \in U$ is $k^{1/3}$-good if more than $k^{1/3}$ vertices in $T \cap U$ are reachable from u in $G(U)$, by a directed path of length at most br^*.

Input: graph $G = (V, E)$, a set T of terminals and a source $s \in V$,
Output: a $k^{1/3}$ partition.
Procedure $Comp - Par$

1. $I \leftarrow \emptyset$;
2. $U \leftarrow T$;
3. $Roots \leftarrow \emptyset$; /* The roots are later used to inform I. */
4. **While** $G(U)$ admits a $k^{1/3}$−good vertex u **Do:**
5. Let $Cl(u)$ be the $\lfloor k^{1/3} \rfloor$ closest to u in $G(U)$ of the $T \cap U$ terminals (namely, $Cl(u)$ are the first $\lfloor k^{1/3} \rfloor$ terminals in the vector of terminals ordered by their distance from u in $G(U)$). Let T_u be the shortest path tree leading from u to $Cl(u)$ in $G(u)$.
6. $I \leftarrow I \cup V(T_u)$, $U \leftarrow U \setminus V(T_u)$, $Roots \leftarrow Roots \cup \{u\}$
7. Output $(I, U, Roots)$

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

Lemma 4.5. *The pair (I, U) output by $Comp - Par$ is a $k^{1/3}$ partition*

Lemma 4.6. *The set Roots has cardinality $|Roots| = O(k^{2/3})$*

4.5 The broadcast algorithm

In this subsection the pieces are put together and an $O((\log k + k^{1/3}) \cdot br^* + k^{2/3})$ rounds schedule is derived.

Input: graph $G = (V, E)$, a set T of terminals and a source $s \in V$.
Output: an $O((\log k + k^{1/3}) \cdot br^* + k^{2/3})$ rounds schedule.

Procedure $Comp - Sch$

1. Apply Procedure $Comp - Par$ to derive $(I, U, Roots)$.
2. Use a shortest path tree T leading from s to $Roots$. Apply the busy schedule to send the message from s to $Roots$.
3. For $u \in Roots$, let T_u be the tree as defined in Line 5 in Procedure $Comp - Par$. Every vertex $u \in Roots$ broadcasts over T_u (in parallel) using the busy schedule.
4. Use Procedure $Pro - Bip$ of Lemma 4.3 to inform $T \cap U$.

Properties of the algorithm:

Lemma 4.7. *The number of rounds to inform I (via Roots) in Lines (2) and (3) of Procedure $Comp - Sche$ is $O(k^{2/3} + br^*)$.*

Theorem 4.1. *Procedure $Comp - Sche$ sends the message to T in $O((\log k + k^{1/3}) \cdot br^* + k^{2/3})$ rounds*

References

[1] S. Arora, C. Lund. Hardness of Approximation, in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum Ed., pp. 399-440, (PWS Publishing Company), 1997.

[2] A. Bar-noy, S. Guha, J. Naor and B. Schieber. Multicasting in Heterogeneous Networks In *Proc. of ACM Symp. on Theory of Computing* 1998.

[3] M. Elkin and G. Kortsarz. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem, *Proc. of ACM Symp. on Theory of Computing* 2002, pages 438–447.

[4] 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, pages 440-451, 2001.

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

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

[7] L. K. Lenstra and D. Shmoys and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines", *Math. Programming*, 46, 259-271. (1990).

[8] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS '94)*, 202-213 (1994).

[9] R. Raz, S. Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. in *Proc. of ACM Symp. on Theory of Computing* 1997, pages 475-484.

[10] P. Raghavan and C. Thompson. Randomized Rounding, *Combinatorica*, volume 7, pages 365-374, 1987.