SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 401–427, August 1995

APPROXIMATION ALGORITHMS FOR MINIMUM-TIME BROADCAST *

GUY KORTSARZ† AND DAVID PELEG†

Abstract. This paper deals with the problem of broadcasting in minimum time in the telephone and message-passing models. Approximation algorithms are developed for arbitrary graphs as well as for several restricted graph classes.

In particular, an $O(\sqrt{n})$-additive approximation algorithm is given for broadcasting in general graphs, and an $O(\log n/\log\log n)$ (multiplicative) ratio approximation is given for broadcasting in the open-path model. This also results in an algorithm for broadcasting on random graphs (in the telephone and message-passing models) that yields an $O(\log n/\log\log n)$ approximation with high probability.

In addition, the paper presents a broadcast algorithm for graph families with small separators (such as chordal, k-outerplanar, bounded-face planar, and series-parallel graphs), with approximation ratio proportional to the separator size times $\log n$. Finally, an efficient approximation algorithm is presented for the class of graphs representable as trees of cliques.

Key words. broadcast, approximation

AMS subject classifications. 05C85, 68Q25, 90C27, 90B35

1. Introduction.

1.1. Minimum-time broadcast. One of the most important efficiency measures of a communication network is the speed by which it delivers messages between communicating sites. Therefore, much of the research in this area concentrates on developing techniques for minimizing message delay.

This work concerns efficient algorithms for broadcast in a communication network. The network is modeled by a connected graph and assumes the *telephone* communication model (cf. [HHL88]). In this model messages are exchanged during *calls* placed over edges of the network. A round is a series of calls carried out simultaneously. Each round is assumed to require one unit of time, so round t begins at time $t-1$ and ends at time t. A vertex may participate in at most one call during a given round, however, there are no limitations on the amount of information that can be exchanged during a given call. At a given round, if a call is placed over an edge e, we say that e is *active* in this round, otherwise it is *idle*. The set of rules governing the activation of edges at each round is called a *schedule*.

A *broadcasting* problem refers to the process where a distinguished vertex v originates a message M that has to become known to all other processors. The efficiency of a broadcast scheme is usually measured by the number of time units it takes to complete the broadcast. Given a scheme S for broadcasting in a graph G, denote the broadcasting time from v using S by $b(v,G,S)$. Define $b(v,G)$, the *broadcast time* of a vertex v in G, as the minimum time for broadcasting a message originating at v in

* Received by the editors March 22, 1993; accepted for publication (in revised form) June 1, 1994.

† Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel. The research of the second author was supported in part by an Allon Fellowship, a Bantrell Fellowship, a Walter and Elise Haas Career Development Award, and a grant from the Israeli Basic Research Foundation.

G, i.e.,

$$b(v, G) = \min_S \{b(v, G, S)\}.$$

We denote it simply by $b(v)$ when the context is clear. We denote

$$b(G) = \max_v \{b(v, G)\}.$$

Given a network $G = (V, E)$ and an originator u, the *minimum broadcast time* (MBT) problem is to broadcast the message from u to the rest of the vertices in $b(u)$ time units. This problem has received considerable attention in the literature. For example, broadcasting in trees is studied in [SCH81], and broadcasting in grid graphs is studied in [FH78]. For a comprehensive survey on the subject of gossiping and broadcasting see [HHL88].

The MBT problem in general graphs is NP-complete (cf. [GJ79]) and, thus, is unlikely to be solved exactly. However, several approaches toward coping with the MBT problem have been considered in the literature. A dynamic programming formulation for determining $b(v)$ and a corresponding broadcast scheme for an arbitrary vertex v are proposed in [SW84]. Since the exact algorithm is not efficient for large networks, several heuristics are presented in [SW84] for achieving a broadcast scheme with good performance.

Note that at each round, the informed vertices transmit the message outside to the uninformed vertices. Since calls are placed along nonadjacent edges, the set of active edges at each round constitutes a *matching* between the informed and the uninformed vertices. In general it may be preferable to transmit the message first to vertices with certain properties, e.g., to vertices whose degrees are maximal among the uninformed vertices and thus are likely to be able to inform a large number of vertices. Thus the approach of [SW84] is based on assigning weights to the vertices and looking for a matching that also maximizes the sum of the weights of the vertices. The drawback of such a heuristic approach is that it provides no guarantee on the performance of the algorithm.

Consequently, in this paper we consider *approximation schemes* for broadcast, namely, algorithms that may not give an exact solution but still give a solution that is *guaranteed* to be "not too far" from the optimum.

1.2. Related communication models and primitives. Most of our results in what follows are formulated for the broadcast operation in the telephone model. However, a number of these results apply directly, or with minor changes, to some other related communication primitives and models. Let us now introduce these alternate models and primitives.

A generalization of broadcast that we consider is to assume the set V_0 of informed vertices at the beginning of the run need not consist of a single vertex but can be an arbitrary subset of V. Denote this problem by SMBT, and denote the time needed to broadcast from V_0 by $b(V_0, G)$ or $b(V_0)$. As mentioned in [GJ79], SMBT is NP-complete even for $k = 4$ where k is the bound on the time for completing the broadcast.

Another important and well-studied communication primitive is the *gossip* operation. A gossip problem refers to the process of message dissemination, where each vertex v originates a message m_v and all messages have to become known to all vertices. The problem of performing the gossip primitive in minimum time is called the *minimum gossip time* (MGT) problem. The problem of efficient gossiping has also received considerable attention, mainly in the telephone model. For example, the papers [HMS72], [FP80] concern gossiping on the complete graph and on grid graphs, respectively.

Let us now turn to alternative communication models. Several generalizations of the telephone model appear in the literature. In [Far80], Farley suggests reconsidering the assumption that a vertex may call only neighboring vertices. Farley defines a possible variant of the model using long-distance calls, called the *open-path* model. In the open-path model, communication is carried along vertex disjoint paths. At each round, an informed member v may call an uninformed vertex u on an (arbitrarily long) path, adding u to the set of informed vertices. Two paths corresponding to two different pairs must be vertex disjoint. We note the time needed to complete the broadcast from a distinguished vertex v in the graph G in the open-path model, by $b_{\mathrm{op}}(v, G)$ (or $b_{\mathrm{op}}(v)$). We also denote

$$b_{\mathrm{op}}(G) = \max_{v \in V}\{b_{\mathrm{op}}(v, G)\}.$$

The problem of broadcasting from a vertex v in $b_{\mathrm{op}}(v, G)$ time units is referred to in what follows as OMBT.

In fact, Farley defines a second "long-distance" variant named the *open-line* model. This model is similar to the open-path model, except that the paths used in a communication round need only be *edge* disjoint. This model is less interesting for our purposes, since the problem of approximating broadcast in this model is essentially solved up to logarithmic factors. This is because, as shown in [Far80], the open-line model enables broadcast from an arbitrary vertex in $\lceil \log n \rceil$ time units.

Another common model of communication is the *message-passing* model, which is based on the assumption that a processor may send at most one fixed-size message in each time step, along one of its outgoing edges, but the communication pattern need not be a "matching." That is, it is possible that v_1 sends a message to v_2 while during the same round v_2 sends a message to v_3.

1.3. Contributions. In what follows we consider approximation schemes for broadcast (and some related primitives). Formally, we call an algorithm A for broadcasting on a family of graphs $\mathcal{F}$ a *k-approximation scheme* if for every $G \in \mathcal{F}$ and vertex $v \in V$,

$$b(v, G, A) \leq k \cdot b(v, G).$$

We say that a scheme S has a *(k, k')-approximation ratio* if

$$b(v, G, S) \leq k \cdot b(v, G) + k'.$$

A ratio is *k-additive* if it is an $(O(1), k)$-approximation ratio.

In this paper, we give approximation schemes for broadcast on several networks classes and analyze their approximation ratio.

The next two sections are dedicated to preparing the background for our approximation algorithms. Section 2 introduces the basic notions and definitions concerning transmission schedules and broadcast. Next, §3 presents some of our main technical tools. Specifically, it defines the *minimum weight cover* (MWC) problem and its close variant named the *minimum vertex weight cover* (MVWC) problem and provides a pseudopolynomial algorithm for solving them. The usefulness of this algorithm for handling transmission scheduling problems is demonstrated via a "toy example" of the *bipartite edge scheduling* (BES) problem. The MWC algorithm is used in virtually all our subsequent approximations for MBT.

In §4 we turn to approximations for the broadcast problem itself. To motivate the need for approximations, we examine three heuristics proposed in [SW84] for the

MBT problem, analyze their behavior on the wheel graph (see Fig. 2), and show that their output solution could be away from the optimum by a factor as high as $\sqrt{n}$ (on the n-vertex wheel). We then give an approximation algorithm for general n-vertex graphs with an $O(\sqrt{n})$-additive ratio.

Next, we show that in the open-path model, the OMBT problem has an approximation algorithm on arbitrary n-vertex graphs with a (multiplicative) approximation ratio $O(\log n/\log\log n)$. This algorithm has the additional desirable property that it solves the MBT problem (in the original telephone model) on random graphs (taken from the class $G_{n,p}$) and yields an approximation ratio $O(\log n/\log\log n)$ with high probability.

Section 5 presents an approximation scheme for broadcasting on graphs enjoying small separators, with approximation ratio $\varphi(n)\cdot\log n$ for graph families with separators of size $\varphi(n)$ (on n-vertex graphs). In particular, this algorithm yields an $O(\log n)$ approximation for *chordal* and $O(1)$-*separable* n-vertex graphs (including *series-parallel* graphs and k-*outerplanar* graphs for fixed k) and an $O(n^{1/4}/\sqrt{\log n})$ approximation for n-vertex *bounded-face* planar graphs.

Finally, in §6 we consider a special family of graphs called *trees of cliques,* generalizing trees, and give an approximation scheme for broadcasting on such graphs with additive-$O(\log^2 n)$ ratio.

Our results for the broadcast operation carry over to the gossip operation as well. To see this, note that in the telephone model, any scheme for broadcast can be used to perform the gossip operation. This is done as follows. First, fix a root vertex v and perform a *convergecast* operation (which is the opposite primitive to broadcast), collecting all the messages from the rest of the vertices to v, using the broadcast scheme in reverse. Then v, knowing all the information, performs a broadcast of the combined message. It follows that our results for MBT hold for MGT as well.

Although we formulate our statements in the telephone model, virtually all our results for broadcast hold also for the message-passing model, since as far as the broadcast operation is concerned, these two models are equivalent in power. (This is no longer the case for more involved operations, such as gossip.)

2. Preliminaries.

2.1. Graph definitions. Throughout this paper we use the following terminology to describe the behavior of the network. Our network is represented by a graph $G=(V,E)$, and we denote the number of vertices of a graph G by n and the number of edges by m.

Given a graph $G=(V,E)$ and two vertices $v,w\in V$, we denote the number of edges in a shortest path between v and w by $\mathrm{dist}(v,w)$. We denote $\mathrm{Diam}(v)=\max_w\{\mathrm{dist}(v,w)\}$. The diameter of the graph G is $\mathrm{Diam}(G)=\max_v\{\mathrm{Diam}(v)\}$.

A *cluster* in a graph G is a subset V' of the vertices such that the subgraph induced by V' is *connected.* Two clusters V',V'' are said to be *disjoint* if $V'\cap V''=\emptyset$. Two disjoint clusters C_1 and C_2 are said to be *independent* if there is no edge connecting a vertex of C_1 to a vertex of C_2.

DEFINITION 2.1. *Let $G=(V,E)$ be a graph, and let $S\subset V$ be a subset of the vertices. A subtree $T=(V_1,E_1)$ of G rooted at a distinguished vertex v_0 is a* shortest paths tree (*SPT*) leading from v_0 to S *iff $S\subseteq V_1$, each path from v_0 to $v_i\in S$ in T is a shortest path in G, and every leaf of T belongs to S. Denote an SPT leading from a vertex v_0 to a set S by $SPT(v_0,S)$.*

We now state the definition of a *control graph* of a subset $V_0\subseteq V$, in a graph $G=(V,E)$. This definition will be useful in most of our approximation algorithms.

DEFINITION 2.2. *Suppose that the clusters (i.e., connected components) formed when extracting V_0 from the graph G are $\{C_1, \ldots, C_k\}$. The* control graph *of V_0 in G is a bipartite graph $D_{V_0,G} = (V_1, V_2, A)$, where $V_1 = V_0, V_2 = \{C_1, \ldots, C_k\}$, and A contains an edge (v, C_i) iff there is an edge between v and some vertex of C_i in G.*

2.2. Schedules and broadcast.

DEFINITION 2.3. *Given a graph $G = (V, E)$, two edges $e_1, e_2 \in E$ are called* adjacent *iff they share* exactly *one vertex and* nonadjacent *otherwise. A subset of edges $E' \in E$ is an* independent set *(of edges) iff every two edges $e_1, e_2 \in E'$ are nonadjacent.*

Fact 2.4. At each round t, the set of active edges is independent.

The MBT problem is formally defined as follows. Let $v_0 \in V$ be a distinguished vertex. A broadcast from v_0 is a sequence

$$\{v_0\} = V_0, E_1, V_1, E_2, \ldots, E_k, V_k = V$$

such that for $1 \leq i \leq k$, the following hold.

1. $V_i \subseteq V$ and $E_i \subseteq E$.
2. Each edge in E_i has exactly one end point in V_{i-1}.
3. The set E_i is an independent set of edges.
4. $V_i = V_{i-1} \cup \{v : (u, v) \in E_i\}$.

In this case we say that the broadcast is performed in k time units. The MBT problem concerns looking for the minimal k such that broadcasting in k time units is possible, for a given graph G and vertex v_0. This k is denoted $b(v_0, G)$.

For any connected graph G of n vertices and originator $u, b(u) \geq \lceil \log n \rceil$,[1] since in each time unit the number of informed vertices can at most double. Another simple lower bound for $b(v)$ for an arbitrary v is $b(v) \geq \text{Diam}(v)$, since a vertex may only send information to a neighboring vertex at each round. An example of a graph for which $b(G) = \lceil \log n \rceil$ is K_n, the complete graph of n vertices.

In any connected graph G, a broadcast from a vertex u determines a spanning tree rooted at u. The parent of a vertex v is the vertex w that transmitted the message to v. Clearly, one may assume that such a vertex is unique. Even when using an arbitrary spanning tree, it is clear that at each step the set of informed vertices grows by at least one. Thus for each network $G, b(G) \leq n - 1$. We cannot always improve upon this result. For example, in S_n, the star of n vertices, the broadcast time is $b(S_n) = n - 1$. We summarize this discussion as follows.

Fact 2.5.

1. For every graph $G = (V, E)$ and vertex $v \in V, \lceil \log n \rceil \leq b(v) \leq n - 1$.
2. $b(K_n) = \lceil \log n \rceil$.
3. $b(S_n) = n - 1$.
4. For every graph $G = (V, E)$ and vertex $v \in V, b(v, G) \geq \text{Diam}(v)$.

Note that Fact 2.5 holds in the open-path model as well, except of course for claim 4; we cannot argue that $b_{\text{op}}(G) \geq \text{Diam}(G)$.

3. The minimum-weight cover problem. The MWC problem is a basic tool we use in our approximation algorithms for MBT. In this section we define the problem and provide a solution for it.

3.1. The problem. In order to describe the MWC problem we need some preliminary definitions. Let $G = (V_1, V_2, A, \omega)$ be a bipartite graph with bipartition

[1] All logs in this paper are taken to base 2.

(V_1, V_2), edge set A, a weight function $\omega : A \mapsto \mathbb{Z}^+$ on the edges, and no isolated vertices. A feasible solution to the MWC problem is a *control* function $F : V_2 \to V_1$, where $F(v_2) = v_1$ implies $(v_1, v_2) \in A$. Each vertex $v_1 \in V_1$ is called a *server,* and each vertex $v_2 \in V_2$ is called a *customer.* If $F(v_2) = v_1$ we say that v_1 *controls* (or *dominates*) v_2.

We adopt the following notational convention.

DEFINITION 3.1. *Consider a bipartite graph $G = (V_1, V_2, A, \omega)$ as above and a control function F. For each server $v \in V_1$, we denote the clients dominated by v by $\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)$ and the edges connecting them to v by $e_i^v = (v, \mathcal{D}_i(v))$. Furthermore, we assume (without loss of generality) that these vertices are ordered so that $\omega(e_i^v) \geq \omega(e_{i+1}^v)$ for every i.*

The weight *of F is defined as*

$$\mathcal{W}(F) = \max_{v \in V_1}\{\max_i\{i + \omega(e_i^v)\}\}.$$

The MWC problem is now defined as follows. Given a bipartite graph $G = (V_1, V_2, A, \omega)$ as above, determine a control function $F : V_2 \to V_1$ whose weight $\mathcal{W}(F)$ is minimal. We call this function F the *minimum control function* for G (or just the minimal function).

It is important to note that in all the applications to the MWC problem given in this paper, the weights satisfy $\omega(e) \leq n$. Thus, in order to use this problem as a basic auxiliary tool for the study of MBT, a pseudopolynomial solution for it will suffice.

A special variant of the MWC problem arises when for each $v_2 \in V_2$ the weights of the edges entering v_2 are identical. In this case, we might as well associate the weight with v_2 itself. We call this variant of the problem the MVWC problem. If *all* the weights are identical (thus without loss of generality all the weights are 0), a solution only needs to minimize the maximal number of vertices dominated by a single vertex of V_1, i.e., minimize the size of the largest inverse image of F; hence, it is possible to use the modified weight function

$$\omega'(F) = \max_{v_1 \in V_1} |\{v_2 \in V_2 : F(v_2) = v_1\}|.$$

3.2. An algorithm for MWC. This subsection presents a pseudopolynomial solution to the MWC problem. The algorithm is based on a procedure FLOW, which given an instance $G = (V_1, V_2, A, \omega)$ and a positive integer j, checks if there exists a control function F for G of weight $\mathcal{W}(F) \leq j$. This procedure is then used to search for the minimum control function by going over the possible j values.

The solution method employed by Procedure FLOW for this problem involves flow techniques. Specifically, the procedure constructs a modified *flow graph* $\hat{G}_j$ based on G and j, with the property that G has a control function F with weight $\mathcal{W}(F) \leq j$ iff it is possible to push $|V_2|$ units of flow from the source to the sink on $\hat{G}_j$.

The graph G is modified by Procedure FLOW into $\hat{G}_j$ as follows. Create a source vertex s and a sink vertex t. Notice that for the function F to be of weight j, a server v cannot dominate a customer u such that $\omega(v, u) \geq j$. Assume that ω_v is the maximal weight that is less than or equal to $j - 1$ of an edge incident to $v \in V_1$. Duplicate v into $\omega_v + 1$ different copies and arrange the copies in an arbitrary order $v_1, \ldots, v_{\omega_v+1}$. For v_1, the first copy of v, create a directed edge (s, v_1) with capacity $j - \omega_v$ and a directed edge (v_1, u) with capacity 1, from v_1 to every customer $u \in V_2$ such that $(v, u) \in A$. For v_i the ith copy of $v, i \geq 2$, create a directed edge (s, v_i) with capacity 1

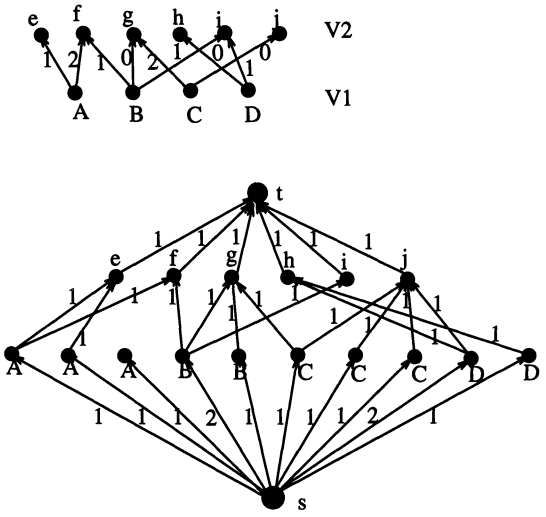

FIG. 1. *A bipartite graph G with its weights, and the corresponding flow graph* $\hat{G}_3$.

and a directed edge (v_i, u) with capacity 1 to all the customers u such that $(v, u) \in A$ and $\omega(v, u) \leq \omega_v - i + 1$. Finally create for each customer $u \in V_2$ a directed edge (u, t) with capacity 1. See Fig. 1 for an example of a graph G and the associated flow graph $\hat{G}_3$.

PROCEDURE FLOW

Input: a graph $G = (V, E)$ and an integer j.

1. Construct the flow graph $\hat{G}_j$.
2. Compute the maximal flow on $\hat{G}_j$ from s to t.

Since there are exactly $|V_2|$ edges entering t and each of them is of capacity 1, the maximal flow from s to t cannot exceed $|V_2|$. We now claim the following lemma.

LEMMA 3.2. *Consider an MWC problem on a given graph G and the corresponding flow graph* $\hat{G}_j$ *constructed by procedure* FLOW *for some integer j. Then the maximal flow from s to t on* $\hat{G}_j$ *is* $|V_2|$ *iff there exists a control function F for G such that* $\mathcal{W}(F) \leq j$. *Furthermore, the required control function F for G can be computed efficiently from the flow assignment for* $\hat{G}_j$.

Proof. For the first direction, we assume that there exists an integral flow function for $\hat{G}_j$ with $|V_2|$ flow units entering t and show how to (efficiently) construct the required control function F.

Let us note that since each edge pointing from a vertex $v_2 \in V_2$ to t has capacity 1 and all the edges entering v_2 have capacity 1 as well, only a single edge entering v_2 may carry positive flow of one unit. Hence since the total flow is $|V_2|$, one such edge must exist for each vertex $v_2 \in V_2$. Consequently, define $F(v_2) = v_1$ such that v_1 is the vertex for which the flow through (v_1, v_2) is 1.

We need to show that this definition of F meets the requirement, namely, $\mathcal{W}(F) \leq j$. This is shown as follows. Note that since the total flow entering all the copies of a vertex $v \in V_1$ does not exceed j, each server of V_1 may dominate no more than j customers in V_2. For every server $v \in V_1$ and integer m, denote the number of v's customers $\mathcal{D}_l(v)$ such that $\omega(e_l^v) \geq m$ by $r_m(v)$; alternatively, recalling that the edges

e_l^v are ordered in nonincreasing weight order, we have

$$r_m(v) = \max\{l : \omega(e_l^v) \geq m\}.$$

We argue the following.

CLAIM 3.3. $r_m(v) \leq j - m$.

Proof. Consider first the value $m = \omega_v$. Since only the first copy of v can dominate vertices u such that $\omega(v,u) = \omega_v$, v does not dominate more than $j - \omega_v$ such vertices.

Now consider $m < \omega_v$. Note that only the copies number $2, 3, \ldots, \omega_v - m + 1$ can aid in increasing the value of $r_m(v)$, each by at most 1. Thus $r_m(v) \leq j - \omega_v + (\omega_v - m + 1 - 2) + 1 = j - m$. □

COROLLARY 3.4. *Every* $1 \leq i \leq k$ *satisfies* $i \leq j - \omega(e_i^v)$.

Proof. Let $m = \omega(e_i^v)$. By the last claim, $r_m(v) \leq j - m$, so it remains to show that $i \leq r_m(v)$. This follows readily from the definition of $r_m(v)$. □

The remainder of the proof of this direction follows in a straightforward way from Definition 3.1: since by the last corollary $\max_i\{i + \omega(e_i^v)\} \leq j$ for every j, it follows that $\mathcal{W}(F) \leq j$ as well.

To prove the other direction, assume that there exists a control function F for G such that $\mathcal{W}(F) \leq j$. Thus each server dominates no more than j customers, and furthermore v does not dominate more than $j - i$ customers whose corresponding weights are i or larger. We now use F to define a flow function for the flow graph $\hat{G}_j$ as follows.

Consider a server v. As before, order its customers $\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)$ by nonincreasing weights of their edges. Augment the flow through v_1, the first copy of v, by $j - \omega_v$, adding a flow of 1 from v_1 to each of $\mathcal{D}_1(v), \ldots, \mathcal{D}_{j-\omega_v}(v)$. The weight of the next client of v, namely, $\mathcal{D}_{j-\omega_v+1}(v)$, is smaller than ω_v, and therefore there is a directed edge from the second copy, v_2, to that vertex. Thus it is still possible to augment the flow by 1 through v_2. In a similar way, it is possible to continue this process and augment the flow by k through all of v's clients, $\{\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)\}$. Since this is true for any server, it follows that in total, a maximal flow of $|V_2|$ can be attained. □

The minimum weight control function itself can now be found by using procedure FLOW within a binary search on the possible range of j values. Formally, we do the following.

PROCEDURE MWC

1. Start with $j_1 \leftarrow \min_e\{\omega(e)\} + 1$ and $j_2 \leftarrow \max_e\{\omega(e)\} + |V_2|$.
2. **repeat**
   (a) $j \leftarrow \lceil \frac{j_1 + j_2}{2} \rceil$.
   (b) Apply Procedure FLOW to G with j.
   (c) If the maximal flow is $|V_2|$
   /* hence there exists a control function F of weight j */
   then set $j_1 \leftarrow j$, else set $j_2 \leftarrow j$.
   until $j = j_2 \leq j_1 + 1$.
3. Return the minimum control function F corresponding to the maximal flow computed on $\hat{G}_{j_1}$.

Clearly, when this process terminates, we are left with a minimum control function F, whose weight is $\mathcal{W}(F) = j_1$ for the value of j_1 upon termination.

The flow computation is performed for at most a polynomial number of times. Note, however, that a vertex may be duplicated in a number of copies that can equal

the maximal number in the input; hence the solution is not strongly polynomial in the input. To summarize, we have established the following.

THEOREM 3.5. *There exists a pseudopolynomial algorithm for solving the MWC problem.*

Since MVWC is a special case of MWC, we can deduce the following corollary.

COROLLARY 3.6. *There exists a pseudopolynomial algorithm MVWC for solving the MVWC problem.*

3.3. The bipartite edge scheduling problem. Before embarking on our main task of approximating broadcast problems, let us first try to demonstrate the way in which the MWC algorithm can be used for this purpose. To do that, we introduce a model problem called the BES problem, which will serve as an illustrative example for the usefulness of MWC.

The BES problem is defined as follows. Assume that we are given a bipartite graph $G = (V_1, V_2, A)$ where the vertices of V_1 know an `initiation` message that must be broadcast to the vertices of V_2. That is, the initial set of informed vertices is V_1. Again, call each vertex in V_1 a *server* and each vertex in V_2 a *customer.*

Further suppose that each customer $v_2 \in V_2$ has a task t_{v_2} to perform. The customer must first receive the `initiation` message before it can start performing its task. Moreover, the *length* of the task t_{v_2} (i.e., the time it takes v_2 to complete it) depends upon the *source* of the `initiation` message, namely, which vertex of V_1 transmits the message to v_2. Formally, for each edge $e = (v_1, v_2) \in A$ there is a weight $\omega(e) \in \mathbb{Z}^+$, such that if v_2 receives the `initiation` message from v_1, then it takes $\omega(e)$ time units for v_2 to complete the task t_{v_2}, starting from the arrival time of the message.

The BES problem is to minimize the completion time of the entire process, namely, the time by which every vertex in V_2 completes its job. The *bipartite vertex scheduling* (BVS) problem is the variant of BES in which every vertex $v' \in V_2$ has entering edge weights that are identical (and hence can be thought of as vertex weights, $\omega(v')$).

To see the relationship between the BES problem and the previously discussed MWC problem, note that MWC is in fact the main component in solving the problem, since once a control function F is defined for the graph G, it is intuitively most efficient for each server v to send the `initiation` message along its edges in nonincreasing order of weights. It is therefore easy to see that the following (pseudopolynomial) procedure will be optimal for BES.

ALGORITHM BES

1. Apply Procedure MWC to compute a minimal control function F for G.
2. Every server $v \in V_1$ sends the `initiation` message to its clients $\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)$ in this order.
   /* recall that clients are ordered by nonincreasing edge weights */
3. Each customer $v' \in V_2$ starts performing its task immediately after it is informed.

Fact 3.7. Algorithm BES optimally schedules any BES instance.

Since BVS is a special case of BES, we can deduce the following corollary.

COROLLARY 3.8. *There exists an optimal pseudopolynomial algorithm for BVS.*

Example. In order to demonstrate the relevance of the BES for broadcasting, let us consider the following restricted variant of SMBT. Let V_0 be the base set of an SMBT instance. Suppose that when extracting V_0 from the graph, the remainder of the graph becomes disconnected and breaks into k (connected) clusters $C_1, \ldots, C_k$.

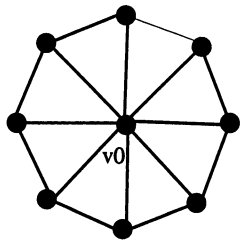

FIG. 2. *The wheel of nine vertices.*

Furthermore, suppose that every cluster C_i contains a *representative* vertex v_i, such that every vertex $v \in V_0$ is either connected to v_i or not connected to any vertex of C_i. Finally, assume that we can optimally compute the broadcast times $b(v_i, C_i)$ for every cluster C_i (e.g., when the clusters C_i are, for example, trees or grids or of logarithmic size).

Intuitively, we can view this variant of the broadcasting problem as a special case of the BVS problem, as follows. Form the control graph $D_{V_0,G}$ of V_0 in G as in Definition 2.2. Define the weight $\omega(C_i)$ on a vertex $C_i \in V_2$ as the time required by the representative vertex, v_i, for broadcast in C_i. Now apply the BVS algorithm to this graph, taking V_1 to be the base set and V_2 to be the collection of clusters $\{C_1, \ldots, C_k\}$ and regarding the *task* of each vertex C_i in V_2 as performing broadcast in the cluster by the representative vertex v_i. Since in each of the clusters only the representative vertex is connected to the "outside world," the vertices outside the cluster cannot aid in this internal broadcast process; hence, the vertex weights on V_2 are defined appropriately.

It is straightforward to show that the solution to the BVS problem yields exactly the minimum broadcast time from V_0. Further, note that since the vertex weights in this problem represent broadcast times in a cluster of the graph, it follows from Fact 2.5 that they are no larger than $n-1$, and hence the BVS instance can be solved polynomially.

4. Approximating broadcast in general graphs. In this section we give an approximation scheme for broadcasting in general graphs, both in the telephone model and in the open-path model. We begin by motivating the need for approximation schemes for broadcasting.

4.1. The heuristic approach. In this subsection we demonstrate the fact that the natural heuristics proposed in [SW84] might be inadequate in some simple cases. The example considered is a *wheel,* i.e., a cycle of $n-1$ vertices numbered $1, \ldots, n-1$, arranged by increasing order of indices, with an extra vertex v_0 connected to all the vertices in the cycle (Fig. 2). Let us first give tight bounds on the minimum broadcast time from v_0.

LEMMA 4.1. *In the n-vertex wheel,* $b(v_0) = \Theta(\sqrt{n})$.

Proof. First let us show that $b(v_0) = \Omega(\sqrt{n})$. Assume that the broadcast takes k time units. The vertex v_0 can inform up to k different vertices. Therefore there is a vertex w whose distance on the ring from the vertices directly informed by v_0 is at least $\lfloor \lceil (n-1)/k \rceil / 2 \rfloor$. Thus the time for informing w is at least $\lfloor \lceil (n-1)/k \rceil / 2 \rfloor + 1$. The minimum broadcast time is achieved when $\lfloor \lceil (n-1)/k \rceil / 2 \rfloor + 1 = k$, in which case $k \geq \sqrt{(n-1)/2}$, yielding the desired lower bound on the broadcast time.

To prove that $b(v_0) = O(\sqrt{n})$, note that if v_0 informs all vertices whose index

is congruent to 1 mod$\lceil\sqrt{n-1}\rceil$ and then these cycle vertices inform the rest of the vertices, the total time for broadcasting is no more than $3 \cdot \lceil\sqrt{n-1}\rceil/2 + 1$ time units. □

Now consider the following three heuristics suggested in [SW84] for MBT. The first heuristic is based on defining, for a set of vertices $V' \subseteq V$,

$$D_G(V') = \sum_{v \in V'} d(v).$$

At each round i, select from all possible maximum matchings between the set of informed vertices and the set of uninformed vertices, a matching satisfying that the set $V'' \subseteq V \setminus V_i$ of "receiving" end vertices in the matching has maximal $D_G(V'')$. Let us describe a possible broadcast scenario. At the first round, v_0 delivers the message to the vertex 1. At round 2, v_0 calls $n-1$ and 1 calls 2. It is easy to see that starting from the third round one can choose a maximal matching that enlarges the set of informed vertices by 3 at each round. For example, at the third round v_0 may call $n-3, n-1$ calls $n-2$, 2 calls 3, and so on. Note that the above scenario conforms with the given heuristic, since the degree of all the vertices in the wheel (except v_0) is 3. Thus using this heuristic, it might take $\Omega(n)$ time units to complete the broadcasting.

A second heuristic approach suggested in [SW84] is the following. Define the *eccentricity* of a set $V' \subset V$ to be

$$\mathrm{Dist}(V') = \sum_{v \in V'} \mathrm{Diam}(v).$$

Choose, among all the possible maximal matchings, a matching for which the eccentricity of the set of newly informed vertices is maximal. In a way similar to that above, it is easy to see that there are cases in which this approach leads to a broadcast time of $\Omega(n)$. The last heuristic suggested in [SW84] is a combination of the former two, and it, too, may lead to an $\Omega(n)$ broadcast scheme.

It follows that, for this example, all of these heuristics may yield broadcast times that are $\Omega(\sqrt{n})$ times worse than optimal. We do not know whether this ratio is guaranteed by any of the three heuristics.

4.2. Approximating MBT. In this subsection we consider approximation schemes for broadcasting in general graphs. By Fact 2.5, the scheme that chooses an arbitrary broadcast tree is at worst an $(n-1)/\lceil\log n\rceil$ approximation scheme. We improve upon this approximation ratio and present an algorithm that guarantees an $O(\sqrt{n})$-*additive* ratio.

The method used for the approximation is based on dividing the set of vertices into clusters of size $\lceil\sqrt{n}\rceil$ and broadcasting separately on those clusters. Let us start by describing the various tools used by the algorithm. At the heart of our scheme is the following decomposition lemma.

LEMMA 4.2. *The set of vertices of any graph $G = (V, E)$ can be (polynomially) decomposed into two sets of clusters $\mathcal{A}$ and $\mathcal{B}$, such that $|\mathcal{A}| \leq \sqrt{n}$, the clusters in $\mathcal{A} \cup \mathcal{B}$ are pairwise disjoint, $(\bigcup \mathcal{A}) \cup (\bigcup \mathcal{B}) = V$, the size of each cluster $C' \in \mathcal{A}$ is $|C'| = \lceil\sqrt{n}\rceil$, the size of each cluster $C' \in \mathcal{B}$ is bounded by $|C'| \leq \sqrt{n}$, and the clusters in $\mathcal{B}$ are pairwise independent.*

Proof. The proof the lemma follows by direct construction. Let us next present the decomposition procedure. The algorithm maintains three different sets of clusters $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$.

PROCEDURE DECOMPOSE

1. At the start $\mathcal{C} \leftarrow \{V\}, \mathcal{A} \leftarrow \emptyset, \mathcal{B} \leftarrow \emptyset$.
2. **repeat**
   (a) Choose a cluster C in $\mathcal{C}$ and an arbitrary (connected) subcluster C' of C such that $|C'| = \lceil \sqrt{n} \rceil$.
   (b) Remove C' from C, and set $\mathcal{A} \leftarrow \mathcal{A} \cup \{C'\}$.
   /* Now $C \setminus C'$ is composed of several independent clusters. */
   Add the clusters $\{C'' : C'' \subseteq C \setminus C', |C''| = \lceil \sqrt{n} \rceil\}$ to $\mathcal{A}$.
   Add the clusters $\{C'' : C'' \subseteq C \setminus C', |C''| > \lceil \sqrt{n} \rceil\}$ to $\mathcal{C}$.
   Add the remaining clusters in $C \setminus C'$ to $\mathcal{B}$.
   until $\mathcal{C} = \emptyset$.

It is clear from the construction that all the clusters in $\mathcal{A}$ have exactly $\lceil \sqrt{n} \rceil$ vertices. Thus the number of clusters in $\mathcal{A}$ cannot exceed $\sqrt{n}$. It is also clear that the number of vertices in each cluster in $\mathcal{B}$ is no more than $\sqrt{n}$.

To prove that the clusters in $\mathcal{B}$ are independent, it is sufficient to claim that at each stage, all the clusters of $\mathcal{B} \cup \mathcal{C}$ are such. The proof is by induction on the stage number. At the basis $\mathcal{B} = \mathcal{C} = \emptyset$ and the claim follows vacuously. Now assume that after stage i all the clusters in $\mathcal{C}$ and $\mathcal{B}$ are pairwise independent. In the next stage we exclude a subset of vertices from some cluster $C \in \mathcal{C}$; thus the remaining vertices decompose into independent clusters. The clusters added to $\mathcal{B}$ and $\mathcal{C}$ are a subset of those clusters; thus the claim follows by this fact and the induction hypothesis. □

Note that the construction of Procedure DECOMPOSE in the proof of Lemma 4.2 allows us to find such a decomposition in $O(E \cdot \sqrt{V})$ time, since checking which are the newly formed clusters at each stage takes $O(E)$ time.

Our next tool exploits the use of a minimum control function for broadcast. Let $G = (V, E)$ be a graph and $V' \subset V$ subset of the vertices. Form the control graph of V' in $G, D_{V',G} = (V_1, V_2, A)$, as in Definition 2.2. Let the weight of each edge be 0. In this scenario we claim the following.

LEMMA 4.3. *Let F be a minimum control function for $D_{V',G}$. Then $\mathcal{W}(F) \leq b(V', G)$.*

Proof. Consider an optimal communication scheme S from V' on G. For every cluster $C_i \in V_2$, choose *one* of the vertices that is among the *first* to transmit the message to a vertex in C_i. Since the clusters C_i are pairwise independent, such a vertex is in V' for each i. The above determines a function F' from V_2 to V_1.

Consider an arbitrary vertex $v \in V'$. If v dominates j clusters, then there is a cluster $C_i \in V_2$ such that v transmits the message to C_i in the jth round (or later). By the way F' was chosen, $j \leq b(V', G)$. Thus by definition, $\mathcal{W}(F') \leq b(V', G)$. However, the weight of F satisfies $\mathcal{W}(F) \leq \mathcal{W}(F')$. The proof follows. □

Our final tool involves broadcasting on a tree. Recall that given a tree $T = (V_1, E_1)$ and a vertex $v \in V_1$ it is easy to compute the optimal scheme for broadcasting on T from v [SCH81]. Let us call the optimal scheme for broadcasting in a tree the *OT scheme.*

In the next lemma we use a shortest paths tree $\mathrm{SPT}(v, S)$ rooted at a vertex v and leading to a set S of vertices (see Definition 2.1). Note that it is easy to construct such a tree in time polynomial in $|E|$ using a shortest path tree algorithm; simply construct a shortest path tree T spanning all the graph vertices, and iteratively exclude from it each leaf not belonging to S, until no such leaf exists.

LEMMA 4.4. *Transmitting a message from a vertex v to a subset $V' \subseteq V, |V'| = l$ of the graph, can be performed in no more than $l - 1 + \mathrm{Diam}(v)$ time units.*

Proof. Construct an arbitrary tree SPT(v, V') rooted at v and leading to the l vertices of V'. Use the OT scheme to broadcast the message to all the members of the tree. Consider any leaf u. We would like to show that u gets the message within the specified time bounds. This is done by "charging" each time unit elapsing until u gets the message to a distinct vertex of the tree and then bounding the number of charges.

Consider the situation immediately after an ancestor v' of u receives the message. The vertex v' is currently the lowest ancestor of u that knows the message. Thereafter v' starts delivering the message to its children. When v' delivers the message to a child whose subtree T' does not include u, choose arbitrarily a leaf in T' and charge this time unit to the leaf. When v' delivers the message to the ancestor of u, charge this time unit to v'.

Note that at every time unit we charge a single vertex, on account of u, and thus the total number of units charged is exactly the time before the message reaches u. Also note that no leaf is charged twice and that u is not charged. Finally note that every ancestor of u (except u itself) is charged once. Thus the time it takes the message to reach u is bounded by Diam(v) plus the number of leaves in T beside u. Since each leaf is a member of V', the proof follows. □

We are now ready to combine the three tools discussed in the above lemmas into an algorithm, named APPROX_MBT, for approximate broadcast on general graphs.

ALGORITHM APPROX_MBT

Input: a graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.

1. Decompose the vertices of V into two sets of clusters $\mathcal{A}$ and $\mathcal{B}$ using Procedure DECOMPOSE of Lemma 4.2.
2. Choose for each cluster C in $\mathcal{A}$ a single representative vertex v_C. Let R denote the set of representatives, $R = \{v_C | C \in \mathcal{A}\}$.
3. Transmit the message from v_0 to all the vertices of R by choosing an arbitrary tree SPT(v_0, R) leading from v to R and applying the OT scheme to the tree.
4. Choose for each cluster $C \in \mathcal{A}$ an arbitrary spanning tree rooted at its representative v_C, and broadcast (in parallel) in the clusters of $\mathcal{A}$ according to the OT scheme.
5. Construct the bipartite control graph $D_{V_0,G}$ where $V_0 = (\bigcup \mathcal{A}) \cup \{v_0\}$. Compute a minimum control function F on $D_{V_0,G}$ using Procedure MWC. Assume that a vertex v' dominates clusters $C_1, \ldots, C_k \in \mathcal{B}$. Choose for each C_i an arbitrary vertex $v_i \in C_i$ connected to v', and deliver the message from v' to $v_1, \ldots v_k$ (in arbitrary order). This is done in parallel for all the dominating vertices of V_0.
6. Choose for each cluster in $\mathcal{B}$ an arbitrary spanning tree rooted at an informed vertex, and transmit the message in parallel to all the vertices in the clusters of $\mathcal{B}$ using the OT scheme in each cluster.

THEOREM 4.5. *The broadcast time of Algorithm* APPROX_MBT *from a vertex* v_0 *in a graph* G *is bounded by* $3 \cdot \sqrt{n} + \text{Diam}(v_0) + b(v_0)$ *time units.*

Proof. By Lemmas 4.4 and 4.2 the time it takes to complete stage 2 is bounded by $\sqrt{n} - 1 +$ Diam(v). The fact that each cluster in $\mathcal{A}$ has exactly $\lceil \sqrt{n} \rceil$ vertices and Fact 2.5 imply that stage 3 takes no more than $\sqrt{n}$ time units. By Lemma 4.3, $\mathcal{W}(F) \leq b((\bigcup \mathcal{A}) \cup \{v_0\}) \leq b(v_0)$; thus stage 4 takes no more than $b(v_0)$ time units. Finally, the fact that the clusters in $\mathcal{B}$ are of size no larger than $\sqrt{n}$ implies that stage 5 takes no more than $\sqrt{n} - 1$ time units. □

The ratio guaranteed by the theorem is $O(\sqrt{n})$-additive. Consequently, whenever the broadcast time of a network is $\Omega(\sqrt{n})$, e.g., in the wheel of n vertices, the scheme of Algorithm APPROX_MBT is a constant approximation scheme. For example, for each network whose diameter is at least $\sqrt{n}$, the above is a 5 approximation scheme. However, in the general case, the optimal broadcasting scheme may achieve time that is close to $\lceil \log n \rceil$. Thus, in the general case our method is a $(3 \cdot \sqrt{n}/\lceil \log n \rceil + 2)$-approximation scheme and also an $O(\sqrt{n}/\mathrm{Diam}(G))$-approximation scheme. In order to improve upon this it may be necessary to achieve, say, a good approximation scheme for networks of "small" diameter.

4.3. Broadcasting in the open-path model. Let us turn to the open-path communication model. Algorithm APPROX_MBT can be generalized to give a good approximation scheme for the open-path broadcasting problem. It is easy to see that Lemma 4.3 holds even in the open-path model.

LEMMA 4.6. *Let T be a tree rooted at v, with up to k leaves. Then it is possible to broadcast a message from the root v to all the vertices of the tree in the open-path model, in no more than $2 \cdot k + \log n$ time units.*

Proof. We first recall the following fact, proven in [Far80].

Fact 4.7 [Far80]. Broadcasting in the open model on a path of m vertices can be completed in $\lceil \log m \rceil$ time units.

To prove Lemma 4.6 we give a two-stage procedure. In the first stage we proceed in the following recursive fashion. As soon as a vertex v' is informed, it checks if the subtree $T_{v'}$ rooted at it contains a vertex of degree at least 3, i.e., with at least two children. If no such vertex exists (i.e., the subtree $T_{v'}$ is a path), then v' does nothing. Otherwise, assume that the highest such vertex in the tree rooted at v' is v''. That is, the subtree $T_{v'}$ is composed of a path connecting v' to v'', plus the subtree $T_{v''}$ (note that possibly $v'' = v'$). Then v' makes a long-distance call to all the *children* of v'', in arbitrary order.

It is easy to see that when this first stage is finished, the tree can be decomposed into a union of disjoint paths where for each path, one end vertex knows the message. During the second stage, each informed end vertex of each path informs the rest of the vertices in the path as in Fact 4.7.

In analyzing the delays occurring before a message reaches a leaf u, we separate our analysis to the first and second stages. This first stage is handled by a charging rule somewhat similar to that used in the proof of Lemma 4.4. Note that at each time step t, there is a unique ancestor $\mathrm{low}(u,t)$ of u that is responsible for it, namely, the lowest ancestor currently holding the message. At time step t, $\mathrm{low}(u,t)$ sends the message to some child v'' of a vertex v' with at least two children. If u does not belong to the subtree $T_{v''}$, charge a delay to an arbitrary leaf in $T_{v''}$. When the message is sent to a child v'' of a vertex v' such that v'' is a ancestor of u, charge the delay to v'. Note that only leaves and vertices with degree at least 3 are charged, and none of them is charged more than once. The number of vertices of degree 3 or higher is no more than $k-2$; hence, the total delay in stage 1 is bounded by $2 \cdot k - 2$.

In the second stage the broadcast takes place along vertex disjoint paths, each with no more than n vertices. Hence by Fact 4.7 this stage can be completed in $\lceil \log n \rceil$ time units. In total, the communication delay is bounded by $2 \cdot k - 2 + \lceil \log n \rceil$. □

This discussion motivates the following approach for approximating OMBT. First we define sets of representatives, $\{R_1, \ldots, R_f\}$, where $R_1 = V, |R_f| \leq \log n$, and $f \leq \log n / \log \log n$. To each set R_j and vertex $v \in R_j$ there is a corresponding tree T_j^v, containing at least $\lceil \log n \rceil$ vertices of R_{j-1}. The trees corresponding to different

vertices in R_j are *vertex disjoint.*

The main algorithm operates in f stages. The first stage informs the vertex set R_f. The algorithm then proceeds to inform the sets R_j in reverse order of indices, i.e., at the end of stage i, the message is known by the set R_{f-i+1}, and the goal of the next stage is for R_{f-i+1} to inform R_{f-i}.

We next present Procedure CHOOSE_REP, whose task is to choose the sets R_i of representatives and the corresponding trees $T_i^v, v \in R_i$. After that, we give the main algorithm that uses Procedure CHOOSE_REP to approximate OMBT.

Throughout the execution of Procedure CHOOSE_REP we extract trees from G.

PROCEDURE CHOOSE_REP

Input: A graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.

1. $R_1 \leftarrow V, i \leftarrow 1$.
2. **repeat**
   (a) $\mathcal{C} \leftarrow \{V\}, R_{i+1} \leftarrow \emptyset$.
   (b) **while** $\mathcal{C} \neq \emptyset$ **do:**
      (i) Choose cluster $A \in \mathcal{C}$. Select $\lceil \log n \rceil$ vertices in $A \cap R_i$ arbitrarily, except that if $v_0 \in A$, take v_0 as one of them. Let the chosen vertices be $v_1, \ldots, v_{\lceil \log n \rceil}$, such that we set $v_1 = v_0$ if $v_0 \in A$. Select in A a subtree $T_i^{v_1}$ leading from v_1 to $\{v_2, \ldots, v_{\lceil \log n \rceil}\}$.
      (ii) Extract $T_i^{v_1}$ from A.
      (iii) Set $\mathcal{C} \leftarrow \mathcal{C} \cup \{B : B$ is a connected component of $A \setminus T^{v_1}, |B \cap R_i| > \lceil \log n \rceil\}$.
   end-while
   (c) For every tree $T_i^{v_1}$ obtained in stage (b), add its root v_1 to R_{i+1}.
   (d) $i \leftarrow i + 1$.
   until $|R_i| \leq \lceil \log n \rceil$.

Let us first state the following properties of Procedure CHOOSE_REP. The proof follows directly from the algorithm.

CLAIM 4.8. *Let the number of stages in Procedure* CHOOSE_REP *be* f. *Then*

1. $v_0 \in R_f$,
2. $|R_i| \leq n/\log^i n$, *and*
3. $f \leq (\log n / \log \log n)$.

We now proceed to define the main algorithm. Throughout the algorithm we maintain a set R of informed vertices that equals R_j for some j. The point is that j decreases by one in each iteration, thus at the end $R = R_1 = V$.

ALGORITHM APPROX_OMBT

Input: A graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.

1. Apply Procedure CHOOSE_REP on G and v_0. Assume that the sets of representatives are $\{R_1, \ldots, R_f\}$.
2. Choose an arbitrary tree leading from v_0 to the other vertices of R_f, and inform all the vertices in R_f using the scheme suggested in the proof of Lemma 4.6.
3. $R \leftarrow R_f, i \leftarrow f$.
4. **repeat**
   (a) Each vertex $u \in R_i$ informs (in parallel) all the vertices in T_i^u using the scheme suggested in the proof of Lemma 4.6.
   (b) Let $G_i' = G \setminus \bigcup_{u \in R_i} T_i^u$. Let $C_1^i, \ldots, C_s^i$ denote the clusters in the graph induced by G_i'.

(c) The vertices $\bigcup T_i^u$ inform a vertex v_j in C_j^i, for each $1 \leq j \leq s$, using a minimum control function computed by Procedure MWC, as in step 5 of Algorithm APPROX_MBT.
(d) Choose for each j, a tree TL_j leading from v_j to the vertices in $R_{i-1} \cap C_j^i$.
(e) The vertices v_j inform the vertices of TL_j using the scheme of Lemma 4.6.
(f) $R \leftarrow R_{i-1}, i \leftarrow i-1$.
until $i = 1$.

It is clear from the algorithm that when stage 4(e) of Algorithm APPROX_OMBT is completed, all the vertices of R_{i-1} know the message. It follows that at the end all the vertices are informed.

THEOREM 4.9. *Algorithm* APPROX_OMBT *is a* $O(\log n / \log\log n)$ *approximation scheme for OMBT.*

Proof. By the definition of Procedure CHOOSE_REP, $|R_f| \leq \log n$. Thus it follows by this fact and Lemma 4.6 that step 2 of Algorithm APPROX_OMBT takes $O(\log n)$ time units. Let us now analyze the communication time of stages 4(a) to 4(e) for a fixed i. Since for every $u \in R_i$, every leaf in T_i^u belongs to R_{i-1} and $|T_i^u \cap R_{i-1}| = \lceil \log n \rceil$, it follows that the number of leaves in T_i^u is bounded by $\lceil \log n \rceil$; thus, by Lemma 4.6 the communication time of informing the vertices of T_i^u in step 4(a) is bounded by $O(\log n)$ time units. In step 4(c) the vertices in $\bigcup T_i^u$ inform a vertex of each clusters C_j^i. It follows by a similar argument to Lemma 4.3 that the number of time units is bounded by $b_{\text{op}}(v_0)$. Finally, it follows from step (b) of Procedure CHOOSE_REP that for every $j, C_j^i \cap R_{i-1} \leq \log n$; thus, by Lemma 4.6, step 4(e) takes no more than $O(\log n)$ time units. Summarizing, for a fixed i, the communication complexity of the scheme is bounded by $O(b_{\text{op}}(v_0) + \log n) = O(b_{\text{op}}(v_0))$. By Claim 4.8, $f \leq \log n / \log\log n$. Hence the desired result follows. □

Let us remark that we choose trees with $\lceil \log n \rceil$ leaves since this is the only lower bound known on the broadcasting time. If, however, it is known that for some particular family of input instances the broadcast time is bounded below by $n^{1/k}$ for some k smaller than $\log n / \log\log n$, then it is possible to construct trees with $n^{1/k}$ leaves and get an $O(k)$-approximation scheme for $k < \log n / \log\log n$.

4.4. Broadcasting on random graphs. The method of Algorithm APPROX_OMBT can be used to deal with the MBT problem as well. However, at each stage, broadcasting in a tree T may take $O(\log n + h(T))$ time units, where $h(T)$ is the height of T. Since the diameter of a subcluster of a graph G may largely increase, this may not be a good approximation scheme in the worst case.

However, it is instructive to consider the behavior of Algorithm APPROX_OMBT on random inputs. Let us consider a random graph $G \in G_{n,p}$. The graph consists of n vertices, where for each pair of vertices $v, w \in V$, the edge $(v, w) \in E$ is drawn with probability p, where p is constant, $0 < p < 1$. It is well known that the diameter of each vertex-induced subcluster of $G_{p,n}$ is bounded by $O(\log n)$ with high probability [Bol85]. For such graphs the scheme of Algorithm APPROX_MBT yields only an $O(\sqrt{n}/\log n)$-approximation ratio. In contrast, Algorithm APPROX_OMBT is an $O(\log n / \log\log n)$-approximation scheme for random graphs with high probability.

COROLLARY 4.10. *There exists a polynomial algorithm that broadcasts on a random graph* $G \in G_{n,p}$ *in no more then* $O(\log n / \log\log n) \cdot b(G)$ *time units with high probability.*

5. Separator-based strategies for broadcasting. An important method for dealing with optimization problems on graphs is the *divide-and-conquer* approach

[AHU74]. The idea is to find a small set of vertices whose removal splits the graph into connected components of roughly equal size and then to solve the problem recursively by handling each of the components separately. Unfortunately, general graphs do not necessarily have small separators. However, some important families of graphs do. The notion of a separator is formalized in the following definition.

DEFINITION 5.1 *Let $\varphi(n)$ be a nondecreasing function, and let y and ρ be fixed numbers such that $0 < \rho < 1$.*

1. *A graph $G = (V, E)$* has a $\langle \rho, y \rangle$-separator *if there exists a set $S \subset V$ such that the removal of S leaves no connected component of size greater than $\rho \cdot n$, and $|S| \leq y$.*
2. *A graph $G = (V, E)$ is* $\langle \rho, \varphi(n) \rangle$-separable *if every vertex-induced subgraph $G' \subset G$ of n' vertices has a $\langle \rho, \varphi(n') \rangle$-separator.*

Given a $\langle \rho, \varphi(n) \rangle$-separable graph, denote the corresponding separator of every subgraph G' by $\mathrm{sep}(G')$. This section examines the idea of using the separability property of a graph in order to achieve fast approximation schemes for broadcasting.

5.1. Broadcasting schemes for separable graphs. In order to develop a separator-based broadcasting scheme, we first need to generalize Lemma 4.3. Suppose that a graph G contains a set V_0 of informed vertices. Denote the clusters created when extracting V_0 from the graph by $C_1, \ldots, C_k$. Choose for each C_i an arbitrary nonempty subset $C'_i \subset C_i$. We can use the fact that in broadcasting it is necessary to inform the vertices of C'_i to achieve a lower bound on the best possible time for broadcasting. As before, we use Procedure MWC developed in §3.2. Let us first define a MWC instance $B' = \mathrm{MWC}(V_0, \{C'_1, \ldots, C'_k\})$ as follows.

1. Form the control graph $D_{V_0,G}$ of V_0 in G.

2. Put weights on the edges as follows. For an arbitrary vertex $v \in V_0$ connected to a vertex in C_i, choose a vertex $v' \in C_i$ connected to v that is closest to the set C'_i. Attach a weight $d^v_{C_i} \equiv \mathrm{dist}(v', C'_i)$ to the edge (v, C_i).

(Note that in Lemma 4.3 the construction is similar, except that we choose as subset $C'_i = C_i$, for every i.) We make the following claim.

LEMMA 5.2. *If F is a minimal control function for B', then $\mathcal{W}(F) \leq b(V_0, G)$.*

Proof. First we argue the following technical claim, implicitly used also in [SCH81].

CLAIM 5.3. *Let $d_i, t_i \in \mathbb{Z}^+$, for $1 \leq i \leq k$, such that $d_k \leq d_{k-1} \leq \cdots \leq d_1$, and the t_i's are all distinct. Then $\max_i\{i + d_i\} \leq \max_i\{t_i + d_i\}$.*

To any communication scheme from a base set V_0 such that $|V_0| > 1$, there is a corresponding spanning forest of G, where each tree in the forest is rooted at a vertex v of V_0 and represents the set of edges that carried the message from v (i.e., the nodes of the tree are those that received their copy of the message along a path originating at v). Assume that S is an optimal communication scheme for broadcasting from the base set V_0. Denote the forest corresponding to S by $\mathcal{F}$. Every vertex $v \in V_0$ that sends the message to some vertex in one of the C_i clusters roots a tree $T_v \in \mathcal{F}$. Let us define a function $F' : \{C_1, \ldots, C_k\} \to V_0$ as follows. Pick a vertex $w_i \in C'_i$ that is among the first in C'_i to receive the message (breaking ties arbitrarily). Suppose that $w \in T_v$; then set $F'(C_i) = v$. Now consider a vertex $v \in V_0$ that dominates a nonempty set of clusters. Say that v controls C_i (i.e., $F'(C_i) = v$), and assume that u_i is the (unique) child of v in T_v that is an ancestor of w_i. Clearly, we can assume that $u_i \in C_i$.

Say that v sends the message to u_i at time t_i. The time that passes before u_i can inform any vertex in C'_i (specifically w_i) is at least $d(u_i, C_i)$, thus by the fact that w_i

is one of the first vertices in C'_i that received the message, $t_i + \text{dist}(u, C_i) \leq b(V_0, G)$, thus $t_i + d^v_{C_i} \leq b(V_0, G)$, and so

$$\max_i \{t_i + d^v_{C_i}\} \leq b(V_0, G).$$

Thus from Claim 5.3 we deduce that

$$\max_i \{i + d^v_{C_i}\} \leq b(V_0, G).$$

Since this holds for every vertex, we conclude $\mathcal{W}(F') \leq b(V_0, G)$. Since F is a minimum control function, $\mathcal{W}(F) \leq \mathcal{W}(F') \leq b(V_0, G)$. □

Note that we can use an algorithm similar to Algorithm BES of §3.3 to establish the following fact.

Fact 5.4. In the above scenario, it is possible to inform at least one vertex in C'_i, for every i, in no more than $\mathcal{W}(F)$ time units.

It is possible to use Fact 5.4 in order to construct schemes for broadcasting from a distinguished vertex v in a graph with a "small" separator. Let G be a $\langle \rho, \varphi(n) \rangle$-separable graph. Throughout the run, the set V_0 will denote the set of already informed vertices.

ALGORITHM APPROX_SEP

1. $V_0 \leftarrow \{v\}$.
2. Construct a separator $\text{sep}(G)$ for G.
3. Build an arbitrary tree $\text{SPT}(v, \text{sep}(G)) = (V_1, E_1)$ rooted at v and leading to the members of $\text{sep}(G)$. Broadcast the message to the vertices of the tree using the OT scheme.
4. $V_0 \leftarrow V_0 \cup V_1$.
5. **Repeat**
   (a) Assume the clusters formed when extracting V_0 from the graph are $C_1, \ldots, C_k$. Each C_i has a separator $\text{sep}(C_i) = C^i_1 \cup C^i_2 \cup \cdots \cup C^i_{l_i}$, where $C^i_1, C^i_2, \ldots, C^i_{l_i}$ are $\text{sep}(C_i)$'s connected components.
   repeat
   (i) For each i pick the lowest index $j(i)$ that has not been chosen yet (for i).
   (ii) Build the instance $B' = \text{MWC}(V', \{C^i_{j(i)} : 1 \leq i \leq k\})$ of the MWC problem, as described.
   (iii) Compute a minimum control function F for B' using Procedure MWC.
   (iv) Send the message to at least one vertex of $C^i_{j(i)}$ for every i and j, using the minimal function F and the scheme suggested in Fact 5.4.
   until the C^i_j's clusters are exhausted for every i and j.
   (b) For every i and j, broadcast (in parallel) the message within C^i_j using the best known scheme for C^i_j. (If no good known scheme exists for the kind of graph C^i_j is, broadcast using an arbitrary tree.)
   (c) $V_0 \leftarrow V_0 \cup \text{sep}(C_1) \cup \cdots \cup \text{sep}(C_k)$.
   Until $V_0 = V$.

It is easy to see that when Algorithm APPROX_SEP terminates, all the vertices in V are informed.

LEMMA 5.5. *On a $\langle \rho, \varphi(n) \rangle$-separable graph, Algorithm* APPROX_SEP *terminates the broadcast process from a vertex v in $O(\log n) \cdot \varphi(n) \cdot b(v)$ time units.*

Proof. Clearly, the number of times the external loop is performed, is bounded by $O(\log n)$. Stage 3 takes no more than $\varphi(G) + \text{Diam}(v)$ time units. We next must bound the number of times that the internal loop is performed. Note that this number is bounded by the maximal number of connected components of the separator $\text{sep}(C)$ of any of the clusters C. Further, note that after the external loop has been executed i times, the size of any separator of any cluster C is bounded by $\varphi(\rho^i \cdot n) \leq \varphi(n)$. Thus, the number of connected components in the graph induced by $\text{Sep}(C)$ is also bounded by $\varphi(n)$. Thus every internal loop is carried out for no more than $\varphi(n)$ times.

Next we bound the maximal time taken by each iteration of the internal loop. By Lemma 5.2 and Fact 5.4, each execution of the repeat loop of stages 5(a)(i)–(iii) takes no more than $b(V_0, G) \leq b(v, G)$ time units. Thus the total time spent in stages 5(a)(i)–(iii) (which is no more than the number of external loops times the number of internal loops times the maximum time taken by an execution of an internal stage) is bounded by $O(\log n) \cdot \varphi(n) \cdot b_{\text{op}}(v)$.

We now bound the number of time units spent in step 5(b). After the external loop took place i times, the size of the separator and, hence, the size of every connected component of a separator are bounded by $\varphi(\rho^i \cdot n) \leq \varphi(n)$. Thus this is also a bound on the time spent in step 5(b), for a fixed i. Summing up this bound for every i, we conclude that the total time spent in step 5(b) is bounded by

$$\sum_{1 \leq i \leq O(\log n)} \varphi(\rho^i \cdot n) \leq O(\varphi(n) \cdot \log n).$$

Thus, in total, the broadcast time is bounded by $O(\log n) \cdot \varphi(n) \cdot b(v, G)$. □

Further, it can be shown that if we can assure that every separator $\text{sep}(C)$ is connected and $b(\text{sep}(C)) \leq k$ for some integer $k < \varphi(n)$, then the bound is improved to

$$O(\log n) \cdot (b(v) + k). \tag{1}$$

THEOREM 5.6. *Algorithm* APPROX_SEP *is an* $O(\log n) \cdot \varphi(n)$*-approximation scheme over* $\langle \rho, \varphi(n) \rangle$*-separable graphs.*

5.2. Applications. In this subsection we give some examples of graph families for which Algorithm APPROX_SEP can be applied. The first example is that of *chordal graphs.* A *chord* in a cycle of at least four vertices is an edge connecting two vertices that are not adjacent in the cycle. A *chordal graph* is a graph with the property that every cycle of four vertices or more has a chord. The following theorem is shown in [GRE84] regarding chordal graphs.

THEOREM 5.7 [GRE84]. *Every* n*-vertex chordal graph* G *contains a (polynomially computable) maximal clique* C*, such that if the vertices in* C *are deleted from* G*, every connected component in the graph induced by any of the remaining vertices is of size at most* $n/2$.

An $O(|E|)$-time algorithm for finding a separating clique that satisfies the condition of the theorem is also given in [GRE84].

Thus chordal graphs always have separating sets that are connected (and moreover, are cliques). Since it is possible to broadcast a message in a clique of m vertices in $\lceil \log m \rceil$ time units, it follows from (1) that the time needed to broadcast in a chordal graph using the scheme of Algorithm APPROX_SEP is no more than

$$\log n \cdot (b(v, G) + \lceil \log n \rceil) + \text{Diam}(v).$$

Consequently, Algorithm APPROX_SEP is a $(2\log n + 1)$-approximation scheme for broadcasting in chordal graphs.

COROLLARY 5.8. *There exists a polynomial* $(2\log n + 1)$*-approximation scheme for broadcasting on chordal graphs.*

A second example is the family of a c-separable graphs, consisting of graphs for which $\varphi(n) = c$ for some constant c. These graphs were considered, for instance, in [FJ90]. It follows from Theorem 5.6 that Algorithm APPROX_SEP is an $O(\log n)$-approximation scheme for broadcasting in such graphs.

We now present two examples of $O(1)$-separable graph families. The class of *k-outerplanar graphs* is defined as follows. Consider a plane embedding of a planar graph. The nodes on the exterior face are termed layer 1 nodes. For $i > 1$, the layer i nodes are those that lie on the exterior face of the embedding resulting from the deletion of all layer j nodes, $j < i$. A plane embedding is k-outerplanar if it contains no node with layer number larger than k. A planar graph is k-outerplanar if it has a k-outerplanar embedding. A graph is called *outerplanar* if it is a 1-outerplanar graph, i.e., a graph that can be embedded in the plane such that all the vertices lie on one face [Har69].

In [FJ90], Frederickson and Janardan show that any k-outerplanar graph is $\langle 2/3, 2\cdot k\rangle$-separable. An $O(n)$-time algorithm to find the separator is given in [FJ90]. Thus Algorithm APPROX_SEP can be used to broadcast in a k-outerplanar graph achieving an $O(k\log n)$-approximation scheme. Thus we have the following theorem.

THEOREM 5.9. *There exists an* $O(k\log n)$*-approximation scheme for broadcasting on a k-outerplanar graph.*

COROLLARY 5.10. *There exists a polynomial* $O(\log n)$*-approximation scheme for broadcasting on the family of outerplanar graphs.*

The third example is the well-known family of series-parallel graphs. Two edges in a graph are in "series" if they are the only edges incident to a node and "parallel" if they join the same pair of nodes. The definition of a series-parallel graph is recursive. First, an edge is a series-parallel graph. Next, the graph obtained by replacing any edge in a series-parallel graph either by two series edges (adding a vertex) or by two parallel edges is series-parallel. In [FJ90] it is shown that every series-parallel graph is $\langle 2/3, 2\rangle$-separable and the separator can be found in $O(n)$ time. Thus Algorithm APPROX_SEP is a polynomial $O(\log n)$-approximation algorithm for broadcasting on a series-parallel graph.

THEOREM 5.11. *There exists a polynomial* $O(\log n)$*-approximation scheme for broadcasting on a series-parallel graph.*

The last example is of the family of *bounded-face planar graphs.* The size of a face of a planar graph is the number of vertices in the face, counting multiple visits when traversing the boundary (cf. [Mil86]). The following theorem is shown in [Mil86].

THEOREM 5.12 [Mil86]. *Every planar graph with bounded-face size is* $\langle 2/3, O(\sqrt{n})\rangle$*-separable, and the separator can be chosen to be a simple cycle or a single vertex.*

A linear time algorithm to find the separating cycle or vertex is also given in [Mil86]. We use this to derive the following theorem.

THEOREM 5.13. *There exists an* $O(n^{1/4}/\sqrt{\log n})$*-approximation scheme for broadcasting on bounded-face planar graphs.*

Proof. Use the algorithm of [Mil86] to compute the separators needed for Algorithm APPROX_SEP. At each step of Algorithm APPROX_SEP, if the separator is a cycle, instead of broadcasting to all the vertices of the cycle separator, choose arbitrarily a "starting" vertex in the cycle and give it an index 1, while giving indices to

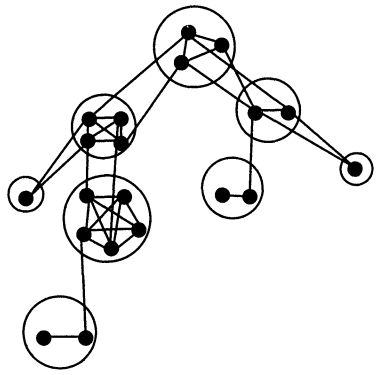

FIG. 3. *A tree of cliques (TOC) G. The large circles represent the vertices $\tilde{V}$ of the corresponding tree $T(G)$.*

the rest of the vertices in increasing order, according to their clockwise position.

Broadcast the message to every vertex whose index is congruent to 1 mod $\lceil n^{1/4} \cdot \sqrt{\log n} \rceil$, in every separating cycle. The number of recipients of the message in each cycle is $O(n^{1/4}/\sqrt{\log n})$. This is done as in Algorithm APPROX_SEP by using the technique for MWC problems. After the corresponding cycle vertices get the message, they can clearly inform the rest of the cycle vertices in no more than $O(n^{1/4} \cdot \sqrt{\log n})$ time units. Finally note that informing the vertices of the first cycle, i.e., the cycle separator of the graph itself, can be done in similar way. The broadcast time using this scheme is no more than

$$\sum_{i=1}^{\log_{3/2} n} O\left(\left(\frac{2}{3}\right)^i \cdot \frac{n^{1/4}}{\sqrt{\log n}}\right) \cdot b(v) + \sum_{i=1}^{\log_{3/2} n} \left(\frac{2}{3}\right)^i \cdot n^{1/4}\sqrt{\log n}$$
$$+ O(n^{1/4}\sqrt{\log n}) + \text{Diam}(G) = O(n^{1/4}/\sqrt{\log n}) \cdot b(v). \qquad \square$$

This is slightly better than the result given for general graphs in Theorem 4.5 (in the worst case).

6. Broadcasting in a tree of cliques. In this section we present a broadcast scheme for graphs that are in a "tree of clusters" form. We illustrate this method by giving an approximation scheme for a special kind of graph family called *trees of cliques,* generalizing the family of trees.

6.1. Trees of cliques.

DEFINITION 6.1 (see Fig. 3). *A graph $G = (V, E)$ is a* tree of cliques (TOC) *if*

1. *the vertex set V can be decomposed into a disjoint union of sets $C_1, \ldots, C_k$ such that each C_i induces a clique (i.e., a complete graph) in G, and*

2. *the auxiliary graph $T(G) = (\tilde{V}, \tilde{E})$ whose vertices are $\tilde{V} = \{C_1, \ldots, C_k\}$ and whose edges are*

$$\tilde{E} = \{(C_i, C_j) : \text{ there is an edge } (v_i, v_j) \in E, \text{ for } v_i \in C_i, v_j \in C_j\}$$

is a tree.

To broadcast a message from a vertex v in a TOC, we use the following idea. In order to deliver the message between vertices of different cliques (i.e., from cliques to their clique children), we use the techniques developed for MVWC problems. It follows

that the total broadcast complexity spent while delivering a message between cliques is bounded by $O(b(v))$. We can achieve an efficient method, since there is an efficient method for message delivery in a clique. We then develop an alternative method for delivering the message inside the cliques. In this method, every vertex participates in the message delivery in its clique only for a small (fixed) number of rounds and is thus free sooner to help in sending the message down the tree to its clique children. Using this method we establish some improved bounds in restricted cases.

6.2. The broadcast scheme. Let G be a TOC. The notions of child, parent, height, and so on are defined in G as in the tree $T(G)$; specifically, the parent of a clique C is denoted by $p(C)$, the height of a rooted TOC G is denoted by $h(G)$, the subtree rooted at a given clique C is denoted by G_C, and $T(G)_C$ is defined accordingly.

Let G be a rooted TOC. The *parent index,* $PI(C)$, of a nonroot clique C in G is defined to be the number of vertices in C that are connected to at least one vertex in the parent clique $p(C)$. The parent index of the root is defined to be 1. Similarly the *child index,* $\mathrm{CI}(C)$, of a nonleaf clique C is the number of vertices in C that are connected to at least one of the vertices of the children of C.

A TOC G is *parent c-restricted* if it is possible to root G at a clique C such that $PI(C') \leq c$ for every clique C'. Note that the fact that a TOC is parent c-restricted does not preclude the possibility that *every* vertex in $P(C)$ will have an arbitrarily large (total) number of adjacent vertices in the clique children. A *child c-restricted* TOC is defined similarly. Note that if a TOC is child c-restricted, there are no more than c vertices in $p(C)$ that can inform the vertices in its clique children. It follows that it seems easier to approximate the broadcast problem on a TOC if it is child c-restricted than if it is parent c-restricted.

We next give a broadcast scheme on a TOC. We assume that the clique partition of the TOC is given. Before presenting the approximation algorithm, let us give two definitions. The first definition is of the *rank* of a clique C in a rooted TOC G. This definition induces a definition of a controlling vertex $\tilde{F}(C)$ of C, such that $\tilde{F}(C) \in p(C)$, for any nonroot clique C. Recall that for a clique C in the tree, T_C is the subtree rooted by C.

DEFINITION 6.2. *Define* $\mathrm{rank}(C) = 0$ *for a leaf C in G. Define inductively the rank of a nonleaf clique C in $T(G)$ as follows.*

1. *Form the control graph $D_{C,T_C} = (V_1, V_2, A)$ of C in $T(G)_C$.*
2. *Compute recursively the ranks of the children of C.*
3. *Set* $\mathrm{rank}(C) = \mathcal{W}(F)$, *where F is the minimal function with respect to the MVWC problem resulting by the construction of step* 1 *taking the weight of a clique child vertex C' to be* $\omega(C') = \mathrm{rank}(C')$.
4. *Define $\tilde{F}(C') = F(C')$, for every child C' of C.*

This definition of rank tries to capture the minimal degrees needed for the cliques to control their clique children. It also identifies those vertices that dominate children cliques in the TOC. Denote

$$\mathrm{Dom}(C) = \{w \in C : \text{ there exists a child } C_i \text{ of } C \text{ such that } \tilde{F}(C_i) = w\}.$$

Our second definition concerns the *degree* of cliques in the TOC and attempts to capture the *number* of vertices there are in a subtree rooted at a clique C.

DEFINITION 6.3. *Let G be a rooted TOC. The* degree *of a leaf C in G is* $\deg(C) = 0$. *The degree of a nonleaf clique C in $T(G)$ is defined recursively as follows:*

1. *Compute the degrees of C's children in G.*

2. *Let* $C_1, \ldots, C_k$ *denote* C*'s children in* G*, ordered by nonincreasing degrees, i.e.,* $\deg(C_i) \geq \deg(C_{i+1})$ *for every* i.

3. *Define* $\deg(C) = \max_i\{\lceil \log\lceil i/PI(C)\rceil\rceil + \deg(C_i)\}$.

Let us now describe a scheme called the *Fibonacci method,* for message dissemination within a clique. In this scheme we try to save time in informing the vertices within the clique so that a vertex will be able to start sooner to deliver the message to its clique children. Let $v_1, \ldots, v_k$ be k vertices in a clique C. We assume that C contains an informed vertex v_0. Our goal is to broadcast the message from v_0 to $\{v_1, \ldots, v_k\}$. This is done as follows:

1. In the first two steps, v_0 sends the message to v_1 and v_2.
2. Now define a delivery scheme for $v_i, i \geq 2$, as follows: each vertex v_i spends the first two rounds after it gets the message on delivering the message within the clique. The delivery scheme prefers vertices v_m with lower index. In each round j, the l vertices that are required to participate in the delivery within the clique in this round send the message to the next lowest index l vertices among $\{v_1, \ldots, v_k\}$ that did not get the message yet.

For instance, in round 3, v_1 and v_2 send the message to v_3 and v_4; in round 4, v_1, v_2, v_3, v_4 send the message to v_5, v_6, v_7, v_8; and in round 5, $v_2, \ldots, v_8$, send the message to $v_9, \ldots, v_{15}$.

Let G be a TOC, and let C_0 be a clique in G and $v_0 \in C_0$ a distinguished vertex. The goal is to broadcast the message from v_0 to all the vertices in G. We next give two recursive approximation algorithms for the problem.

ALGORITHM APPROX_TOC$_{2a}$

Input: A TOC G and a root clique C_0 and an informed vertex $v_0 \in C_0$.

1. Compute ranks and define a dominating vertex $\tilde{F}(C_i) \in C$ for any clique children C_i of any clique $C \in G$ as indicated by Definition 6.2, using Algorithm MVWC.
2. As soon as a clique C contains an informed vertex $v \in C, v$ sends the message to all the vertices in the clique C, using an optimal procedure for the clique.
3. Each vertex $w \in \text{Dom}(C)$ starts sending the message to a single arbitrary vertex in each of the cliques it controls. The delivery is performed in nonincreasing order of ranks; i.e., if $\text{rank}(C') > \text{rank}(C'')$, then w sends the message to a vertex in C' before it sends it to a vertex in C''.

Next, let us modify Algorithm APPROX_TOC$_{2a}$ to get Algorithm APPROX_TOC$_{2b}$. Instead of delivering the message within the clique using all the vertices through the entire process, as done in step 2 of Algorithm APPROX_TOC$_{2a}$, we use the Fibonacci delivery scheme. Thus in Algorithm APPROX_TOC$_{2b}$ step 2 is replaced by the following step:

2. Let C be a clique in G containing an informed vertex $v \in C$. Assume that $\text{Dom}(C) = \{v_1, \ldots, v_k\}$. For every vertex v_i, let $C_{\max}(v_i)$ be the clique dominated by v_i with maximal degree. Assume without loss of generality that for every $i, \deg(C_{\max}(v_i)) \geq \deg(C_{\max}(v_{i+1}))$. Apply the Fibonacci delivery method within the clique, from the informed vertex v to $\{v_1, \ldots, v_k\}$.

It is easy to see that when the execution of this modified version of Algorithm APPROX_TOC$_{2a}$ terminates, every clique contains at least one informed vertex (however, not necessarily all the vertices in all the cliques are informed, since in any clique a vertex is informed only if it controls a nonempty set of children cliques). Thus to terminate the broadcast, in every clique the informed vertices deliver (in paral-

lel) the message to the rest of the vertices in the clique, using the optimal delivery procedure for the clique. We call this modified version of the algorithm Algorithm APPROX_TOC$_{2b}$.

6.3. The broadcast complexity of the scheme. We now analyze the broadcast complexity of the scheme. Before stating the next claim, which speaks about the Fibonacci delivery method, recall the sequence of Fibonacci numbers defined as follows: $z_1 = z_2 = 1, z_{i+2} = z_{i+1} + z_i$ for $i \geq 1$ (cf. [HW56]).

LEMMA 6.4. *Let C be a clique, and suppose that $v, v_1, \ldots, v_k \in C$ and v uses the Fibonacci method to send a message to $v_1, \ldots, v_k$. Then i rounds after v_1 receives the message from v, there are exactly z_{i+2} informed vertices in $\{v_1, \ldots, v_k\}$.*

Proof. The claim holds for $i = 1, i = 2$, and $i = 3$ by a direct inspection. Now assume it is true for $i \geq 3$ and that at times $i-2, i-1$, and i the number of informed vertices in $\{v_1, \ldots, v_k\}$ is z_i, z_{i+1}, and z_{i+2}, respectively.

The number of vertices that delivered the message only once within the clique is $z_{i+1} - z_i$. The number of vertices that have not yet broadcast the message at all is $z_{i+2} - z_{i+1}$. Thus the total number of informed vertices in the next round is $z_{i+2} + (z_{i+2} - z_{i+1}) + (z_{i+1} - z_i) = z_{i+2} + z_{i+1} = z_{i+3}$. □

Let us now study the time needed to inform k vertices $v_1, \ldots, v_k$ in the Fibonacci method. Since $z_i = (((1+\sqrt{5})/2)^i - ((1-\sqrt{5})/2)^i)/\sqrt{5}$ (cf. [HW56]),

$$z_i \geq \frac{((1+\sqrt{5})/2)^i/\sqrt{5}) - 1}{\sqrt{5}}.$$

Thus the number of rounds before v_i is informed in the Fibonacci scheme is no greater than $\lceil 1.441 \cdot \log i \rceil + 2$.

We now make the following claim.

LEMMA 6.5. *For any tree of cliques $G = (V, E)$ rooted at a clique C_0, $\text{rank}(C_0) \leq b(C_0, G)$.*

Proof. We prove the lemma by induction on the height of the TOC. If $h(T(G)) = 0$ the claim holds trivially since $\text{rank}(C) = 0$. Assume the claim for height k. Consider a tree of height $k+1$. Assume that the children of C_0 in $T(G)$ are $C_1, \ldots, C_l$. Consider an optimal scheme S for broadcasting from the base set C. For each clique C_i choose a vertex $v_i \in V$ that is among the first vertices that transmit the message to a vertex in C_i. Since the clusters C_i are independent, all of the selected vertices v_i are in C_0. We have defined a function F' from the children of C to the vertices of C. Denote the subtree corresponding to C_i by T_i and the corresponding subgraph of G by G_i.

Assume that v is first to deliver the message to the cliques $C'_1, \ldots, C'_j$, that $F'(C'_i) = v$ for every i, and without loss of generality that the cliques C'_i are arranged by nonincreasing order of ranks. Further, assume that v delivers the message to a vertex in C'_i at time t_i. We claim that

$$\max_i \{t_i + b(C'_i, G_i)\} \leq b(C, G),$$

since t_i is the first time that a subset of the vertices of C'_i receive the message. Thereafter they must deliver the message to the rest of the vertices of G_i, and this clearly takes at least $b(C'_i, G_i)$ time units. Since $h(G_i) \leq k$ for every i, by the induction hypothesis,

$$\max_i \{t_i + \text{rank}(C'_i)\} \leq b(C, G).$$

By Lemma 5.3,

$$\max_i\{i + \text{ rank}(C_i')\} \leq b(C, G).$$

Since this is true for any vertex $v, \mathcal{W}(F') \leq b(C, G)$. Thus it follows from Definition 3.1 plus the fact that $\text{rank}(C_0)$ is the weight of the minimal function that $\text{rank}(C) \leq b(C_0, G)$. □

LEMMA 6.6. *Let G be a TOC rooted at C_0 and v_0 a vertex in C_0. Then* $\deg(C) \leq b(v)$.

Proof. Consider an optimal scheme S for broadcasting from v in G. At the first round to where a vertex of a clique C' in G receives the message, there are at most $q = PI(C')$ informed vertices in C'. Note that the vertices of $T_{C'}$ cannot receive the message through any alternative route other than via the vertices adjacent to $p(C')$; it follows by the doubling argument of Fact 2.5(1) that for any $l > q$, it will take at least $\lceil \log(l/q) \rceil$ rounds until l vertices of $T_{C'}$ are informed. Suppose that C' has l children $C_1, \ldots, C_l$ ordered by nonincreasing order of degrees. For any $l, 1 \leq l \leq j$, the first time that all the first l cliques $C_1, \ldots, C_l$ contain an informed vertex is at least $t_0 + \lceil \log l/q \rceil$. The rest of the proof follows in a straightforward way by induction on $h(G)$. □

We are ready to analyze the complexity of the broadcast scheme. Let us divide the delays encountered by a message before it reaches a vertex in some leaf clique C' into the following two possible types:

1. Delays encountered when a predecessor clique C'' of C' delivers the message to another subtree rather than the one containing C', and

2. Delays encountered by the message when it is being broadcast within a predecessor clique of C'.

We bound delays of the first type by proving the following (where v_0 is the originator of the message.)

LEMMA 6.7. *There are no more than $b(v_0) - d(C_0, C')$ delays of the first type, where $d(C_0, C')$ is the distance between C_0 and C' in the tree $T(G)$.*

Proof. The proof follows by Lemma 6.5 and straightforward induction on $h(T(G))$. □

We now consider second-type delays. Let us first analyze the number of such delays in Algorithm APPROX_TOC$_{2a}$. In this case, the number of type 2 delays encountered by a message before it reaches a leaf C' is bounded by $\sum_i \lceil \log C_i \rceil$, where C_i is the ith clique in the path connecting C and $p(C')$ in $T(G)$. If there are h cliques in the path, then the number of type 2 delays is bounded by $h \cdot \log(n/h)$.

Since both $\log n$ and h are lower bounds on the broadcast time, the worst case is when $h = \log n$, and in this case the number of type 2 delays is bounded by $\log n \cdot (\log n - \log\log n)$. Thus we have the following theorem.

THEOREM 6.8. *Algorithm* APPROX_TOC$_{2a}$ *is an additive* $\log n \cdot (\log n - \log\log n)$*-approximation scheme for broadcasting in a* TOC.

Let us now analyze the situation in Algorithm APPROX_TOC$_{2b}$. Let $C_1, \ldots, C_l$ be the children of C, ordered by nonincreasing order of degrees, and let $\text{Dom}(C) = \{v_1, \ldots, v_k\}$. Assume without loss of generality that the vertices v_i are ordered such that $\deg(C_{\max}(v_i)) \geq \deg(C_{\max}(v_{i+1}))$ for every i. Since C_i is dominated by one of the first i vertices, the number of type 2 delays encountered by the message before it

is sent to C_i is no more than

$$\begin{aligned} 1 + \lceil 1.441 \cdot \log i \rceil + 2 + 2 &= (\lceil 1.441 \cdot \log i \rceil - \lceil 1.441 \cdot \log PI(C_i) \rceil) \\ &\quad + \lceil 1.441 \cdot \log PI(C_i) \rceil + 5 \\ &= O\left(\frac{\log i}{PI(C_i)}\right) + O(\log PI(C_i)). \end{aligned}$$

Thus the next claim follows from Lemma 6.6 and by induction on $h(G)$. Suppose that the ith clique in the path in $T(G)$ from the root to a leaf C' is C_i.

CLAIM 6.9. *The number of second-type delays encountered by the message before it reaches C' in Algorithm* APPROX_TOC$_{2b}$ *is bounded by*

$$O(\deg(C)) + \sum_i \lceil \log PI(C_i) \rceil) \rceil + O(\mathrm{Diam}(v)) \leq O(b(v)) + O\left(\sum_i \log PI(C_i)\right).$$

For example, consider a parent c-restricted TOC G for a constant c. It follows that Algorithm APPROX_TOC$_{2b}$ is a *constant* approximation scheme for broadcasting in such a TOC. (If the goal is to broadcast the message from a vertex v' that is not in C, where C is the clique that determines the fact that G is c-restricted, simply deliver the message to a vertex v in C, by a shortest path, and then use the APPROX_TOC$_{2b}$ scheme to broadcast from v. The fact that $b(v)$ and $b(v')$ differ by at most $\mathrm{Diam}(G)$ guarantees that the scheme is still an $O(\log c)$-approximation scheme.) As one can easily check, the scheme is also an $O(\log c)$-approximation scheme in the case of a child c-restricted TOC. We summarize this discussion in the following theorem.

THEOREM 6.10. *Algorithm* APPROX_TOC$_{2b}$ *is a* $\min\{O(\log c_1), O(\log c_2)\}$*-approximation scheme for broadcasting in a child c_1-restricted, parent c_2-restricted* TOC.

Note that similar methods can be used to broadcast in a more general class of a "tree of clusters" graphs as long as there exists a fast approximation scheme for broadcasting in the clusters.

Note added in proof. Recently, a new approximation algorithm was presented for the minimum broadcast time problem [R94]. That algorithm has (*multiplicative*) approximation ratio $O(\log^2 n / \log\log n)$. Hence the new algorithm improves on the result of Theorem 4.5 for graphs whose broadcast time is $O(\sqrt{n} \log\log n / \log^2 n)$ or smaller. For graphs with larger broadcast time, our $\sqrt{n}$*-additive* algorithm still yields better approximation.

REFERENCES

[AHU74] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[Bol85] B. BOLLOBAS, *Random Graphs*, Academic Press, New York, 1985.

[Far80] A. M. FARLEY, *Minimum time line broadcast network*, Networks, 10 (1980), pp. 59–70.

[FH78] A. FARLEY AND S. HEDETNIEMI, *Broadcasting in grid graphs*, in Proceedings of the Ninth Southern Conference on Combinatorics, Graph Theory, and Computing, 1978, pp. 275–288.

[FJ90] G. FREDERICKSON AND R. JANARDAN, *Space-efficient message routing in c-decomposable networks*, SIAM J. Comput., 19 (1990), pp. 164–181.

[FP80] A. M. FARLEY AND A. PROSKUROWSKI, *Gossiping in grid graphs*, J. Combin. Inform. System Sci., 15 (1980), pp. 161–162.

[GJ79] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, CA, 1979.

[GRE84] J. R. GILBERT, D. J. ROSE, AND A. EDENBRANT, *A separator theorem for chordal graphs*, SIAM J. Alg. Discrete Methods, 5 (1984), pp. 306–313.

[Har69] F. HARARY, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

[HHL88] S. HEDETNIEMI, S. HEDETNIEMI, AND A. LIESTMAN, *A survey of gossiping and broadcasting in communication networks*, Networks, 18 (1988), pp. 319–349.

[HMS72] A. HAJNAL, E. C. MILNER, AND E. SZEMEREDI, *A cure for the telephone disease*, Canad. Math. Bull., 15 (1972), pp. 447–450.

[HW56] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, Oxford University Press, London and New York, 1956.

[Mil86] G. MILLER, *Finding small simple cycle separators for 2-connected planar graphs*, J. Comput. System Sci., 32 (1986), pp. 265–279.

[R94] R. RAVI, *Rapid rumor ramification: Approximating the minimum broadcast time*, Proc. 35th IEEE Symp. on Foundations of Computer Science, 1994, pp. 202–213.

[SCH81] P. J. SLATER, E. J. COCKAYNE, AND T. HEDETNIEMI, *Information dissemination in trees*, SIAM J. Comput., 10 (1981).

[SW84] P. SCHEUERMAN AND G. WU, *Heuristic algorithms for broadcasting in point-to-point computer networks*, IEEE Trans. Comput., 33 (1984), pp. 804–811.