# Approximation Algorithms for Minimum Time Broadcast

Guy Kortsarz [*] David Peleg[*] [†]

August 4, 2014

**Abstract**

This paper deals with the problem of broadcasting in minimum time. Approximation algorithms are developed for arbitrary graphs, as well as for several restricted graph classes.

[*]Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel.

[†]Supported in part by an Allon Fellowship, by a Bantrell Fellowship and by a Walter and Elise Haas Career Development Award.

## 1 Introduction

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. [?]). 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, else it is *idle.* The rule 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 originated at $v$ in $G$, 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 [?], and broadcasting in grid graphs is studied in [?]. For a comprehensive survey on the subject of gossiping and broadcasting see [?].

The MBT problem in general graphs is NP-complete (cf. [?]) and thus it 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 [?]. Since the exact algorithm is not efficient for large networks, several heuristics are presented in [?] for achieving a broadcast scheme with good performance.

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

In this paper we consider *approximation schemes* for broadcast, namely, algorithms that do not give an exact solution, but rather give a solution that is not "too far" from the optimum. More formally, we call a scheme $A$ 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. We give approximation schemes for broadcast on several networks classes, and analyze their approximation ratio. In particular, we give an approximation algorithm for general $n-$vertex graphs with $O(\sqrt{n})-$additive ratio. For *chordal* and $O(1)$-separable $n-$vertex graphs we have a scheme with approximation ratio $O(\log n)$. We also give an efficient approximation scheme for broadcasting in a special family of graphs called *trees of cliques*, and some tools for broadcasting in a bounded-face planar graph.

Our results for the broadcast operation carry over to another important and well-studied operation, namely, *gossip*. 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 too has received considerable attention, mainly in the telephone model. For example, the papers [**?**, **?**] concern gossiping on the complete graph and on grid graphs, respectively.

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 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, based on the assumption that a processor may send at most one fixed size message in each time stamp, 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.)

Several generalizations of the MBT problem appear in the literature. In [?], Farley suggests to reconsider the assumption that a vertex may call only neighboring vertices. Farley defines two possible variants of the model using long distance calls, the *open path* and the *open line* models. In the open path model, communication is carried along vertex disjoint paths. At each round, an informed member $v$ may call a noninformed 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 denote the time needed to complete the broadcast from a distinguished vertex $v$ in the graph $G$ in the open path model, by $b_{op}(v, G)$ (or $b_{op}(v)$). We also denote

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

We call the problem of broadcasting from a vertex $v$ in $b_{op}(v, G)$ time units, OMBT. In this paper we give an approximation algorithm for OMBT on arbitrary $n-$vertex graphs with approximation ratio $O(\log n / \log \log n)$. (The open line model is similar, except that the paths used in a communication round need only be *edge* disjoint. The problem of approximating broadcast in this model is essentially solved up to logarithmic factors, since as shown in [?] the open line model enables broadcast from an arbitrary vertex in $\lceil \log n \rceil$ time units.)

Returning to the telephone model, another generalization we consider is to assume that 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 [?], SMBT is NP-complete even for $k = 4$ where $k$ is the bound on the time for completing the broadcast.

In order to approximate the MBT problem, we define in the sequel a problem called *Minimum Weight Cover* (MWC) which will be shown to be related to MBT. We give a pseudo polynomial solution to MWC, and use it in virtually all our approximations for MBT.

# 2 Preliminaries

## 2.1 Schedules and Broadcast

Throughout the sequel 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 $m$.

**Definition 2.1** *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 2.2** *At each round $t$, the set of active edges is independent.*

The *Minimum Broadcast Time* (MBT) problem is formally defined as follows. Let $v_0 \in V$ be a distinguished vertex. A broadcast from $v_0$ is a sequence

$$\{v_0\} = V_0, E1, V_1, E_2, \ldots, E_k, V_k = V$$

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

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 and
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 the 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 $b(v) \geq Diam(v)$, 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,

[1] All log's in this paper are taken to base 2

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 can not 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 2.3**

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 $v \in V$, $b(v, G) \geq Diam(v)$.*

Note that Fact 2.3 holds in the open path model as well, except of course for claim (4); we can not argue that $b_{op}(G) \geq Diam(G)$.

## 2.2 Graph definitions

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 $dist(v, w)$. We denote $Diam(v) = \max_w \{dist(v, w)\}$. The diameter of the graph $G$ is $\max_v \{Diam(v)\}$.

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 2.4** *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 2.5** *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 $G$.*

## 3 The Minimum Weight Cover problem

The *Minimum Weight Cover* (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 $G = (V_1, V_2, A)$ be a bipartite graph with a weight function $w : A \mapsto Z^+$ on the edges. A feasible solution to the MWC problem is a *control* function $F : V_2 \longrightarrow V_1$, 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 costumer. If $F(v_2) = v_1$ we say that $v_1$ *controls* (or *dominates*) $v_2$. Suppose that $v$ controls $u_1, ..., u_k$. Denote $e_i^v = (v, u_i)$ and without loss of generality assume that the vertices are ordered so that $w(e_1^v) \geq w(e_2^v) \geq \ldots \geq w(e_k^v)$. For a given function $F$, denote

**Definition 3.1** *The* weight *of a control function $F$ is defined as*

$$w(F) = \max_v\{\max_i\{i + w(e_i^v)\}\}.$$

Thus the MWC is defined as follows. Given a bipartite graph $G = (V_1, V_2, A)$ with no isolated vertices, and a weight $w(e) \in Z^+$ for every edge $e \in A$, determine a function $F : V_2 \longrightarrow V_1$ whose weight $w(F)$ 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 given in this paper to the MWC problem, the weights satisfy $w(e) \leq n$. Thus in order to use this problem as a basic auxiliary tool for the study of MBT, a pseudo polynomial solution 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 *Minimum Vertex Weight Cover* (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, i.e., minimize the size of the largest inverse image of $F$, $\max_v |\{u : F(u) = v\}|$.

## 3.2 An algorithm for MWC

This subsection presents a pseudo polynomial solution to the MWC problem. The algorithm is based on the following procedure. Given an instance $G = (V_1, V_2, A, w)$ and a positive integer $j$, we check if there exists a control function $F$ for $G$ of weight $w(F) \leq j$. The solution method for this problem involves flow techniques . The original graph is modified as follows. Create a source vertex $s$ and a sink vertex $t$. For a function to be of weight $j$, a server $v$ can not dominate a customer $u$ such that $w(v,u) \geq j$. Assume that $w_v$ 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 $w_v + 1$ different copies and arrange the copies in an arbitrary order $v_1, ..., v_{w_v+1}$. For $v_1$, the first copy of $v$, create a directed edge $(s, v_1)$, with capacity $j - w_v$, and create 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 $i$'th copy of $v$, $i \geq 2$, create a directed edge $(s, v_i)$ with capacity 1, and a directed edge $(v_i, u)$ with capacity 1 to all the customers $u$ such that $(v,u) \in A$ and $w(v,u) \leq w_v - i + 1$. Finally create for each customer $u \in V_2$ a directed edge $(u,t)$ with capacity 1. Call the resulting graph $F_{j,G}$.

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

**Lemma 3.2** *Given a MWC problem on a given graph $G$ and the corresponding construction $F_{j,G}$, the maximal flow is $|V_2|$ iff there exists a control function $F$ such that $w(F) \leq j$.*

**Proof:** First assume that there exists an integral flow function for $F_{j,G}$ with $|V_2|$ flow units entering $t$. 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. Since the total flow is $|V_2|$, such an edge exists for each vertex $v_2 \in V_2$. Define $F(v_2) = v_1$ such that $v_1$ is the vertex for which the flow through $(v_1, v_2)$ is 1.

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 $i$, denote the number of customers $u$ dominated by $v$ such that $w((v,u)) \geq i$ by $r_i(v)$. We argue the following.

**Claim 3.3** *$r_i(v) \leq j - i$.*

**Proof:** Consider first the value $i = w_v$. Since only the first copy of $v$ can dominate vertices $u$ such that $w((v,u)) = w_v$, $v$ does not dominate more than $j - w_v$ such vertices. Now consider $i < w_v$. Note that only the copies number $2, 3, ..., w_v - i + 1$ can aid in increasing $r_i(v)$, each by at most 1. Thus $r_i(v) \leq j - w_v + (w_v - i + 1 - 2) + 1 = j - i$. ▮

The remainder of the proof of this direction follows in a straightforward way from the definition of the weight of a function.

For proving the other direction, assume that there exists a function $F$ such that $w(F) \leq j$. Thus each server dominates no more then $j$ customers, and furthermore $v$ does not dominate more than $j - i$ customers whose corresponding weights are $i$ or larger. Consider a server $v$. Order the customers $u_1, ..., u_k$ dominated by $v$ by non increasing weights. Augment the flow through $v_1$ by $j - w_v$ adding a flow of 1 from $v_1$ to each of $u_1, ..., u_{j-w_v}$. The weight of the next vertex dominated by $v$, namely, $u_{j-w_v+1}$, is smaller than $w_v$, thus there is a directed edge from $v_2$ to that vertex. Thus it is still possible to augment the flow by 1 through $v_2$. In a similar way it follows that it is possible to augment the flow by $k$ through $\{u_1, \ldots, u_k\}$. 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 be found by using binary search. That is, start with $j_1 = \min_e\{w(e)\} + 1$, and $j_2 = \max_e\{w(e)\} + |V_2|$, and check if there is a function $F$ of weight $j = \lceil \frac{j_1+j_2}{2} \rceil$. If such a function exists, then set $j_1 \leftarrow j$, else set $j_2 \leftarrow j$, and then recursively repeat this test. Clearly, when this process terminates, we are left with a minimum control function (whose weight is $j_1$). 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 times that can equal the maximal number in the input. To summarize, we have established the following.

**Theorem 3.4** *There exists a pseudo polynomial algorithm for solving the MWC problem.*

Since MVWC is a special case of MWC we can deduce:

**Corollary 3.5** *There exist a pseudo polynomial algorithm for solving the MVWC problem.*

## 3.3 The BES problem

The bipartite edge scheduling problem, will serve as an illustrative example for the usefullnes of MWC. Assume that we are given a bipartite graph $G = (V_1, V_2, A)$ where the vertices of $V_1$ know a message that has to be broadcasted to the vertices of $V_2$. I.e., the initial set of informed vertices is $V_1$. We 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, and the *length* of the task (i.e., the time it takes to complete it) depends upon which vertex of $V_1$ transmits the message to $v_2$. I.e., for each edge $e = (v_1, v_2) \in A$ there is a weight $w(e) \in Z^+$, such that if $v_1$ transmites the message to $v_2$ then it takes $w(e)$ time units for $v_2$ to complete the task

$t_{v_2}$, starting from the arrival time of the message (and of course, no vertex of $V_2$ can start performing its task before it is informed). It is required to minimize the completion time of the entire process, namely, the time by which every vertex in $V_2$ completes its job. This scheduling problem is called *Bipartite Edge Scheduling* (BES). We call this problem *Bipartite Vertex Scheduling* when for every vertex of $V_2$ its entering edge weights are identical. Once a control function is defined, intuitively, it is most efficient for $v$ to send the message along its edges, in decreasing order of weights. It is therefore easy to see that the following (pseudo polynomial) procedure will be optimal to $BES$.

**Algorithm 3.6** *An algorithm for BES*

1. Compute a minimal control function $F$.
2. Suppose that $v \in V_1$ dominates the vertices $v_1, \ldots, v_k \in V_2$, and without loss of generality assume that $w(v, v_i) \geq w(v, v_{i+1})$, for each $i$.
3. Every vertex $v$ sends the message to the vertices $v_i$ in increasing order of indices.
4. A vertex $v' \in V_2$ starts performing its task, immediately after it is informed.

**Fact 3.7** *Procedure 3.6 optimally schedules any BES instance.*

Since BVS is a special case of BES we can deduce:

**Corollary 3.8** *There exist an optimal pseudo polynomial 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 the clusters formed when extracting $V_0$ from the graph are $C_1, \ldots, C_k$. Suppose that every cluster $C$ contains a *representative* vertex $v_C$ such that every vertex $v \in V_0$ is either connected to $v_C$, or not connected to any vertex of $C$. Further, suppose that we can optimally compute the broadcast times $b(v_C, C)$ for any cluster $C$ (e.g., when the clusters $C_i$ are trees or grids or of logarithmic size etc). For example, if $V_0$ contains a single vertex and the clusters are trees then $G$ is itself a tree.

Intuitively, we regard the task of each representative, as performing broadcast in its corresponding cluster. Since in any cluster only the single vertex representative is connected to the "outside world", the vertices outside the cluster can not aid in this broadcast process.

The above special case of the SMBT problem can be solved by applying the solution method for the MVWC problem as follows. Form the control graph $D_{V_0,G}$ of $V_0$ in $G$ as in Def. 2.5. The weight on a vertex $C_i \in V_2$ is the broadcast times of the representative of $C_i$, in $C_i$. It is straightforward to show that the solution to the MVWC problem is exactly the broadcast time from $V_0$. Further, note that since the edge weights in this problem represent broadcast times in subcluster, it follows from Fact 2.3 that they are no larger than $n-1$, and the MVWC 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 [?] 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. 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 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.

For proving 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 [?]. The first heuristic is based on defining for a set of vertices $V' \subseteq V$,

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

At each round $i$, select from all possible maximum matchings between the set of informed vertices and the set of noninformed ones, 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 $n-2$ and 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 [?] is the following. Define the *eccentricity* of a set $V' \subset V$ to be

$$Dist(V') = \Sigma_{v \in V'} Diam(v)$$

Choose, among all the possible maximal matchings, a matching for which the eccentricity of the set of newly informed vertices is maximal. In a similar way to the above it is easy to see that there are cases in which this approach leads to broadcast time of $\Omega(n)$.

The last heuristic suggested in [?] 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 time that is $\Omega(\sqrt{n}\,)$ 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 2.3, 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 which 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. This scheme is based upon the following lemma.

**Lemma 4.2** *The set of vertices of any graph $G = (V, E)$ can be (polynomialy) 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:** By direct construction. Let us next present the decomposition algorithm. The algorithm maintains three different sets of clusters $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$.

**Algorithm 4.3**

```
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 $\{C'' \mid C'' \subseteq C \setminus C', |C''| = \lceil \sqrt{n} \rceil\}$ to $\mathcal{A}$.
        Add the clusters $\{C'' \mid C'' \subseteq C \setminus C', |C''| > \lceil \sqrt{n} \rceil\}$ 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 $\mathcal{A}$ can not 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 $C \in C$, 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 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.

Let $G = (V, E)$ be a graph and $V' \subset V$ subset of the vertices. Form the control graph of $V'$, $D_{V',G} = (V_1, V_2, A)$, in $G$ as in Def. 2.5. Let the weight of each edge be 0. In this scenario we claim:

**Lemma 4.4** *Let $F$ be a minimum control function for $D_{V',G}$. Then $w(F) \leq b(V', G)$.*

**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 $C_i \in V_2$ s.t. $v$ transmits the message to $C_i$ in the $j$'th round (or later). By the way $F'$ was chosen, $j \leq b(V', G)$. Thus by definition, $w(F') \leq b(V', G)$. However, the weight of $F$ satisfies $w(F) \leq w(F')$. The proof follows. ▮

In the next lemma we use a shortest paths tree $SPT(v, S)$ rooted at a vertex $v$ and leading to a set $S$ of vertices (see Def. 2.4). 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. 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 $v$ ([**?**]). Let us call the optimal scheme for broadcasting in a tree the *OT scheme.*

**Lemma 4.5** *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 $l - 1 + Diam(v)$ 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. ▮

Now we are ready to use the above lemmas to approximate broadcast on general graphs.

**Algorithm 4.6** *Approximating broadcast in general graphs*
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 $\mathcal{B}$ as in 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 $F$. Assume that a vertex $v'$ dominates clusters $C_1, .., C_k \in \mathcal{B}$. Choose for each $C_i$ an arbitrary vertex $v_i \in C_i$ connected to $v'$ and deliver the message from $v'$ to $v_1, .., v_k$ (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 4.7** *The broadcast time of* Alg. 4.6 *from a vertex $v_0$ in a graph $G$, is bounded by $3 \cdot \sqrt{n} + Diam(v_0) + b(v_0)$ time units.*

**Proof:** By Lemmas 4.5 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 2.3 imply that stage 3 takes no more than $\sqrt{n}$ time units. By Lemma 4.4, $w(F) \leq b((\bigcup \mathcal{A}) \cup \{v_0\}) \leq b(v_0)$, 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 Alg. 4.6 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 $O(\sqrt{n}/Diam(G))$ approximation scheme. In order to improve upon that it may be needed 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 4.6 can be generalized to give a good approximation scheme for the open path broadcasting problem. It is easy to see that Lemma 4.4 holds even in the open path model.

**Lemma 4.8** *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 fact proven in [?].

**Fact 4.9** [?] *Broadcasting in the open model on a path of $m$ vertices can be completed in $\lceil \log m \rceil$ time units.* ▮

For proving Lemma 4.8 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 $v''$. I.e., 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 to a union of disjoint paths where for each path, one end vertex knows the message. The second stage, each informed end vertex of each path, informs the rest of the vertices in the path as in Fact 4.9.

In analyzing the delays occurring before a message reaches a leaf $u$, we separate our analysis to the first and second stage. This first stage is handled by a charging rule somewhat similar to that used in the proof of Lemma 4.5. Note that at each time step $t$, there is a unique ancestor $low(u, t)$ of $u$ that is responsible for it, namely, the lowest ancestor currently holding the message. At time step $t$, $low(u, t)$ 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 4.9 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 $Choose - Rep$ to approximate OMBT.

Throughout the execution of procedure $Choose - Rep$ we extract trees from $G$.

**Algorithm 4.10** *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) $i \leftarrow i + 1$.
3. **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 4.11** *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. $f \leq \frac{\log n}{\log\log n}$ ▮

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 4.12** *Approximation algorithm for 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 4.8.
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 4.8.
   (b) Let $G'_i$ denote, $G'_i = G \setminus \bigcup_{u \in R_i} T_i^u$. 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, as in step 5 of Alg. 4.6.
   (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 4.8.
   (f) $R \leftarrow R_{i-1}, i \leftarrow i - 1$.
5. **until** $i = 1$.

It is clear from the algorithm that when stage 4(e) of Alg. 4.12 is completed, all the vertices of $R_{i-1}$ know the message. It follows that at the end all the vertices are informed.

**Theorem 4.13** Alg. 4.12 *is a* $O(\log n/\log\log n)$ *approximation scheme for OMBT.*

**Proof:** By the definition of $Choose - Rep$, $|R_f| \leq \log n$. Thus it follows by this fact and Lemma 4.8 that step 2 of Algorithm 4.12 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 $\lceil \log n \rceil$, and thus by Lemma 4.8 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 4.4 that the number of time units used is bounded by $b_{op}(v_0)$. Finally, it follows from step (b) of Alg. 4.10 that for every $j$, $C_j^i \cap R_{i-1} \leq \log n$, thus by Lemma 4.8, 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 $O(b_{op}(v_0) + \log n) = O(b_{op}(v_0))$. By Claim 4.11, $f \leq \log n/\log\log n$. Hence the desired result follows. ▮

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 by $n^{1/k}$ for some $k$ smaller than $\log n/\log\log n$, it is possible to construct trees with $n^{1/k}$ leaves and get an $O(k)-$approximation scheme for $k < \log n/\log\log n$.

The method of Alg. 4.12 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 Alg. 4.12 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 [?]. For such graphs the scheme of Alg. 4.6 yields only an $O(\frac{\sqrt{n}}{\log n})$ approximation ratio. In contrast, Alg. 4.12 is an $O(\log n/\log\log n)$ approximation scheme for random graphs, with high probability.

**Corollary 4.14** *There exists a polynomial approximation algorithm that broadcasts in* $G_{n,p}$ *in no more then* $O(\log n/\log\log n) \cdot b(G_{n,p})$ *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 [**?**]. The idea is to find a small set of vertices whose removal splits the graph into connected components of roughly equal size, and then 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 $\rho$ 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 $sep(G')$. 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 4.4. 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 $C_1, ..., C_k$. Choose for each $C_i$ an arbitrary nonempty subset $C'_i \subset C_i$. We can use the fact that in broadcasting it is needed to inform the vertices of $C'_i$ to achieve a lower bound on the best possible time for broadcasting. We use the technique developed for solving MWC problems in Subsection 3.2. Let us first define a MWC instance $B' = MWC(V_0, \{C'_1, \ldots, C'_k\})$ 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 $d^v_{C_i} \equiv dist(v', C'_i)$ to the edge $(v, C_i)$.

(Note that in Lemma 4.4 the construction is similar, except that we choose as subset $C_i' = C_i$, for every $i$.) We claim the following:

**Lemma 5.2** *If $F$ is a minimal control function for $B'$, then $w(F) \leq b(V_0, G)$.*

**Proof:** First we argue the following technical claim, used also in [?]..

**Claim 5.3** *Let $d_i, t_i \in Z^+$, for $1 \leq i \leq k$, such that $d_k \leq d_{k-1} \leq \ldots \leq d_1$, 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 originated 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 non empty 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, $t_i + dist(u, C_i) \leq b(V_0, G)$, 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 $w(F') \leq b(V_0, G)$. Since $F$ is a minimum control function, $w(F) \leq w(F') \leq b(V_0, G)$. ▮

Note, that we can use an algorithm similar to Alg. 3.6 to establish:

**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 $w(F)$ 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 5.5**

```
1. $V_0 \leftarrow \{v\}$.
2. Construct a separator $sep(G)$ for $G$.
3. Build an arbitrary tree $SPT(v, sep(G)) = (V_1, E_1)$ rooted at $v$ and leading to the
   members of $sep(G)$. 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 $C_1, ..., C_k$.
       Each $C_i$ has a separator $sep(C_i) = C_1^i \cup C_2^i \cup \ldots C_{l_i}^i$, where $C_1^i, C_2^i, \ldots, C_{l_i}^i$ are
       $sep(C_i)$'s connected components.
       repeat
          i. For each $i$ pick the lowest index $j(i)$ that hasn't been chosen yet (for $i$).
          ii. Build the instance $B' = MWC(V', \{C_{j(i)}^i : 1 \leq i \leq k\})$ of the MWC problem,
              as described.
          iii. Send the message to at least one vertex of $C_{j(i)}^i$ for every $i$ and $j$, using a
               minimal function $F$ and the scheme suggested in Fact 5.4.
       until the $C_j^i$'s clusters are exhausted for every $i$ and $j$.
   (b) For every $i$ and $j$, broadcast (in parallel) the message within $C_j^i$ using the best
       known scheme for $C_j^i$. (If no good known scheme exists for the kind of graph $C_j^i$
       is, broadcast using an arbitrary tree.)
   (c) $V_0 \leftarrow V_0 \cup sep(C_1) \cup \ldots \cup sep(C_k)$.
   Until $V_0 = V$.
```

It is easy to see that when Alg. 5.5 terminates, all the vertices in $V$ are informed.

**Theorem 5.6** Alg. 5.5 *terminates the broadcast process 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 $\varphi(G) + Diam(v)$ 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 $sep(C)$ 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 $Sep(C)$ 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 $O(\log n) \cdot \varphi(n) \cdot b_{op}(v)$.

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, is bounded by $\varphi(\rho^i \cdot n) \leq \varphi(n)$. Thus by this is also a bound on the time spent in step 5(b), for a fixed $i$. Summing up this bounds for every $i$ we conclude that the total time spent in step 5(b) is bounded by

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

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 $sep(C)$ is connected and $b(sep(C)) \leq k$ for some integer $k < \varphi(n)$, then the bound is improved to

$$O(\log n) \cdot (b(v) + k). \quad (1)$$

## 5.2 Applications

In this subsection we give some examples of graph families for which Algorithm 5.5 can be applied. The first example is the one of *chordal graphs*. A *chord* in a cycle of at least 4 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 [?] regarding chordal graphs.

**Theorem 5.7** [?] *Every $n$-vertex chordal graph $G$ contains a (polynomialy 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 satisfying the condition of the theorem is also given in [?].

Thus chordal graphs always have separating sets that are connected (and moreover, are cliques). Since it is possible broadcast a message in a clique of $m$ vertices in $\lceil \log m \rceil$ time units, it follows from Eq. (1) that the time needed to broadcast in a chordal graph using the scheme of Alg. 5.5 is no more than

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

Consequently, Alg. 5.5 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 [?]. It follows from Theorem 5.6 that Alg. 5.5 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 a $k$-*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 [?].

In [?], 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 [?]. Thus Alg. 5.5 can be used to broadcast in a $k-$outerplanar graph achieving an $O(k \log n)$ approximation scheme. Thus we conclude.

**Theorem 5.9** *There exists an* $O(k \log n)$ *approximation algorithm for broadcasting in 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 [?] 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 Alg. 5.5 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 algorithm 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. [?]). The following theorem is shown in [?].

**Theorem 5.12** [?] *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 [?]. We use this to derive the following theorem.

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

**Proof:** Use the algorithm of [?] to compute the separators needed for Alg. 5.5. At each step of Alg. 5.5, 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 the rest of the vertices in increasing order, according to their clockwise ordering.

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 5.5 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

$$\sum_{i=1}^{\log_{3/2} n} O((\frac{2}{3})^i\cdot\frac{n^{1/4}}{\sqrt{\log n}}\,)\cdot b(v) + \sum_{i=1}^{\log_{3/2} n}(\frac{2}{3})^i\cdot n^{1/4}\sqrt{\log n} + O(n^{1/4}\sqrt{\log n}) + Diam(G)$$
$$= O(n^{1/4}/\sqrt{\log n}\,)\cdot b(v). \quad ▮$$

This is slightly better than the result given for general graphs in Theorem 4.7 (in the worst case).

# 6 Broadcasting in a tree of cliques

In this section we wish to present a broadcast scheme for graphs which 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** *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 $G$.*
2. *The auxiliary graph $T(G) = (\tilde{V}, \tilde{E})$ whose vertices are $\tilde{V} = \{C_1, \ldots, C_k\}$ and whose edges are*

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

   *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 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 etc. 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*, $CI(C)$, 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 $rank(C) = 0$ 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 $rank(C) = w(F)$, where $F$ is the minimal function w.r.t the MVWC problem resulting by the construction of step 1 taking the weight of a clique child vertex $C'$ to be $w(C') = rank(C')$.*
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

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

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 $C_1, \dots C_k$ denote $C$'s children in $G$, ordered by non increasing 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 $v_1, \dots, v_k$ 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 $\{v_1, \dots, v_k\}$. 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, for 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 $\{v_1, \dots, v_k\}$ 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 $v_5, v_6, v_7, v_8$; in round 5 $v_2, \dots, v_8$, send the message to $v_9, \dots, v_{15}$, etc.

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 6.4** *Approximation scheme $A_{2a}$*
Input: A TOC $G$ and a root clique $C_0$ and an informed vertex $v_0 \in C_0$.

1. 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.
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 $w \in Dom(C)$ 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 $rank(C') > rank(C'')$ then $w$ sends the message to a vertex in $C'$ before it sends it to a vertex in $C''$

Next, let us modify Alg. $A_{2a}$ to get Alg. $A_{2b}$. Instead of delivering the message within the clique using all the vertices through the entire process, as done in step 2 of Alg. $A_{2a}$, we use the Fibonacci delivery scheme. Thus in Algorithm $A_{2b}$ 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 $Dom(C) = \{v_1, \ldots, v_k\}$. For every vertex $v_i$, let $C_{max}(v_i)$ be the clique dominated by $v_i$ with maximal degree. Assume w.l.o.g 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 modifies version of Alg. $A_{2a}$ terminates, every clique contains at least one informed vertex (but not necessarily all the vertices in all the cliques are informed, since in any clique a vertex is informed only if it controls a non empty set of children cliques). Thus for terminating the broadcast, in every clique, the informed vertices, deliver (in parallel) 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, Alg. $A_{2b}$.

## 6.3 The broadcast complexity of the schemes

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: $z_1 = z_2 = 1$, $z_{i+2} = z_{i+1} + z_i$.

**Lemma 6.5** *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 hold for $i = 1$, $i = 2$ and $i = 3$ by a direct inspection. Now assume it is true for $i \geq 3$ and assume 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 $z_i = \frac{((1+\sqrt{5})/2)^i - ((1-\sqrt{5})/2)^i}{\sqrt{5}}$

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

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 claim the following:

**Lemma 6.6** *For any tree of cliques $G = (V, E)$ rooted at a clique $C_0$, $rank(C_0) \leq b(C_0, G)$.*

**Proof:** We prove the lemma by induction on the height of the TOC. If $h(T(G)) = 0$ the claim holds trivially since $rank(C) = 0$. Assume the claim for height $k$. Consider a tree of height $k+1$. Assume that the children of $C_0$ in $T(G)$ are $C_1, ..., C_l$. 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 $C'_1, ..., C'_j$, and $F'(C'_i) = v$ for every $i$, and without loss of generality assume that the cliques $C'_i$ are arranged by non increasing 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,

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

By Lemma 5.3,

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

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

**Lemma 6.7** *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 $t_0$ that 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 then via the vertices adjacent to $p(C')$, it follows by a the doubling argument of Fact 2.3(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 non increasing 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 two possible types:

1. delays encountered when a predecessor clique $C''$ of $C'$ delivers the message to other subtree, rather then 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 6.8** *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:** By Lemma 6.6 and straightforward induction on $h(T(G))$. ▮

We now consider second type delays. Let us first analyze the number of such delays in Algorithm $A_{2a}$. 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 $i$'th 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 conclude.

**Theorem 6.9** *Algorithm $A_{2a}$ 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 $A_{2b}$. Let $C_1, \ldots, C_l$ be the children of $C$, ordered by non increasing order of degrees, and let $Dom(C) = \{v_1, \ldots, v_k\}$. 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
$1 + \lceil 1.441 \cdot \log i \rceil + 2 + 2 =$
$(\lceil (1.441 \cdot \log i) \rceil - \lceil (1.441 \cdot \log PI(C_i)) \rceil) + \lceil (1.441 \cdot \log PI(C_i)) \rceil + 5$
$= O\left(\frac{\log i}{PI(C_i)}\right) + O(\log PI(C_i))$.

Thus the next claim follows from Lemma 6.7 and by induction on $h(G)$. Suppose that the $i$'th clique in the path in $T(G)$ from the root to a leaf $C'$ is $C_i$.

**Claim 6.10** *The number of second type delays encountered by the message before it reaches $C'$ in Alg $A_{2b}$ is bounded by*

$$O(deg(C)) + \sum_i \lceil \log PI(C_i) \rceil) \rceil + O(Diam(v)) \leq O(b(v)) + O(\sum_i \log PI(C_i)). \quad \blacksquare$$

For example, consider a parent $c$-restricted TOC $G$ for a constant $c$. It follows that Alg. $A_{2b}$ 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 than use the $A_{2b}$ scheme to broadcast from $v$. The fact that $b(v)$ and $b(v')$ differ by at most $Diam(G)$, 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 6.11** *Algorithm $A_{2b}$ 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.

## References