Energy Efficient Broadcast in Distributed Ad-Hoc Wireless Networks

Subhas Kumar Ghosh
Honeywell Technology Solutions Lab.,
151/1, Doraisanipalya, Bannerghatta Road,
Bangalore, INDIA-560076

E-mail: `Subhas.Kumar@honeywell.com`

Abstract

In this work we present algorithms for Minimum Energy Consumption Broadcast Subgraph *(*MECBS*) problem. First, we focus on designing distributed algorithms for* MECBS*. To our knowledge, this is the first work looking into the aspects of designing a distributed approximation algorithm for the* MECBS*. Our algorithm has approximation ratio of $2H_{n-1}$ with time complexity $\mathcal{O}(n \cdot \Lambda(\mathcal{G}))$, where $\Lambda(\mathcal{G})$ denotes the diameter of the communication graph $\mathcal{G}$. Second, we present an improved approximation algorithm for the* MECBS *problem with arbitrary asymmetric power requirement having performance ratio $1.5(\ln(n-1)+1)$, hence improving all known results for* MECBS *problem in most general case. Our improvement in* MECBS *problem also implies that there is a $1.5\ln(n-1)+2.5$ – approximation algorithm for strong connectivity with asymmetric power requirements.*

1 Introduction

In the context of ad-hoc wireless networks the problem of constructing a minimum energy consumption broadcast subgraph (MECBS) has received significant attention over last few years [6, 14, 3, 5, 7, 10, 2, 4]. Broadcast is a fundamental protocol for building applications over ad-hoc wireless networks, which allows a source node s to transmit a message to all other nodes in the network. In this work we focus on designing algorithms for building energy efficient broadcast subgraph of the network graph. Objective of the algorithm is to assign transmission power to nodes in the network such that a communication from a source s to all other nodes in the network can occur using directed (possibly multi-hop) paths, while energy consumption in one such message broadcast is minimized. Formally, we model the network as a complete directed graph $G = (V, E)$ with a cost function $c : E \to Q^+$ associated with its edges (we will say that cost is symmetric, if $c(u,v) = c(v,u)$). A node $s \in V$ is the source. A weight (transmission power) assignment $\omega : V \to Q^+$ to the nodes of G induces the directed graph G_ω as follows. It has the same set of nodes with G and a (directed) edge (u,v) belongs to G_ω if the weight assigned to node u is at least the cost of the edge (u,v), i.e., $\omega(u) \geq c(u,v)$. The *Minimum Energy Consumption Broadcast Subgraph* (MECBS) problem is following:

> MECBS:
>
> INPUT: $G = (V, E), c : E \to Q^+, s \in V$
>
> OUTPUT: A weight assignment ω to the nodes of V so that for any node $u \in V \setminus \{s\}$, the induced graph G_ω has a directed path from s to u.
>
> MEASURE: Total weight $\sum_{u \in V} \omega(u)$.
>
> GOAL: Minimize total weight.

In following we start by noting few important known results for MECBS problem.

Theorem 1.1. [8]: *Unless* NP $\subseteq$ DTIME$\left(n^{\log\log n}\right)$, MECBS *problem is not approximable within $(1-\epsilon)\ln n$ factor.*

Theorem(1.1) was obtained by presenting an approximation factor preserving reduction of minimum SET-COVER problem to MECBS problem.

Theorem 1.2. [2]: *There exists a polynomial time $2H_{n-1}$ factor approximation algorithm for* MECBS *problem with symmetric cost function.*

Theorem 1.3. [3]: *There exists a polynomial time $2 + 2\ln(n-1)$ factor approximation algorithm for* MECBS *problem with asymmetric cost function.*

Recently, in [4] authors presented a polynomial time $2 + 2\ln(n-1) - 2\ln 2$ factor approximation algorithm for MECBS problem with symmetric cost function.

In this paper we first present distributed approximation algorithm for general MECBS problem with symmetric cost function. To our knowledge, this is the first work looking into the aspects of designing a distributed approximation algorithm for MECBS problem. While it is possible to centrally compute the power assignment and then inform a node at which power level it should transmit, it is quite impractical for several reasons. Central computing station need to gather either location information of deployed nodes, or edge cost for each pair of nodes to construct the input instance, and by the time such information is gathered, network topology may have changed. Any distributed implementation also provides greater degree of fault-tolerance. Importantly, distributed implementation in this case provides certain flexibility. The algorithm we present has two distinct phases. First phase of this algorithm constructs a weakly connected graph and this construction is independent of the input s, and hence the result of the first phase can be reused – as long as the network topology remains unchanged.

We also present an improved approximation algorithm for the MECBS problem extending the algorithm of [3], and using the ideas from [8]. This algorithm has performance ratio $\frac{3}{2}(\ln(n-1)+1)$, hence improving all known results for MECBS problem. In [3], the result of MECBS problem was used to obtain results for several other connectivity problems. Importantly, Strong connectivity: When induced graph G_ω is strongly connected. Our improvement in MECBS problem also implies that there is a $1.5\ln(n-1)+2.5$ – approximation algorithm for Strong connectivity with asymmetric power requirements.

Organization: Rest of the paper is organized as follows. In section 2, we present and analyze our distributed algorithm for MECBS problem. This algorithm is based on the greedy algorithm of [2], and achieves approximation ratio of $2H_{n-1}$ with time complexity $\mathcal{O}(n\cdot\Lambda(\mathcal{G}))$, where $\Lambda(\mathcal{G})$ denotes the diameter of the communication graph $\mathcal{G}$. Then in section 3 we present an improved approximation algorithm for the MECBS problem extending the algorithm of [3], and using the ideas from [8]. This algorithm has performance ratio $\frac{3}{2}(\ln(n-1)+1)$, hence improving all known results for MECBS problem. Finally we conclude the paper in section 4.

2 Distributed algorithm

In this section we present our distributed approximation algorithm for general MECBS problem where edge cost is symmetric.

Model and notations: We assume the classical $\mathcal{LOCAL}$ distributed model of computation (cf. [11, Chapter 2]). In specific, we model the network as a connected graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$. Vertices $\mathcal{V}$ of the graph are nodes having limited information (i.e. small part of the input is given to each vertex). Edges of this graph is weighted as $c:E\to Q^+$, and a distinguished vertex $s\in V$ is known to every node in networks.

We assume that communication is synchronous, i.e. each node follows a global clock. Thus round of the distributed algorithm is determined by the global clock. In each round nodes can send and receive message of unlimited size and local computation in nodes are atomic. Time complexity of the algorithm is determined by the rounds as measured by the global clock in worst case.

We shall denote the vertex set V by $\{1,\ldots,n\}$, where n is the number of nodes in the network. Following notations of [2] for a vertex i let $\mathcal{C}_i=\cup_{j\in V}\{c(i,j)\}$ denote the set of all possible power levels allowing i to reach j directly using edge (i,j). For a vertex $i\in V$ and any transmission power $\omega\in\mathcal{C}_i$, let $\mathcal{F}_\omega(i)=\{j\in V|j\neq i\wedge\omega\geq c(i,j)\}$ be the set of reachable vertices j from i with transmission power of vertex i set to ω (call them follower of vertex i). If there exists a vertex $j\in V$ such that $i\in\mathcal{F}_\omega(j)$, for some $\omega\in\mathcal{C}_j$, then we set $\mathcal{L}_\omega(i)=\{j|i\in\mathcal{F}_\omega(j)\}$ (call j, a leader of vertex i). Let $\mathcal{P}$ be a partition of the vertex set into disjoint family of vertex sets, then by $\mathcal{P}_\omega(i)=\{X\in\mathcal{P}|X\cap\mathcal{F}_\omega(i)\neq\emptyset\}$ we will denote the elements of the partition that has at least one vertex reachable from vertex i with transmission power of vertex i set to ω. Given a partition of the vertex set $\mathcal{P}$, a vertex i and a transmission power $\omega\in\mathcal{C}_i$ we define the cost-effectiveness $\rho_\omega(i,\mathcal{P})$ of ω at i, w.r.t. $\mathcal{P}$ as

$$\rho_\omega(i,\mathcal{P})\triangleq\begin{cases}\frac{\omega}{|\mathcal{P}_\omega(i)|} & \text{when } |\mathcal{P}_\omega(i)|>0\\ \infty & \text{otherwise}\end{cases}$$

For a distributed algorithm, the view of a global partition may be inconsistent, i.e. each vertex may have different local view of partition in a round. Hence, in following $\mathcal{P}(i)$ will denote the view of partition from a vertex i. Given a transmission power assignment ω, for each vertex we associate $\omega(i)$ as assigned power to vertex i. Let, $G_\omega[X]$ denote the subgraph of G_ω induced by a set $X\subseteq V$ w.r.t ω. Note that G_ω could be weakly connected, thus, let $\Pi(s,G_\omega)$ denote the set of vertex i, connected to a vertex s through a directed path from s to i using the edges of G_ω.

2.1 Algorithm

The algorithm we present here is a distributed version of the greedy assignment algorithm described in [2]. Our distributed algorithm has two distinct phases. In the first phase the algorithm constructs a weakly connected graph $G_\omega=G_\omega[V]$ spanning the vertex set V. In the second phase of the algorithm power assignment ω is modified so that at the end of this phase $\Pi(s, G_\omega) = V$. It is interesting to note that if the network topology do not change, then this two phases can be executed separately – as first phase is insensitive to input s. Thus, for a static topology, nodes can store the assignment ω after first phase, and to broadcast from a distinct node s second phase can be executed as required. In sequel, we first describe these two phases of the algorithm.

Forming weakly connected graph: In this phase, input for every vertex is $\mathcal{C}_i$. Message m from a vertex i to a set of vertices $S \subseteq V$ is denoted as $i \longrightarrow S : \langle m \rangle$. The procedure for vertex i is described in Figure-1 in detail, and what follows is a high level description. The procedure has several *states*, and a vertex moves from one state to another in a round, starting at Initialize state (which is executed one-time at start). In every round in Propose state, a vertex sends proposal to components in $\bar{\mathcal{P}}$ corresponding to its minimum cost-effectiveness $\bar{\rho}$, which is computed with respect to $\mathcal{P}(i)$, the local view of partition of vertices in V as observed by vertex i. In this case we assume that a node maintains a data structure where id of the leader is stored for each component. Note that this is nodes local view, however, it will be apparent that whenever a node updates it view of partitions, it always receives the information on components current leader explicitly. Thus in the presentation of the algorithm, we will not distinguish between a component and its leader while sending a message, i.e. when a node i sends a message to a component $X \in \mathcal{P}_i$, its sends the message to the leader of X, as stored in the local view of node i, but we will denote this as $i \longrightarrow X : \langle m \rangle$. As we have noted, global view of weakly connected component may not be available to a vertex. To address this a vertex maintains a leader l^*. When a vertex i receives a proposal, and vertex i is part of a larger component, i forwards the proposal to its leader l^*. Let $\mathcal{R}(j) = \{\langle j, \bar{\rho}_j, \bar{\omega}_j \rangle\}$, be the set of all proposals received from vertex j by a component. We ensure that there is a single leader in each component, and hence a leader can recompute the cost-effectiveness of j as:

$$\rho'_j \triangleq \frac{\bar{\omega}_j}{\left(\frac{\bar{\omega}_j}{\bar{\rho}_j} - |\mathcal{R}(j)| + 1\right)}$$

If $\rho'_j \neq \rho_j$, then vertex j has inconsistent view of the global partition, and node j is informed about this. In Consistency state a node updates its view on partition, re-computes $\bar{\rho}$, and re-proposes in the next round. If all received proposals are consistent then a node selects proposal from j with minimum ρ_j and sends back accept. When a vertex i observes that all its proposals in a round has been accepted, it sends sends *'Greet'* message to components merged by this new power assignment of i. Updating partition and managing local view od partition is closely related. A node updates its view based on the information on merged components it receives as a part of the message. Local view is updated when

1. A vertex sends *'Greet'* message.
2. Receives *'Greet'* message.
3. Upon receiving *'Inconsistent'* message.
4. On receiving *'Leader'* message.

Group management requires that in a merged component every vertex has the same leader as l^*. When a vertex has more than one incoming edges, it selects the vertex with minimum id as its leader. Any change in leader is informed to all vertices in a component. Finally, Terminate state is executed by a vertex when it observes that a single component is formed and the graph is weakly connected.

Forming directed spanning tree: The objective of this phase is to adjust transmission powers so that given a distinct vertex $s \in V$ the transmission graph G_ω contains a directed tree rooted at s and every vertex is reachable from s using a directed path. Since the edge cost is assumed to be symmetric and output of the first phase is a weakly connected graph – this requires at most doubling the existing edges induced by ω. The algorithm at each vertex repeatedly checks if there is a vertex $j \in \mathcal{L}_\omega(i) \,|\, j \in \Pi(s, G_\omega)$. Surely if this is satisfied then vertex can halt. Alternatively, it checks if $\exists j \in \mathcal{F}_\omega(i) \,|\, j \in \Pi(s, G_\omega)$, while if this is satisfied then assigning $\omega(i)$ as power of vertex j doubles the edge and vertex $i \in \Pi(s, G_\omega)$ is satisfied. The algorithm has been given in Figure-1.

2.2 Analysis of the algorithm

In this section we first show the correctness of the algorithm. This amounts to showing that at the end of second phase the induced graph G_ω with power assignment $\forall i \in V, \omega(i)$ contains a directed spanning tree rooted at s. We will prove this first by establishing few invariant properties of the algorithm. Let $\mathcal{P}_i^{(k)}$ denote the partition of vertices at the end of k-th round as observed by vertex i, also let $\mathcal{P}^{(k)}$ denote the partition of vertices at the end of k-th round as observed globally. By $\omega_1^{(k)}$ and $\omega_2^{(k)}$ we will denote the partial power assignment at the end of k-th round in first and second phase of the algorithm respectively. Also let $\mathcal{P}^{(k)}_{\omega_1^{(k)}(i),i}(i) = \left\{ X \in \mathcal{P}_i^{(k)} | X \cap \mathcal{F}_{\omega_1^{(k)}(i)}(i) \neq \emptyset \right\}$, and let $\mathcal{P}^{(k)}_{\omega_1^{(k)}}$ denote the partition of vertices at the end of k-th round as observed globally under partial power assignment at the end of k-th round in first phase.

```
Initialize:
$\mathcal{P}(i) := \{\{1\}, \{2\}, \ldots, \{n\}\}$.
$\omega(i) := 0$.
$\mathcal{F}_\omega(i) = \mathcal{L}_\omega(i) := \emptyset$.
$l^* = i$.
Propose:
$\bar{\rho} := \min_{\omega \in \mathcal{C}_i} \{\rho_\omega(i, \mathcal{P}(i))\}$.
$i \longrightarrow \bar{\mathcal{P}} : \langle i, \bar{\rho}, \bar{\omega} \rangle$.
Receive:
$\rho'_j := \bar{\omega}_j / \left(\bar{\omega}_j / \bar{\rho}_j - |\mathcal{R}(j)| + 1\right)$.
if $\exists j \,|\, \rho'_j \neq \rho_j$ then
    $i \longrightarrow j : \langle i, \text{'Inconsistent'}, \mathcal{S}(j) \rangle$.
else
    Select proposal from $j$ with minimum $\rho_j$.
end if
Update $\mathcal{P}(i)$ using $\bar{\mathcal{P}}$.
Greet/Regret:
if $\forall j \in \bar{\mathcal{P}}, ack_i(j) = true$ then
    $\omega(i) := \bar{\omega}$.
    $i \longrightarrow j \in \bar{\mathcal{P}} : \langle i, \bar{\mathcal{P}}, \text{'Greet'} \rangle$.
end if
if $\exists j \,|\, ack_j(i) = true \wedge j \in \mathcal{L}_\omega(i)$ then
    $i \longrightarrow \bar{\mathcal{P}} : \langle i, \text{'Regret'} \rangle$.
end if
Terminate:
if $|\mathcal{P}(i)| = 1$ then
    Halt.

end if
```

(a) Phase-I

```
procedure TREE $(\omega(i), s, \mathcal{L}_\omega(i), \mathcal{F}_\omega(i))$    ▷ Let the value of variable $flag$ denote the answer to predicate $v \in \Pi(s, G_\omega)$, which is set to true for vertex $s$ and to false for all $i \in V \setminus \{s\}$ before this phase starts.
    if $\exists j \in \mathcal{L}_\omega(i) \,|\, j \in \Pi(s, G_\omega)$ then
        $flag :=$ true.
    else if $\exists j \in \mathcal{F}_\omega(i) \,|\, j \in \Pi(s, G_\omega)$ then
        $i \longrightarrow j : \langle \omega(i), \text{'Adjust-power'} \rangle$.
        $flag :=$ true.
    end if

    Adjust:
    Let $\bar{\omega} = \max\{\omega(j) \,|\, \langle \omega(j), \text{'Adjust-power'} \rangle\}$
    $\omega(i) := \bar{\omega}$.

end procedure
```

(b) Phase-II

Figure 1. Distributed algorithm for MECBS

Lemma 2.1. *Let $\mathcal{P}^{(k)}$ denote the partition of vertices at the end of k-th round as observed globally, and $X \in \mathcal{P}^{(k)}$. For any $i, j \in V$, such that $i \in X$ and $j \in V$, any proposal $j \longrightarrow i : \langle j, \bar{\rho}_j, \bar{\omega}_j \rangle$ reaches a distinct leader vertex $l_X^{(k)} \in X$.*

Proof. For $k = 0$ this is true. Consider any $k > 0$. Observe that all proposal for i and from $\mathcal{F}_\omega(i)$ is forwarded to $l^* \neq i$. On the other hand, if $l^* \neq i$, then l^* is selected from a vertex in $\mathcal{L}_\omega(i) \neq \emptyset$. Hence, if component X is formed as directed tree with a single root, then all proposals to vertices in component X reaches this root. On the other hand, if there is a vertex $i^* \in X$ such that $|\mathcal{L}_\omega(i^*)| > 1$, then i^* selects the vertex with smallest id in $\mathcal{L}_\omega(i)$ as leader. This change is communicated recursively to the set $\{\mathcal{L}_\omega(i) \cup \mathcal{F}_\omega(i) \,|\, i \in \mathcal{L}_\omega(i^*) \cup \mathcal{F}_\omega(i^*)\}$. Surely, this set is same as X. □

Second important invariant property of the algorithm is monotonically non-decreasing power assignment to any vertex.

Lemma 2.2. *For any vertex $i \in V$ and for any integers k, l, m, q such that $k < l$ and $m < q$, $\omega_1^{(k)} \leq \omega_1^{(l)} \leq \omega_2^{(m)} \leq \omega_2^{(q)}$.*

Proof. Initially, $\forall i \in V, \omega(i) = \omega_1^{(0)}(i) = 0$. Let $\omega(i)$ be set to $\omega_1^{(k)}(i)$ at the kth round. For any $h > k$ and $\omega' \in \mathcal{C}_i$ such that $\omega' \leq \omega_1^{(k)}(i)$, we have

- $\mathcal{F}_{\omega'}(i) \subset \mathcal{F}_{\omega_1^{(k)}(i)}(i)$.
- $\mathcal{P}_{\omega'}^{(h)}(i) = \emptyset$ and hence $\rho_\omega\left(i, \mathcal{P}^{(h)}(i)\right) = \infty$, and
- If $\mathcal{P}_{\omega',i}^{(h)}(i) \neq \emptyset$ then i receives *'Inconsistent'* message from the leaders of all $X \in \mathcal{P}_{\omega',i}^{(h)}(i)$ and eventually makes the view $\mathcal{P}_{\omega',i}^{(h)}(i)$ consistent with global view. This is guaranteed by lemma 2.1, as all vertices in X forwards message to same leader vertex.

This implies for any admissible power assignment in round l such that $k < l$ we have $\omega_1^{(k)} \leq \omega_1^{(l)}$.

In the second phase of the algorithm, let us assume that $\omega(i) = \omega_2^{(0)}(i) = \omega_1^{(r)}(i)$, where r is the number of rounds in first phase of the algorithm. For any $m \geq 1, \omega(i)$ remains unchanged if $i = s$ or $\mathcal{L}_\omega(i)$ has a vertex j such that $j \in \Pi(s, G_\omega)$. On the other hand if $\exists j \in \mathcal{L}_\omega(i)$ such that $j \notin \Pi(s, G_\omega)$, and i is already reachable by s, i.e. $flag = true$ for i, then $\omega(i)$ is changed to $\omega(j)$. In this case it must be there exists an edge $(j, i) \in G_\omega$, but no (i, j) edge, and it must be $\omega(j) > \omega_2^{(m)}(i)$. □

Final invariant property of the algorithm is that induced graph is a colection of weakly connected components:

Lemma 2.3. *For each set $X \in \mathcal{P}^{(k)}, G_{\omega_1^{(k)}}[X]$ is weakly connected.*

Proof. For $k = 0$ this is true. Consider any $k > 0$. By lemma 2.2, for all $i \in V, \omega_1^{(k)}(i)$ is monotonically non-decreasing. In the kth round, power for each vertex i, $\omega_1^{(k)}(i)$ is chosen such a way that for each $X \in \mathcal{P}_{\omega_1^{(k)}(i),i}^{(h)}(i)$, X has at least one vertex $j \in \mathcal{F}_{\omega_1^{(k)}(i)}(i)$. This choice is valid if and only if view of i for all $X \in \mathcal{P}_{\omega_1^{(k)}(i),i}^{(k)}(i)$ is consistent, i.e. each X is also present in global view $X \in \mathcal{P}_{\omega_1^{(k)}}^{(k)}$, as otherwise i receives inconsistent message. While if this choice is valid and is most cost effective for all the sets in $\mathcal{P}_{\omega_1^{(k)}(i),i}^{(k)}(i)$ then $\omega_1^{(k)}(i)$ is assigned to i. Thus, in the view of i $G_{\omega_1^{(k)}(i)}[X]$ is weakly connected, and since all these sets X are also present in the global view $G_{\omega_1^{(k)}}[X]$ is also weakly connected. □

Using these invariant properties we can prove that at the end of second phase the induced graph G_ω contains a directed spanning tree rooted at s, with power assignment $\forall i \in V, \omega(i)$.

Theorem 2.1 (Correctness)**.** *The output ω of the two phase algorithm induces a graph G_ω such that in G_ω every vertex $i \in V \setminus \{s\}$ is reachable from s by a directed path.*

Proof. Let us start by assuming that in first phase all vertices eventually reaches the terminate condition and halts, and in second phase for all $i \in V$, value of the variable $flag$ is eventually set to *true*.

When first phase terminates, it is guaranteed that network has a weakly connected component spanning all vertices as induced by ω, this is a direct consequence of terminating condition $|\mathcal{P}(i)| = 1$, and global condition proved in lemma 2.3.

In the second phase, when all vertices reaches terminate condition, G_ω has directed path from s to all $i \in V \setminus \{s\}$. This can be observed by following. By lemma 2.2, we have $\Pi(s, G_\omega)$ always expanding and initially $\Pi(s, G_\omega) \neq \emptyset$, namely $s \in \Pi(s, G_\omega)$. Finally, if there exists an i such that $i \notin \Pi(s, G_\omega)$, then it must not be in terminate condition.

In following, we prove that in first phase all vertices eventually reaches the terminate condition and halts, hence completing the proof. By lemma-2.2, again power assignment of every vertex is monotonically non-decreasing in this phase. Also by lemma 2.1, if the local view of a vertex i has $|\mathcal{P}^{(k)}_{\omega_1^{(k)}(i),i}(i)| > 1$ for $k > 0$ and $\omega_1^{(k)}(i)$, while globally $|\mathcal{P}^{(k)}_{\omega_1^{(k)}}| = 1$, then all proposals made by by vertex i, reaches a single leader, and i receives *'Inconsistent'* message from the leader, making its view eventually consistent. □

Let OPT be the cost of an optimal solution ω^*, for a given instance $x = \langle G, c, s\rangle$ of MECBS problem. We will show following:

Theorem 2.2 (Performance Guarantee)**.** *Distributed algorithm produces a solution ω such that $\sum_{u \in V} \omega(u) \leq 2 \cdot H_{n-1} \cdot \mathsf{OPT}$.*

Proof. In every round $r \geq 0$ of the first phase of the algorithm, we have a set of vertex and corresponding power assignment pairs that is selected.

Let, $\mathcal{Q}^{(r)} = \{\langle i_1, \omega_1^{(r)}(i_1)\rangle, \ldots, \langle i_p, \omega_1^{(r)}(i_p)\rangle\}$ be the set of such pairs at the end of round r, that is selected by the algorithm. Our first observation is that the sets $\mathcal{F}_{\omega_1^{(r)}(i_j)}(i_j)$ corresponding to each i_j are disjoint. This is by construction, as each component in $\mathcal{F}_{\omega_1^{(r)}(i_j)}(i_j)$ selects i_j as minimum cost effective proposal sender (breaking ties arbitrarily), and i_j sends *'Greet'* message iff all proposed components send acknowledge message. Next we observe that each i_j corresponding to sequence of pairs in $\mathcal{Q}^{(r)}$ is not in $\mathcal{F}_{\omega_1^{(r)}(j)}(j)$ for any $j \in V$ in round r. This is again a consequence of the construction, as otherwise i_j sends *'Regret'* message.

With these two observations we can claim that all components $\cup_{j=1}^{p} \mathcal{P}^{(r)}_{\omega_1^{(r)}(i_j),i_j}(i_j)$ that are merged in round r, are disjoint from one another. Also, the cost of each $X_{i_j,t} \in \mathcal{P}^{(r)}_{\omega_1^{(r)}(i_j),i_j}(i_j)$ can be written as $cost(X_{i_j,t}) = \rho_{\omega_1^{(r)}(i_j)}(i_j, \mathcal{P}^{(r)}_{\omega_1^{(r)}(i_j),i_j}(i_j))$. Hence, we can write the cost incurred in round r as sum of the costs of these disjoint sets. Now, observer that the components that are merged in round r and $r+1$ for $r \geq 0$ are also disjoint from one another. Thus, for sequence $\mathcal{Q} = \bigcup_{r=0}^{k} \mathcal{Q}^{(r)}$ and corresponding components that are merged we have a family of sets

$$\mathcal{T}_{\mathcal{Q}} = \bigcup_{r=0}^{k} \left(\bigcup_{j=1}^{p} \mathcal{P}^{(r)}_{\omega_1^{(r)}(i_j),i_j}(i_j) \right),$$

over which we can compute the cost of the algorithm. Number these sets in the order in which they are merged by the algorithm, and breaking ties for the sets that are merged in the same round arbitrarily. Next we show that for kth set $X_k \in \mathcal{T}_{\mathcal{Q}}$, we have $cost\,(X_k) \leq \mathsf{OPT}/\,(n-k)$. First, note that $|\mathcal{T}_{\mathcal{Q}}| \leq n-1$, as in every round at least one vertex merges at least two components with its assigned power. Let us assume that $X_k \in \mathcal{P}^{(r)}(j)$, corresponding to vertex j in round r, having cost $\rho_{\omega_1^{(r)}(j)}\left(j, \mathcal{P}^{(r)}(j)\right)$. For any such k, the remaining components of the partition can be merged into single component by a sequence of optimal assignments of cost at most OPT. Thus there must be an assignment of cost effectiveness at most $\mathsf{OPT}/\,(n-k)$. In the round X_k was merged, there must be at least $n-k$ components remaining and we have that:

$$\begin{aligned} cost\,(X_k) &= \rho_{\omega_1^{(r)}(j)}\left(j, \mathcal{P}^{(r)}(j)\right) \\ &\leq \frac{\mathsf{OPT}}{n - |\mathcal{T}_{\mathcal{Q}^{(n-k)}}|} \leq \frac{\mathsf{OPT}}{n-k} \end{aligned}$$

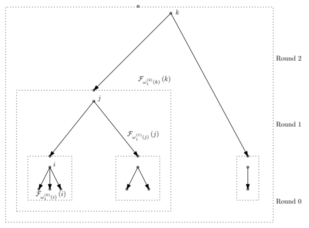
It follows that:

$$\begin{aligned} cost\,(\omega_1) &= cost\,(\mathcal{T}_{\mathcal{Q}}) \leq \sum_{k=1}^{n-1} cost\,(X_k) \\ &\leq \sum_{k=1}^{n-1} \frac{\mathsf{OPT}}{n-k} \leq \mathsf{OPT} \cdot H_{n-1} \end{aligned}$$

In the second phase of the algorithm, if a vertex receives *'Adjust-power'* request from $S \subseteq V$, then it selects $\max\{\omega(j)\,|j \in S\}$ as its new power. Once $\omega(i)$ is set to this $\omega(j)$ the for all $j' \in S$, $j' \in \Pi(s, G_\omega)$ is satisfied. Hence, each $\omega(i)$ is chosen at most once for assigning it to some other vertex i, this is maximum when number of such vertex j is at most $|V|/2$, and it follows that $cost\,(\omega_2) \leq 2 \cdot cost\,(\omega_1) \leq 2 \cdot \mathsf{OPT} \cdot H_{n-1}$. □

Theorem 2.3 (Time Complexity)**.** *Time complexity of the two phase distributed algorithm is $\mathcal{O}(n \cdot \Lambda(\mathcal{G}))$, where $\Lambda(\mathcal{G})$ denotes the diameter of graph $\mathcal{G}$.*

Proof. As in the proof of theorem 2.2, let $\mathcal{Q}$ be the sequence of pairs of vertex and power assignment, and $\mathcal{T}_{\mathcal{Q}}$ be the corresponding family of sets. The basic idea to obtain the bound on time complexity will be to view the execution of the algorithm (first phase) as a tree (see figure-2). The tree is formed as follows. In round r for each vertex i_j that merges components in $\mathcal{F}_{\omega_1^{(r)}(i_j)}(i_j)$, i_j is considered as root, and $X_{i_j,t} \in \mathcal{F}_{\omega_1^{(r)}(i_j)}(i_j)$ as leaves. This forms a tree because all components that are merges in a round are disjoint from the other, and denoted as level r nodes of the tree. It can be seen that the time complexity in the



k
Round 2
$\mathcal{F}_{\omega_1^{(2)}(k)}(k)$
j
Round 1
$\mathcal{F}_{\omega_1^{(1)}(j)}(j)$
i
$\mathcal{F}_{\omega_1^{(0)}(i)}(i)$
Round 0

Figure 2. Execution tree

first phase is proportional to the depth of this tree. Now at each level $1 \leq r \leq d$ of the tree, at most $\Lambda(\mathcal{G})$ rounds are spend in the step $\mathsf{Manage-Group}$ to communicate the current leader in the merged group. Finally, the worst case depth of the tree is bounded by $|\mathcal{T}_{\mathcal{Q}}| \leq n-1$. hence time complexity of the first phase of the distributed algorithm is $\mathcal{O}(n \cdot \Lambda(\mathcal{G}))$. Since this dominates that time complexity of the second phase, the bound on the algorithm follows. □

3 Improved approximation algorithm

In this section we present an improved approximation algorithm for the MECBS problem, which has performance ratio $\frac{3}{2}(\ln(n-1)+1)$. The algorithm is by extending the greedy algorithm of [3], and using the ideas from [8]. In every iteration i of the algorithm a set of strongly connected components are included in the larger arborescence (possibly one which has $s \in V$ as root). The induced directed graph (seen as a set of directed edges and the vertex set V) in iteration i is denoted as H_i. Strongly connected components of H_i, that are not reachable from s, and have no incoming edge are called *unhit components* of H_i. The algorithm stops when there are no unhit component in H_i for some i. While, if there is an unhit component, the algorithm determines a power assignment to construct a structure (seen as a set of edges induced by the power assignment) called a *spider* of minimum cost effectiveness. Where, cost effectiveness of a spider is defined as ratio of number of unhit components it merges into one larger arborescence divided by its cost. It can be seen that when H_i for some i has no more unhit components, then it implies that there is a directed path from s to every vertex $v \in V \setminus \{s\}$.

Algorithm of [3] allows spiders that hits one unhit component, restricting every step of the algorithm to larger spiders (the theme is same as the modification suggested in [8] as compared to [9]), provides an algorithm with better approximation factor.

Definition 3.1. *A* spider *S is an arborescence defined over vertex set $X \subseteq V$ with one vertex $h \in X$ called* head*, and a set of directed paths (called* legs*), to set of vertices called* feet *of the spider. Cost of a spider $cost(S)$ is the weight of head $\omega(h) = \max\{c(h,u) : u \in X\}$ plus the sum of the cost of remaining edges of S (see figure 3(a)).*

Recall, a strongly connected components of H_i, that are not reachable from s, and have no incoming edge are called *unhit components* of H_i. We will pick a vertex in a unhit components and call it a *representative*.

Definition 3.2. *A $l+$ spider is one that hits at least l unhit components among its feet, i.e. it has at least l representatives among its feet (see figure 3(c)). We will denote a $l+$ spider by S_{l+}*

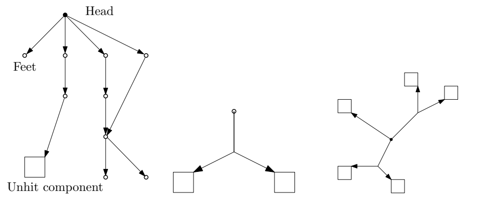
A 2+-spider is one having at least two representative among its feet(see figure 3(b)). In [3], it was shown that an optimal solution T^* of and MECBS instance, can be decomposed into a vertex disjoint 2+-spider. We will show that a decomposition of an optimal solution T^* of and MECBS instance with $l+$ spider with $l \geq 3$ is also possible.

Definition 3.3. *The shrink factor $\delta(S)$ of a spider S with head h is the number of representatives among its feet if h is reachable from the root s, or if h is not reachable from any of its feet, or the number of representatives among its feet minus one, otherwise.*

What follows is the greedy algorithm:

```
1: procedure GREEDY − SPIDER $(G, c, s)$
2:     $H = \emptyset$
3:     while $H$ has one unhit component do
4:         $(*)$ Find a spider $S_{3+}$ which minimizes the ratio
           $cost(S_{3+})/\delta(S_{3+})$ with respect to $H$.
5:         $H \leftarrow H \cup S_{3+}$
6:     end while
7: end procedure
```

Algorithm $\mathsf{GREEDY-SPIDER}$, chooses a spider $S_{l+} : l \geq 3$ in every step that has minimum ratio of cost to the number of unhit components that it hits. We will show how to implement step 4 (marked as $(*)$) of the algorithm $\mathsf{GREEDY-SPIDER}$ in lemma 3.3, but before that we first



(a) A spider. (b) A 2+-spider. (c) A 3+-spider.

Figure 3. Spider gtaphs

show that a good $l+$ spider for $l \geq 3$ exists by lower bounding the OPT.

Lemma 3.1. *Let H_i be the induced directed graph (seen as a set of directed edges and the vertex set V) in iteration i with $u(H_i)$ number of unhit components obtained from H_i, there exists a spider $S_{l+} : l \geq 3$ such that:*

$$\frac{cost(S)}{\delta(S)} \leq \frac{3 \cdot \mathsf{OPT}}{2 \cdot u(H_i)}$$

Proof. Let T^* be the optimal solution of cost OPT. Thus T^* is an optimum arborescence from root s reaching every vertex in $V \setminus \{s\}$ using a directed path. Let H_i be the induced directed graph (seen as a set of directed edges and the vertex set V) in iteration i with $u(H_i)$ number of unhit components obtained from H_i. Let R be the set of representatives from unhit components (i.e. one vertex from each, and $|R| = u(H_i)$). The depth of a node is the distance from root s using shortest directed path. Choose a vertex v in T^* of maximum depth (by postorder traversal) such that sub-arborescence rooted at v has at least three representatives from unhit components. Since no proper decendents of v has three representatives from unhit components it defines a S_{3+} spider. Remove v and its decendents from T^* and repeat. Let v be the last such vertex removed from T^* along with its its decendents, after which there are at most two representatives from unhit components remained. Let v' be the vertex (possibly s) which has path to v and both of these remaining vertices. Then we form the last spider connecting v and remaining vertices to v'.

Let $\{S_i\}_{i=1}^{q}$ be the set of S_{3+} spiders thus obtained in sequence. We have $\sum_{i=1}^{q} cost(S_i) \leq \mathsf{OPT}$. With $r(S_i)$ denoting number of representatives in S_i, we also have $\sum_{i=1}^{q} r(S_i) = u(H_i)$. Now observe that

$$2/3 \cdot r(S_i) \leq r(S_i) - 1 \leq \delta(S_i), \text{ for all } 1 \leq i \leq q-1.$$

This holds as $r(S_i) \geq 3$, it follows that $r(S_i) - 1 \geq 2/3 \cdot r(S_i)$. On the other hand, by the definition of $\delta(S_i)$, we have $\delta(S_i) \geq r(S_i) - 1$. Thus, $3/2 \cdot \sum_{i=1}^{q} \delta(S_i) \geq \sum_{i=1}^{q} r(S_i) = u(H_i)$. Hence, for the S_{3+} spider S_j with highest ratio satisfies

$$\frac{cost(S_j)}{3/2 \cdot \delta(S_j)} \leq \frac{\mathsf{OPT}}{u(H_i)}$$

□

In following we show that algorithm gives an improved approximation algorithm for the MECBS problem, which has performance ratio $\frac{3}{2}(\ln(n-1)+1)$. We use following lemma from [3]:

Lemma 3.2 **([3]).** *For a spider S, and induced directed graph H, $u(H \cup S) \leq u(H) - \delta(S)$, where $u(H)$ is the number of unhit components obtained from H, and $\delta(S)$ is the shrink factor of spider S.*

Theorem 3.1. *Algorithm $\mathsf{GREEDY-SPIDER}$ is a $1.5(\ln(n-1)+1)$–approximation for MECBS problem with arbitrary asymmetric power requirement.*

Proof. Let, $\phi_i = u(H_i)$ be the number of unhit components of H_i (thus ϕ_0 is $n-1$ for instance), and let S_i be the spider selected in this iteration in step $(*)$ such that $cost(S_i) = w_i$, and $\delta(S) = \delta_i$. Using lemma 3.1, we have $w_i/\delta_i \leq (3 \cdot \mathsf{OPT})/(2 \cdot \phi_i)$. Using lemma 3.2, we have that $\phi_{i+1} \leq \phi_i - \delta_i$, for all $0 \leq i \leq m$, where m is the number of iterations algorithm runs. It follows that

$$\phi_{i+1} \leq \phi_i - \delta_i \leq \phi_i \left(1 - \frac{2 \cdot w_i}{3 \cdot \mathsf{OPT}}\right) \qquad (1)$$

using recurrence relation of equation 1 we get,

$$\phi_{m-1} \leq \phi_0 \prod_{i=0}^{m-2} \left(1 - \frac{2 \cdot w_i}{3 \cdot \mathsf{OPT}}\right) \qquad (2)$$

Using $\ln(1+x) \leq x$, and taking natural logarithm on both sides of equation 2, we have,

$$\ln\left(\frac{\phi_0}{\phi_{m-1}}\right) \geq \frac{2 \cdot \sum\limits_{i=0}^{m-2} w_i}{3 \cdot \mathsf{OPT}}$$

Now observe that $\phi_0 = n-1$ and $\phi_{m-1} \geq 1$, hence $\sum\limits_{i=0}^{m-2} w_i \leq 1.5 \cdot \mathsf{OPT} \cdot \ln(n-1)$. We also have $w_{m-1} \leq 1.5 \cdot \mathsf{OPT}$ by lemma 3.1. Noting that cost of the solution produced by the algorithm is $\sum\limits_{i=0}^{m-1} w_i$ we have $\sum\limits_{i=0}^{m-1} w_i \leq 1.5 \cdot \mathsf{OPT} \cdot (\ln(n-1)+1)$. □

Finally we present how to implement step $(*)$ of the algorithm $\mathsf{GREEDY-SPIDER}$ in lemma 3.3. Due to limitation of space we present a sketch of the procedure here.

Lemma 3.3. *There exists a polynomial time procedure to compute a $S_{l+}, l \geq 3$ of minimum ratio.*

Proof. Let R be the set of representatives from unhit components of H. For all possible head h, and all possible $\omega \in \mathcal{C}_h$ we will do following. Let $\mathcal{F}_\omega(h)$ be children of h with ω as chosen power. For each representative r_i compute shortest path P_i, starting at $j \in \mathcal{F}_\omega(h)$ and ending at r_i (some j for which P_i is shortest). Arrange this paths in nondecreasing order. Let R_1 be the set of representatives that can reach h, and R_2 which can not, otherwise set $R_1 = R, R_2 = \emptyset$. For a choice of $l \geq 3$, and split $j_1 + j_2 = l$, use the algorithm for finding a minimum weight 3+ branch-spider with input (R_1, h, j_1), and (R_2, h, j_2) as presented in [8], using h–r_i path P_i obtained earlier in the calculation of the weight of spider to obtain S_{l+}. These two spider can be merged at h to get another S_{3+} if required. We will try all possible l, which is bounded by $|R|$, and possible integers $j_1 + j_2 = l$ to obtain best spider with a given power value, iterating over all possible choice of h. The minimality in ratio follows by argument similar to one presented in [8, lemma 3.2]. □

Using techniques of [3, Theorem 2], we have following result:

Theorem 3.2. *There is a* $1.5 \ln(n-1) + 2.5$ *– approximation algorithm for Strong connectivity with asymmetric power requirements.*

4 Concluding remarks

In this work we have presented a $2H_{n-1}$ factor distributed approximation algorithm for MECBS problem [6] with symmetric edge cost. Time complexity of the two phase distributed algorithm is $\mathcal{O}(n \cdot \Lambda(\mathcal{G}))$, where $\Lambda(\mathcal{G})$ denotes the diameter of the communication graph $\mathcal{G}$. To our knowledge, there is no previously known distributed algorithm for this problem. In order to obtain a distributed algorithm, we consider the approximation algorithms presented in [2], which is very similar in its spirit to that of well known greedy approximation algorithm for SET-COVER problem (cf. [13, Chapter 2, Algorithm 2.2]).

We believe that time complexity of this algorithm can be reduced, possibly using some efficient distributed datastructures. We would like to explore the possibility if sparse spanners ([12, 1]) specifically constructed for MECBS problem provide logarithmic approximation factor, as that could be a new avenue to design faster distributed approximation algorithm for this problem.

References

[1] B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[2] V. Bilò and G. Melideo. An improved approximation algorithm for the minimum energy consumption broadcast subgraph. In M. Danelutto, M. Vanneschi, and D. Laforenza, editors, *Euro-Par*, volume 3149 of *Lecture Notes in Computer Science*, pages 949–956. Springer, 2004.

[3] G. Calinescu, S. Kapoor, A. Olshevsky, and A. Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In G. D. Battista and U. Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 114–126. Springer, 2003.

[4] I. Caragiannis, M. Flammini, and L. Moscardelli. An exponential improvement on the mst heuristic for minimum energy broadcasting in ad hoc wireless networks. In *34th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[5] I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. A logarithmic approximation algorithm for the minimum energy consumption broadcast subgraph problem. *Inf. Process. Lett.*, 86(3):149–154, 2003.

[6] A. E. F. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. On the complexity of computing minimum energy consumption broadcast subgraphs. In *STACS '01: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, pages 121–131, London, UK, 2001. Springer-Verlag.

[7] M. Flammini, A. Navarra, R. Klasing, and S. Pérennes. Improved approximation results for the minimum energy broadcasting problem. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 85–91, New York, NY, USA, 2004. ACM Press.

[8] S. Guha and S. Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Inf. Comput.*, 150(1):57–74, 1999.

[9] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[10] D. Li, X. Jia, and H. Liu. Energy efficient broadcast routing in static ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 3(2):144–151, 2004.

[11] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[12] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[13] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[14] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Networks*, 8(6):607–617, 2002.