Center-oriented Algorithms for the Minimum Energy Broad- and Multicast Problem in Wireless Ad hoc Networks

Joanna Bauer [a,1,*], Kemal Altinkemer [b] Dag Haugland [a]

[a] *Department of Informatics, University of Bergen, PB. 7800, 5020 Bergen, Norway, Tel: +47 55584094, Fax: +47 55584199*

[b] *Krannert School of Management, Purdue University, 403 W. State Street, West Lafayette, IN 47907, USA*

Abstract

Quickly finding low-energy multicast routings is vital for a wireless system's energy efficiency. Therefore, key aspects for heuristics for the minimum energy multicast problem (MEMP) are time complexity (measured in the numbers $|V|$ and $|A|$ of networking devices and their possible power assignments, respectively) and deviation from optimum. Following a center-oriented approach, we develop the STSuS and STESuS algorithms (time complexity $O(|V|^2)$ and $O(|V|^2 \log |V|)$, respectively), and analyze their performance in numerical simulations. They deviate from optimum by only $\approx 11\%$ and $\approx 7.5\%$, respectively, and thereby outperform the well-known MIP ($O(|V|^2)$, $\approx 22\%$ deviation) and many other algorithms significantly.

Key words: Wireless Ad-hoc Network, Minimum Energy Multicast Problem, Approximation Algorithm

1 Introduction

Wireless ad hoc (i.e. "off-the-cuff") mesh networks enable wireless devices to communicate without access to a backbone structure as for example the Internet or base stations. This enables a wide range of powerful applications, from instant conferencing between notebook PC users to emergency and military services as for example disaster rescue work, disease outbreak detection and monitoring, forest fire detection, weather forecasting, control of smart home appliances, ubiquitous computing, and homeland security (Perkins, 2000). Warneke et al. (2001) introduced the idea of *smart dust*, devices equipped with a sensor and wireless communication, which would, following Moore's Law (Moore, 1998) eventually become as small and inexpensive that they can be scattered over an area to monitor for example temperature or movements. In such a scenario, energy is a scarce resource, as such tiny devices can only carry a tiny energy source, and not be recharged. The Minimum Energy Multicast Problem (MEMP) addressed in this paper is of high importance to any application seeking to set up an energy efficient wireless network without base infrastructure. Three specific applications and the state of the technical prerequisites are explained in detail in Section 7.

In many applications of wireless systems, a minimum energy multicast routing has to be computed repeatedly and quickly. To establish a multicast routing, a transmission power must be assigned to each network unit. The coverage area

* Corresponding author
Email addresses: Joanna.Bauer@ii.uib.no (Joanna Bauer), kemal@purdue.edu (Kemal Altinkemer), Dag.Haugland@ii.uib.no (Dag Haugland).

[1] Supported by The Research Council of Norway under contract 160233/V30

of a unit is given by the power assigned to it, and therefore the power needed to cover a set of receiving units is not the sum, but the maximum of the power needed to cover any of them. The power needed at one unit to cover another unit grows at least quadratically with the distance. Consequently, computing a minimum energy multicast routing is NP-hard (Cagalj et al., 2002). Therefore, the energy efficiency of applications depend on efficient routing algorithms. The efficiency of an algorithm is in general evaluated by the solution quality, and, as better solutions can be obtained by more computations, by its time complexity. Both aspects of the minimum energy multicast problem have therefore attracted intensive research. An overview can be found in the survey of Guo and Yang (2007).

A common approach is to represent the network as a graph $G = (V, A)$ and identify a routing arborescence, which is a directed tree with all arcs oriented away from the root. Thus, a directed path (the routing) from the root to any destination node is defined. The power consumption of a node in the arborescence is determined by the length of the most expensive outgoing arc.

Most proposed heuristics follow the scheme of constructing an arborescence by starting with the source and in every step greedily adding at least one node according to a certain decision rule. The most frequently cited such algorithm is the Multicast Incremental Power (MIP) algorithm presented by Wieselthier et al. (2002). MIP is called BIP for broadcast. The average performance ratio of MIP is approximately 1.22. MIP has $O(|V|^2)$ time complexity (Bauer et al., 2009). To the best of our knowledge, MIP is the algorithm with the best average performance ratio within this time complexity.

An inherent weakness of MIP is that its greedy nature makes it biased towards

overly high arborescences. In this paper, we strive to overcome this weakness by a center-oriented algorithms. We first study a simple algorithm which we call Star. It assigns to the source a sufficiently high power to cover all destinations, the solution arborescence thus being a star. Our experiments show that if the source is located centrally in the network, the star arborescence is more energy efficient than the solution produced by MIP. However, when the source is located more peripherally, the star gives a weak solution. These observations call for an approach referred to as *center-oriented algorithms*. The idea is to first choose a node which in some sense is positioned centrally in the network. A route between the source and this center node is established, usually by constructing the shortest path from the source to the center node. Then, a routing from the center node to all remaining destinations is determined. This algorithm, which we call ST, outputs arborescences with about half the power consumption of the Star arborescences. We improve the ST arborescence by applying the successive shrink (SuS) algorithm, which was presented by Yuan et al. (2008), to the center node. SuS subsequently disconnects the most power demanding child from the center node and assigns a new parent to it. This yields algorithm STSuS. It has $O(|V|^2)$ time complexity, and a better average performance ratio than MIP. We improve the STSuS arborescence by introducing an enhanced SuS move: After a new parent for the most power demanding child of the center is chosen, this new parent is allowed to increase its power further if this results in a total power saving. This yields algorithm STESuS of time complexity $O(|V|^2 \log |V|)$.

Kang and Poovendran (2004) present a first approach to center-oriented broadcast algorithms (COBRA). They propose several algorithms, of which a version named COBRA-EWMA performs best. The main difference between center-

oriented algorithms and greedy tree-construction algorithms like MIP is that the latter construct arborescences neglecting the location of the destinations. Thus, they can run off in a wrong direction. MIP's worst-case-instance by Wan et al. (2004) is a drastic example of this behavior. Our algorithms, on the contrary, start off with a reasonably good solution, which then is refined.

The presentation begins with introducing notation in Section 2. In Section 3, we develop the simple center-oriented algorithm ST and review (Kang and Poovendran, 2004). In Sects. 4 and 5, we subsequently refine ST to STSuS and STESuS. All sections discuss the time complexity of the proposed algorithms. Section 6 presents the results of our numerical experiments and the resulting observations, and Section 7 shows possible applications. The paper is ended with concluding remarks.

2 Preliminaries

A problem instance is given by a directed graph $G = (V, A)$, where the nodes represent the networking units, a source $s \in V$, a set of destinations $D \subseteq V$, and power requirements $c \in \mathbb{R}^A$. We assume G to be complete.

A solution to an instance can be given by an s-arborescence $T = (V_T, A_T)$ with node set $V_T \subseteq V$ and arc set $A_T \subseteq A$. An s-arborescence is a directed tree where $s \in V_T$, and all arcs are oriented away from s. In T, every $v \in V_T \setminus \{s\}$ has a parent $\pi_v(T)$, and every $v \in V_T$ has a (possibly empty) set $\Gamma_v(T)$ of children and a (possibly empty) set $\Delta_v(T)$ of descendants. A node $v \in V_T$ is called *active*, if $v \in D$ or $D \cap \Delta_v(T) \neq \emptyset$. For any set $S \subseteq V_T$, we denote by

$\widetilde{S}(T)$ the subset of S containing exactly the active nodes in S, and we define

$$p_v(S) = \begin{cases} 0 & \text{if } \widetilde{S}(T) = \emptyset \\ \max_{w \in \widetilde{S}(T)} \{c_{vw}\} & \text{otherwise} \end{cases}$$

We use $p_v(T)$ as a short hand notation for $p_v(\Gamma_v(T))$, that is, $p_v(T)$ is the power necessary for v to reach all its active children in T. Consequently, we define the cost of T as $p_T = \sum_{v \in V_T} p_v(T)$. The minimum energy multicast problem can then be formulated as

[**MEMP**] Find an s-arborescence T such that $D \subseteq V_T$ and p_T is minimized.

An algorithm *links* node w to node v by adding (v, w) and w to the current s-arborescence T. If w is active, this may lead to an increase in $p_v(T)$. If v has been inactive, additional power might also be required at upstream nodes of v. The total additional power needed at all nodes in T if v becomes active is called the *induced power* $I_v(T)$ of v in T. If v is active, we have $I_v(T) = 0$. We furthermore call the node $\gamma_v(T) \in \Gamma_v(T)$ that determines the power assignment at v (that is, $p_v(T) = c_{v\gamma_v(T)}$) the *critical child* of v. Ties are broken by a consistent tie-braking rule, such that every node has at most one critical child. We let $h_v(T)$ be the number of nodes of the s-v-path in T.

We denote by $\Phi_v = (V_{\Phi_v}, A_{\Phi_v})$ the shortest s-v-path in G for every node $v \in V$, by $p_{\Phi_v} = \sum_{(i,j) \in A_{\Phi_v}} c_{ij}$ the power consumption of Φ_v, and by $\Phi = (V, \bigcup_{v \in V} A_{\Phi_v})$ the shortest-path-arborescence of G with root s.

Table B.1 on page 34 summarizes the introduced terms and abbreviations, and Table C.1 on page 36 summarizes the introduced notation.

2.1 Test Instances

In order to evaluate the presented algorithms, we apply them to generated test instances. The set "$|V|/|D|$" refers to 100 instances with $|V|$ nodes and $|D|$ destinations, generated by distributing the nodes on a square using the standard C++ pseudo-random number generator `rand()`. A realistic cost function for MEMP (Wieselthier et al., 2000) is to set $c_{vw} = \kappa d_{vw}^{\alpha}$, where d_{vw} is the distance between v and w, κ is a positive constant and α is an environment-dependent parameter in $[2, 4]$. We use this cost function for all test instances. We set κ to 1 and α to 2, as for these values the lower bound of $|V| - 2 - o(1)$ on the approximation ratio of MIP was derived by Wan et al. (2004). However, the algorithms presented in this paper can be applied to instances with arbitrary cost functions.

For any specific instance $\mathcal{I}$, we let $p_{\text{opt}}(\mathcal{I})$ and $p_{\mathcal{A}}(\mathcal{I})$ denote the power consumption of the optimal solution to $\mathcal{I}$ and the solution to $\mathcal{I}$ determined by algorithm $\mathcal{A}$, respectively.

3 The Shortest Trunk Algorithm

3.1 Motivation

The simplest algorithm for solving MEMP is the Star algorithm. It links every destination to the source, the resulting s-arborescence being a star. It has $O(|D|)$ time complexity. Clearly, the performance of the Star can be quite bad, as for example $p_{\text{Star}}(\mathcal{S}) = 2.55 p_{\text{opt}}(\mathcal{S})$ for the sample instance $\mathcal{S}$ taken

out of the test instance set |25|/|5|. In $\mathcal{S}$, node 0 is the source, and nodes $1, \ldots, 5$ are the destinations (Fig. 1).

However, it is likewise clear that the bad performance of the Star applied to $\mathcal{S}$ is due to the position of s, at the boundary of the deploy region of the instance. If the position of s is favorable to the Star, then the results are surprisingly good: We generated test sets |25|/|5|, |25|/|12|, and |25|/|24|, where we let the source be positioned at the center of the square in which the instance is deployed. For every instance, we determined the optimal solution using integer programming (Bauer et al., 2008), and the multicast arborescences generated by MIP and Star. In Table 1, the averaged power consumptions are given in columns 3-5. The table shows that the simple Star outperforms MIP for the instance sets |25|/|12| and |25|/|24|. This makes the Star a promising base for developing a MEMP algorithm.

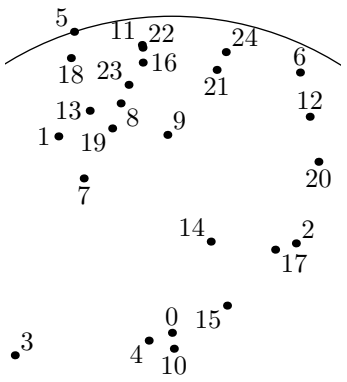


Fig. 1. Star($\mathcal{S}$)

Table 1. Performance of MIP and Star on instances with source in the center

$\|V\|$	$\|D\|$	opt	MIP	Star
25	5	19.81	25.61	29.13
	12	27.68	38.55	34.99
	24	32.53	48.40	38.61

3.2 The Shortest Trunk (ST) Algorithm

A straightforward idea to improve the Star to handle general locations of the source is to determine a node z which is positioned "centrally" in the instance,

connect it to s, and make z the center of a star covering the remaining nodes.

We define as an (s, z)-*trunk* any s-arborescence where all edges are incident to a node on the path from s to z. For G, s, z and c, we define as a *shortest* (s, z)-*trunk* any (s, z)-trunk where the path from s to z equals a shortest s-z-path in G with cost function c.

A simple MEMP-algorithm is given by finding the s-arborescence T that minimizes p_T subject to the constraints:

- T covers D
- for some $z \in V$, T consists of a shortest (s, z)-trunk and a star centered at z.

For every $z \in V$, we let D_z be the set of destinations covered by the shortest (s, z)-trunk defined by Φ: $D_z = \{d \in D | \exists (v, w) \in A_{\Phi_z} : c_{vw} \geq c_{vd}\}$. Then, the s-arborescence satisfying the above constraints has cost $p_{\Phi_z} + p_z(D \setminus D_z)$. This leads to the *Shortest-Trunk* (ST) algorithm.

ST$(G = (V, E), s \in V, D \subset V, c \in \mathbb{R}^E)$

1 construct Φ
2 find $z \in \text{argmin}\{p_{\Phi_z} + p_z(D \setminus D_z) : z \in V\}$
3 $T = (V_T, A_T) \leftarrow (\{s\}, \emptyset)$
4 let $s = v_0, \ldots, v_k = z$ be the nodes on Φ_z
5 **for** $i \leftarrow 0 \ldots k - 1$
6 $A_T \leftarrow A_T \cup \{(v_i, v_{i+1})\}$
7 $V_T \leftarrow V_T \cup \{v_{i+1}\}$
8 **for all** $d \in D \setminus V_T$
9 **if** $c_{v_i v_{i+1}} \geq c_{v_i d}$

10 $A_T \leftarrow A_T \cup \{(v_i, d)\}$

11 $V_T \leftarrow V_T \cup \{d\}$

12 **for all** $d \in D \setminus V_T$

13 $A_T \leftarrow A_T \cup \{(z, d)\}$

14 $V_T \leftarrow V_T \cup \{d\}$

15 **return** T

In Step 2, we choose z such that $p_{\Phi_z} + p_z(D \setminus D_z)$ is minimized. In Steps 4-7, the node z is connected to s via the shortest path between s and z. While the path is constructed, every node on the path adds to its children all destinations it covers that do not yet have a parent (Steps 8-11). In Steps 12-14, node z finally becomes the parent of all destinations that do not yet have a parent.

The power assigned to the nodes in $\mathcal{S}$ by ST is visualized in Fig. 2. ST first determines node 7 as center node. The shortest path from node 0 to node 7 is (0–15–14–9–19–7). As the shortest (0,7)-trunk covers destinations 1, 2, and 4, node 7 only needs to cover destinations 3 and 5. Of those, node 3 is furthest away from 7 and thus determines the power assignment at 7. In Fig. 2, the nodes 6, 12, and 20 are therefore uncovered. ST reduces the power consumption of the Star by about one third to $p_{\mathrm{ST}}(\mathcal{S}) = 1.61 p_{\mathrm{opt}}(\mathcal{S})$.

3.3 Time complexity of the Shortest Trunk Algorithm

Theorem 1 *The ST algorithm has* $O(|V||D| + |A| + |V| \log |V|)$ *time complexity.*

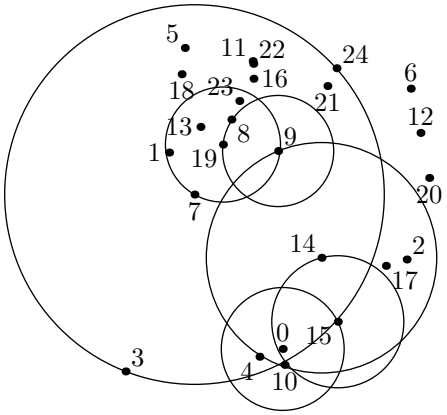


Fig. 2. ST($\mathcal{S}$)

PROOF. See Appendix A.1.

In the remainder of the paper, we refer by ST to a slightly altered version of the algorithm, which serves as a base for the two subsequently presented algorithms: In Steps 8 and 12, we substitute $D \setminus V_T$ by $V \setminus V_T$, such that ST returns an s-arborescence T with $V_T = V$. We also let ST not only return T, but (T, z). This version of ST has $O(|V|^2)$ time complexity.

3.4 Previous Work: The COBRA Scheme and COBRA-EWMA

A preliminary approach to solve the Minimum Energy Broadcast Problem (MEBP) by center-oriented algorithms was made by Kang and Poovendran (2004). The authors argued that intuitively, the center of a network instance is the best place to take advantage of MEBP's node-oriented objective function, and proposed the following general scheme for center-oriented broadcast routing algorithms.

COBRA($G = (V, A), s \in V, c \in \mathbb{R}^A$)

1 choose center node $z \in V$

2 connect z to s

3 apply a broadcast routing algorithm at z

For Step 1, Kang and Poovendran (2004) suggested to choose for z the node closest to the geometrical center of the region over which the network is distributed. For Step 2, they suggested to use the shortest path in G with arc costs c. However, in Step 3, they disregard that the nodes on the path and possibly some other nodes are already covered, and let all nodes be covered by a broadcast routing algorithm started at z. For Step 3, they experimented with several possibilities, among which the broadcast algorithm EWMA (Cagalj et al., 2002) performs best. EWMA executes Prim's MST algorithm followed by a local search. Kang and Poovendran (2004) named the resulting algorithm COBRA-EWMA.

We adapt the COBRA scheme to multicast as follows. We split Step 3 into the two steps that usually are executed by a routing algorithm, namely tree construction and local search.

COMA($G = (V, A), s \in V, D \subseteq V, c \in \mathbb{R}^A$)

1 choose $z \in V$

2 connect z to s

3 cover all nodes in D

4 improve the solution by local search

4 The Shortest Trunk Successive Shrink Algorithm

4.1 Previous Work: The IMBM Algorithm

Li and Nikolaides (2001) presented the algorithm IMBM, which is to the best of our knowledge the first algorithm based on the idea to first construct a star arborescence and then improve it by local search. However, IMBM is not a center-oriented algorithm, as the idea of a center node is not integrated.

In the local search of IMBM, it is checked whether for any critical child $\gamma_v(T)$, there is a node f satisfying $c_{vf} + c_{f\gamma_v(T)} < c_{v\gamma_v(T)}$. Of all such possible moves (v, f) that actually lead to power improvement (they might not, as the node-oriented objective function is not taken into account), one is selected randomly and executed. If this leads to an increase in $p_f(T)$, it is then checked whether f covers any other critical children, and if so, they are linked to f.

Li and Nikolaides (2001) suggest to execute this local search until no further improvement is found, and execute IMBM $|V|$ times (which, because of the random move selection, leads to different solutions), and then finally select the best outcome. The time complexity is $O(|V|^4)$. Li and Nikolaides (2001) report that IMBM outperforms BIP for a majority of their test instances.

In the forthcoming algorithms, we consider moves (v, f) satisfying

$$(1) \qquad I_f(T) + c_{f\gamma_v(T)} - p_f(T) < c_{v\gamma_v(T)} - p_v(\Gamma_v(T) \setminus \{\gamma_v(T)\}) \ .$$

This criterion is more closely related to the objective function than the evaluation criterion of Li and Nikolaides (2001). We consider as v only the center node z, as the highest savings can be obtained at this node. This reduces the

time complexity by one order of magnitude. Furthermore, in every step we select the move yielding the highest power saving (steepest descent). We then need to execute the algorithm only once, which reduces the time complexity by another order of magnitude. Using a careful implementation, we can evaluate (1) within $O(|V|^2)$ time. In Section 4.2, we give this implementation, which was presented as part of the SuS heuristic by Yuan et al. (2008). In Section 4.3, we show how it can be combined with the ST algorithm, yielding the STSuS algorithm. In the subsequent discussion of the time complexity of STSuS, we also strengthen the result on the time complexity of the SuS algorithm given by Yuan et al. (2008).

4.2 The Successive Shrink (SuS) Procedure

SuS$(G = (V, A), D \subset V, c \in \mathbb{R}^A, T = (V_T, A_T), v \in V_T)$

1 $T' = (V_{T'}, A_{T'}) \leftarrow T$

2 sort $\Gamma_v(T) = \{g_1, \ldots, g_k\} : c_{vg_i} \geq c_{vg_{i+1}} \forall i = 1, \ldots, k-1$

3 **for** $i \leftarrow 1, \ldots, k$

4 $g \leftarrow g_i$

5 $F \leftarrow V \setminus \{v\} \setminus \Delta_g(T)$

6 **if** $F \neq \emptyset$

7 find $f \in \operatorname{argmin}\{I_{f'}(T) + c_{f'g} - p_{f'}(T) : f' \in F\}$

8 $A_T \leftarrow A_T \setminus \{(v, g)\} \cup \{(f, g)\}$

9 **if** $p_T < p_{T'}$

10 $T' \leftarrow T$

11 **return** T'

Given an s-arborescence T and a node v, the SuS procedure in Step 2 sorts $\Gamma_v(T)$ with respect to c, resulting in $\Gamma_v(T) = \{g_1, g_2, \ldots, g_k\}$, where $c_{vg_i} \geq c_{vg_{i+1}}$, and $k = |\Gamma_v(T)|$. Then, the SuS procedure considers reducing $\Gamma_v(T)$ by disconnecting g_1 from v and linking it to another node $f \notin \Delta_{g_1}(T)$. In general, this leads to a power increase at f by $c_{fg_1} - p_f(T)$ and possibly to a power increase at upstream nodes of f, if they were inactive. However, if there is a node f for which

$$I_f(T) + c_{fg_1} - p_f(T) < c_{vg_1} - p_v(\Gamma_v(T) \setminus \{g_1\}) \ , \tag{2}$$

then the total power consumption of T can be reduced by substituting the link $(v, g_1) \in A_T$ by (f, g_1). If there are several nodes that fulfill (2), then the node f minimizing $I_f(T) + c_{fg_1} - p_f(T)$ is chosen as new parent for g_1. Finding a new parent can then be repeated for $g_2, \ldots, g_k$ (Steps 3-8). For every g_i, all nodes except v and descendants of g_i are possible parents (Step 5).

The SuS procedure moves all children one by one, even if this does not yield an immediate power reduction, as power saving might occur in a later step. During execution, the SuS procedure keeps track of the lowest cost arborescence found so far (Steps 9-10), which is output at the end (Step 11).

Yuan et al. (2008) introduce the SuS heuristic (see below), which executes the SuS procedure (see above) for every node in V. The SuS heuristic is a local search heuristic for improving a given multicast s-arborescence. It copies a given s-arborescence T for every node $v \in V_T$, and then executes the SuS procedure on T and v. The best encountered s-arborescence is output at the end.

SuS$(G = (V, E), s \in V, D \subset V, c \in \mathbb{R}^E, T = (V_T, A_T))$

1 $T^* \leftarrow T$

2 **for all** $v \in V$

3 $T' \leftarrow$SuS(G, s, D, c, T, v)

4 **if** $p_{T'} < p_{T^*}$

5 $T^* \leftarrow T'$

6 **return** T^*

Theorem 2 *The SuS heuristic presented by Yuan et al. (2008) has* $O(|V|^2)$ *time complexity.*

PROOF. See Appendix A.2.

Corollary 3 *The SuS procedure has* $O(|V|^2)$ *time complexity.*

PROOF. See Appendix A.3.

4.3 The Shortest Trunk Successive Shrink Algorithm

The SuS procedure can be applied to the center node of the s-arborescence constructed by ST. This is done by the STSuS algorithm.

STSuS$(G = (V, A), s \in V, D \subset V, c \in \mathbb{R}^A)$

1 $(T, z) \leftarrow$ ST(G, s, D, c)

2 **return** SuS(G, s, D, c, T, z)

The power assigned to the nodes of instance $\mathcal{S}$ by STSuS is given in Fig. 3(a). After arriving at the final state of ST (recall Fig. 2), STSuS first disconnects node 3 from 7 and links it to node 4. The power assignment at node 7 is accordingly reduced to $c_{7,5}$. Then, node 5 is disconnected from 7. It is linked to node 23, as thus the power assignment at node 7 can be reduced to 0, and the power assignment at node 19 can be reduced to $c_{19,1}$. As the center node 7 is an inactive node in STSuS's solution, it was not the perfect choice as center node. However, STSuS managed to correct this at least partially by reducing the power at node 7 to 0.

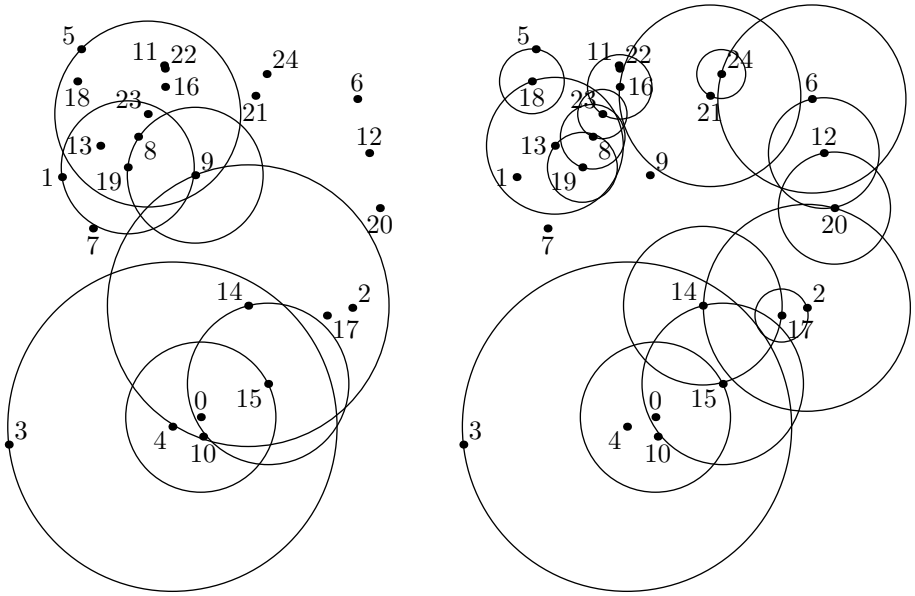
Figure 3(b) shows the power assigned to the nodes to $\mathcal{S}$ by MIP. Comparing the solutions, we see a distinct example of MIP building an arborescence into a wrong direction, as it does not take the positions of the destinations into account. This is one of the main reasons why STSuS on average outputs better solutions than MIP: The total power consumptions of STSuS and MIP are $p_{\mathrm{STSuS}}(\mathcal{S}) = 1.29 p_{\mathrm{opt}}(\mathcal{S})$ and $p_{\mathrm{MIP}}(\mathcal{S}) = 1.53 p_{\mathrm{opt}}(\mathcal{S})$.

Corollary 4 *The STSuS algorithm has $O(|V|^2)$ time complexity.*

PROOF. See Appendix A.4.

5 The Shortest Trunk Enhanced Successive Shrink Algorithm

The SuS procedure chooses the new parent h for g_i according to the current power levels of the nodes in T. When at a later point g_j $(j > i)$ is linked to f and $p_f(T)$ increases, this possibly yields $I_f(T) + c_{fg_1} - p_f(T) < p_h(T) - p_h(\Gamma_h(T) \setminus \{g_i\})$, which means that f has become a better parent for g_i than



(a) STSuS($\mathcal{S}$) (b) MIP($\mathcal{S}$)

Fig. 3. Power assignment determined by STSuS and MIP for $\mathcal{S}$

h. The STSuS algorithm can therefore be improved by checking whether p_T can be reduced by increasing the power assigned to f further.

To implement this idea, we use the enhanced sweep (ES) improvement procedure for MEMP arborescences, which was presented by Yuan et al. (2008). Given an s-arborescence T and a node f, ES considers increasing $p_f(T)$ in order to reduce the power assignment at other nodes and thus the total power assignment. Steps 5-8 comprise the sweep procedure at node f as introduced by Wieselthier et al. (2000), that is, node f becomes the parent of all nodes it covers (except nodes on the path s-f), which might yield power saving at other nodes. Steps 5-6 together with Steps 9-10 implement the "enhanced sweep", which incrementally raises $p_f(T)$ and checks whether doing so saves power at other nodes. The best encountered arborescence is returned.

ES$(G = (V, A), D \subset V, c \in \mathbb{R}^A, T = (V_T, A_T), f \in V_T)$

1 $\quad T' = (V_{T'}, A_{T'}) \leftarrow T$

2 $K \leftarrow V_{T'} \setminus \{f\} \setminus \Gamma_f(T') \setminus \{w : f \in \Delta_w(T')\}$

3 **if** $K \neq \emptyset$

4 sort $K = \{g_1, \ldots, g_m\} : c_{fg_i} \leq c_{fg_{i+1}}$

5 **for** $j \leftarrow 1, \ldots, m$

6 $A_{T'} \leftarrow A_{T'} \setminus \{(\pi_{g_j}(T), g_j)\} \cup \{(f, g_j)\}$

7 **if** $c_{fg_j} \leq p_f(T)$

8 $T \leftarrow T'$

9 **else if** $p_{T'} < p_T$

10 $T \leftarrow T'$

11 **return** T

We allow f to adopt every node v

- of which f is not a descendant and
- which is not a child of z and
- for which the number of hops between s and v is not smaller than the number of hops between s and f.

Accordingly, we alter Step 2 of the ES procedure to

$$K \leftarrow V'_T \setminus \{f\} \setminus \Gamma_f(T') \setminus \{w : f \in \Delta_w(T')\} \setminus \Gamma_z(T') \setminus \{v : h_v(T) < h_f(T)\} \ .$$

We denote the altered ES procedure by ES2. This leads to the STESuS algorithm. STESuS is identical to STSuS except for executing ES2 after Step 8 in the SuS procedure (see page 14).

STESuS$(G = (V, E), s \in V, D \subset V, c \in \mathbb{R}^E)$

1 $(T, z) = ((V_T, A_T), z) \leftarrow \mathrm{ST}(G, s, D, c)$

2 $T' \leftarrow T$

3 sort $\Gamma_z(T) = \{g_1, \ldots, g_k\} : c_{zg_i} \geq c_{zg_{i+1}} \forall i = 1, \ldots, k-1$

4 **for** $i \leftarrow 1, \ldots, k$

5 $g \leftarrow g_i$

6 $F \leftarrow V_T \setminus \{z\} \setminus \Delta_g(T)$

7 **if** $F \neq \emptyset$

8 find $f \in \text{argmin}\{I_{f'}(T) + c_{f'g} - p_{f'}(T) : f' \in F\}$

9 $A_T \leftarrow A_T \setminus \{(z, g)\} \cup \{(f, g)\}$

10 $T \leftarrow \text{ES2}(G, D, T, f)$

11 **if** $p_T \leq p_{T'}$

12 $T' \leftarrow T$

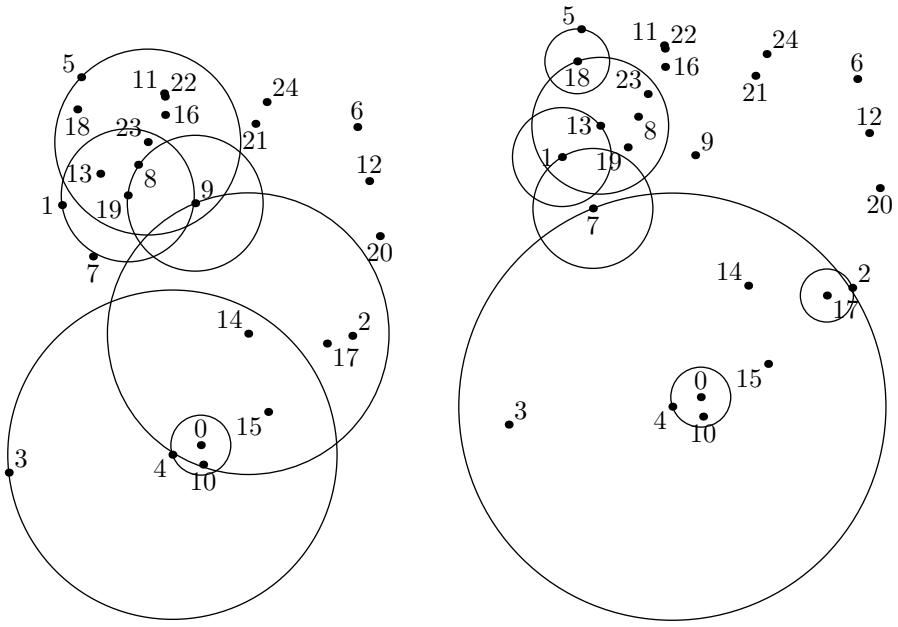
13 **return** T'

The output of STESuS applied to instance $\mathcal{S}$ is given in Fig. 4(a). For comparison, Fig. 4(b) shows the optimal solution to $\mathcal{I}$. The total power consumption of the STESuS solution is $p_{\text{STESuS}}(\mathcal{S}) = 1.12 p_{\text{opt}}(\mathcal{S})$.

Remark The local search of EWMA considers only a subset of the moves considered by ES2.

Theorem 5 *The STESuS Algorithm has $O(|V|^2 \log |V|)$ time complexity.*

PROOF. See Appendix A.5.



(a) STESuS($\mathcal{S}$) (b) opt($\mathcal{S}$)

Fig. 4. Power assignment determined by STESuS and the optimal solution for $\mathcal{S}$

6 Numerical Experiments

In this section, we study the computational performance of the algorithms presented in this paper. The performance of MIP and, when available, the optimal solution are used as reference points. As explained in Section 2.1, we price a direct transmission from node v to node w at their squared distance, that is, $c_{vw} = d_{vw}^2$. Since only the relative costs compared with each other are relevant for our studies, we omit the squared length units from the tables.

6.1 Comparing the star-based Algorithms

As described in Section 2.1, we generated sets $|V|/|D|$ of network instances with $|V| \in \{25, 50, 100\}$, and $|D| \in \{5, \left\lceil \frac{|V|-1}{2} \right\rceil, |V|-1\}$. For every instance, we generated multicast arborescences using MIP, Star, ST, STSuS and STESuS. In Table 2, the averaged power consumptions are given in columns 4-8, re-

spectively. For instances with $|V| = 25$ or $|D| = 5$, we can obtain the optimal solution by integer programming as described by Yuan et al. (2008). For those instance sets, the averaged optimal power consumption is given in column 3, and columns 9-13 give the averaged performance ratios $\rho_{\mathcal{A}} = \sum_{i=1}^{100} \frac{p_{\mathcal{A}}(\mathcal{I}_i)}{p_{\mathrm{MIP}}(\mathcal{I}_i)}$ for $\mathcal{A} \in \{\text{opt, Star, ST, STSuS, STESuS}\}$.

Table 2

Performance of MIP, Star, ST, STSuS and STESuS

$\|V\|$	$\|D\|$	opt	MIP	Star	ST	STSuS	STESuS	ρ_{MIP}	ρ_{Star}	ρ_{ST}	ρ_{STSuS}	ρ_{STESuS}
25	5	24.24	29.67	63.81	31.53	25.98	25.39	1.22	2.62	1.30	1.07	1.04
	12	32.60	39.94	76.36	42.82	35.65	34.31	1.23	2.31	1.32	1.10	1.05
	24	39.46	49.44	85.06	49.33	42.63	41.27	1.26	2.12	1.26	1.08	1.05
50	5	19.42	24.19	66.33	29.28	21.97	21.22	1.25	3.38	1.51	1.13	1.09
	25		40.69	88.86	46.34	37.30	35.98					
	49		47.42	94.77	49.95	41.86	40.99					
100	5	14.50	20.10	63.62	26.61	17.18	16.66	1.40	4.38	1.83	1.18	1.14
	50		40.05	91.04	46.25	37.00	35.68					
	99		45.84	96.42	49.98	40.29	39.54					

Table 2 shows that ST about halves the average power consumption of the Star algorithm. Compared to ST, STSuS reduces the power consumption by on average 20%. Finally, STESuS reduces the power consumption by about 3% compared to STSuS. In summary, every improvement of the algorithm yields a significant reduction in power consumption.

As optimal solutions are not available for large networks, it is general practice to report MEMP algorithm performance in comparison to MIP, which we do in Table 3. Columns 3-7 give the average $\mu_{\mathcal{A}} = \sum_{i=1}^{100} \frac{p_{\mathcal{A}}(\mathcal{I}_i)}{p_{\text{MIP}}(\mathcal{I}_i)}$, where $\mathcal{A} \in \{\text{opt, Star, ST, STSuS, STESuS}\}$. Column 8 compares MIP and STSuS by the number of instances in which one algorithm determined an arborescence with lower power consumption than the other algorithm, and column 9 shows the same ratio for MIP and STESuS.

Table 3

Comparing STSuS and STESuS to MIP

$\|V\|$	$\|D\|$	μ_{opt}	μ_{Star}	μ_{ST}	μ_{STSuS}	μ_{STESuS}	MIP:STSuS	MIP:STESuS
25	5	0.83	2.22	1.07	0.88	0.86	13:83	8:89
	12	0.82	1.92	1.09	0.90	0.86	22:78	6:94
	24	0.80	1.71	1.01	0.87	0.84	8:92	2:98
50	5	0.82	2.79	1.24	0.92	0.89	27:73	18:82
	25		2.19	1.15	0.92	0.89	17:83	8:92
	49		2.00	1.06	0.88	0.87	5:95	2:98
100	5	0.74	3.24	1.36	0.87	0.85	17:83	14:86
	50		2.28	1.16	0.93	0.89	5:95	2:98
	99		2.11	1.09	0.88	0.86	1:99	1:99

For comparing the star-based algorithms to MIP, it is most interesting to

compare STSuS to MIP, as they have the same time complexity if $|A| \in \Theta(|V|^2)$ in the underlying graph. STSuS beats MIP for 87% of the instances, and the averaged performance ratio of STSuS is significantly smaller than the one of MIP. The probability of STSuS and STESuS outperforming MIP seems to increase with the proportion of destinations among the nodes.

6.3 Effect of the weak points of the star on the star-based Algorithms

The Star algorithm has two weak points. As seen in Section 3, its performance depends heavily on the relative position of the source. Second, its performance deteriorates with decreasing width to height ratio of the rectangle in which the instance is deployed. In the next sets of experiments, we study whether and to what extent these weak points affect STSuS and STESuS. To this end, we study networks of 25 nodes with $|D| \in \{5, 12, 24\}$. In the following tables, the averaged power consumptions of the optimal solutions, MIP, Star, ST, STSuS and STESuS are given in columns 3-8, respectively, and columns 9-13 list the averaged performance ratio of MIP, Star, ST, STSuS and STESuS.

Let ϱ denote the width to height ratio of the rectangle on which the instance is distributed randomly, and assume $\varrho \leq 1$ without loss of generality. We generate a test set of every combination of $|D|$ and $\varrho \in \{1, 0.5, 0.3, 0.25\}$. The results are given in Table 4. As expected, the performance of both Star and ST significantly deteriorates with decreasing ϱ. Nevertheless, the performance of STSuS and STESuS is stable, indicating that SuS and ES2 are well suited to adapt the ST solution to instances with small ϱ. In contrast to the star-based algorithms, the performance of MIP increases with decreasing ϱ. The reason for this is that with decreasing ϱ, the optimal solution arborescence becomes

more pathlike than starlike, and MIP typically outputs arborescences that are more pathlike than starlike (the worst-case instances by Bauer et al. (2009) and Wan et al. (2004) illustrate this behavior). MIP is nevertheless outperformed by both STSuS and STESuS.

Table 4

Performance for different values of ϱ

$\lvert D\rvert$	ϱ	opt	MIP	Star	ST	STSuS	STESuS	ρ_{MIP}	ρ_{Star}	ρ_{ST}	ρ_{STSuS}	ρ_{STESuS}
5	1.00	24.24	29.67	63.81	31.53	25.98	25.39	1.22	2.62	1.30	1.07	1.04
	0.50	13.49	15.74	43.12	18.68	14.65	14.21	1.17	3.11	1.38	1.08	1.05
	0.30	9.72	11.03	39.74	15.33	10.48	10.25	1.14	3.92	1.57	1.08	1.05
	0.25	8.88	9.90	39.24	14.54	9.60	9.49	1.12	4.23	1.63	1.08	1.07
12	1.00	32.60	39.94	76.36	42.82	35.65	34.31	1.23	2.31	1.32	1.10	1.05
	0.50	17.41	20.89	51.87	26.56	19.56	18.59	1.21	2.92	1.53	1.13	1.07
	0.30	12.26	13.93	48.34	22.54	13.49	12.85	1.14	3.87	1.85	1.10	1.05
	0.25	11.03	12.33	47.84	21.66	12.09	11.66	1.12	4.26	1.98	1.10	1.06
24	1.00	39.46	49.44	85.06	49.33	42.63	41.27	1.26	2.12	1.26	1.08	1.05
	0.50	20.52	24.93	57.85	30.99	23.33	22.04	1.22	2.79	1.51	1.14	1.07
	0.30	14.00	16.09	53.97	26.83	15.75	14.88	1.15	3.84	1.93	1.13	1.06
	0.25	12.49	14.13	53.43	25.63	13.89	13.17	1.13	4.26	2.07	1.11	1.05

To examine the effect of the position of the source, we generate two sets of 25

nodes for every $|D| \in \{5, 12, 24\}$ and $\varrho = 1$. For the first set, the source is in the geometric center of the square in which the instance is deployed, and for the second set, the source is in one of the corners of the square. The results are given in Table 5 and provide some interesting insights. Examining the performance ratios of the algorithms, we first see that MIP performs significantly better on instances with the source in a corner than with the source in the center. This is in accordance with the above discussed characteristic of MIP of favoring pathlike solution arborescences, and such solutions are more likely to be good when the position of the source is peripheral rather than central. It is also illustrated by the stronger performance of the Star for instances with the source in the center, when compared to the weak performance on the other instances. As discussed in Section 3, the Star outperforms MIP for instances with the source in the center and $|D| \in \{12, 24\}$. This partly explains why STSuS and STESuS perform better than MIP, as they adapt the Star algorithm to general locations of the source. That this is done in a suitable manner can be seen from the last two columns, as there is almost no difference in the performance ratios for instances with the source in the center or in a corner.

7 Applications

A specific application of MEMP is given by the much spoken of "One Laptop Per Child" (OLPC) project (see the web page `http://www.laptop.org/en/` of the project for more information). It seeks to provide children in developing countries with inexpensive laptops for education. The laptops are specifically geared towards use in remote areas and for home learning. Every laptop is equipped with a wireless network antenna, and all laptops within range of

Table 5

Performance for different positions of the source

$\|D\|$	source	opt	MIP	Star	ST	STSuS	STESuS	ρ_{MIP}	ρ_{Star}	ρ_{ST}	ρ_{STSuS}	ρ_{STESuS}
5	center	19.81	25.61	29.13	25.50	20.84	20.52	1.28	1.51	1.30	1.06	1.04
	corner	33.20	39.00	119.97	39.70	34.84	34.46	1.17	3.65	1.20	1.05	1.04
12	center	27.68	38.55	34.99	33.45	28.84	28.24	1.39	1.28	1.22	1.04	1.02
	corner	40.99	48.59	142.62	51.32	43.88	42.70	1.19	3.53	1.26	1.07	1.04
24	center	32.53	48.40	38.61	37.60	33.75	33.08	1.49	1.19	1.16	1.04	1.02
	corner	47.43	56.10	157.90	57.44	50.27	49.06	1.18	3.36	1.22	1.06	1.04

each other (the range is about 2 km in flat areas without buildings) form a wireless ad hoc mesh network. In a teaching situation, various multicasts of for example a text by the teacher to a group of his students are necessary. This has to be done in an energy-efficient way, as the only source of power might be solar power or power the children generate by a dynamo. Although the children are mobile, they can be assumed to remain at their position during class or in the evenings. These are exactly the preconditions of MEMP.

Another example is the company TerraNet, which provides mobile telephony to areas not covered by conventional telecommunication, for example remote or disaster areas. All TerraNet handhelds within range of each other form a wireless ad hoc mesh network, and communication between two devices out of reach of each other is relayed by other devices in the network. When one device seeks to establish a connection, it needs to broadcast to find the destination

device and the shortest path to it. To prolong the network's lifetime, the broadcast needs to be energy efficient, which gives the preconditions of MEBP.

Parameswaran et al. (2008) outline a business model for distributing multimedia content of high quality and thus having high bandwith requirements over the wired internet. They single out multicasting as delivery solution, and elaborate the need of heuristics to compute multicast routings. Using the heuristics we presented, their model can be directly transferred to distribute pay as you go content over wireless ad hoc networks. Wireless ad hoc mesh networking is standardized by IEEE 802.11s. All devices implementing this standard are able to form a wireless network with each other. Among others, OlPC and Google sponsor the implementation of IEEE 802.11s for the operating system Linux in order to promote wireless ad hoc networking for Linux devices. The standard is implemented in Linux since kernel version 2.6.26, which enables devices running Linux to wireless ad hoc networking.

In order to apply the algorithms discussed in this paper to a wireless ad hoc network one needs to know the energy consumption for a direct connection between two devices. This information can be gathered readily: For example, a device joining an existing network can gradually increase its transmission energy asking other devices to acknowledge it as soon as they hear it. The joining device thus knows by which transmission energy it reaches which devices. As the transmission energy depends only on the distance between the devices, another approach is to learn the coordinates of every device. In the OLPC homeschooling case, the coordinates of each child can be identified simply by using Google maps. For disaster relief situations like after earthquakes or tsunamis, devices can be equipped with a GPS tracker.

8 Conclusions

In this paper, we have studied the possibility of solving MEMP by center-oriented algorithms, and developed the efficient STSuS and STESuS algorithms. We have shown that center-oriented algorithms keep up with greedy arborescence constructions for developing algorithms for MEMP. The center-oriented STSuS outperforms the well-known MIP at equal time complexity and implementation simplicity.

Furthermore, as there are distributed algorithms for the shortest path problem available (e.g. Chandy and Misra (1982)), our algorithms are a promising starting point for developing distributed algorithms for MEMP.

Another direction for future research is how the routing arborescences can be optimally restored in the case of node failure. A simple strategy is to just redetermine a routing arborescence from scratch, which due to the short running time is a sensible approach for networks of up to 100 nodes. For larger methods, a more sophisticated approach is called for, both to minimize the time needed for network recover, and because the failure of one node within a large network will in most cases only call for local amendments.

References

Bauer, J., Haugland, D., and Yuan, D. Analysis and computational study of several integer programming formulations for minimum-energy multicasting in wireless ad hoc networks. *Networks*, 52, 2, 2008, 57–68.

Bauer, J., Haugland, D., and Yuan, D. New results on the time complex-

ity and approximation ratio of the broadcast incremental power algorithm. *Information Processing Letters*, 109, 12, 2009, 615–619.

Cagalj, M., Hubaux, J., and Enz, C. Energy-efficient broadcasting in all-wireless networks, *Wireless Networks*, 30, 1–2, 2005, 177–188.

Chandy, K. and Misra, J. Distributed computation on graphs: Shortest path algorithms. *Communications of the ACM*, 11, 1982, 833–837.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2001.

Guo, S. and Yang, O. Energy-aware multicasting in wireless ad hoc networks: A survey and discussion. *Computer Communications*, 30, 2007, 2129–2148.

Kang, I. and Poovendran, R. COBRA: Center-oriented broadcast routing algorithms for wireless ad hoc networks. In *Wireless Communications and Networking Conference, 2004. WCNC. IEEE*, 2, March 21-25, 2004, 813–818.

Li, F. and Nikolaides, I. On minimum-energy broadcasting in all-wireless networks. In *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN 2001)*, November 14-16, 2001, 193–202. IEEE Computer Society, Washington, DC.

Moore, G. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86, 1, 1998, 82–85.

Parameswaran, M., Stallaert, J., and Whinston, A. Shared distribution of digital products using multicast. *Electronic Commerce Research and Applications*, 7, 1, 2008, 93–104.

Perkins, C. E. *Ad Hoc Networking*. Pearson Education, Indianapolis, IN, 2000.

Wan, P.-J., Călinescu, G., and Yi, C.-W. Minimum-power multicast routing in static ad hoc wireless networks. *IEEE/ACM Transactions on Networking*,

12, 2004, 507–514.

Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34, 1, 2001, 44–51.

Wieselthier, J. E., Nguyen, G. D., and Ephremides, A. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *IEEE INFOCOM*, 2000, 585–594.

Wieselthier, J. E., Nguyen, G. D., and Ephremides, A. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Networks and Applications*, 7, 6, 2002, 481–492.

Yuan, D., Bauer, J., and Haugland, D. Minimum-energy broadcast and multicast in wireless networks: An integer programming approach and improved heuristic algorithms. *Ad Hoc Networks*, 6, 5, 2008, 696 – 717.

A Proofs

A.1 Proof of Theorem 1

Step 1 has $O(|A| + |V| \log |V|)$ time complexity using Dijkstra's Algorithm (Cormen et al., 2001). To identify a node $z \in \text{argmin}\{p_{\Phi_z} + p_z(D \backslash D_z) : z \in V\}$, Depth First Search (DFS) is applied to Φ. When the DFS reaches node v, the set D_v is determined by $D_v = D_{\pi_v(T)} \cup \{d \in D : c_{\pi_v(T)d} \leq c_{\pi_v(T)v}\}$ in $O(|D|)$ time. As DFS has $O(|V|)$ time complexity (Cormen et al., 2001), Step 2 has $O(|V||D|)$ time complexity. As the Steps 6, 7, 9-11, 13 and 14 run in constant time, Steps 5-14 have $O(|V||D|)$ time complexity. As Step 15 has $O(|V|)$ time complexity, the result follows. □

A.2 Proof of Theorem 2

Yuan et al. (2008) proved that the SuS heuristic has $O(|V|^2)$ time complexity, provided that for all nodes $v \in V$ all nodes $w \in V \setminus \{v\}$ are sorted according to c_{vw}. However, for performing SuS, it is not necessary to sort $V \times V$ completely. It is only necessary to sort $\Gamma_v(T)$ for every $v \in V$. As there are $|V| - 1$ children in a tree, the necessary sorting operations have total time complexity $O(|V| \log |V|)$. □

A.3 Proof of Corollary 3

Since the SuS procedure is a part of the SuS heuristic, which has $O(|V|^2)$ time complexity, the SuS procedure also has $O(|V|^2)$ time complexity. □

A.4 Proof of Corollary 4

Since both ST and the SuS procedure have $O(|V|^2)$ time complexity (Thm. 1 and Cor. 3), STSuS as a combination of ST and the SuS procedure also has $O(|V|^2)$ time complexity. □

A.5 Proof of Theorem 5

As Step 1 performs ST, and Steps 2-9 together with Steps 11-13 perform the SuS procedure, these steps have $O(|V|^2)$ time complexity. Yuan et al. (2008) provided an implementation of the ES procedure that has $O(|V| \log |V|)$ time complexity. Altering ES to ES2 does not affect this: Removing $\Gamma_z(T)$ and

$\{v : h_v(T) < h_f(T)\}$ from K can be done using depth first search, which has $O(|V|)$ time complexity. As ES2 is part of the for-loop in Step 4, it is executed at most $|V| - 1$ times. Thus, the time complexity of STESuS is $O(|V|^2 \log |V|)$. $\square$

B Key Terms

Table B.1

Definition of Abbreviations and Terms used

Term	Definition	Comments
arborescence	tree with arcs oriented away from the root	
BIP	Broadcast Incremental Power (Algorithm)	MIP in the *broadcast* case
broadcast	one-to-all communication of a given *source*	all nodes except *source* are *destinations*
destination	device which the *source* wants to communicate with	
MEBP	Minimum Energy Broadcast Problem	*broadcast* case of MEMP
MEMP	Minimum Energy Multicast Problem	Problem of finding an energy efficient routing from a given *source* to a given set of *destinations*
MIP	Multicast Incremental Power (Algorithm)	best-known algorithm for MEMP by Wieselthier et al. (2002)
s-arborescence	*arborescence* with root s	
source	network device with a message for at least one other device	s, given in the problem instance

continued on next page

Term	Definition	Comments
SuS heuristic	Successive Shrink heuristic	local search heuristic to improve a multicast arborescence, Yuan et al. (2008)
SuS procedure	Successive Shrink for one node	the *SuS heuristic* consists of the SuS procedure for every node in the arborescence
Star	*arborescence* with all nodes connected directly to the *source*	the simplest MEMP algorithm
ST	Shortest Trunk (Algorithm)	MEMP algorithm constructing an *(s, z)-trunk*
STESuS	Shortest Trunk Enhanced Successive Shrink (Algorithm)	MEMP algorithm combining ST and an enhanced version of SuS
STESuS	Shortest Trunk Successive Shrink (Algorithm)	MEMP algorithm combining ST and SuS
(s, z)-trunk	*s-arborescence* where every node is either on or directly connected to some node on the shortest (s, z)-path in the underlying graph	

C Notation

Table C.1

Notation associated with a node v in an s-arborescence T

$\Delta_v(T)$	is the set of descendants of node $v \in V_T$ in T
$\gamma_v(T)$	is the node for which $p_v(T) = c_{v\gamma_v(T)}$
$\Gamma_v(T)$	is the set of children of node $v \in V_T$ in T
Φ	is the shortest-path-arborescence of G with root s
Φ_v	is the shortest s-v-path in G
$\pi_v(T)$	is the parent of $v \in V_T$ in T
$h_v(T)$	is the number of hops between s and $v \in V_T$ in T
$I_v(T)$	is the sum of the additional power needed at nodes in V_T if v becomes active
$p_v(S)$	is the power required for v to cover all active nodes in $S \subseteq V$
$p_v(T)$	$= p_v(\Gamma_v(T))$ is the power required at v by T
p_T	$= \sum_{v \in V_T} p_v(T)$ is the total power required by a broadcast described by T
$p_{\text{opt}}(\mathcal{I})$	the power consumption of the optimal solution to $\mathcal{I}$
$p_{\mathcal{A}}(\mathcal{I})$	the power consumption of the solution to $\mathcal{I}$ determined by algorithm $\mathcal{A}$
$\|V\|/\|D\|$	is a set of 100 test instances with $\|V\|$ nodes and $\|D\|$ destinations