

Linköping University Post Print

A fast local search method for minimum energy broadcast in wireless ad hoc networks

Joanna Bauer, Dag Haugland and Di Yuan

N.B.: When citing this work, cite the original article.

Original Publication:

Joanna Bauer, Dag Haugland and Di Yuan, A fast local search method for minimum energy broadcast in wireless ad hoc networks, 2009, OPERATIONS RESEARCH LETTERS, (37), 2, 75-79.

<http://dx.doi.org/10.1016/j.orl.2009.01.004>

Copyright: Elsevier Science B.V., Amsterdam.

<http://www.elsevier.com/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-17741>

A Fast Local Search Method for Minimum Energy Broadcast in Wireless Ad Hoc Networks

Joanna Bauer^a, Dag Haugland^{a,*}, Di Yuan^b

^a*Department of Informatics, University of Bergen, PB. 7800, N-5020 Bergen, Norway*

^b*Department of Science and Technology, Linköping University, SE-601 74 Norrköping, Sweden*

Abstract

Local search methods are often used to reduce the power consumption of broadcast routing in wireless networks. For a classic such method, **sweep**, the best available time complexity result is $O(|V|^4)$. We present an $O(|V|^2)$ -time method, which exhaustively removes unnecessary transmissions yielding a solution comparable to that of **sweep**.

Key words: Wireless Ad Hoc Network, Minimum Energy Broadcast, Local Search, Sweep.

1. Introduction

In many applications of wireless systems, a minimum energy broadcast routing from a given source unit has to be computed repeatedly and quickly. To establish a broadcast routing, a transmission power must be assigned to each network unit. The power assignment corresponds to a transmission range assignment, implying that 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. Furthermore, the power needed at one unit to cover some other unit grows at least quadratically with the distance [8], and hence, computing a minimum energy routing is NP-hard [2]. Therefore, the energy efficiency of broadcast applications depends on efficient routing heuristics.

A common approach (e.g. [5], [8]) is to represent the network as a graph $G = (V, A)$ and greedily construct a routing arborescence. The power of every node is equal to that of the most power demanding outgoing arc in the arborescence. Due to the greedy construction, the solutions often contain unnecessary transmissions. That is, the power assignment of a node v is determined by the arc (v, w) in the routing arborescence, although w is within the transmission range of some other node f . If w is not on the path between the source and f , then node w can be linked to f and the power assignment at v can be reduced. The total transmission power can thus be reduced by removing all such unnecessary transmissions. In [8], the local search heuristic **sweep** was introduced. For networks consisting of 20 nodes, applying **sweep** to an arborescence constructed by the well-known Broadcast Incremental Power (BIP) algorithm [8], for example, reduces the total power consumption from about 1.3 to about 1.17 times the optimal power consumption [9].

*Corresponding author

Email addresses: Joanna.Bauer@ii.uib.no (Joanna Bauer), Dag.Haugland@ii.uib.no (Dag Haugland), diyua@itn.liu.se (Di Yuan)

One important characteristic of local search heuristics is whether the search is done *exhaustively*. This means that the search is continued until it arrives at a local optimum in respect to the search neighborhood. Most other suggested improvement heuristics (e.g. [4], [7], and [9]) do not perform exhaustive search. They rather execute a pre-defined number of search moves in order to be able to prove a certain time complexity (in general, $O(|V|^3)$), or simply adopt a maximum time limit. Also, they do not only remove unnecessary transmissions, but also allow the routing arborescence to be altered by increasing the power previously assigned to a node.

For performing **sweep** exhaustively, we are not aware of any stronger time complexity results than the $O(|V|^4)$ bound proved in [9]. This is in contrast to arborescence construction heuristics with low time complexity like BIP ($O(|A| + |V| \log |V|)$, [1]).

In this paper, we propose the new local search algorithm, *Bottom-Up Sweep* (BUS), and prove that an exhaustive search by BUS has $O(|V|^2)$ time complexity. None of **sweep** (if performed exhaustively) and BUS can improve the output of the other. To the best of our knowledge, BUS is the lowest-time-complexity exhaustive local search for the Minimum Energy Broadcast Problem. We apply both this method and **sweep** to arborescences constructed by BIP. The experiments show that the power reductions found by our method are comparable to those found by **sweep**.

2. Preliminaries

A problem instance is given by a complete directed graph $G = (V, A)$, where the nodes represent the networking units, a source $s \in V$, and the power requirements $c \in \mathbb{R}_+^{|A|}$. To avoid the need for tie-breaking rules, we assume that all c_{vw} are unique, although the results presented do not depend on this assumption.

A solution to a problem instance can be given as an s -arborescence $T = (V, A_T)$ with arc set $A_T \subseteq A$. An s -arborescence is a directed tree where all arcs are oriented away from s . We let \mathcal{T} denote the set of all possible s -arborescences on V . Let $\Gamma_v^+(G) = \{w : (v, w) \in A\}$ and $\Gamma_v^-(G) = \{w : (w, v) \in A\}$ denote respectively the sets of out-neighbors and in-neighbors of node v in G . For any $T \in \mathcal{T}$ and any $v \in V$, we let $\Delta_v(T)$ denote the (possibly empty) set of descendants of v in T , and we define $\Delta'_v(T) = \Delta_v(T) \cup \{v\}$. The s -arborescence T defines a power assignment $p_v(T)$ to all nodes $v \in V$. If $\Gamma_v^+(T) = \emptyset$ we have $p_v(T) = 0$, and otherwise $p_v(T) = c_{v\gamma_v(T)}$, where $\gamma_v(T) \in \Gamma_v^+(T)$ is defined by $\max_{w \in \Gamma_v^+(T)} c_{vw} = c_{v\gamma_v(T)}$ and referred to as the *critical child* of v in T . Thereby, the cost of T is $p_T = \sum_{v \in V} p_v(T)$. We say that v reaches w if $p_v(T) \geq c_{vw}$. The minimum energy broadcast routing problem can then be formulated as

[MEBP] Find an s -arborescence T such that p_T is minimized.

3. The BUS Local Search Algorithm

Any local search algorithm is characterized by the search neighborhood, the update strategy, that is the strategy of selecting what neighbor solution to move to, and by whether or not the search is done exhaustively.

3.1. The Local Search Neighborhood

Many local search neighborhoods for MEBP have been proposed. In [4], the authors define two general neighborhoods, which include s -arborescences where nodes may be assigned higher power than they are in the current one. The neighborhood searched by **sweep** in [8], on the contrary, is more restricted: Within one move, one node adopts all non-ancestor nodes within its transmission range. Such a move is accepted by **sweep** if it reduces the total

power consumption. We introduce a neighborhood that is similar to the one of **sweep** in the sense that no node can increase its power assignment, but differs from the **sweep** neighborhood in that only one arc can be exchanged in a single move. For any $T \in \mathcal{T}$, we define the neighborhood as the set $\mathcal{N}(T)$ of s -arborescences $T(v, f)$ that can be obtained by disconnecting $\gamma_v(T)$ from v and linking it to some $f \in V$, without increasing the power assigned to f . That is, $p_f(T(v, f)) = p_f(T)$ and

$$\mathcal{N}(T) = \left\{ T(v, f) \in \mathcal{T} : T(v, f) = (V, A_T \setminus \{(v, \gamma_v(T))\} \cup \{(f, \gamma_v(T))\}) : f \neq v \wedge p_f(T) \geq c_{f\gamma_v(T)} \right\}.$$

As $T(v, f)$ is an s -arborescence, f cannot be a descendant of $\gamma_v(T)$ in T .

In the update strategy of **sweep**, no attempt is made to reduce node power in any particular order of the nodes. This is unfortunate, as it is then difficult to prove any time complexity lower than $O(|V|^4)$ for exhaustive search. Wieselthier et al. suggested in [8] to execute **sweep** only for a predefined small number of iterations, which reduces the time complexity to $O(|V|^2)$. Obviously, this does not guarantee that all superfluous transmissions are removed.

In our method, we process the nodes in such an order that tight bounds on both the number of moves and the computational work within each move are achieved. Consequently, a local optimal solution can be found within $O(|V|^2)$ time.

In the following, we first present the search strategy. We then introduce the concept of a labeled reachability graph on which our algorithm is based. Finally, correctness of the algorithm and the running time are proved.

3.2. The Bottom Up Strategy

Consider an s -arborescence T where the node w is reached by one of its descendants

$f \in \Delta_w(T)$. Although f is assigned a sufficiently high power level to become the new parent of w , the move is prohibited due to the ancestor relation between f and w . But if a later move assigns a new parent to f , and this parent does not descend from w , f may become the parent of w . In order to conclude that w cannot be assigned a new parent by any later move, we therefore need to know that neither can any descendant of w . This calls for an approach where the nodes are processed in an upward direction, where the leaves are the first and the child nodes of s are the last to be assigned to new parents.

Definition 1. The rank $\rho_v(T)$ of node v in the arborescence T is defined as the height of the subarborescence of T rooted at v .

For any arborescence $T_0 \in \mathcal{T}$ given as input to the local search algorithm, define $\bar{\rho}_v = \rho_v(T_0)$ for all $v \in V$, and let the partition (V^0, \dots, V^h) of V be defined by $V^k = \{v \in V : \bar{\rho}_v = k\}$, $k = 0, \dots, h$, where h is the height of T_0 . Ordering the nodes by non-decreasing rank in T_0 , we get $V = \{v_1, \dots, v_{|V|}\}$, where $\bar{\rho}_{v_i} \leq \bar{\rho}_{v_j}$ for $1 \leq i < j \leq |V|$. This is the order in which the algorithm below processes the nodes, and hence the order is fixed before and throughout the processing.

Since processing a node amounts to reducing its power by transferring child nodes to new parents, the leaves V^0 of T_0 need no processing. In the first $|V^1|$ iterations of the algorithm, we process all nodes v whose child nodes all are leaves. As long as we can find a node $f \in V$ with sufficiently high power to reach the critical child w of $v \in V^1$, we assign f as the new parent to w . Hence another node becomes the critical child of v , and the process continues until all child nodes have been assigned new parents, or the critical child is reached only by v . Note that since any child node w of v is a leaf, it is not necessary to verify that f does not

descend from w . When v has been processed, we can conclude that further moves can never reduce the power level at v (as no node will ever increase its power during the procedure). Therefore, we can consider this power level as fixed, and so is the parent assignment to all child nodes of v that were not moved.

In subsequent iterations, we process the nodes in V^2, \dots, V^h in an analogous manner. As the child nodes w of v are not necessarily leaves, current descendants of w must be disregarded even if their power can reach w . This could be done by e.g. depth first search, which would result in a running time of $O(|V|^3)$. A faster method is achieved by organizing the search with the help of a reachability graph.

3.3. Reachability Graphs

In [6], Mavinkurve et al. introduced the *reachability graph* associated with T , which contains the arc (u, w) if and only if the power assignment to u is sufficient for u to reach w . The reachability graph is equivalent to the term *topology* in [3].

Our definition of a *labeled reachability graph*, to be given in precise terms below, resembles the definition in [6] in that the presence of an arc (u, w) implies that some node is assigned sufficient power to become the new parent of w . Unlike the definition in [6], however, it does not imply that the node reaching w is u itself. If $u \in F$, where $F \subseteq V$ is a set of nodes referred to as *fixed*, the signification of the arc is that w is reached by some node ℓ_{uw} in the subarborescence of T rooted at u . Hence the arc label ℓ_{uw} is used to identify a new parent of w .

Definition 2. A *labeled reachability graph* of an s -arborescence $T = (V, A_T)$, is a quadruple $H = (V, A_H, \ell, F)$, where $A_H \subseteq V \times V$ is the arc set, $\ell : A_H \mapsto V$ defines a node ℓ_{uw} for each $(u, w) \in A_H$, and where $F \subseteq V$, satisfying the properties:

- (1) $(u, u) \notin A_H \ \forall u \in V$,
- (2) A_H does not contain any arc (u, w) where $w = \gamma_v(T)$ for some $v \in F$,
- (3) for all arcs $(u, w) \in A_H$, $\ell_{uw} \in \Delta'_u(T)$ and ℓ_{uw} reaches w ,
- (4) for all nodes $w \in V$ reached by some $f \in V \setminus \Delta_w(T)$ other than its parent, there exists an arc $(u, w) \in A_H$ where $u \in F \cup \{f\}$ and $f \in \Delta'_u(T)$.

The purpose of the set F is to store nodes where power cannot be reduced. Accordingly, property (2) in Def. 2 states that A_H should not contain an arc pointing at the critical child of any such node. In the subsequent text, whenever we apply graph notation to H (e.g. $\Gamma_v^-(H)$), we refer to the directed graph (V, A_H) .

If the parent of $v \in F$ is not fixed, whereas $\Delta_v(T) \subseteq F$, we refer to the subarborescence of T rooted at v as a *maximal fixed subarborescence*.

By letting $F \subseteq V^0$, we see that Def. 2 subsumes the reachability graph suggested in [6].

Property (3) in Def. 2 implies that if $(u, w) \in A_H$, then there is a node in the subarborescence rooted at u reaching w , and the label ℓ_{uw} is one such node.

Property (4) ensures that when w is reached by some non-descendant f , this is reflected by some arc in A_H of which w is the end node. The start node of this arc is either f itself ($f \notin F$) or a fixed node u of which f is a descendant.

An important characteristic of our algorithm (see Lemma 5), is that all fixed start nodes of reachability arcs are roots of maximal fixed subarborescences. This implies that information concerning reachability from nodes in maximal fixed subarborescences is aggregated at the roots, and thus there is no need to search the subarborescences for reachability arcs.

3.4. The BUS Local Search Algorithm

The *Bottom Up Sweep* (BUS) algorithm is given in Tab. 1. We say that *BUS is processing node v_i* when it is executing any of the Steps 9-19 and $|F| + 1 = i$. All nodes in $V \setminus V^0$ are processed exactly once, and (see Steps 4 and 20) F contains exactly the nodes in V^0 and those already processed.

In the following, we prove local optimality of the solution returned by BUS and the algorithm's time complexity. To this end, we first prove five lemmata, three of which are based on induction over the while-loop spanning Steps 8-20 of the algorithm.

Lemma 3. *At the end of every iteration of the while-loop 8-20 in Tab. 1, we have*

$$\bar{\rho}_w < \bar{\rho}_v \quad \forall w \in \Delta_v(T), \quad v \in V \setminus F \quad (1)$$

PROOF. Immediately after the initialization of H in Steps 4-7, inequalities (1) hold by definition. We prove that if the inequalities hold at the start of any iteration of the loop 8-20, they also do at the end of the iteration.

When processing node $v \in V$, BUS updates T only in Step 13, where a node $w \in \Gamma_v^+(T)$ may be assigned a new parent ℓ_{uw} . Let y be any node on the path from the root to ℓ_{uw} in T . We need to show that either y is fixed or all its new descendants $\Delta'_w(T)$ have lower initial rank. If y is already processed, then $y \in F$. Otherwise, since the nodes are processed in an order where $\bar{\rho}$ is non-decreasing, we have $\bar{\rho}_y \geq \bar{\rho}_v$. By the induction hypothesis, we thus get $\bar{\rho}_x < \bar{\rho}_v \leq \bar{\rho}_y \quad \forall x \in \Delta'_w(T)$, and Lemma 3 follows. \square

Corollary 4. *When BUS is processing node v , we have $\Delta_v(T) \subseteq F$.*

PROOF. It follows from Lemma 3 and the node processing order (non-decreasing $\bar{\rho}_v$), that while node v is being processed, all current descendants of v are already processed, and hence $\Delta_v(T) \subseteq F$. \square

Lemma 5. *At the end of every iteration of the while-loop 8-20 in Tab. 1, we have for all $(u, w) \in A_H$ that u is the root of a maximal fixed subarborescence if $u \in F$, and $u = \ell_{uw}$ otherwise.*

PROOF. Immediately after the initialization of H , the statement is obviously true since $u \notin F$ and $u = \ell_{uw} \quad \forall (u, w) \in A_H$. We prove that if the statement is true at the start of an iteration of the while-loop 8-20, it also is after the updates made to F (Step 20) and A_H (Steps 15 and 19).

The statement holds for all $u \notin F$, since no nodes leave F , and no reachability arcs are transferred to nodes outside F .

In Step 15, we transfer reachability arcs to node $u \in F$, which has been identified as the start node of some arc in A_H . Hence the induction hypothesis implies that u is the root of a maximal fixed subarborescence.

In Step 19, we transfer reachability arcs from w to v . By Cor. 4, we have that $\Delta_v(T) \subseteq F$. Since $v \notin F$, all $w \in \Gamma_v^+(T)$ are roots of maximal fixed subarborescences. Thus Step 20 makes v the root of a new maximal fixed subarborescence, ensuring that the statement holds also at the end of the iteration. \square

Lemma 6. *At the end of every iteration of the while-loop 8-20 in Tab. 1, T is an s -arborescence of which H is a labeled reachability graph.*

PROOF. Obviously, H is a labeled reachability graph of $T \in \mathcal{T}$ immediately after Steps 4-7. Assume this is true at the start of the iteration in which BUS processes node v .

We first prove that T remains an s -arborescence after its update in Step 13. As $|A_T|$ is unchanged, it suffices to show that T remains acyclic. Assume that in Step 12, BUS identifies an arc $(u, w) \in A_H$ producing a cycle in T when arc (ℓ_{uw}, w) is added to A_T in

Table 1: The Bottom Up Sweep (BUS) Algorithm

```

BUS( $G = (V, A)$ ,  $s \in V$ ,  $c \in \mathbb{R}_+^{|A|}$ ,  $T = (V, A_T)$ )
1  for all  $v \in V$ 
2     $\bar{\rho}_v \leftarrow$  the height of the subarborescence rooted at  $v$ 
3  Sort  $V$  into  $(v_1, \dots, v_{|V|})$  such that  $\bar{\rho}_{v_1} \leq \dots \leq \bar{\rho}_{v_{|V|}}$ 
4   $F \leftarrow \{v \in V : \bar{\rho}_v = 0\}$  // Steps 4-7 initialize  $H \leftarrow (V, A_H, \ell, F)$ 
5   $A_H \leftarrow \{(u, v) \in A : c_{uv} \leq p_u(T)\}$ 
6  for all  $(u, v) \in A_H$ 
7     $\ell_{uv} \leftarrow u$ 
8  while  $F \neq V$ 
9     $v \leftarrow v_{|F|+1}$ 
10   while  $\Gamma_v^+(T) \neq \emptyset \wedge \Gamma_{\gamma_v(T)}^-(H) \setminus \{v\} \neq \emptyset$ 
11      $w \leftarrow \gamma_v(T)$ 
12      $u \leftarrow$  ChooseReachingNode( $T, H, w, |F| + 1$ ) // Find new (preferably fixed) parent
13      $A_T \leftarrow A_T \setminus \{(v, w)\} \cup \{(\ell_{uw}, w)\}$ 
14     if  $u \in F$  // The new parent has (an ancestor with) fixed power
15       TransferReachability( $A_H, u, w$ )
16     for all  $w \in \Gamma_v^+(H) : c_{vw} \geq p_v(T)$  // For  $w =$  critical child or a node beyond reach
17        $A_H \leftarrow A_H \setminus \{(v, w)\}$  // delete reachability arc to  $w$ 
18     for all  $w \in \Gamma_v^+(T)$  // Transfer reachability information from children to  $v$ 
19       TransferReachability( $A_H, v, w$ )
20      $F \leftarrow F \cup \{v\}$ 
21 return  $T$ 

TransferReachability( $A_H, v, w$ )
22 for all  $x \in \Gamma_w^+(H)$  // For all  $x$  reached by some node in  $\Delta'_w(T)$ 
23   if  $x \neq v \wedge (v, x) \notin A_H$  //  $x$  is neither  $v$ , nor already reached by some node in  $\Delta'_v(T)$ 
24      $A_H \leftarrow A_H \cup \{(v, x)\}$  // Add the reachability arc  $(v, x)$ 
25      $\ell_{vx} \leftarrow \ell_{wx}$ 
26      $A_H \leftarrow A_H \setminus \{(w, x)\}$  // Delete the reachability arc  $(w, x)$ 

ChooseReachingNode( $T, H, w, i$ )
27 if  $(\Gamma_w^-(H) \setminus \Gamma_w^-(T)) \cap F \neq \emptyset$ 
28   return some  $u \in (\Gamma_w^-(H) \setminus \Gamma_w^-(T)) \cap F$ 
29 else
30   return  $v_{j'}$ , where  $j' = \max \{j = i + 1, \dots, |V| : v_j \in \Gamma_w^-(H) \setminus \Gamma_w^-(T)\}$ 

```

Step 13. Then the new parent of w , ℓ_{uw} , currently must be in the subarborescence of T with node set $\Delta'_w(T)$. As $\Delta'_w(T) \subseteq \Delta_v(T)$, we have $\Delta'_w(T) \subseteq F$ by Cor. 4. Because $v \notin F$, w is the root of a maximal fixed subarborescence containing ℓ_{uw} .

By the induction hypothesis, H is a labeled reachability graph, and property (3) in Def. 2 implies $\ell_{uw} \in \Delta'_u(T)$. As $\ell_{uw} \in \Delta'_w(T) \subseteq F$, we have by Lemma 5 that $\ell_{uw} \in \Delta_u(T)$ with $u \in F$ being the root of a maximal fixed subarborescence containing ℓ_{uw} .

Thus, both w and u are roots of some maximal fixed subarborescence containing ℓ_{uw} , and as this subarborescence is unique, we have $u = w$. But since $(u, w) \in A_H$, property (1) in Def. 2 gives a contradiction. Hence T remains acyclic.

We next prove that H remains a labeled reachability graph of T , by showing that the properties in Def. 2 remain satisfied at the end of the iteration processing node v .

(1): Since new arcs are added to H only in Step 24, the property follows directly from the first condition of the if-statement of Step 23.

(2): We have to show that at the end of the iteration, $S = \bigcup_{u \in V} \Gamma_u^+(H)$ contains no critical child nodes of fixed nodes. Since an arc is added to A_H (Step 24) only if it replaces an arc with the same end node (Step 26), S is never extended. When v is processed, the critical child nodes of all nodes but v remain unchanged. By the induction hypothesis, S hence does not contain $\gamma_{v'}(T)$ for any $v' \in F \setminus \{v\}$.

After the while-loop 10-15, either v has no critical child ($\Gamma_v^+(T) = \emptyset$), or $\Gamma_w^-(H) \subseteq \{v\}$, where w is the new critical child of v . But then $c_{vw} = p_v(T)$, and if $\Gamma_w^-(H) = \{v\}$ the arc (v, w) is removed from A_H in Step 17.

(3): For all nodes w such that $p_v(T) < c_{vw}$, i.e., nodes no longer reached by v , we delete (Step 17) the corresponding arc in A_H . For all new arcs (v, x) added to A_H (Step 24), we have

by Step 25 and the induction hypothesis that $\ell_{vx} \in \Delta_v(T)$ and that ℓ_{vx} reaches x .

(4): When w is assigned a new parent in Step 13, an arc $(u, w) \in A_H$ has been identified in Step 12. By the induction hypothesis, either $u \notin F$ or u is the root of a maximal fixed subarborescence. The transfer of reachability arcs in Step 15 ensures that for all nodes u' reached by some node in $\Delta'_w(T)$, $u \in \Gamma_{u'}^-(H)$. By the induction hypothesis and Cor. 4, the children of v are roots of maximal fixed subarborescences. Thus, after the reachability arcs of the remaining children of v have been transferred to v (Step 19), and v becomes fixed (Step 20), the property remains satisfied. \square

The set of nodes for which power reduction is possible is

$$R_T = \left\{ v \in V : \exists u \in V \setminus \Delta'_{\gamma_v(T)}(T) \setminus \{v\} \right. \\ \left. c_{u\gamma_v(T)} \leq p_u(T) \right\}.$$

Hence, T is locally optimal with respect to $\mathcal{N}(T)$ if and only if $R_T = \emptyset$.

Lemma 7. *If (V, A_H, ℓ, F) is a labeled reachability graph of T , then $R_T \subseteq V \setminus F$.*

PROOF. Since Properties (4) and (2) in Def. 2 state, respectively, that all nodes reached by a non-descendant (other than their parents) have an entering reachability arc, and that the critical children of fixed nodes do not have any entering reachability arc, the result follows. \square

Lemma 8. *BUS assigns a new parent to each node in T at most twice.*

PROOF. Since the algorithm never processes fixed nodes, it follows that if $w \in V$ is assigned a fixed parent ℓ_{uw} in Step 13, then this is the final parent assignment to w . Assume that $\Gamma_w^-(H) = \{v_{i_1}, \dots, v_{i_m}\} \subseteq V \setminus F$, where $i_1 < \dots < i_m$. Hence $v_{i_m} \notin F$ becomes the new

parent of w . If another parent assignment to w is made, the next occurs while BUS is processing v_{i_m} . In that case, the new parent must be one of $v_{i_1}, \dots, v_{i_{m-1}}$. When processing v_{i_m} , we have $v_{i_1}, \dots, v_{i_{m-1}} \in F$, and the result follows. \square

Theorem 9. *Given a network instance $(G = (V, A), s, c)$ and an s -arborescence $T = (V, A_T)$ as input, the BUS algorithm transforms T to an $\mathcal{N}(T)$ -locally optimal s -arborescence in $O(|V|^2)$ running time.*

PROOF. It follows from Lemma 6 that upon termination of the algorithm, H is a labeled reachability graph of some s -arborescence T . Since also $F = V$, it follows from Lemma 7 that T is locally optimal with respect to $\mathcal{N}(T)$.

For the proof of the running time, assume that T and A_H are represented by adjacency matrices such that node/arc retrieval, insertion and deletion all run in constant time, whereas checking existence of arcs entering/leaving a node (see Step 10) runs in linear time.

All initialization operations in Steps 1-7 run in $O(|V|^2)$ time. The condition in Step 10 is evaluated $|V \setminus V^0|$ times to false, and, by Lemma 8, at most $2|V|$ times to true. Lemma 8 also implies that Steps 11-15, each of which runs in $O(|V|)$ time, are executed at most $2|V|$ times. Finally, since an arc is never reintroduced once it is removed from A_H , Steps 23-26 are applied at most once to each $(w, x) \in A$, and the total running time of Steps 16-20 is also $O(|V|^2)$. \square

4. Numerical Experiments

In this section, we compare the numerical performance of the BUS algorithm to the **sweep** algorithm. We are interested in how much the total power consumption can be reduced, and how efficiently the algorithms move through the search neighborhood.

We generated 100 instances of 1000 nodes and 100 instances of 5000 nodes by distributing the nodes on a square using the standard C++ pseudo-random number generator **rand()**. We let $c_{uv} = d_{uv}^2$ for all $(u, v) \in A$, where d_{uv} is the Euclidean distance between nodes u and v . For every instance, we let BIP [8] construct a broadcast arborescence, which is used as the starting point for both BUS and **sweep**. We apply **sweep** iteratively to its own output until it reaches a local optimum.

The efficiency of the search is measured by two indicators. The first is the number of arc exchanges (Step 13 of BUS) executed before the algorithm reaches a local optimum. This number is given in the second row of Tab. 2. In the table, all numbers are averaged over the 100 instances.

The original **sweep** [8] lets the new parent f take over all non-ancestor nodes w within its reach. If w is not the critical child of its current parent, the total power consumption is not reduced by making w a child of f . Thus, **sweep** may execute unnecessary arc exchanges, especially as nodes may repeatedly be handed around between several potential parents. For the purpose of fair comparison, we therefore modify **sweep** to an algorithm called **sweepCritical**. Within one move of this algorithm, a node becomes the new parent of exactly one other node, which has to be a critical child. Consequently, **sweepCritical** is likely to perform fewer arc exchanges than **sweep**, but also likely to need more iterations before it reaches a local optimum.

The second indicator, given in the third row of Tab. 2, shows how many times the algorithms check whether one node can take over children of other nodes. For BUS, the number of such node checks equals $|V \setminus V^0|$ in every instance. For **sweep** and **sweepCritical**, it equals $|V|$ times the number of iterations necessary to reach a local optimum.

The power consumptions resulting from each

of the three algorithms are given in the first row of Tab. 2, where for every instance the cost of BIP's solution is normalized to 100. No significant difference in the three algorithms' solutions can be observed.

The number of executed arc exchanges necessary to reach a local optimum is smaller for BUS than for `sweep` and `sweepCritical`, although the difference between BUS and `sweepCritical` is marginal. Hence BUS and `sweepCritical` have the desired effect of converging to a local optimum that, in terms of the number of arc exchanges, is closer to the initial s -arborescence. The `sweep` algorithm has no such built-in mechanism, and the resulting local optimum is more arbitrary.

The number of node checks is, as expected, far smaller for BUS than for `sweep` and `sweepCritical`, and higher for `sweepCritical` than for `sweep`.

Our experiments also confirm the statement in [8] that about three rounds of iterations of `sweep` are needed to reach a local optimum. The average values are 2.73 and 3.11 for networks of 1000 and 5000 nodes, respectively. Even for 5000 nodes, no more than four `sweep` rounds are needed for any instance. `SweepCritical` needs an average of 5.60 and 6.73 rounds for instances of 1000 and 5000 nodes, respectively, and the maximum numbers of rounds are 9 and 10, respectively.

5. Acknowledgments

The authors wish to thank the reviewer for the constructive comments. The work of the two first authors is supported by The Research Council of Norway under grant 160233/V30, and the work of the third author is supported by the Swedish Research Council under grant 621-2004-3902.

Table 2: Comparing BUS and sweep

	1000 nodes			5000 nodes		
	sweep	sweepCritical	BUS	sweep	sweepCritical	BUS
power	94.574	94.562	94.559	94.652	94.632	94.644
# arc exchanges	190.24	133.89	133.17	939.76	658.19	657.39
# node checks	2730.00	5600.00	660.86	15550.00	33650.00	3294.49

References

- [1] J. Bauer, D. Haugland, D. Yuan, New results on the time complexity and approximation ratio of the broadcast incremental power algorithm, submitted to *Information Processing Letters* (Revision submitted in August 2008).
- [2] M. Cagalj, J. Hubaux, C. Enz, Energy-efficient broadcasting in all-wireless networks, *Wireless Networks* 11 (2005) 177–188.
- [3] I. Kang, R. Poovendran, Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks, *Mobile Networks and Applications* 10 (2005) 879–896.
- [4] I. Kang, R. Poovendran, Iterated local optimization for minimum energy broadcast, in: *Proceedings of the 3rd IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2005, pp. 332–341.
- [5] R. Klasing, A. Navarra, A. Papadopoulos, S. Pérennes, Adaptive broadcast consumption (abc), a new heuristic and new bounds for the minimum energy broadcast routing problem, in: *NETWORKING*, vol. 3042 of Lecture Notes in Computer Science, Springer, 2004.
- [6] P. Mavinkurve, H. Ngo, H. Mehra, MIP3S: Algorithms for power-conserving multicasting in wireless ad hoc networks, in: *Proceedings of the 11th IEEE International Conference on Networks (ICON)*, 2003.
- [7] G. D. Nguyen, General algorithms for construction of broadcast and multicast trees with applications to wireless networks, *Journal of Communications and Networks* 7 (2005) 263–277.
- [8] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, Energy-efficient broadcast and multicast trees in wireless networks, *Mobile Networks and Applications* 7 (2002) 481–492.
- [9] D. Yuan, J. Bauer, D. Haugland, Minimum-energy broadcast and multicast in wireless networks: An integer programming approach and improved heuristic algorithms, *Ad Hoc Networks* 6 (2008) 696–717.