Minimum-Energy Broadcast and Multicast in Wireless Networks: An Integer Programming Approach and Improved Heuristic Algorithms

Di Yuan, Joanna Bauer and Dag Haugland

Minimum-Energy Broadcast and Multicast in Wireless Networks: An Integer Programming Approach and Improved Heuristic Algorithms

Di Yuan [a,*]

[a] *Department of Science and Technology, Linköping University, SE-601 74 Norrköping, Sweden*

Joanna Bauer [b]   Dag Haugland [b]

[b] *Department of Informatics, University of Bergen, PB. 7800, N-5020 Bergen, Norway*

Abstract

Both integer programming models and heuristic algorithms have been proposed for finding minimum-energy broadcast and multicast trees in wireless ad hoc networks. Among heuristic algorithms, the broadcast/multicast incremental power (BIP/MIP) algorithm is most known. The theoretical performance of BIP/MIP has been quantified in several studies. To assess the empirical performance of BIP/MIP and other heuristic algorithms, it is necessary to compute an optimal tree or a very good lower bound of the optimum. In this paper we present an integer programming approach as well as improved heuristic algorithms. Our integer programming approach comprises a novel integer model and a relaxation scheme. Unlike previously proposed models, the continuous relaxation of our model leads to a very sharp lower bound of the optimum. Our relaxation scheme allows for performance evaluation of heuristics without having to compute optimal trees. Our contributions to heuristic algorithms consist of the power-improving algorithm successive power adjustment (SPA), and improved time complexity of some previously suggested algorithms. We report extensive numerical experiments. Algorithm SPA finds better solutions in comparison to a host of other algorithms. Moreover, the integer programming approach shows that trees found by algorithm SPA are optimal or near-optimal.

Key words: Algorithm, Broadcast, Integer programming, Lagrangean relaxation, Minimum energy, Multicast, Performance evaluation, Wireless networks

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

1 Introduction

We study the problem of minimum-energy multicast and broadcast in wireless ad hoc networks. Multicast refers to the process of routing messages from a source node to a set of destination nodes. Broadcast is the special case of multicast where the set of destination nodes comprises all nodes other than the source.

In a wireless environment, broadcast is an inherent characteristic of signal propagation. As long as an omni-directional antenna is used, transmission power corresponds to coverage range in all directions. It is thus sufficient to transmit once in order to deliver a message to all devices within the range. In [34,35], this property is referred to as the "wireless multicast advantage". As a consequence of the property, power required to reach a set of devices is not the sum, but the maximum of the power for reaching all of them.

The total power requirement of a multicast session is the sum of the transmission power used by the source and the nodes involved in forwarding messages. Minimum-energy multicast refers to the problem of finding node transmission power such that the total power is minimized. In the subsequent text, we use MEMP to denote this problem. The problem of minimum-energy broadcast is a special case of MEMP. It has been proved that MEMP is *NP*-hard [5,13]. Wieselthier et al. [34,35,37] presented several heuristic algorithms for MEMP. Among these, one is referred to as the broadcast/multicast incremental power (BIP/MIP) algorithm. This is the most cited algorithm for MEMP.

Research on MEMP has been very active. Wan et al. [31–33] examined the performance ratio of several algorithms (including BIP and MIP) in Euclidean metric space. Analysis of approximating the optimal broadcast tree by minimum spanning tree (MST) are provided in [3,14,19,27]. Various algorithms for solving MEMP have been proposed in [4,5,7,10–12,15,19,20,22,24,26,28,33]. Some of these algorithms [4,5,7,10–12,19,20,22,24,26] are intended for solving the broadcast case of MEMP; others [15,28,33] deal with the more general case of multicast. For numerical evaluation of algorithm performance, BIP and MIP trees are often used as benchmark.

Another standpoint of studying MEMP is integer programming. Research following this direction is reported in [2,9]. The aspect of distributed computation has been addressed in, for example, [5,6,28,36], and many other aspects of energy-efficient broadcast are discussed in a number of surveys [17,21,30].

A generalization of MEMP is the use of directional antennae. Guo and Yang [15] proposed heuristic algorithms and an integer programming model for solving this generalization of MEMP. The authors of [8] provided a simulation study of several algorithms for MEMP with directional antennae.

In this paper we present an integer programming approach and new results on heuristic algorithms for MEMP. Our contributions to applying integer programming to MEMP are as follows.

- We present a novel integer programming model for MEMP. The most important feature of our model in comparison to previously suggested models is that its continuous relaxation gives a very sharp lower bound of the optimum. This feature is extremely useful for performance evaluation of heuristic algorithms when network size does not admit computing optimal trees.
- We propose to use Lagrangean relaxation and subgradient optimization to approximate the continuous relaxation of the integer model. This relaxation scheme allows us to conduct performance evaluation when it becomes time-consuming to solve the continuous relaxation.

We use the terms integer programming and integer programming approach in a broad sense. They are hence not limited to setting up and solving an integer model via a solver. By an integer programming approach, we mean any procedure that makes use of an integer programming model, such as a relaxation scheme that computes a good bound of the integer optimum. A number of previously proposed integer programming models [2,9,15,25] can be applied to MEMP. The main feature of our model is that it yields a very sharp lower bound of the optimum. Applying our Lagrangean relaxation scheme to the model, we are able to evaluate the performance of heuristics without having to compute optimal trees for networks of non-trivial size. Therefore our model is more powerful from the performance evaluation standpoint. In Section 4.2 we compare our model to those in [2,9,15,25] in more detail. In the same section we provide a numerical illustration of the strength of continuous relaxation in comparison to these models.

In addition to the integer programming approach, we contribute to heuristic algorithms for MEMP. These contributions are summarized below.

- We improve the time complexity of several known heuristics. Let N denote the number of nodes, the improved complexity results are as follows.

(1) BIP/MIP can be implemented to run in $O(N^2 \log N)$ time instead of the frequently quoted (e.g., [37]) $O(N^3)$ time.

(2) Wieselthier et al. [34,35,37] suggested a power-improving operation called sweep. Recently, the authors of [15,28,38] presented algorithms adopting some enhancements of sweep. We prove that, assuming node power levels are sorted, finding the best power-improving move of an enhanced version of sweep can be done in $O(N^2)$. Our analysis strengthens the complexity results of algorithm B in [28] by one magnitude, from $O(N^4)$ to $O(N^3)$.

(3) For omni-directional antennae, our complexity analysis of successive shrink (see below) improves the complexity of algorithm D-MIDP in [15] from

$O(N^3)$ to $O(N^2)$.

- We propose a power-improving operation, called successive shrink. This operation is a generalization of the algorithm in [11]. As will be clear in Section 6.2, successive shrink is a very natural complement to enhanced sweep in [28]. Furthermore, we prove that, once node power levels are sorted, finding the best power-improving move of successive shrink takes $O(N^2)$ for broadcast as well as the general multicast version of MEMP.
- We suggest to combine enhanced sweep and successive shrink. The combination results in a strong heuristic algorithm, which we refer to as successive power adjustment (SPA). Algorithm SPA has a complexity of $O(N^3)$.
- We present results of extensive computational experiments. In the experiments we examine the performance of SPA versus the previously suggested tree-improving algorithms sweep, 1-shrink [11], and B2 [28], based on trees constructed by BIP/MIP [34,35,37] and algorithm B and M [28]. We apply our integer programming approach to evaluate all these algorithms. Our experiments show that SPA finds lowest-power broadcast and multicast trees. Moreover, through the integer programming approach, we are able to show that the trees found by SPA are close to optimal.

The remainder of the paper is organized as follows. In Section 2, we give the network model and some notation. Section 3 is devoted to a review of the BIP and MIP algorithm, and the sweep operation [34,35,37]. Our integer programming model and the Lagrangean relaxation scheme are presented in Sections 4 and 5, respectively. In Section 6, we detail algorithm SPA. We present computational experiments in Section 7. In Section 8, we give some concluding remarks and propose future research directions.

2 Preliminaries

We use a set of nodes $V = \{1, \ldots, N\}$ to represent networking devices in a wireless network. The network itself is modeled using a graph $G = (V, A)$, where A is the set of (directed) potential communication links. Without loss of generality, we assume that G is complete, i.e., $A = \{(i, j) : i, j \in V, i \neq j\}$. Let p_{ij} denote the (minimum) power required at node i to reach node j. Typically, p_{ij} is a function of the distance between i and j. Denoting this distance by d_{ij}, a widely-used formula is $p_{ij} = \kappa d_{ij}^{\alpha}$, where α is an environment-dependent parameter (typically $2 \leq \alpha \leq 4$), and κ is a constant.

A multicast session is characterized by a source node $s \in V$ and a set of destination nodes $D \subset V$. Broadcast is the special case of multicast where $D = V \setminus \{s\}$. A destination node in D receives messages from s either directly, or via some other nodes. Connectivity between nodes is determined by transmission power. Due to the wireless multicast advantage, node i can, via a single transmission, send a message to all nodes $j \in V$ for which p_{ij} is less than or equal to the transmission power at i. Therefore, the power necessary to reach a set of nodes equals the maximum power needed to reach all of them. Problem MEMP amounts to determining transmission power of the nodes to enable message routing from s to all nodes in D, with the objective of minimizing the total power.

Remark 1. If power levels are not given in ascending order in the input data, we need a preprocessing step for sorting, which has a complexity of $O(N^2 \log N)$. In the paper we present complexity results for a number of algorithms. If sorting is necessary to obtain a complexity result, the associated $O(N^2 \log N)$ running time will be taken into account. □

It is easy to prove that an optimal solution to MEMP can be represented by a directed tree (i.e., an arborescence) rooted at s. For this reason, all heuristic algorithms for MEMP construct a solution having a tree structure. Throughout the paper, we use T as a general notation of a tree spanning V and rooted at s. Tree T is stored as a vector of length N. The ith element of this vector, $T(i)$, denotes the parent node of i. A multicast session uses a subset of the links of T. Links in this subset, and the end nodes of these links, are called active. Given D and T, the sets of active links and nodes can be uniquely determined. For broadcast, all nodes and links in T are active.

Although vector T is sufficient to represent a tree, our discussion of heuristic algorithms necessitates some other structures. These are presented in Table 1.

Table 1
Additional structures of tree T.

Notation	Meaning
U_T^i	The set of upstream nodes of node i. Node $j \in U_T^i$ if j is in the path from s to i.
C_T^i	The set of child nodes of i, that is, $C_T^i = \{j \in V : T(j) = i\}$.
S_T^i	The set of descendant nodes of i. Set S_T^i can be derived from U_T^i, as $S_T^i = \{j \in V : i \in U_T^j\}$.
A_T	The set of active nodes. Node i is active if it lies in the path between s and some destination in D. Specifically, $i \in A_T$ if $i \in D$ or $D \cap S_T^i \neq \emptyset$.
$P_T(i)$	Transmission power at node i. Node i is required to reach all of its active child nodes, hence $P_T(i) = \max_{j \in C_T^i \cap A_T} p_{ij}$.
c_T	Total power of tree T, that is, $c_T = \sum_{i \in V} P_T(i)$.

Remark 2. We use the concept of active nodes in computing node power for multicast, because this concept simplifies our analysis of time complexity. There are other ways to describe the process of calculating node power. In [28],

```
(1)  $V_T = \{s\}$; $T(i) = null\ \forall i \in V$; $P_T(i) = 0\ \forall i \in V$;
(2)  **while** $V_T \neq V$
(3)      $\delta^* = \infty$;
(4)      **for** all $i \in V_T$
(5)          **for** all $j \in V \setminus V_T$
(6)              **if** $p_{ij} - P_T(i) < \delta^*$
(7)                  $i^* = i$; $j^* = j$; $\delta^* = p_{ij} - P_T(i)$;
(8)      $V_T = V_T \cup \{j^*\}$; $T(j^*) = i^*$; $P_T(i^*) = \max\{p_{i^*j^*}, P_T(i^*)\}$;
(9)  Compute sets $A_T$ and $C_T^i\ \forall i \in V$;
(10) $P_T(i) = \max_{j \in C_T^i \cap A_T} p_{ij}\ \forall i \in V$;
(11) $c_T = \sum_{i \in V} P_T(i)$;
(12) **return** $T$ and $c_T$;
```

Fig. 1. Algorithms BIP and MIP.

for example, the author uses a general function to represent the evaluation of tree power. In [15,34,35,37], the corresponding procedure is called pruning. □

Remark 3. The structures in Table 1 are introduced for the purpose of making the presentation of algorithms precise and compact. A solution to MEMP, as the output of any heuristic algorithm, consists of the tree vector and its total power. The other structures are used in our presentation when necessary. In our discussion of algorithm implementation, many of the structures in Table 1 are updated implicitly to improve readability. □

3 BIP/MIP and Sweep

Wieselthier et al. [34,35] proposed several heuristic algorithms. One of them, developed for solving the broadcast case of MEMP, was named the broadcast incremental power (BIP) algorithm. For multicast, the corresponding algorithm is consistently called multicast incremental power (MIP). MIP is identical to BIP except that the former uses additional steps to calculate the power for multicast.

BIP/MIP is a tree-construction algorithm. It adapts the Prim's algorithm to MEMP. Starting from the source s, algorithm BIP/MIP builds up a tree by adding the node requiring a minimum amount of incremental power in every step. A formal description of both BIP and MIP is given in Figure 1. In the figure, V_T denotes the set of nodes that have been included in the tree.

The MIP algorithm comprises all steps in Figure 1, whereas the BIP algorithm does not contain Steps (9)–(10). These two steps calculate node power for a multicast session. The two steps correspond to a phase called pruning in [34,35,37]. A commonly cited running time of BIP and MIP is $O(N^3)$. Below we present an improved complexity result for BIP and MIP.

```
(1)  for all $i \in V$
(2)      $T' = T$;
(3)      for all $j \in V \setminus \{i\}$
(4)          if $j \notin U_T^i$ and $p_{ij} \leq P_T(i)$
(5)              $T'(j) = i$;
(6)      $P_{T'}(i) = \max_{j \in C_{T'}^i} p_{ij}\ \forall i \in V$; $c_{T'} = \sum_{i \in V} P_{T'}(i)$;
(7)      if $c_{T'} < c_T$
(8)          $T = T'$; $c_T = c_{T'}$;
(9)          Compute sets $U_T^n\ \forall n \in V$;
(10) return $T$ and $c_T$;
```

Fig. 2. Algorithm sweep.

Proposition 1. If node power levels are sorted in the input data, then BIP and MIP can be implemented to run in $O(N^2)$. Otherwise the complexity is $O(N^2 \log N)$.

Proof. See Appendix A. □

Analysis of the theoretical performance of BIP and MIP in Euclidean metric space has been conducted in a number of studies. Wan et al. [32] together with Klasing et al. [19] showed that the approximation ratio of BIP is in the range $[13/3, 12.15]$. Recently, Ambühl [3] improved the upper bound from 12.15 to 6. For MIP, Wan et al. [33] showed that the approximation ratio can be as bad as $N - 2 - O(1)$. Note that the results in [3,19] are derived for the MST heuristic, but these results apply to BIP as well due to a lemma in [32].

In addition to BIP, Wieselthier et al. [34,35,37] presented a power-improvement algorithm, called sweep, for the broadcast case of MEMP. The idea of sweep is as follows. At node i, the sweep algorithm finds out those nodes that lie within i's transmission range, but are currently not child nodes of i. Assigning such nodes (except those being upstream nodes of i) to be child nodes of i may result in power saving at the current parents of these nodes. Algorithm sweep is formalized in Figure 2. In the figure we use T' to denote a temporary tree being examined by sweep.

Remark 4. In our presentation of algorithms, we prioritize clarity over details of implementation, although the latter is crucial for algorithm complexity. In the appendices, we will detail the implementation of an algorithm when its complexity is being analyzed. □

Algorithm sweep goes through all nodes in V according to some order (e.g., the natural order 1, 2, ..., N). Applying sweep more than once may lead to further improvement. Typically, however, the improvement obtained from additional rounds of sweep is very small [34]. Moreover, algorithm sweep has a low time complexity of $O(N^2)$ [37]. In case any improvement is found in a round of sweep, then the power level of at least one node is reduced by at least one step. Because there are no more than N^2 power levels in total, running multiple rounds of sweep until no improvement can be found has a complexity of $O(N^4)$, which is much more expensive than the $O(N^2)$ complexity of sweep.

4 An Integer Programming Approach

4.1 The model

We use multi-commodity flow variables to represent paths being taken to reach destinations. A second set of variables is used to express transmission power. Below we give variable definitions as well as the model, denoted by MEMP-IP.

$$x_{ij}^d = \text{Flow from source to destination } d \text{ on link } (i,j).$$

$$z_{ij} = \begin{cases} 1 \text{ if the transmission power of node } i \text{ equals } p_{ij}, \\ 0 \text{ otherwise.} \end{cases}$$

$$\text{[MEMP-IP]} \quad P^* = \min \sum_{(i,j)\in A} p_{ij} z_{ij}$$

$$\text{s. t.} \sum_{j:(i,j)\in A} x_{ij}^d - \sum_{j:(j,i)\in A} x_{ji}^d = \begin{cases} 1 \text{ if } i = s \\ -1 \text{ if } i = d \\ 0 \text{ otherwise.} \end{cases} \quad \forall i \in V, \forall d \in D, \tag{1}$$

$$\sum_{j:(i,j)\in A} z_{ij} \leq 1 \quad \forall i \in V, \tag{2}$$

$$\sum_{k\in V:k\neq i, p_{ik}\geq p_{ij}} x_{ik}^d \leq \sum_{k\in V:k\neq i, p_{ik}\geq p_{ij}} z_{ik} \quad \forall (i,j) \in A, \forall d \in D, \tag{3}$$

$$x_{ij}^d \geq 0 \quad \forall (i,j) \in A, \forall d \in D, \tag{4}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A. \tag{5}$$

Constraints (1) ensure that every destination receives one unit of flow from the source. A solution in the x-variables is a so called multi-commodity flow, because flows to various destinations are indexed separately. The transmission power of a node should be zero or equal to that of one of the outgoing links (as otherwise the solution cannot be optimal). This observation leads to the definition of the z-variables, constraints (2), as well as the objective function.

Constraints (3) define the relationship between the two sets of variables. Note that for any $d \in D$, the flow variable x_{ij}^d indicates whether or not arc (i,j) is on the unique transmission path from s to d, and thus the left hand side of (3) cannot exceed one. Furthermore, $x_{ik} = 1$ for some k for which $p_{ik} \geq p_{ij}$ only if the power at i is at least p_{ij}. Since the latter condition is identical to $\sum_{k:p_{ik} \geq p_{ij}} z_{ik} = 1$, the constraints follow.

4.2 A comparison to some other models

Integer programming models have been presented for MEMP in several works [2,9,15]. It is therefore interesting to compare our model MEMP-IP to those suggested in these references. Das et al. [9] proposed three integer programming models. These models apply to both broadcast and multicast versions of MEMP. The first formulation is a hop-indexed model. The second formulation uses subtour elimination constraints. In the third formulation, connectivity is ensured by network flow equations. We are not aware of research work reporting implementation of the first two models. The third, flow-based model, on the other hand, has been extended to formulate MEMP with directional antennae by Guo and Yang [15], who also presented computational results of applying the model. We consider omni-directional antennae in MEMP, for which the model by Guo and Yang [15] reduces to the flow-based model by Das et al. [9].

Our model MEMP-IP and the flow-based model by Das et al. [9] (as well as the model in [15]) are similar in terms of the use of flow conservation equations. However, MEMP-IP differs from the model in [9,15] in several key aspects. The differences, which are presented below, significantly strengthen the continuous relaxation.

- Whereas the model in [9,15] uses single-commodity flow, MEMP-IP adopts a multi-commodity flow representation of a feasible solution.
- The flow model in [9,15] uses node power variables. Some constraints are used to connect these variables to link-selection variables. Our model describes node power in an implicit way by exploiting the fact that the optimal power of a node equals zero, or the power required by one of the outgoing links.
- MEMP-IP uses inequalities (3) to connect flow variables to power-selection variables. This representation of the relation of variables was not present in [9,15].

Integer programming has been applied to the symmetric power assignment problem (which is different from MEMP) by Montemanni and Gambardella in [25]. The authors proposed two models, among which a model based on network flow has a structure similar to that of MEMP-IP and the model by Das et al. [9]. It is not difficult to adapt this model in [25] to MEMP.

Concerning the strength of continuous relaxation, the model by Montemanni and Gambardella performs better than the one by Das et al., because the former uses the same representation of node power as in MEMP-IP. However, the continuous relaxation of the adapted model of [25] is still very weak in comparison to that of MEMP-IP.

In Table 2 we justify numerically the above claim regarding the sharpness of the lower bound obtained from the continuous relaxation of MEMP-IP. Table 2 uses three groups of 10-nodes, randomly generated networks to compare the continuous relaxations of our model, the flow-based model by Das et al. [9] (which is the same as the one by Guo and Yang for omni-directional antennae), and the model by Montemanni and Gambardella (adapted to MEMP). The power parameters are generated using $\alpha = 2$. There are 2, 5, and 9 destination nodes in the three groups, respectively. Each group contains 100 networks. The table shows values averaged over the networks in each of the groups. All values are normalized with respect to the integer optimum, of which the normalized value is always 100.

Table 2
A numerical comparison between continuous relaxations. (Integer optimum = 100.)

Network Group	Continuous Relaxations		
$(N, \|D\|)$	Model in [9,15]	Adapted Model of [25]	MEMP-IP
(10, 2)	29.84	43.93	99.95
(10, 5)	15.72	26.63	99.88
(10, 9)	10.56	18.33	99.79

From Table 2, it is clear that the continuous relaxation of the model in [9,15] gives a poor lower bound of the integer optimum – The former is less than 30% of the latter for multicast, and the value drops to 10% for broadcast. The model by Montemanni and Gambardella leads to a better continuous relaxation, which, however, is still too weak to be able to estimate the value of the integer optimum. Our model MEMP-IP, in contrast, results in a very sharp bound. Thus MEMP-IP outperforms considerably previously suggested models in the strength of continuous relaxation.

Remark 5. A strong continuous relaxation is a very important feature when integer optimum is out of reach. It should be pointed out that, in order to evaluate heuristics, *any* integer model can be used, as long as the model can be solved to integer optimality within reasonable time. (This is, for example, the case for the networks in Table 2.) However, since MEMP is NP-hard, any exact algorithm for MEMP, including solving an integer model by a solver, can be no faster than exponential, unless $P = NP$. Once finding the integer optimum is no longer computationally feasible due to network size, a model having a weak continuous relaxation is not useful for performance evaluation. In contrast, because the continuous relaxation of MEMP-IP is so tight, we can apply a (computationally efficient) relaxation scheme to obtain a very good lower bound of the optimum in order to examine the numerical performance of heuristics. Thus our model can be used for performance evaluation *without* computing the integer optimum, whereas the previously suggested model cannot. At present, optimal trees of MEMP can be computed for networks of 50 nodes or less. For larger networks, performance evaluation has been conducted between heuristics only (e.g., [15]). In the next section we present a Lagrangean relaxation scheme, which enables performance evaluation when network size does not admit computing optimal trees. The relaxation scheme allows us to show via computational experiments, that the algorithm we present in Section 6 finds close-to-optimal solutions for networks for which optimal trees cannot be obtained. We also remark that an interesting topic for further investigation is to generalize our model to handle MEMP with directional antennae. □

For the broadcast case of MEMP, the authors of [2] presented an integer programming model of set-covering type, and proposed a solution approach based on Lagrangean heuristic. In [2], some computational experiments have been conducted to compare this approach to solving the flow-based model of [9,15]. The results show that, within the same amount of computing time, the approach in [2] leads to better trees (when optimum could not be obtained because of network size).

The model in [2] does not use an explicit tree representation. Instead, the model relies on an implicit set that contains all spanning trees. A Lagrangean heuristic is a suitable approach for trees spanning N nodes (i.e., broadcast), because generating a tree is handled in one of the subproblems. More precisely, the heuristic involves solving a sequence of the minimum arborescence problem, which is polynomially solvable. For the multicast version of MEMP, however, this subproblem becomes NP-hard, as it turns out to be a type of prize-collecting Steiner tree problem. For this reason, it is difficult to apply the approach in [2] to the general case of MEMP.

5 A Lagrangean Relaxation Scheme

When there are many nodes and destinations, even the continuous relaxation of MEMP-IP becomes out of reach of a solver. To overcome this difficulty, we propose a Lagrangean relaxation scheme aimed at approximating efficiently the continuous relaxation. Our numerical results in Section 7 demonstrate that this scheme approximates the continuous relaxation very well, and thereby gives a high-quality bound of the integer optimum.

We relax the flow conservation constraints (1) using Lagrangean multipliers $\lambda_i^d, i \in V, d \in D$. For $\lambda = \bar{\lambda}$, the resulting relaxation reads

$$L(\bar{\lambda}) = \min \sum_{(i,j)\in A} p_{ij}z_{ij} + \sum_{d\in D}\sum_{i\in V} \bar{\lambda}_i^d \left(\sum_{j:(i,j)\in A} x_{ij}^d - \sum_{j:(j,i)\in A} x_{ji}^d \right) + \sum_{d\in D} (\bar{\lambda}_d^d - \bar{\lambda}_s^d)$$
$$\text{s. t. } (2), (3), (4), \text{ and } (5).$$

By defining $\bar{c}_{ij}^d = \bar{\lambda}_i^d - \bar{\lambda}_j^d$ for all $(i,j) \in A$, the relaxation can be stated as $L(\bar{\lambda}) = \sum_{d\in D} \bar{c}_{ds}^d + \min \sum_{(i,j)\in A} p_{ij}z_{ij} + \sum_{d\in D}\sum_{(i,j)\in A} \bar{c}_{ij}^d x_{ij}^d$, subject to (2), (3), (4), and (5). It is then clear that the relaxation decomposes by node. At node i the relaxation amounts to solving the following subproblem.

$$L_i(\bar{\lambda}) = \min \sum_{j:(i,j)\in A} p_{ij}z_{ij} + \sum_{d\in D}\sum_{j:(i,j)\in A} \bar{c}_{ij}^d x_{ij}^d \tag{6}$$
$$\text{s. t. } \sum_{j:(i,j)\in A} z_{ij} \le 1,$$
$$\sum_{k\in V:k\ne i, p_{ik}\ge p_{ij}} x_{ik}^d \le \sum_{k\in V:k\ne i, p_{ik}\ge p_{ij}} z_{ik} \quad \forall j:(i,j)\in A, \forall d \in D,$$
$$x_{ij}^d \in \{0,1\} \quad \forall j:(i,j)\in A, \forall d\in D,$$
$$z_{ij} \in \{0,1\} \quad \forall j:(i,j)\in A.$$

The above subproblem can be solved to optimality by a greedy-type algorithm. Because at most one of the z-variables can be one, we can compute $L_i(\bar{\lambda})$ by enumerating the set of possible power levels at node i, that is, setting $z_{ij} = 1$ for one j at a time and minimizing (6) in the flow variables. Due to the second set of constraints, at most one x-variable of any destination can be one. This set of constraints also enforces that, if $z_{ij} = 1$, then variable x_{ik}^d can be set to one only if $p_{ik} \le p_{ij}$. Among such flow variables, it is optimal to select the one having the most negative coefficient in the objective function. Performing this computation for all z-variables gives $L_i(\bar{\lambda})$. In Figure 3 we give a formal description of the algorithm for solving the Lagrangean relaxation. In the figure we use $\{\bar{x}, \bar{z}\}$ to denote an optimal solution for $\lambda = \bar{\lambda}$.

Proposition 2. The procedure in Figure 3 can be implemented to run in $O(N^2|D|)$, provided that node power levels are sorted.

Proof. To establish the result, observe first that Steps (8)–(12) are executed at most once for every $i \in V$. Among these steps, the most expensive one is (10), which is of $O(N)$. Since Step (10) repeats for every $d \in D$, the complexity of Steps (8)–(12), for all $i \in V$, is of $O(N^2|D|)$. Next, note that none of (1) or (6) is the computational bottleneck. Therefore, what remains to be analyzed is the for-loop (4). Assume that the nodes in $V \setminus \{i\}$ are processed in the order $j_1, j_2, \ldots, j_{N-1}$ such that $p_{ij_1} \le p_{ij_2} \le \ldots \le p_{ij_{N-1}}$. Denoting the result of the inner minimization in Step (5) by $q_{ij}^d = \min_{k\in V\setminus\{i\}, p_{ik}\le p_{ij}} \bar{c}_{ik}^d$, Step (5) amounts to computing $q_{ij_l}^d = \min\{q_{ij_{l-1}}^d, \bar{c}_{ij_l}^d\}$ for every $d \in D$, followed by computing

```
(1)  $L(\bar{\lambda}) = \sum_{d \in D} \bar{c}^d_{ds}$; $\bar{z}_{ij} = 0\ \forall (i,j) \in A$; $\bar{x}^d_{ij} = 0\ \forall (i,j) \in A, \forall d \in D$;
(2)  for all $i \in V$
(3)      $L_i(\bar{\lambda}) = 0$;
(4)      for all $j \in V \setminus \{i\}$
(5)          $\hat{p}_{ij} = p_{ij} + \sum_{d \in D} \min\{0, \min_{k \in V: k \neq i, p_{ik} \leq p_{ij}} \bar{c}^d_{ik}\}$;
(6)      Find $j^* \in \text{argmin}_{j \in V: j \neq i}\ \hat{p}_{ij}$;
(7)      if $\hat{p}_{ij^*} < 0$
(8)          $\bar{z}_{ij^*} = 1$; $L_i(\bar{\lambda}) = \hat{p}_{ij^*}$; $L(\bar{\lambda}) = L(\bar{\lambda}) + L_i(\bar{\lambda})$;
(9)          for all $d \in D$
(10)             Find $k^* \in \text{argmin}_{k \in V: k \neq i, p_{ik} \leq p_{ij^*}} \bar{c}^d_{ik}$
(11)             if $\bar{c}^d_{ik^*} < 0$
(12)                 $\bar{x}^d_{ik^*} = 1$;
(13) Return $L(\bar{\lambda})$, $\bar{x}$, and $\bar{z}$;
```

Fig. 3. The procedure for solving the Lagrangean relaxation.

$\hat{p}_{ij_l} = p_{ij_l} + \sum_{d \in D} \min\{0, q^d_{ij_{l-1}}\}$. Hence Step (5) runs in $O(|D|)$ times, and the result follows. □

For arbitrary values of λ, $L(\lambda)$ is a lower bound of the integer optimum. To obtain a best possible bound, we would like to maximize $L(\lambda)$ over λ (i.e., to solve the Lagrangean dual). Note that, because the integrality requirement does not affect $L(\lambda)$, the maximum of $L(\lambda)$ equals the optimum of the continuous relaxation.

We use a subgradient optimization procedure to maximize $L(\lambda)$. A subgradient of $L(\lambda)$ at $\bar{\lambda}$ is $\bar{\xi} = \{\bar{\xi}^d_i, i \in V, d \in D\}$, where

$$\bar{\xi}^d_i = \sum_{j:(i,j) \in A} \bar{x}^d_{ij} - \sum_{j:(j,i) \in A} \bar{x}^d_{ji} - b^d_i. \tag{7}$$

In (7), b^d_i is 1 if $i = s$, -1 if $i = d$, and 0 otherwise. The Lagrangean multipliers are then updated by taking a step in the direction of the subgradient:

$$\bar{\lambda}' = \bar{\lambda} + \gamma \frac{\bar{P} - L(\bar{\lambda})}{||\bar{\xi}||^2} \bar{\xi}. \tag{8}$$

In (8), the step size, $\gamma(\bar{P} - L(\bar{\lambda}))/||\bar{\xi}||^2$, follows a commonly-used formula (e.g., [1]). Here, $\bar{P}$ is an upper bound of the optimum of the Lagrangean dual, and γ is between zero and two.

Since subgradient optimization converges asymptotically, the standard stopping criterion is a maximum allowed number of iterations. We use RELAX to denote the algorithm of solving Lagrangean relaxation repeatedly in subgradient optimization. Algorithm RELAX delivers a lower bound of the integer optimum. We formalize algorithm RELAX in Figure 4. In the figure, R denotes

```
(1) $l = 0$; $\lambda_i^d = 0 \; \forall i \in V, \forall d \in D$; $L^* = 0$;
(2) while $l < R$
(3)     Solve the Lagrangean relaxation $L(\lambda)$.
(4)     if $L^* < L(\lambda)$
(5)         $L^* = L(\lambda)$;
(6)     Compute a subgradient according to equation (7).
(7)     Update $\lambda$ according to equation (8).
(8)     $l = l + 1$;
(9) return $L^*$;
```

Fig. 4. Algorithm RELAX.

the (pre-defined) number of iterations in subgradient optimization.

It follows from equations (7)-(8) and Proposition 2 that one iteration of RELAX runs in $O(N^2|D|)$ time. Since sorting is required prior to running algorithm RELAX, the overall algorithm complexity is of $O(N^2 \times \max(\log N, |D|))$.

Remark 6. To obtain convergence, R may have to be large. However, because R is a pre-defined constant, it is not involved in the analysis of algorithm complexity. The values of R used in our computational experiments are specified in Section 7. □

To achieve good convergence in subgradient optimization, $\bar{P}$ should be close to the optimum of the Lagrangean dual. If $\bar{P}$ equals this optimum, γ should be set to one. In our implementation, $\bar{P}$ is the total transmission power of the solution found by the MIP algorithm. The initial value of γ is set to 1.0, and the final value is 0.001. We reduce the value of γ in each subgradient iteration by a constant scaling factor, which is derived from the initial and final values of γ, and the number of subgradient iterations R. Computational experiments show that L^* found by our subgradient optimization procedure almost equals the optimum of the continuous relaxation of MEMP-IP.

6 A Strong Heuristic Algorithm for MEMP

6.1 Enhanced sweep

We discuss two enhancements that can be used to improve the sweep algorithm. Throughout the rest of the paper, we refer to sweep with the two enhancements as enhanced sweep.

The first enhancement is the use of a search space (for tree construction or power improvement) in which a node may increase its transmission power to generate test trees. Raising the power at node i, the node can reach additional nodes. If these nodes become child nodes of i, the total power reduction at their ex-parent nodes may pay off the extra power required at i. Clearly, improvement of this type is not possible in the original sweep algorithm.

Several previously suggested algorithms adopt the idea to examine whether the total power can be reduced by allowing one node to increase its power. For broadcast, increasing the power of a node is used in the local search heuristic presented by Kang and Poovendran in [18], two algorithms (named B and B2) proposed by Nguyen in [28], as well as the algorithm presented by Yuan in [38]. The idea is also useful for multicast (to which algorithm B2 in [28] is applicable).

In Figure 5 we illustrate the first enhancement for broadcast in a small network of 10 nodes. In this example node 7 is the source. Figure 5(a) shows the initial tree. Two examples of increasing node power are shown in Figures 5(b) and 5(c), respectively. In Figure 5(b), node 7 increases its power to reach node 4. As a result, five nodes, 2, 3, 6, 9, and 10, can reduce their power. In Figure 5(c), node 3 extends its transmission range to cover node 8. Power saving occurs at nodes 2, 6, and 7.

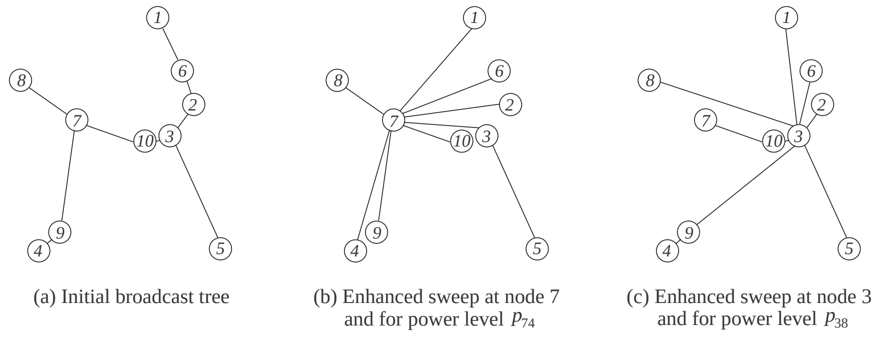
1
6
8
2
7
10
3
9
4
5
(a) Initial broadcast tree
(b) Enhanced sweep at node 7 and for power level p_{74}
(c) Enhanced sweep at node 3 and for power level p_{38}

Fig. 5. Examples of the first enhancement of sweep.

The second enhancement of sweep addresses multicast. An algorithm adopting this enhancement keeps a tree spanning all N nodes, but evaluates the tree power based on the multicast group, i.e., computes the power necessary to reach nodes in D. This idea has been discussed by Guo and Yang [15] in their algorithm, called D-MIDP, for MEMP with directional antennae.

The original sweep evaluates tree power under the broadcast assumption. Pruning, which correctly determines the power for multicast, is invoked after sweep in [34]. Thus a straightforward way to implement the second enhancement is to embed the MIP Steps (9)–(10) in Figure 1 into a tree-search procedure. This strategy is used by algorithm D-MIDP in [15].

In Figure 6 we formalize the procedure of applying one round of enhanced sweep (i.e., examining all nodes and power levels), and choosing the *best power-improving* enhanced sweep to obtain a new tree. In the figure, T' is a temporary

```
(1)  $T^* = T$; $c_{T^*} = c_T$;
(2)  for all $i \in V$
(3)      for all $j \in V \setminus \{i\}$
(4)          $T' = T$; $P_{T'} = P_T$;
(5)          for all $k \in V \setminus \{i\}$
(6)              if $p_{ik} \leq p_{ij}$ and $k \notin U_T^i$
(7)                  $T'(k) = i$;
(8)          Compute sets $A_{T'}$ and $C_{T'}^n$, $\forall n \in V$;
(9)          $P_{T'}(n) = \max_{m \in C_{T'}^n \cap A_{T'}} p_{nm}$ $\forall n \in V$; $c_{T'} = \sum_{n \in V} P_{T'}(n)$;
(10)         if $c_{T'} < c_{T^*}$
(11)             $T^* = T'$; $c_{T^*} = c_{T'}$;
(12) return $T^*$ and $c_{T^*}$;
```

Fig. 6. The procedure of computing the best power-improving enhanced sweep.

tree under evaluation, and the tree returned by the procedure is denoted by T^*.

Remark 7. Algorithms B and B2 in [28] are mainly intended for the broadcast case of MEMP. Algorithm B is not a tree-improvement algorithm, but a tree-construction algorithm. In addition, algorithm B uses the strategy of selecting the best power-improving move (obtained from sweep with the first enhancement). Algorithm B2, on the other hand, performs a tree update as soon as a power-improving move is detected. Applying algorithm B2 to multicast involves computing the total power of multicast trees. When generating a test tree, it may happen that a node does not have any destination as descendant, or its most power-demanding child node does not lead to any destination. In such cases computing the correct power level corresponds to pruning. In this sense the second enhancement is also present in algorithm B2. We also remark that the D-MIDP algorithm in [15] does not use the first enhancement (which is not directly applicable to directional antennas), but a one-link exchange strategy to search in the solution space. □

When it comes to time complexity, the procedure in Figure 6 gives an impression of $O(N^3)$. This complexity has been used to derive the overall complexity of algorithm B in [28]. Our contribution to enhanced sweep is a new complexity result. Provided that node power levels are sorted (see Remark 1, Section 2), there is a faster, $O(N^2)$ implementation of the procedure in Figure 6.

Proposition 3. Finding the best power-improving move of enhanced sweep can be implemented to run in $O(N^2)$ for broadcast and multicast.

Proof See Appendix B. □

The above $O(N^2)$ result leads to an improvement of the complexity of algorithm B in [28]. The currently known complexity result of this algorithm is $O(N^4)$. If the $O(N^2)$ implementation of enhanced sweep is used, the complexity of algorithm B is reduced to $O(N^3)$.

Corollary 4. Algorithm B in [28] can be implemented to run in $O(N^3)$.

Proof Algorithm B constructs a tree successively from scratch. Assume that the current tree spans n_k nodes. To expand the tree, algorithm B applies enhanced sweep to all the n_k nodes. At every node, the algorithm considers power levels greater than or equal to the current power, and generates new trees using enhanced sweep. Among the new trees, algorithm B selects the one having minimum power and including at least one more destination in comparison to the current tree. The algorithm stops when the tree includes all destination nodes. Assume that the numbers of nodes in the sequence of trees constructed by algorithm B are n_1, n_2, $\ldots$, n_k, $\ldots$, n_{k^*}. In Appendix B, we prove that at one node, finding the best power-improving enhanced sweep can be implemented to run in $O(N)$. Therefore, for a tree containing n_k nodes, the complexity of constructing the next tree in the sequence is of $O(N) \times n_k$. Consequently, the overall complexity is $O(N) \times (n_1 + n_2 + \ldots + n_k + \ldots + n_{k^*})$ $\leq O(N) \times \sum_{n=1}^{N} n = O(N^3)$. The $O(N^3)$ result is obviously not affected if node power levels have to be sorted in preprocessing, because sorting has a lower complexity. $\square$

6.2 Successive shrink

Enhanced sweep searches for power saving by actively increasing node power. An alternative search strategy is to decrease the power of a node, and assign those child nodes that become disconnected due to power reduction to some new parent nodes. For the broadcast case of MEMP, the authors of [11] presented a concept called r-shrink, and developed an algorithm for $r = 1$ (i.e., 1-shrink). This 1-shrink algorithm has been embedded into a simulated annealing heuristic in [26]. In 1-shrink, the most power-demanding child of a node is moved to become a child of another node.

We propose a power-improving operation, which we refer to as successive shrink, for both *broadcast and multicast* versions of MEMP. Successive shrink does not perform its shrink operation for any particular r, but moves all child nodes of a node, one by one, to new parents. Given a tree T, successive shrink steps though the nodes one by one, and processes the child nodes of each node. Consider node i, for which the current set of child nodes is $C_i = \{j_1, j_2, \ldots, j_{|C_i|}\}$. Assume that the nodes in C_i are sorted such that $p_{ij_1} \leq p_{ij_2} \leq \ldots \leq p_{ij_{|C_i|}}$. Successive shrink removes node $j_{|C_i|}$ from C_i and assigns a new parent to $j_{|C_i|}$, such that the power increment at the new parent is minimized. All nodes, except i and descendants of $j_{|C_i|}$, are candidate parents. After the new parent has been found, i reduces its power. Next, a new parent is found in the same manner for node $j_{|C_i|-1}$. This process is repeated until j_1 is assigned to a new parent. At this stage i becomes a leaf and its power is reduced to zero. Note, that after r child nodes have been moved, the change in total power can be positive, zero, or negative. When processing the child nodes of a node, successive shrink does not stop even if the total power becomes temporarily worse, because an improvement may be obtained at a later stage.

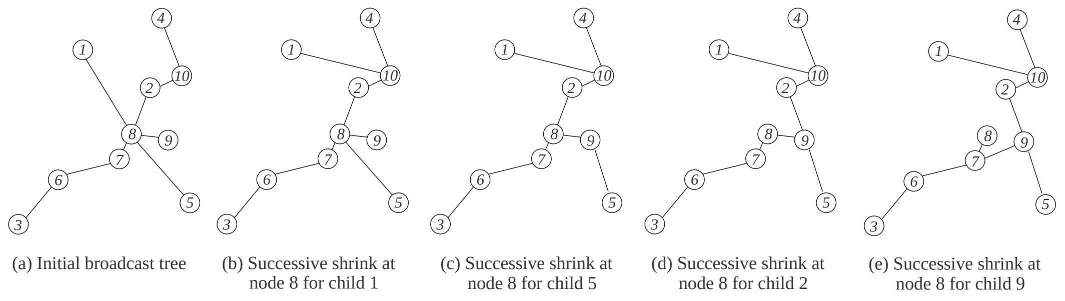
(a) Initial broadcast tree
(b) Successive shrink at node 8 for child 1
(c) Successive shrink at node 8 for child 5
(d) Successive shrink at node 8 for child 2
(e) Successive shrink at node 8 for child 9

Fig. 7. Examples of successive shrink.

In Figure 7 we illustrate successive shrink at node 8 of the depicted network. Node 6 is the source. The initial broadcast tree is shown in Figure 7a, and the child nodes of 8 are moved one by one in Figures 7(b) to 7(e). In this process, the total power can increase as well as decrease relative to the initial tree. For example, in the successive shrink operation for obtaining the tree in Figure 7(b), the power reduction at node 8 is smaller than the power increment at node 10. So the total power increases. The tree in Figure 7(d), on the other hand, improves the initial tree in total power.

At every node, successive shrink processes its child nodes starting from the initial tree T (i.e., T is not updated even though some power-improving moves have been found). We suggest to use the strategy of updating T by the *best power-improving* move of successive shrink. Thus the output is the minimum-power tree found after all nodes and their child nodes have been processed.

Remark 8. Successive shrink is a natural complement to enhanced sweep. In enhanced sweep, a node can become the new parent of multiple nodes. The original parents of these nodes are typically different. Successive shrink acts conversely, i.e., to let the child nodes of a node go to multiple, different parents. A tree returned by successive shrink can seemingly be found using a sequence of enhanced sweep. However, this would require enhanced sweep to accept non-improving moves. Note that enhanced sweep is not designed to accept a tree having worse total power (nor does any previously suggested algorithms using enhanced sweep). Thus a tree generated from successive shrink may not be obtained by any sequence of power-improving enhanced sweep. Consider node i and its most power-demanding child node $j_{|C_i|}$. In successive shrink, assigning a new parent node, say k, to $j_{|C_i|}$ is often equivalent to an enhanced

```
(1)  $T^* = T$; $c_{T^*} = c_T$;
(2)  **for** all $i \in V$
(3)      $T' = T$; $P_{T'} = P_T$;
(4)      Compute sets $S^n_{T'}$ and $C^n_{T'}$ $\forall n \in V$;
(5)      **while** $C^i_{T'} \neq \emptyset$
(6)          Find $k \in \text{argmax}_{n \in C^i_{T'}} p_{in}$;
(7)          $\tilde{T}^*(i) = null$ $\forall i \in V$; $c_{\tilde{T}^*} = \infty$;
(8)          **for** all $j \in V \setminus \{i,k\} \setminus S^k_{T'}$
(9)              $\tilde{T} = T'$; $\tilde{T}(k) = j$;
(10)             Compute sets $A_{\tilde{T}}$ and $C^n_{\tilde{T}}$ $\forall n \in V$;
(11)             $P_{\tilde{T}}(n) = \max_{m \in C^n_{\tilde{T}} \cap A_{\tilde{T}}} p_{nm}$ $\forall n \in V$; $c_{\tilde{T}} = \sum_{n \in V} P_{\tilde{T}}(n)$;
(12)             **if** $c_{\tilde{T}} < c_{\tilde{T}^*}$
(13)                 $\tilde{T}^* = \tilde{T}$; $c_{\tilde{T}^*} = c_{\tilde{T}}$;
(14)         $T' = \tilde{T}^*$; $c_{T'} = c_{\tilde{T}^*}$;
(15)     **if** $c_{T'} < c_{T^*}$
(16)         $T^* = T'$; $c_{T^*} = c_{T'}$;
(17)     Compute sets $S^n_{T'}$ and $C^n_{T'}$ $\forall n \in V$;
(18) **return** $T^*$ and $c_{T^*}$;
```

Fig. 8. The procedure of computing the best power-improving successive shrink.

sweep move at k. However, if the total power becomes worse after assigning $j_{|C_i|}$ to k, then this assignment will be excluded from enhanced sweep. Successive shrink, on the other hand, will (tentatively) accept the new tree, and continue moving the remaining child nodes of i. We also remark that successive shrink, like enhanced sweep, is a generalization of sweep, as any update that can be achieved in sweep is among the moves considered by successive shrink. □

We formalize the procedure of finding the best power-improving move of successive shrink in Figure 8. In the algorithm description, T' is the minimum-power tree after all child nodes of a node have been processed. (At each node, T' is reset to be the initial tree T before the child nodes are processed.) When processing a child, $\tilde{T}$ is used to denote the tree resulting from moving this child to a new trial parent, and $\tilde{T}^*$ is the minimum-power tree obtained in all trials. Finally, tree T^* is the output.

At a first glance, successive shrink, as shown in Figure 8, appears to be computationally expensive. But we can prove that a fast implementation runs in $O(N^2)$. Thus successive shrink has the same complexity as enhanced sweep.

Proposition 5. Provided that node power levels are sorted, finding the best power-improving move of successive shrink can be implemented to run in $O(N^2)$ for broadcast and multicast.

Proof See Appendix C. □

D-MIDP in [15] is an $O(N^3)$ tree-improving algorithm for MEMP with di-

```
(1)  $T^* = T$; $c_{T^*} = c_T$; $l = 0$; status $= true$;
(2)  **while** $l < N$ and status $= true$
(3)      $(T^*_E, c_{T^*_E}) =$ EnhancedSweep$(T^*)$;
(4)      $(T^*_S, c_{T^*_S}) =$ SuccessiveShrink$(T^*)$;
(5)      **if** $c_{T^*_E} < c_{T^*}$ and $c_{T^*_E} < c_{T^*_S}$
(6)          $(T^*, c_{T^*}) = (T^*_E, c_{T^*_E})$;
(7)      **else if** $c_{T^*_S} < c_{T^*}$ and $c_{T^*_S} < c_{T^*_E}$
(8)          $(T^*, c_{T^*}) = (T^*_S, c_{T^*_S})$;
(9)      **else**
(10)         status $= false$;
(11)     $l = l + 1$;
(12) **return** $T^*$ and $c_{T^*}$;
```

Fig. 9. Algorithm SPA.

rectional antennae. The algorithm uses a one-link exchange strategy in its search for new trees. For omni-directional antennae, successive shrink generalizes this one-link exchange strategy. Proposition 5 leads to the following result for algorithm D-MIDP.

Corollary 6. Given an initial tree, algorithm D-MIDP [15] runs in $O(N^2)$ for broadcast and multicast using omni-directional antennae.

Proof See Appendix D. □

Note that in the proof of Corollary 6, we do not need to assume that power levels are sorted. As a consequence of the corollary, for omni-directional antennae, the computational bottleneck in D-MIDP is not its power-improving operation, but rather the procedure used to construct the initial tree, because the latter often has a higher complexity than $O(N^2)$.

6.3 Successive power adjustment

We propose a new tree-improving heuristic algorithm. The algorithm, which we refer to as successive power adjustment (SPA), makes use of both enhanced sweep and successive shrink. The algorithm selects, in every iteration, the best of the trees returned by enhanced sweep and successive shrink. This is repeated for at most N iterations. In Figure 9 we summarize algorithm SPA.

Remark 9. Algorithm SPA has a structure that is not shown in the figure, where the algorithm picks the best tree found by two separate (and quite different) procedures. Note that enhanced sweep at a node can be (and should be) implemented as to set its tentative power to maximum, followed by decreasing the tentative power successively down to the current level. (In fact, this implementation is crucial for proving Proposition 3.) Successive shrink at a node decreases the power from its current level step by step to zero.

Thus a time-efficient implementation of SPA amounts to, at every node, going through all power levels from maximum to zero, giving arise to the name of the algorithm. This implementation merges enhanced sweep and successive shrink at each node. □

Corollary 7. Algorithm SPA runs in $O(N^3)$.

Proof The result follows directly from the time complexity of sorting node power levels ($O(N^2 \log N)$), Propositions 3 and 5, and Figure 9. □

7 Computational Experiments

We use networks of 10, 20, 50, and 100 nodes in the numerical experiments. The number of destination nodes varies from a few (small multicast group) to $N-1$ (broadcast). For every combination of N and $|D|$, 100 networks have been generated using the instance-generation procedure suggested in [34,35]. For each network, we use two sets of transmission power, for $\alpha = 2$ and $\alpha = 4$, respectively. The total number of networks in the experiments is 2,800.

We compare algorithm SPA to a host of heuristic algorithms. For performance evaluation, we use either the optimum or algorithm RELAX. The latter is used if obtaining optimal trees by CPLEX [16] is not computationally feasible. Tables 3-4 give an overview of the heuristic algorithms used in the computational experiments. In each table, algorithms for tree construction and tree improvement are summarized in the upper right and lower left parts, respectively. The tables also show the time complexity of these algorithms. The lower right parts of the tables summarize combinations of algorithms.

Table 3
A summary of the heuristic algorithms for minimum-energy broadcast.

		Tree-Construction Algorithms	
	Algorithm	BIP [34,35,37]	B [28]
	Complexity	$O(N^2 \log N)$	$O(N^3)$
Tree-Improvement Algorithms			
Algorithm	Complexity	Combinations	
Sweep [34,35,37]	$O(N^2)$	BIP + sweep	–
1-shrink [11]	$O(N^4)$	BIP + 1-shrink	–
B2 [28]	$O(N^3)$	BIP + B2	B + B2
SPA	$O(N^3)$	BIP + SPA	B + SPA

Remark 10. Algorithm B2 uses enhanced sweep in its search for better trees. In this algorithm, the tree is updated as soon as a power-improving move

Table 4
A summary of the heuristic algorithms for minimum-energy multicast.

Tree-Construction Algorithms

Algorithm	MIP [34,35,37]	M [28]
Complexity	$O(N^2 \log N)$	$O(N^4) \times \lvert D \rvert$

Tree-Improvement Algorithms

Algorithm	Complexity
Sweep [34,35,37]	$O(N^2)$
B2 [28]	$O(N^3)$
SPA	$O(N^3)$

Combinations

MIP + sweep	M + sweep
MIP + B2	M + B2
MIP + SPA	M + SPA

of enhanced sweep is found. In [28], algorithm B2 was intended mainly for solving minimum-energy broadcast. The author of [28] indicated that algorithm B2 can be adapted to multicast. We have implemented this idea in our computational study, so the algorithm applies to broadcast and multicast. Algorithm M, also presented in [28], is developed specifically for multicast. In one iteration, algorithm M uses a path (consisting of one or multiple links) in order to augment a tree to include at least one more destination and possibly some non-destination nodes. The set of candidate paths contains the shortest paths between nodes, as well as some additional paths derived from the shortest ones. In [28], it has been shown that algorithm M outperforms MIP with sweep. On the other hand, the time complexity of algorithm M is high. □

Remark 11. We do not report results of combining algorithm B with sweep or 1-shrink, because sweep and 1-shrink can rarely improve a tree constructed by algorithm B. For multicast, sweep is not performed at all nodes, but those included in the multicast trees, i.e., trees constructed by MIP (after pruning) or by algorithm M. □

Remark 12. The authors of [11] did not discuss the complexity of 1-shrink. In [11], the suggested implementation stops when no improvement is possible using 1-shrink. This strategy has been used in our implementation as well. From the discussion in [11], it can be easily derived that one round of 1-shrink runs in $O(N^2)$. (One round of 1-shrink means to examine every node once, and assign a new parent to the most power-demanding child node if this leads to an improvement in total power.) The $O(N^4)$ overall complexity follows then from the fact that there will never be more than N^2 rounds. (See the discussion of sweep at the end of Section 3.) However, we note that reducing the maximum allowed number of rounds (e.g., to N) has no impact on algorithm performance in practice. Therefore the same numerical results of 1-shrink can be obtained with a complexity no higher than $O(N^3)$. □

In our experiments, all test networks of 10 and 20 nodes could be solved to optimality by CPLEX. For 50-nodes networks, instances with few destinations ($|D| \leq 10$) permit exact solutions. For the rest of the test networks (more specifically, networks with $N = 50$ and $|D| \geq 25$, and networks with $N = 100$), the solver did not manage to find the optimum, and therefore we use the lower bound provided by algorithm RELAX for performance evaluation. This lower bound tells how much the solutions from the heuristic algorithms deviate from optimum *in the worst case*. We applied algorithm RELAX to small networks as well (for which optimal trees have been computed), in order to examine the quality of the lower bound. The numbers of subgradient iterations in algorithm RELAX are 2,000, 5,000, 10,000, and 50,000 for networks of 10, 20, 50, and 100 nodes, respectively.

In Tables 5 and 6 we present the computational results. A row in the tables shows results averaged over 100 networks. We present the performance of the continuous relaxation of MEMP-IP and algorithm RELAX (both in comparison to optimum) under columns 'LP' and 'RELAX', respectively. For the results of heuristic algorithms, those that are best in performance are highlighted in bold. Examining the results leads to the following observations.

Table 5
Performance evaluation for minimum-energy broadcast.

N	α	LP	RELAX	BIP	BIP +				B	B +	
					Sweep	1-shrink	B2	SPA		B2	SPA
		Deviation from optimum (%)									
10	2	0.21	0.22	13.87	6.74	4.77	1.32	0.68	1.66	0.90	**0.66**
20	2	1.89	1.92	23.93	17.11	13.74	6.45	**1.90**	5.46	4.07	2.90
10	4	0.08	0.09	5.37	1.85	1.39	0.44	**0.14**	0.20	**0.14**	**0.14**
20	4	0.38	0.47	6.42	3.32	2.40	1.71	1.04	1.82	1.00	**0.80**
		Deviation from bound by RELAX (%) (Pessimistic estimation of deviation from optimum)									
50	2	–	–	35.91	28.20	24.64	19.22	**13.26**	17.56	15.91	14.59
100	2	–	–	45.09	36.66	32.74	28.34	21.52	25.37	23.27	**21.45**
50	4	–	–	10.92	7.29	6.40	5.62	**5.14**	6.40	5.72	5.56
100	4	–	–	11.30	7.83	6.68	6.49	**5.52**	6.90	6.03	5.81

- For tree construction, algorithms B and M have significantly better performance than BIP and MIP for broadcast and multicast, respectively. The difference in performance is particularly evident for multicast, for which the trees constructed by MIP are often more than 20% away from optimum, whereas for algorithm M, the deviation from optimum is in most cases less than 10%. (This achievement comes, however, at a cost of considerably higher time complexity.)
- For tree-improvement, algorithm SPA has the best performance. For broadcast, sweep and 1-shrink improve considerably the trees constructed by BIP.

Table 6
Performance evaluation for minimum-energy multicast.

$(N, \lvert D\rvert)$	α	LP	RELAX	MIP	MIP +			M	M +		
					Sweep	B2	SPA		Sweep	B2	SPA
		Deviation from optimum (%)									
(10, 2)	2	0.05	0.06	7.84	5.95	0.41	**0.13**	0.75	0.63	0.49	0.47
(10, 5)	2	0.12	0.12	10.68	6.16	0.59	**0.26**	1.41	1.20	0.40	0.29
(20, 5)	2	0.35	0.38	16.82	13.14	5.15	1.72	3.45	3.23	2.20	**1.62**
(20, 10)	2	1.08	1.09	22.78	16.41	5.91	**1.73**	5.26	5.21	3.98	2.75
(50, 5)	2	1.38	1.43	28.71	25.62	11.16	4.51	5.82	5.62	4.02	**2.99**
(50, 10)	2	3.15	3.18	28.01	23.20	9.93	**3.87**	6.31	5.96	5.09	4.31
(10, 2)	4	0.00	0.02	4.66	3.45	0.26	**0.00**	0.42	0.07	0.07	0.07
(10, 5)	4	0.04	0.08	5.15	2.62	0.23	0.18	0.70	0.67	0.18	**0.02**
(20, 5)	4	0.10	0.12	4.68	3.18	1.06	0.63	1.11	1.04	0.65	**0.48**
(20, 10)	4	0.31	0.36	7.43	4.70	2.12	1.37	2.28	2.25	1.37	**1.15**
(50, 5)	4	0.21	0.25	13.04	11.76	5.46	5.14	2.74	2.57	1.98	**1.87**
(50, 10)	4	0.68	0.72	8.76	6.55	3.63	2.63	1.71	1.62	1.31	**1.27**
		Deviation from bound by RELAX (%) (Pessimistic estimation of deviation from optimum)									
(50, 25)	2	–	–	31.95	25.59	16.63	**10.79**	13.57	13.37	12.64	11.71
(100, 5)	2	–	–	39.27	37.18	16.07	9.43	7.31	7.12	6.13	**5.51**
(100, 10)	2	–	–	41.24	37.99	22.07	13.91	12.82	12.65	11.38	**10.26**
(100, 50)	2	–	–	41.77	34.94	26.87	20.37	21.75	21.57	20.41	**19.11**
(50, 25)	4	–	–	9.24	6.67	4.95	4.17	4.07	3.95	3.35	**3.05**
(100, 5)	4	–	–	16.14	15.16	8.71	7.15	3.40	3.15	2.53	**2.17**
(100, 10)	4	–	–	12.58	10.93	6.90	5.30	3.63	3.41	3.08	**2.85**
(100, 50)	4	–	–	10.93	7.95	5.93	5.19	5.64	5.50	4.77	**4.41**

Among these two algorithms, 1-shrink performs better, but the difference is fairly small. For multicast, only small improvement can be obtained using sweep. Both sweep and 1-shrink are outperformed significantly by algorithms B2 and SPA. Comparing further algorithm B2 to algorithm SPA (both having a complexity of $O(N^3)$), the latter yields noticeably better solutions for $\alpha = 2$. For $\alpha = 4$, the difference in performance is small.

- The solutions found by algorithm SPA are near-optimal. In performance evaluation using optimal trees, algorithm SPA produces solutions that deviate from optimum by a few percent or less. For networks where the bound from algorithm RELAX is used for evaluating performance, trees from algorithm SPA are no more than 10% above optimum in most cases, and about 20% in two cases. Note that performance evaluation using the bound by RELAX gives a pessimistic estimation of the deviation from optimum (i.e., the true deviation is smaller).

- One interesting observation is, that for algorithm SPA, best solutions are sometimes found from the trees of BIP/MIP, and sometimes from those constructed by algorithms B and M. Such a behavior is quite common when applying a search heuristic to an NP-hard problem – The final solution is more dependent on the initial solution than the quality of this initial solution. Overall, results from BIP/MIP and SPA together are comparable to those found by combining algorithm B and M to SPA. Algorithm M has a much higher complexity ($O(N^4|D|)$) than MIP's $O(N^2 \log N)$ and SPA's $O(N^3)$, thus algorithms MIP and SPA form a cheaper yet effective approach in comparison to algorithm M (or algorithms M and SPA).
- The continuous relaxation of the integer programming model MEMP-IP gives a very tight lower bound. The deviation from integer optimum is less than 2% in most cases. Furthermore, the bound from algorithm RELAX is almost identical to that of the continuous relaxation. In conclusion, our integer programming approach is effective for evaluating heuristic algorithms.
- A large value of α gives more rapid growth in transmission power with respect to distance, making it easier to identify and exclude non-optimal links. We can therefore expect better results when $\alpha = 4$. This is indeed the case – All algorithms perform considerably better when $\alpha = 4$.

Table 7
A comparison between enhanced sweep (ES) and successive shrink (SuS).

$(N, \|D\|)$	α	BIP/MIP +		B/M +	
		ES	SuS	ES	SuS
(50, 25)	2	**11.69** (87)	17.99 (13)	12.37 (18)	12.02 (25)
(50, 49)	2	**14.03** (89)	22.39 (11)	15.67 (10)	15.02 (30)
(100, 50)	2	21.49 (89)	26.92 (11)	20.13 (31)	**19.90** (40)
(100, 99)	2	22.63 (83)	29.81 (17)	22.68 (22)	**22.44** (42)
(50, 25)	4	4.22 (26)	4.63 (9)	**3.25** (5)	3.30 (10)
(50, 49)	4	**5.24** (33)	5.96 (11)	5.72 (2)	5.59 (9)
(100, 50)	4	5.40 (52)	5.82 (19)	4.67 (8)	**4.51** (20)
(100, 99)	4	**5.63** (52)	6.27 (15)	5.94 (7)	5.88 (15)

In Table 7 we provide results of additional experiments to show the performance of enhanced sweep and successive shrink. The experiments are made for some networks of 50 and 100 nodes. In the experiments we use either enhanced sweep or successive shrink (but not both). In addition to the percentage deviation from the bound by RELAX (as in Tables 5 and 6), Table 7 shows the numbers of networks (among a total of 100) for which a procedure performs better than the other within parentheses. The lowest percentage number of each row is highlighted in bold. We observe that none of two procedures outperforms the other for all network groups. For initial trees constructed by BIP and MIP, the average performance of enhanced sweep is consistently better than that of successive shrink. The latter finds however trees of lower power in some cases. When initial trees come from algorithms B and M, successive shrink performs slightly better for all but one network group.

8 Conclusions

We have presented some advances in computationally approaching optimal or near-optimal solutions to minimum-energy broadcast and multicast in wireless networks. Our computational machinery consists of a novel integer programming model (MEMP-IP), a bounding algorithm (RELAX), and a strong heuristic algorithm (SPA). In addition we have strengthened complexity results of some previously proposed algorithms.

The integer programming model is very useful for performance evaluation of heuristic algorithms, particularly because its continuous relaxation yields a very sharp bound. Moreover, this bound can be approximated effectively and efficiently by algorithm RELAX. Algorithm SPA outperforms a host of other algorithms in the computational experiments. For networks used in the experiments, the solutions found by SPA are near-optimal. Furthermore, the running time of algorithm SPA is comparable to that of some earlier algorithms. Finally, we would like to remark that the research presented in this paper may be of interest in studying problems related to MEMP, such as source-independent broadcast over a single tree [29].

Acknowledgments

We are thankful to two anonymous reviewers for providing us with comments and suggestions, as well as very useful references. Their comments and suggestions have significantly improved the technical content as well as the presentation of the paper. One of the reviewers suggested to study algorithm complexity. Our work following this suggestion has led to several of the complexity results in the paper. We would also like to acknowledge that the work of the first author is supported by the Swedish Research Council (Vetenskapsrådet) under grant 621-2004-3902, and the work of the second and third author is supported by the Research Council of Norway under contract 160233/V30.

References

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows, Theory, Algorithms, and Applications, Princeton Hall, 1993.

[2] K. Altinkemer, F. S. Salman, P. Bellur, Solving the minimum energy broadcasting problem in ad hoc wireless networks by integer programming, in: Proceedings of the second workshop on modeling and optimization in mobile, ad hoc, and wireless Networks (WiOpt '04), April 2004, pp. 48–54.

[3] C. Ambühl, An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks, in: L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi and M. Yung (eds.), Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05), Lecture Notes in Computer Science 3580, 2005, pp. 1139–1150.

[4] M. Čagalj, J.-P. Hubaux, Energy-efficient broadcasting in all-wireless networks, Mobile Networks and Applications (MONET), in press.

[5] M. Čagalj, J.-P. Hubaux, C. Enz, Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues, in: Proceedings of ACM MobiCom '02, September 2002, pp. 172–182.

[6] J. Cartigny, D. Simplot, I. Stojmenovic, Localized minimum-energy broadcasting in ad-hoc networks, in: Proceedings of IEEE INFOCOM '03, April 2003, pp. 2210–2217.

[7] T. Chu, I. Nikoladis, Energy efficient broadcast in mobile ad hoc networks, in: Proceedings of the First International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW), September 2002, pp. 177–190.

[8] F. Dai, Q. Dai, J. Wu, Power efficient routing trees for ad hoc wireless networks using directional antenna, Ad Hoc Networks 3 (2005) 621–628.

[9] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, A. Gray, Minimum power broadcast trees for wireless networks: integer programming formulations, Proceedings of IEEE INFOCOM '03, April 2003, pp. 1001–1110.

[10] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, A. Gray, A cluster-merge algorithm for solving the minimum power broadcast problem in large scale wireless networks, in: Proceedings of IEEE MILCOM '03, October 2003, pp. 416–421.

[11] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, A. Gray, r-Shrink: A heuristic for improving minimum power broadcast trees in wireless networks, in: Proceedings of IEEE GLOBECOM '03, December 2003, pp. 523–527.

[12] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, A. Gray, e-Merge: A Heuristic for improving minimum power broadcast trees in wireless networks, Technical Report, Department of Electrical Engineering, University of Washington, WA, 2003.

[13] Ömer Eğecioğlu, T. F. Gonzalez, Minimum-energy broadcast in simple graphs with limited node power, Proceedings of IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS) '01, August 2001, pp. 334–338.

[14] M. Flammini, R. Klasing, A. Navarra, S. Pérennes, Improved approximation results for the minimum energy broadcasting problem, in: Proceedings of the Joint Workshop on Foundations of Mobile Computing, 2004.

[15] S. G. Guo, O. Yang, Minimum-energy multicast in wireless ad hoc networks with adaptive antennas: MILP formulations and heuristic algorithms, IEEE Transactions on Mobile Computing 5 (2006) 333–346.

[16] ILOG, Ilog CPLEX 7.0, user's manual, August 2000.

[17] F. Ingelrest, D. Simplot, I. Stojmenovic, Energy-efficient broadcasting in wireless mobile ad hoc networks, in: M. Cardei, I. Cardei, D. Zhu (Eds.), Resource Management in Wireless Networking, Kluwer Academic Publishers, 2005, pp. 543–582.

[18] I. Kang, R. Poovendran, Iterated local optimization of minimum energy broadcast, Proceedings of the third international symposium on modeling and optimization in mobile, ad hoc, and wireless Networks (WiOpt '05), April 2005, pp. 332–342.

[19] 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 N. Mitrou, K. Kontovasilis, G. Rouskas, I. Iliadis, L. Merakos (eds.), Networking 2004, Lecture Notes in Computer Science 2042, 2004, pp. 866–877.

[20] W. Liang, Constructing minimum-energy broadcast trees in wireless ad hoc networks, in: Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC '02), June 2002, pp. 112–122.

[21] X. Y. Li, I. Stojmenovic, Broadcasting and topology control in wireless ad hoc networks, in: A. Boukerche, I. Chlamtac (Eds.), Handbook of Algorithms for Mobile and Wireless Networking and Computing, CRC Press, in press.

[22] F. Li, I. Nikolaidis, On minimum-energy broadcasting in all-wireless networks, in: Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN '01), November 2001, pp. 193–202.

[23] S. Lindsey, C. S. Raghavendra, Energy efficient all-to-all broadcasting for situation awareness in wireless ad hoc networks, Journal of Parallel and Distributed Computing 63 (2003) 15–21.

[24] R. J. Marks, A. K. Das, M. El-Sharkawi, P. Arabshahi, A. Gray, Minimum power broadcast trees for wireless networks: optimizing using the viability lemma, Proceedings of the NASA Earth Science Technology Conference, June 2002, pp. 273–276.

[25] R. Montemanni, L. M. Gambardella, Exact algorithms for the minimum power symmetric connectivity problem in wireless networks, Computers & Operations Research, in press.

[26] R. Montemanni, L. M. Gambardella, A. K. Das, The minimum power broadcast problem in wireless networks: a simulated annealing approach, Technical Report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Manno, Switzerland, 2004.

[27] A. Navarra, Tighter bounds for the minimum energy broadcasting problem, in: Proceedings of Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless (WiOpt 2005), 2005, pp. 313–322.

[28] 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.

[29] I. Papadimitriou, L. Georgiadis, Minimum-energy broadcasting in multi-hop wireless networks using a single broadcast tree, Mobile Networks and Applications, 11 (2006) 361–375.

[30] A. Pelc, Broadcasting in wireless networks, in: Handbook of wireless networks and mobile computing, I. Stojmenovic (Ed.), John Wiley and Sons, Inc., New York, 2002, pp. 509–528.

[31] P.-J. Wan, G. Călinescu, X.-Y. Li, O. Frieder, Minimum-energy broadcast routing in static ad hoc wireless networks, in: Proceedings of IEEE INFOCOM '01, April 2001, pp. 1162–1171.

[32] P.-J. Wan, G. Călinescu, X.-Y. Li, O. Frieder, Minimum-energy broadcast routing in static ad hoc wireless networks, Wireless Networks 8 (2002) 607–617.

[33] P.-J. Wan, G. Călinescu, C.-W. Yi, Minimum-power multicast routing in static ad hoc wireless networks, IEEE/ACM Transactions on Networking 12 (2004) 507–514.

[34] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, On the construction of energy-efficient broadcast and multicast trees in wireless networks, in: Proceedings of IEEE INFOCOM '00, March 2000, pp. 585–594.

[35] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, Algorithms for energy-efficient multicasting in static ad hoc wireless networks, Mobile Networks and Applications 6 (2001) 251–263.

[36] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, Distributed algorithms for energy-efficient broadcasting in ad hoc networks, in: Proceedings of IEEE MILCOM '02, October 2002, pp. 820–825.

[37] 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.

[38] D. Yuan, Computing optimal or near-optimal trees for minimum-energy broadcasting in wireless networks, Proceedings of the third international symposium on modeling and optimization in mobile, ad hoc, and wireless Networks (WiOpt '05), April 2005, pp. 323–331.

Appendix A

In Figure 10, we present an $O(N^2)$ implementation for constructing a broadcast tree in BIP/MIP. This implementation requires that power levels are sorted in the input data. In the figure, j^i_k denotes the kth node in the sorted power sequence at node i, i.e., $p_{ij^i_1} \leq p_{ij^i_2} \leq \ldots \leq p_{ij^i_k} \leq \ldots \leq p_{ij^i_{N-1}}$.

The underlying idea of the implementation is to, for the sorted sequence at each node, keep track on the index corresponding to the current node power. In the figure, these indices are stored in vector K. To augment the tree, the algorithm starts from the current index of every node (if the node is in the tree), and moves forward, until a node that is not part of the tree is encountered.

```
 (1) $T(i) = null\ \forall i \in V$; $P_T(i) = 0\ \forall i \in V$; $K(i) = 0, i = 1, \ldots, N-1$; $l = 1$;
 (2) while $l < N$
 (3)     $\delta^* = \infty$;
 (4)     for all $i \in V$
 (5)        if $T(i) \neq null$ or $i = s$
 (6)           $j = j^i_{K(i)}$;
 (7)           while $T(j) \neq null$ or $j = s$
 (8)              $K(i) = K(i) + 1$; $j = j^i_{K(i)}$;
 (9)           if $p_{ij} - P_T(i) < \delta^*$
(10)              $i^* = i$; $j^* = j$; $k^* = K(i)$; $\delta^* = p_{ij} - P_T(i)$;
(11)     $T(j^*) = i^*$; $P_T(i^*) = p_{i^*j^*}$; $K(i^*) = k^*$; $l = l + 1$;
(12) return $T$;
```

Fig. 10. An $O(N^2)$ implementation for constructing a broadcast tree in BIP/MIP.

It is obvious that, except the while-loop (7), all the computational steps run in $O(1)$. The purpose of the while-loop is to find the first non-tree node in the sorted sequence (of node i). Note that the value of $K(i)$ of any i is monotonously non-decreasing, and the sorted sequence at any node has $N-1$ elements in total. So the overall complexity of this loop is of $O(N^2)$ for tree construction. Thus the procedure in Figure 10 runs in $O(N^2)$.

Once the broadcast tree has been constructed, computing its total power for broadcast as well as multicast runs in $O(N)$ [28]. Therefore the overall time complexity of BIP/MIP is of $O(N^2)$. However, if sorting has to be carried out prior to running BIP/MIP, the complexity becomes $O(N^2 \log N)$.

Appendix B

We show that finding the best power-improving move of enhanced sweep can be implemented to run in $O(N)$ at one node. Proposition 3 follows then immediately.

Given a tree T, consider enhanced sweep at node i. To obtain the $O(N)$ running time, the power of node i is first increased to the maximum, and then successively reduced. Setting the power to maximum lets node i take over all nodes (except the upstream nodes of i) as its new child nodes. The new child nodes are then moved, one by one in descending order of their power requirement at i, back to their *original* parent nodes in tree T. Along with moving child nodes from node i, the set of active nodes and tree power are updated accordingly. Algorithm implementation is given in Figure 11. The trial tree is denoted by T'. The set of active nodes is represented by binary vector $A_{T'}$ (i.e., $A_{T'}(j) = true$ if node j is active, otherwise $A_{T'}(j) = false$). Let m be the most power-demanding and active child of node i in tree T. Vector M contains nodes in $V \setminus \{i\}$ that require larger power than that for m. Nodes in M are sorted in ascending order by power. Vector M is empty (i.e., its length is zero) if node i does not have any active child in T. To save space, for some computations in Figure 11 we have used verbal description instead of pseudo-code, as long as the time complexity of the computations is obvious and does not affect the overall complexity. Also, to ease the presentation, we assume that $p_{i,M(0)} = 0$.

```
(1)  $T^* = T; c_{T^*} = c_T; j^* = null; T' = T;$
(2)  Update $T'$: All nodes, except upstream nodes of $i$, become child nodes of $i$;
(3)  Compute vectors $M$, $A_{T'}$, and $P_{T'}$;
(4)  $l = |M|$; $P_{T'}(i) = p_{i,M(l)}$; $c_{T'} = \sum_{j \in V} P_{T'}(j)$;
(5)  while $l \geq 0$
(6)      $j = M(l)$;
(7)      $n = T(j)$; $T'(j) = n$; $c_{T'} = c_{T'} - (P_{T'}(i) - p_{i,M(l-1)})$;
(8)      if $A_{T'}(j) = true$
(9)          repeat
(10)             $A_{T'}(n) = true$;
(11)             if $p_{nj} > P_{T'}(n)$
(12)                 $c_{T'} = c_{T'} + (p_{nj} - P_{T'}(n))$; $P_{T'}(n) = p_{nj}$;
(13)             $j = n$; $n = T'(n)$;
(14)         until $A_{T'}(n) = true$
(15)     if $c_{T'} < c_{T^*}$
(16)         $j^* = j$; $c_{T^*} = c_{T'}$;
(17)     $l = l - 1$;
(18) if $j^* \neq null$
(19)     Construct tree $T^*$ from $T$ and $j^*$;
(20) return $T^*$ and $c_{T^*}$;
```

Fig. 11. Finding the best power-improving move of enhanced sweep at node i in $O(N)$ time.

In Steps (2)-(4), all nodes, except the upstream nodes of node i (and i itself), become child nodes of i. Vectors M, $P_{T'}$, and $A_{T'}$ are constructed accordingly, and the total power is computed. Note that the power of node i is set to reach the last node in list M, no matter this node is active or not. There is no doubt that Steps (2)-(4) run in $O(N)$. In the while-loop starting at Step (5), all nodes in list M are moved back, one after another, to their original parents in the initial tree T. The power of node i decreases by one step in each of the move operations. When node j is moved to become a child node of n (its original parent in T), the power of node n is updated if node j is active. In this case node n is marked as active in Step (10). Moreover, if n was inactive before Step (10), then some of its upstream nodes may be inactive as well, and the state and power of these nodes must be updated. This job is done in the repeat-loop starting at (9).

Tree T^* and parameter j^* are updated in Step (16) in case of improvement. Note that the tree vector T' is not saved (which would require $O(N)$ in complexity) in this step. Instead, the tree vector is constructed later (outside the while-loop) from T and j^* in Step (19). Moreover, it is worth remarking that when all elements of M have been moved, the check performed in Step (15) corresponds to the (original) sweep operation.

From the discussion above, it is clear that the implementation in Figure 11 runs in $O(N)$ if this complexity holds for the while-loop (5). Within this loop, all the computational operations run in $O(1)$. So the only possible obstacle is the repeat-loop (9). For node j, the operations within the repeat-loop may have to be executed multiple times. However, note that the state of at least one node is set to active within the loop, and the loop ends as soon as an active node is encountered. Because there are N nodes, the operations within the loop require no more than $O(N)$ in total. In conclusion, the overall time complexity is of $O(N)$.

For a tree that does not span all nodes in D, a slight modification of the implementation in Figure 11 can be used to find the best move of enhanced sweep that *augments* the tree to include at least one additional destination. The modification involves stopping moving nodes if M no longer contain any *non-tree* destination, or, equivalently, to redefine the content of M. Also, the trial trees should not be compared to the initial tree, but among themselves (i.e., to set $c_{T^*} = \infty$ in Step (1)). The modifications do not change the $O(N)$ complexity. Therefore, for a partial tree T with n_T nodes, tree augmentation using the best move of enhanced sweep runs in $O(N) \times n_T$.

Appendix C

We prove that finding the best power-improving move of successive shrink can be implemented to run in $O(N^2)$. We use T and T' to denote the initial tree and a trial tree, respectively. Tree T' is stored as a vector and as adjacency lists. The latter enables us to update the parent of a node in $O(1)$ and to perform depth-first search (DFS) in $O(N)$ time. In the presentation of the $O(N^2)$ implementation, the adjacency lists are updated implicitly with every update of the vector T' to improve readability.

Our $O(N^2)$ implementation of successive shrink uses some structures and quantities that have not been defined earlier. For convenience, in Table 8 we summarize the key structures and quantities (including those defined previously).

Table 8
Structures and quantities related to tree T'.

Notation	Meaning
$P_{T'}$	A vector containing node transmission power.
$c_{T'}$	Total power used by tree T'.
$A_{T'}$	A binary vector indicating node state (active or inactive).
j_k^i	The kth node in the sorted power sequence at node i, i.e., $p_{ij_1^i} \leq p_{ij_2^i} \leq \ldots \leq p_{ij_k^i} \leq \ldots \leq p_{ij_{N-1}^i}$.
Eligible	A binary vector of length N. A node is eligible if it can be a candidate parent. Consider successive shrink at node i and a new parent is to be found for i's child node j, then all nodes but i, j and the descendants of j are eligible.
Induced	A vector of length N containing induced power. At an active node, the induced power is zero. At an inactive node, induced power is the total additional power needed at its upstream nodes if this node becomes active (i.e., has some destination as descendant). Consider an inactive node i. Starting from i and moving upward to source s, let the sequence of nodes being visited be $i, a, b, c, d, \ldots, s$. Without any loss of generality, assume that d is the first active node in the sequence. The induced power at i equals $p_{ai} + p_{ba} + p_{cb} + \max\{0, p_{dc} - P_{T'}(d)\}$.

For multicast, the notion of induced power is crucial to obtain the $O(N^2)$ running time. Once vector *Induced* has been computed, evaluating a candidate parent runs in $O(1)$.

Figure 12 outlines the $O(N^2)$ implementation. The computational operations are specified by verbal description in the outline. Later we will discuss some of the computations in more detail to analyze time complexity. For convenience, the (suggested) time complexity of some computations is given in brackets.

The steps within the for-loop (2) perform successive shrink at node i. In Step (3), the trial tree T' is reset to be the initial tree T. The child nodes of node i are then moved one by one to new parent nodes in the for-loop (4). This loop goes through all nodes in $V \setminus \{i\}$, in descending order of their power requirement at node i. Due to the check in Step (6), however, only child nodes of i will be considered for the shrinking operation.

The sequence of computations in successive shrink for a child node j is as follows. First, vector *Eligible* is computed. Node j is then detached from node i in Step (8). (After this step, T' is a forest of two trees, one rooted at source s, and the other rooted at node j.) Vectors $A_{T'}$ and *Induced* are then computed.

```
(1)  [$O(N)$] $T^* = T; c_{T^*} = c_T$;
(2)           for all $i \in V$
(3)  [$O(N)$]   $T' = T$;
(4)             for $k = N-1:1$
(5)  [$O(1)$]     $j = j_k^i$;
(6)  [$O(1)$]     if $T'(j) = i$
(7)  [$O(N)$]       Compute vector $Eligible$;
(8)  [$O(N)$]       $T'(j) = null$;
(9)  [$O(N)$]       Compute vector $A_{T'}$;
(10) [$O(N)$]       Compute vector $P_{T'}$;
(11) [$O(N)$]       Compute vector $Induced$;
(12) [$O(1)$]       $\delta^* = \infty$;
(13)                for every eligible node $n$
(14) [$O(1)$]         if $Induced(n) + \max\{p_{nj} - P_{T'}(n), 0\} < \delta^*$
(15) [$O(1)$]           $\delta^* = Induced(n) + \max\{p_{nj} - P_{T'}(n), 0\}$; $n^* = n$;
(16) [$O(1)$]       $T'(j) = n^*$;
(17) [$O(N)$]       Compute vector $A_{T'}$;
(18) [$O(N)$]       Compute vector $P_{T'}$;
(19) [$O(N)$]       $c_{T'} = \sum_{n \in V} P_{T'}(n)$;
(20) [$O(1)$]       if $c_{T'} < c_{T^*}$
(21) [$O(N)$]         $T^* = T'$; $c_{T^*} = c_{T'}$;
(22) return $T^*$ and $c_{T^*}$;
```

Fig. 12. Finding the best power-improving move of successive shrink in $O(N^2)$ time.

Next, all nodes that are eligible to be the new parent of node j are examined in the for-loop (13). At each eligible node, the power increment if node j (i.e., the tree rooted at node j) becomes a child is calculated. The node at which the power increment is minimum is selected as the new parent. The resulting tree is evaluated against the best tree found so far.

If node j is inactive, then the power increment is zero no matter which node becomes its new parent. In our implementation, the new parent node is selected as if node j is active. Doing so minimizes the additional power required at upstream nodes of j, when later on j is considered to be a candidate parent of another child node of i.

The for-loops (2) and (4) in combination with Step (6) define a loop over all links in T'. Thus the steps within the for-loop (2) are executed at most $N-1$ times in total. As a consequence, the implementation runs in $O(N^2)$, provided that the time complexity given in the figure for every individual step is correct. For some of the computational steps in Figure 12, the time complexity is obvious. Below we discuss those computational steps for which the time complexity is less straightforward.

- Step (7): First, all nodes are marked eligible. Then nodes i, j, and descendants of j are marked non-eligible via a DFS of the tree rooted at j.
- Step (8): In this step, j is detached from i, and updating the adjacency lists

accordingly runs in $O(N)$.

- Step (9): Initiate $A_{T'}(n)$ to $false$ for all $n \in V$. We then start from each $d \in D$, set $A_{T'}(d) = true$, and traverse upward in T' (i.e., visit d's upstream nodes one by one). As long as the process of traversing nodes encounters an inactive node n, it sets $A_{T'}(n) = true$. The process stops as soon as it encounters an active node (because all upstream nodes of this node must have been marked active previously), or reaches source s, or reaches node j. Note the similarity between this process and the way of marking active nodes in enhanced sweep, and consequently Step (9) runs in $O(N)$.
- Step (11): Only the tree rooted at source s needs to be considered in this step, because nodes in the tree rooted at j are non-eligible. Starting from s, a DFS is performed, and the vector *Induced* is computed alongside performing the DFS.

Appendix D

The one-link exchange strategy in D-MIDP amounts to removing a tree link and moving the child node that becomes disconnected from the tree to a new parent node. The new parent node is selected such that the total tree power is minimized. When this one-link exchange strategy is applied to omni-directional antennae, it suffices to examine, at each node, the link connecting the node to its most power-demanding active child. (One-link exchange for other links will obviously not lead to any power improvement.)

Successive shrink can be adapted to implement the one-link exchange strategy. For each $i \in V$, the adapted procedure goes through all child nodes of i to find the one that is active and most power-demanding. The search runs in $O(N)$ (without assuming that node powers are sorted). For this particular child, the steps within the if-statement (6) in Figure 12 are executed to find the new parent node. Hence the $O(N^2)$ overall time complexity.