# General Algorithms for Construction of Broadcast and Multicast Trees with Applications to Wireless Networks

Gam D. Nguyen

*Abstract:* **In this paper, we introduce algorithms for constructing broadcasting and multicasting trees. These algorithms are general because they may be used for tree cost functions that are of arbitrary form. Thus, essentially the same algorithmic procedures are used for different tree cost functions. We evaluate the effectiveness of the general algorithms by applying them to different cost functions that are often used to model wired and wireless networks. Besides providing a unifying framework for dealing with many present and future tree-construction applications, these algorithms typically outperform some existing algorithms that are specifically designed for energy-aware wireless networks. These general algorithms perform well at the expense of higher computational complexity. They are centralized algorithms, requiring the full network information for tree construction. Thus, we also present variations of these general algorithms to yield other algorithms that have lower complexity and distributed implementation.**

*Index Terms:* **Arbitrary cost function, broadcast, multicast, tree, wireless networks.**

## I. INTRODUCTION

Structures called trees are fundamental and important because they are required in many applications. For a given tree, we can assign a number that represents the cost of using this tree. The goal of a tree-construction algorithm is to produce a tree that includes a given set of vertices and has the least cost. With traditional approaches, whenever the cost function changes, the algorithm also changes to incorporate and reflect the newer cost definition. In other words, new cost functions require new research efforts to design new algorithms. For example, great research efforts are exerted on the development of tree-construction algorithms for wired communication networks, in which link costs reflect the properties of wired networks. For different types of networks, such as wireless networks, it is expected that research is required to develop newer algorithms that exploit the wireless network properties, resulting in better performance than algorithms previously designed for wired networks. Are there general algorithms (or at least a general framework) that are applicable to any cost functions? In this paper, we introduce such algorithms.

Specifically, we first present an algorithm for constructing broadcast trees, which is general because it may be used for cost functions that are of arbitrary form. That is, we impose no restrictions whatsoever on the cost functions (with the exception, of course, that the costs are non-negative). We then modify the general broadcast algorithm to yield a general multicast algorithm, which is also for cost functions that are of arbitrary form. Whenever a problem arises and seeks to optimize a new tree cost function, which can be of any form, a version of our general algorithms can be used as a heuristic baseline solution to the problem. This baseline solution then can be compared to solutions of other algorithms that are designed to solve the same problem.

We evaluate the effectiveness of the general algorithms by applying them to different cost functions that are often used to model wired and wireless networks, and by comparing their performance to some existing algorithms. For wireless multihop networks (also called infrastructureless, peer-to-peer, or ad hoc wireless networks), we consider tree-construction algorithms for the use of omnidirectional and directional antennas, and we also address the impact of constraints on resources such as energy, transceivers, and frequencies [1]. Besides providing a unifying framework for dealing with many tree-construction applications, these general algorithms typically outperform some other existing algorithms that are specifically designed for energy-aware wireless networks. These general algorithms, which are of centralized nature, perform well at the cost of higher computation complexity. Thus, we also adapt these algorithms to yield algorithms that have lower complexity and distributed implementation. The preliminary version of this paper is [2].

The rest of this paper is organized as follows. In Section II, we first review basic terminologies for trees. We then discuss the cost definitions used in wired and wireless network problems. These important problems will serve as example applications of our general algorithms. In Section III, we state the general algorithm for broadcasting (Algorithm B), and its corresponding multicasting version (Algorithm M). We then in Section IV apply the general algorithms to the problems in wireless networks. In Section V, we present 2 algorithms (Algorithm B1 and Algorithm B2) that have lower complexity than Algorithm B. One algorithm is a distributed algorithm, the other is a centralized one. We summarize the paper in Section VI.

## II. DEFINITIONS OF COST FUNCTIONS

Let $V = \{1, 2, \cdots, N\}$ be the set of vertices, and let $S$ be a subset of $V$. Here we consider trees that must include $S$ and may include vertices in $V - S$. Assume that for a given tree $T$, we can assign a cost $d(T) \geq 0$ (no other restrictions are imposed on the cost function). Denote $d_{ij}$ as the cost of the tree consisting of only 2 vertices $i$ and $j$. This simple tree is also called the edge $(i, j)$, i.e., $d_{ij}$ is the cost of the edge $(i, j)$. Note that $d_{ij}$ may or may not be equal to $d_{ji}$. A tree has a root vertex, other vertices are called its descendants. It is desirable to design an algorithm for constructing a tree $T$ that includes the set $S$, such that the cost $d(T)$ is minimum.

Manuscript received June 21, 2003; approved for publication by Elvino S. Sousa, Division II Editor, November 12, 2004.

The author is with the Information Technology Division, Naval Research Laboratory, Washington DC, 20375, USA, email: nguyen@itd.nrl.navy.mil.

In this paper, we also use terminology from the network design viewpoint. Thus, a node is identified with a vertex, and a link with an edge. The subset $S$ is called the multicast group, which consists of the root vertex (also called the source node) and other vertices (also called the destination nodes). The problem of constructing trees including the nodes of $S$ is called the multicast problem. In particular, the problem is called unicast when $|S| = 2$, and broadcast when $|S| = N$. In summary, we can restate the abstract tree-construction problem in terms of network design applications as follows. Let $V = \{1, 2, \cdots, N\}$ be the nodes of a network, and $S$ be a subset of $V$. Suppose that a member $r$ of $S$ wants to communicate (e.g., to send a message) to all other members of $S$. We can accomplish this communication by constructing a tree $T$ that includes $S$ and is rooted at the source node $r$. In general, the tree may also include other nodes in $V$ as relay nodes. The cost of using a tree $T$ is denoted by $d(T)$. It is desirable to construct a tree of minimum cost. The general algorithms in this paper are applicable to any cost functions. We then evaluate the effectiveness of these algorithms by applying them to wired and wireless networks.

### A. Wired Networks

For a given tree $T$, consider the cost function defined by

$$d(T) = \sum_{i,j} d_{ij} \tag{1}$$

where the summation is over $i$ and $j$ such that $i < j$ and $(i, j)$ is a link of $T$. Here we assume that $d_{ij} = d_{ji}$. Constructing a tree $T$ including $S$ such that (1) is minimized is called a Steiner tree problem, which is a subject of extensive studies [3]–[5]. The broadcast case $|S| = N$ is also called the minimum spanning tree (MST) problem, and the unicast case $|S| = 2$ is also called the shortest path problem. These 2 special cases can be solved in time $O(N^2)$ [3], [4]. For more general $S$, the problem is NP-hard and is not solved in polynomial time by any known algorithms. The Steiner tree problem (1) is often used to model wired networks where, for example, the cost $d_{ij}$ represents the time delay between nodes $i$ and $j$, and $d(T)$ is the total delay by using the tree $T$.

### B. Wireless Networks

Recently, there is increasing interest in constructions of trees that minimize the energy consumption in wireless networks [1], [2], [6]–[14]. While the cost (1) used in wired networks is the sum of link costs, we will see later in (4) that the cost used in wireless networks is the sum of *maximum* link costs. Suppose first that we have a simple 4-node wireless network (Fig. 1), where the source node $m$ wants to send a message directly (i.e., without relaying) to 3 destination nodes $i$, $j$, and $k$. We can model this communication by a tree $T$ spanning these 4 nodes. For the tree $T$ in Fig. 1, the total RF transmission power of node $m$ is

$$P_m = \max(p_{mi}, p_{mj}, p_{mk}). \tag{2}$$

We assume here that the nodes use omnidirectional antennas, $p_{ij} = br_{ij}^a$, where $r_{ij}$ is the distance between nodes $i$ and $j$,

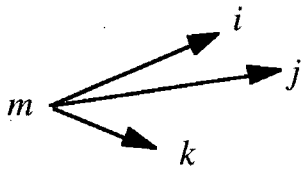



Fig. 1. Using an omnidirectional antenna, node $m$ transmits with power $P_m = \max(p_{mi}, p_{mj}, p_{mk})$.

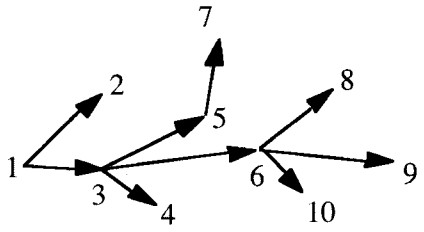



Fig. 2. A tree spanning 10 nodes (source node = 1).

$a \geq 2$ is a constant dependent on the wireless medium, and $b$ is another constant (here we set $b = 1$). Our model mainly consists of only RF transmission power. However, as shown later in Fig. 4, the impact of signal-processing power can also be incorporated into the model. In practice, node transmission power is bounded by $P_{\min} \leq P_m \leq P_{\max}$. However, to allow the algorithms to work over richer spaces, we do not impose such limits, i.e., we let $P_{\min} = 0$ and $P_{\max} = \infty$. Using this simple tree $T$ (Fig. 1), node $m$ can reach its destinations with a single transmission, i.e., when $m$ reaches the farthest node, it also reaches all nearer nodes.

We now consider a tree $T$ spanning a larger 10-node wireless network (Fig. 2). The total transmission power of the tree $T$ is then the sum of the terms of the form (2):

$$\begin{aligned} \text{power} = {} & \max(p_{12}, p_{13}) + \max(p_{34}, p_{35}, p_{36}) + p_{57} \\ & + \max(p_{68}, p_{69}, p_{6,10}). \end{aligned} \tag{3}$$

In general, for a given tree $T$ spanning a subset of nodes in a wireless network, its cost $d(T)$ is defined as follows. First, let $R_T$ be the set of nodes that have children (i.e., the set of nonleaf nodes). For each node $i$ in $R_T$, let $C_T(i)$ be the set of all children of $i$. We then define

$$d(T) = \sum_{i \in R_T} \max\{d_{ij} : j \in C_T(i)\} \tag{4}$$

where $d_{ij} = p_{ij} = r_{ij}^a$. Two algorithms are proposed in [14] for the minimum-energy wireless broadcast problem. The first is based on the familiar MST algorithm, which has computational complexity $O(N^2)$ and is originally developed for cost functions that are appropriate for wired networks [3], [4]. A novel approach is the broadcast incremental power (BIP) algorithm, which incorporates the wireless characteristics into the tree construction and outperforms the MST algorithm, but at the expense of higher complexity $O(N^3)$. Design and simulation performance of these algorithms are provided in [14]. Analytical studies of various aspects of MST and BIP are presented in [9] and [13]. With the exception of the unicast case $|S| = 2$,

which can be solved by Dijkstra's algorithm in time $O(N^2)$, polynomial-time algorithms are not available for finding trees that minimize the cost (4) in wireless networks, even for the special broadcast case $|S| = N$. In fact, recently several authors independently show that the minimum-energy broadcast wireless problem (4) is NP-hard, in addition to their development of newer broadcast algorithms for the same wireless problem [6]–[11].

## III. GENERAL TREE-CONSTRUCTION ALGORITHMS

In this section, we first present a general algorithm for constructing broadcasting trees that include all $N$ nodes of the network. We then modify the broadcasting algorithm to yield a multicasting algorithm for constructing trees that include only a subset of the network nodes. Here we assume that for a given tree $T$, we can evaluate a cost $d(T) \geq 0$ (no other restrictions are imposed on the cost function).

Before presenting the general broadcast algorithm, we need some preliminary ideas. Let $T_{k-1}$ be a tree rooted at source $s$ and has $k-1$ nodes. Note that each non-source node has a unique parent (the source node has no parent). In fact, the tree $T_{k-1}$ is specified by the vector $(A_i, i \neq s)$, where $A_i$ denotes the parent of node $i$, i.e., if $z = A_i$, then node $z$ is the parent of node $i$. Thus, by changing the parents of some nodes of a tree, we can transform it into another tree. Now let $i$ be a node of $T_{k-1}$ and $j$ be another node such that $i$ is not a descendant of $j$. Here $j$ may or may not be a node of $T_{k-1}$. We then let $i$ be the new parent of $j$, i.e., we set $A_j = i$ (i.e., if $j$ has an old parent $x$, then $x$ is replaced by $i$). The result is a new tree $T_k$ that is connected (because $i \in T_{k-1}$) and loop-free (because previously $i$ is not a descendant of $j$). We then say that the tree $T_k$ is *derived* from the tree $T_{k-1}$ and the link $L = (i, j)$. Note that if $j$ is not a node of $T_{k-1}$, then $T_k$ includes $j$ as the new node, i.e., $T_k$ now has $k$ nodes. If $j$ is a node of $T_{k-1}$, then $T_k$ still has $k-1$ nodes as $T_{k-1}$ does (however, the 2 trees are different). For wireless networks, when we say that $i$ becomes the new parent of $j$, it also means that node $i$ adjusts its power to reach node $j$. Due to the wireless medium, any node $m$ with $p_{im} \leq p_{ij}$ also receives this power. Thus, we also let $i$ be the parent of $m$, i.e., $A_m = i$, provided that $i$ is not a descendant of $m$. Thus, $i$ can now be the parent of both $j$ and $m$. Note also that $m$ may or may not be a node of $T_{k-1}$. In summary, as long as not all the nodes are already included in $T_{k-1}$, we can choose a link $L = (i, j)$, where $i \in T_{k-1}$, such that the new tree $T_k$, which is derived from $T_{k-1}$ and $L$, includes at least one new node (which can be $j$ and/or $m$).

The following general algorithms are intended for the cost function $d(\cdot)$, which can be of any form. In Section IV, we will apply these algorithms to specific costs such as (1) and (4) of Section II, as well as (7) and (11) of Section IV.

**Algorithm B:** A node is added to the tree at each step of the algorithm. The tree is initialized as $T_1 = \{\text{source node}\}$. Suppose that during step $(k-1)$, a tree $T_{k-1}$ consisting of $k-1$ nodes is constructed $(k \leq N)$. At step $k$, we decide which new node $j$ will be added next. Let $L$ be a link that starts from a node of $T_{k-1}$. Let $T_k$ be the new tree that is derived from $T_{k-1}$ and $L$. The criterion is that the tree $T_k$ includes $j$ and has the least total cost $d(T_k)$. Because there are $N$ nodes, the algorithm will end after $N$ steps, and $T_N$ is the final broadcast tree.

**Remark 1.** (a) In general, a tree-construction algorithm requires some input information such as the number of nodes, the identity of the source node, or the distribution of nodes. The output of the algorithm is a tree. In this paper, we consider the cost $d(\cdot)$ as part of the input information. Thus, Algorithm B is viewed as a single algorithm whose input consists of the cost function $d(\cdot)$ as well as other conventional information (such as the number of nodes, the identity of the source node, etc.). An alternative viewpoint is to consider Algorithm B not as a single algorithm, but rather a family of algorithms. Each member of this family is indexed by a cost function $d(\cdot)$, i.e., each member of the family is identified as Algorithm B($d$). When Algorithm B is used with a cost function $d(\cdot)$, it may be called Algorithm B($d$). However, this alternative viewpoint is not adopted in this paper, i.e., we simply call Algorithm B, ignoring the cost $d(\cdot)$.

(b) Although the new tree $T_k$ is obtained from the previous tree $T_{k-1}$ and a link $L$, $T_{k-1}$ is not necessarily a subtree of $T_k$ (because some links of $T_{k-1}$ may disappear when $L$ is added). Also, it may not be the case that $d(T_k) \geq d(T_{k-1})$. Further, the 2 trees $T_{k-1}$ and $T_k$ can be very different. When Algorithm B is applied to the cost (4) for wireless networks, at each step of the algorithm, node $i$ (which is the parent of node $j$) is allowed to transmit at a power level that is greater than $p_{ij} = r_{ij}^a$, provided that this will yield lower overall tree power. Thus, in all instances, Algorithm B performs no worse than an algorithm that produces star trees (in a star tree, all destinations are children of the same source).

(c) Let $T_k$ be a tree spanning $k$ nodes $(k \leq N)$, and $f_k(d)$ be the computational complexity of evaluating the quantity $d(T_k)$. Define $f(d) = \max\{f_1(d), \cdots, f_N(d)\}$. As an example, consider the cost $d(T)$ defined by (4), which is used in wireless networks as the total transmission power of a tree $T$. Suppose further that $T$ spans $N$ nodes. It can then be shown that $f(d) = f_N(d) = O(N)$, i.e., $f(d)$ is the computational complexity of evaluating the total transmission power of a tree spanning $N$ nodes. Similarly, it can also be shown that $f(d) = O(N)$ when the cost is defined by (1), which is used in wired networks.

(d) We use the following procedure for wired networks (see Remark 1(e) below for wireless networks). During step $k$ of Algorithm B, we scan all possible links $L = (i, j)$, where $i \in T_{k-1}$ and $j \notin T_{k-1}$. There are $O(N^2)$ such links. Here $j$ is a new node, which is a test child of $i$. We then choose the link $L = (i, j)$ that yields the new tree $T_k$ whose cost $d(T_k)$ is minimum. Computation of $d(T_k)$ can be done in time $f_k(d)$, which is bounded by $f(d)$. Thus, the computational complexity for each step, where one new node $j$ is added into the tree list, is bounded by $O(N^2) \times f(d)$. Because there are at most $N$ steps, the overall complexity of Algorithm B is bounded by $O(N^3) \times f(d)$.

(e) For wireless networks, the following procedure is used in Algorithm B. When looking for a new node to be added into the tree $T_{k-1}$ at step $k$, we scan all possible links $L = (i, j)$, where $i \in T_{k-1}$ and $j \neq i$ (here $j$ may or may not belong to $T_{k-1}$). There are at most $N(N-1) = O(N^2)$ such links. However, we only keep the links $(i, j)$ such that $i$ is not a descendant of $j$. We then let $i$ be the new parent of $j$ (i.e., if $j$ has an old parent $x$,

then $x$ is replaced by $i$). Thus, the new tree $T_k$ (that includes $j$) is connected and has no loop. When we say that $i$ is the new parent of $j$, it also means that node $i$ adjusts its power to reach node $j$, i.e., node $i$ transmits to node $j$ with power $p_{ij}$. Due to the broadcast wireless medium, any node $m$ with $p_{im} \leq p_{ij}$ will hear the transmission of $i$. Thus, we also let $i$ be the new parent of $m$, provided that previously $i$ is not a descendant of $m$. Note that $m$ may or may not belong to $T_{k-1}$. Then the new tree $T_k$ (that includes $j$ and $m$) is also connected and has no loop. In summary, as long as not all the nodes are already included in $T_{k-1}$, we can choose a link $L = (i, j)$, where $i \in T_{k-1}$, such that the new tree $T_k$, which is derived from $T_{k-1}$ and $L$, includes at least one new node (which can be $j$ and/or $m$). We then choose the link $L = (i, j)$ that yields the new tree $T_k$ of minimum cost $d(T_k)$. Determining that node $i$ is not a descendant of nodes $j$ and $m$ can be done in time $\mathrm{O}(N)$. Evaluating the cost $d(T_k)$ can be done in time $f(d)$. Thus, it takes time $\mathrm{O}(N) + f(d)$ to process each link $(i, j)$ (see also Remark 1(d) above). Then the computational complexity for each step, where at least one new node is added into the tree list, is $\mathrm{O}(N^2) \times (\mathrm{O}(N) + f(d))$. Because there are at most $N$ steps, the overall complexity of Algorithm B is bounded by $\mathrm{O}(N^3) \times (O(N) + f(d))$.

(f) For wired networks, from Remark 1(d) above, Algorithm B adds a single node into the tree at each step. However, from Remark 1(e) above, for wireless networks, the algorithm may add multiple new nodes at some steps.

(g) Algorithm B is a centralized algorithm, because it requires the full network information such as the costs between all pairs of nodes and the total tree cost at each step. A distributed version is presented later in Section V-A.2.

**Remark 2.** We now use Algorithm B to construct a tree spanning a wired network. Suppose that the cost is defined by (1) and $|S| = N$ (the broadcast case). We will now show that the resulting solution is the same as the optimal solution obtained by Prim's algorithm for constructing the MST [3, p. 523]. Here, the procedure of Remark 1(d) is used to incorporate a new node into the tree at each step. Let $D_n$ be the cost of the link that is chosen at step $n$, $n \geq 2$. Note that $D_n = d_{ij}$ for some link $(i, j)$. At step 1, the tree contains only the source node. Thus, we define $D_1 = 0$. Then the tree cost at step $k$ is $d(T_k) = \sum_{n=1}^{k} D_n$. At step $(k + 1)$, Algorithm B chooses a new link (with link cost $D_{k+1}$) such that $d(T_{k+1})$ is minimized. But $d(T_{k+1}) = d(T_k) + D_{k+1}$. Thus, minimizing $d(T_{k+1})$ is equivalent to minimizing $D_{k+1}$, which is the procedure used in Prim's algorithm. In summary, when Algorithm B is applied to a wired network that uses the cost (1), it yields the optimal solution that is also obtained by Prim's algorithm.

We now consider the multicast case. Let $S$ be a subset of $\{1, 2, \cdots, N\}$, $S$ is called a multicast group and has $|S|$ group members. Here we want to construct trees that include at least all nodes in $S$. The basic idea, which follows the method used in [5], [12], [15], and [16] for multicast algorithms, is to substitute paths for links (links are used in broadcast algorithms) such that the resulting structure is a tree (i.e., it is connected and has no loop). Note that the following Algorithm M is similar to the previous Algorithm B. The main difference is: Links $L$ are used in Algorithm B, but paths $F$ (consisting of one or more links) are used in Algorithm M.

**Algorithm M:** A group member is added to the tree at each step of the algorithm. The tree is initialized as $T_1 = \{\text{source node}\}$. Suppose that during step $(k - 1)$, a tree $T_{k-1}$ is constructed and has $k - 1$ group members ($k \leq |S|$). At step $k$, we decide which new group member $j$ will be added next. Let $F$ be a path that starts from a node of $T_{k-1}$. Let $T_k$ be the tree that is derived from $T_{k-1}$ and $F$. The criterion is that the tree $T_k$ includes $j$ and has the least total cost $d(T_k)$. Thus, at each step, the tree will include a new group member. Because there are $|S|$ group members, the algorithm will end after $|S|$ steps, and $T_{|S|}$ is the final multicast tree.

**Remark 3.** (a) In parallel to Remark 1(d), we use the following well-known procedure [12], [15]–[17] for wired networks (see Remark 3(c) below for wireless networks). During step $k$, Algorithm M scans all possible pairs $(i, j)$, where $i \in T_{k-1}$ and $j \in S - T_{k-1}$. There are at most $\mathrm{O}(N^2)$ such pairs. Here $i$ and $j$ are terminal nodes of the shortest path between $i$ and $j$. We then choose the pair $(i, j)$ that yields the new tree $T_k$ whose cost $d(T_k)$ is minimum. Computation of $d(T_k)$ can be done in time $f_k(d)$, which is bounded by $f(d)$. Thus, the computational complexity of each step, where a new group member is added into the tree list, is bounded by $\mathrm{O}(N^2) \times f(d)$. Because there are at most $|S|$ steps, the overall complexity of Algorithm M is bounded by $\mathrm{O}(N^2) \times f(d) \times |S|$.

(b) In general, paths used in Algorithm M are not necessarily the shortest paths (e.g., obtained from the Dijkstra or Bellman-Ford algorithms). For example, Algorithm M may use paths that are derived from the shortest paths as follows. Let $F_{ij}$ be the shortest path from node $i$ to node $j$, which can be written as $F_{ij} = (i_1, i_2, \cdots, i_K)$. This shortest path contains $K$ nodes $i_1, i_2, \cdots, i_K$, with $i_1 = i$ and $i_K = j$. From $F_{ij}$, we derive the following $K - 1$ paths

$$F_{ij}^{i_k} = (i_1, i_k, i_{k+1}, \cdots, i_K), \quad 2 \leq k \leq K. \tag{5}$$

Note that $F_{ij}^{j} = (i, j)$, and $F_{ij}^{i_2} = F_{ij}$ is the original shortest path. Further, $F_{ij}^{i_k}$, $3 \leq k \leq K$, are not the shortest paths from $i$ to $j$. Thus, when the paths (5) are used, the trees produced by Algorithm M are always no worse than the star trees. For each pair $(i, j)$, the number of paths in (5) is bounded by $N - 1 = \mathrm{O}(N)$.

(c) For wireless networks, the following procedure is used in Algorithm M. At step $k$, when looking for a new group member to be added into the tree $T_{k-1}$, we scan all possible paths $F_{ij}^{n}$ that are defined in (5), where $i \in T_{k-1}$ and $j \neq i$ (here $j$ may or may not belong to $T_{k-1}$). There are at most $\mathrm{O}(N^3)$ such paths. However, we keep only the *valid* paths, i.e., the paths that do not include any upstream node of $i$ (with respect to $T_{k-1}$). Invalid paths are discarded. For each valid path $F_{ij}^{n}$, let $T_k$ be the tree that is obtained by joining $T_{k-1}$ and $F_{ij}^{n}$. Note that $T_{k-1}$ and $F_{ij}^{n}$ may intersect at nodes other than $i$. Let $y$ be one of such intersecting nodes. Node $y$ may have 2 possible parents: One parent $w$ from $T_{k-1}$, the other parent $z$ from $F_{ij}^{n}$. Then, for the tree $T_k$, we assign $z$ to be the only parent of $y$. Thus, each non-source node of $T_k$ has exactly one parent. Also, $T_k$ has no loop (because $F_{ij}^{n}$ is a valid path) and is connected (because $i \in T_{k-1}$). From definition (5), we can write $F_{ij}^{n} = (i, n, \cdots, j)$. Within this path, $i$ is the new parent of $n$, this also means that node $i$

adjusts its power to reach node $n$, i.e., node $i$ transmits to node $n$ with power $p_{in}$. Due to the broadcast wireless medium, any node $m$ with $p_{im} \leq p_{in}$ will hear the transmission of $i$. Thus, we also let $i$ be the new parent of $m$, provided that previously $i$ is not a descendant of $m$. Note that $m$ may or may not belong to $S - T_{k-1}$. Then the new tree $T_k$ that includes $j$ (and/or $m$) is also connected and has no loop. In summary, as long as not all the group members are already included in $T_{k-1}$, we can choose a path $F_{ij}^n$, where $i \in T_{k-1}$, such that the new tree $T_k$, which is derived from $T_{k-1}$ and $F_{ij}^n$, includes at least one new group member (i.e., $j$ and/or $m$). We then choose the path $F_{ij}^n$ that yields the new tree $T_k$ of minimum cost $d(T_k)$. Determining that the path $F_{ij}^n$ is valid and that node $i$ is not a descendant of $m$ can be done in time $\mathrm{O}(N)$. Evaluating the cost $d(T_k)$ can be done in time $f(d)$. Thus, it takes time $\mathrm{O}(N) + f(d)$ to process each path $F_{ij}^n$. Then the computational complexity for each step, where at least one new group member is added into the tree list, is $\mathrm{O}(N^3) \times (\mathrm{O}(N) + f(d))$. Because there are at most $|S|$ steps, the overall complexity of Algorithm M is bounded by $\mathrm{O}(N^3) \times (\mathrm{O}(N) + f(d)) \times |S|$.

## IV. APPLICATIONS TO ENERGY-AWARE WIRELESS NETWORKS

In this section, we apply Algorithms B and M, which can be used for any cost functions, to broadcasting and multicasting in energy-aware wireless networks. We will show that these algorithms typically improve on some other algorithms (e.g., MST and BIP). Some historical highlights and major design issues for wireless networks are provided in [1], [14], and [18]–[20]. Our goal is to construct trees to support source-initiated sessions in wireless multihop networks. Here we assume that the wireless networks are static, i.e., the node locations are fixed. As in [1], we consider the algorithms in several different contexts: Broadcasting, multicasting, signal-processing cost, omnidirectional antennas, directional antennas, as well as network resources such as energy, transceivers, and frequencies. Our models are more comprehensive than the models in [6]–[8], and [10], which do not consider the issues associated with limited energy, transceivers, and frequencies.

### A. Broadcasting and Multicasting with Omnidirectional Antennas

*1) Broadcasting*: Here we present algorithms for finding energy-efficient broadcast trees with the use of omnidirectional antennas. Recall that the power required to maintain a tree is given by the cost function (4). In contrast to the problem with the cost function (1), which can be solved in polynomial time, the broadcast problem with the cost function (4) is NP-hard [6], [8]–[11]. Thus, heuristic suboptimal polynomial-time algorithms for the wireless network problem are desirable.

A simple heuristic is based on the MST algorithm. First, we obtain the MST using the cost function (1), with $d_{ij} = p_{ij} = r_{ij}^a$, i.e., ignoring the cost definition (4) during the tree construction. After the MST is constructed, definition (4) is used to evaluate its cost. The 2nd algorithm is BIP [14]. The initial BIP tree includes only the source node. At each step, one node is added into the tree as follows. Consider a link $(i, j)$, where $i$ is a node inside the tree and $j$ is an outside node. The link $(i, j)$ is added into the tree if the incremental power $d_{ij} - P_i$ is minimized, where $d_{ij} = p_{ij} = r_{ij}^a$ and $P_i$ is the transmitted power at node $i$. BIP stops after the tree contains all the nodes. The 3rd algorithm is Algorithm B when applied to the cost (4). At step $k$, a new node $j$ is added such that the tree $T_k$, which is derived from $T_{k-1}$ and includes $j$, has the least total cost $d(T_k)$, where $d(\cdot)$ is the cost function defined by (4). Remark 1(e) describes Algorithm B in detail. In summary, we have 3 heuristic algorithms for the same broadcast wireless network problem with the cost function (4): The MST algorithm, BIP, and Algorithm B (see Fig. 12 in Section V for trees constructed by these algorithms).

In this paper, we use the method of [1] and [14] to evaluate the performance of different algorithms (e.g., MST, BIP, and Algorithm B) by comparing their absolute and normalized tree costs (power). Specifically, we generate 100 different networks. Each network has $N$ nodes (one of which is a randomly chosen source) that are located randomly in the same $5 \times 5$ square region.

We then run these algorithms on each of the 100 networks. Thus, we can obtain 100 absolute tree power values for each algorithm (one value for each network). The normalized tree power of an algorithm is defined as the absolute tree power (obtained by this particular algorithm) divided by the least power (obtained among these algorithms). Thus, for a given network, the normalized power of the best algorithm (among these algorithms) will be 1.0. There are 100 normalized tree power values for each algorithm (one value for each network). Next, we compute the average of these absolute tree power values and the average of normalized tree power values. In addition to the absolute and normalized power values, we also compute their variances. For small networks (e.g., $N = 10$) we can find the optimal trees using the OPT algorithm (by exhaustive search). For larger networks (e.g., $N = 100$), it is not feasible to find the optimal trees due to the high complexity of this NP-hard problem.

The performance of the 4 algorithms MST, BIP, Algorithm B, and OPT for $N = 10$ is shown in Fig. 3(a), both with and without the use of the sweep operation [14]. The purpose of the sweep operation is to discover and reduce redundant transmission power at some nodes. Thus, the overall tree power is often reduced. When $a = 2$, the sweep yields about 8.6% and 6.2% absolute power reduction for MST and BIP, respectively, while the absolute power reduction for Algorithm B is less than 0.3%. Thus, the sweep is more beneficial for MST and BIP than for Algorithm B. We observe that better algorithms usually receive less improvement from the sweep. Obviously, the optimal trees cannot be improved by the sweep operation (or by any other operations). Note that Algorithm B underperforms OPT only slightly, i.e., less than 1.53% with or without the sweep operation. Better performance is indicated by lower absolute and normalized power. Clearly, BIP outperforms MST, and Algorithm B outperforms BIP. With the sweep, the Algorithm B trees consume about 8.8% and 17% less power than the BIP and MST trees, respectively. When $a = 4$, the performance difference among the algorithms is reduced, i.e., the difference between the worst algorithm (MST) and the best (OPT) is only

| Algo | without sweep operation | with sweep operation |
|---|---|---|
| propagation loss exponent $a = 2$ | | |
| MST | 13.47 (15.69) | 12.40 (13.41) |
| | 1.303 (0.054) | 1.198 (0.045) |
| BIP | 12.24 (12.53) | 11.53 (11.96) |
| | 1.178 (0.025) | 1.104 (0.015) |
| Algo B | 10.63 (9.639) | 10.60 (9.616) |
| | 1.015 (0.001) | 1.012 (0.001) |
| OPT | 10.47 (9.380) | 10.47 (9.380) |
| | 1.000 (0.000) | 1.000 (0.000) |
| propagation loss exponent $a = 4$ | | |
| MST | 49.67 (1050) | 47.40 (961.7) |
| | 1.095 (0.019) | 1.046 (0.011) |
| BIP | 48.40 (912.2) | 46.52 (871.2) |
| | 1.070 (0.014) | 1.023 (0.005) |
| Algo B | 45.67 (841.4) | 45.64 (841.1) |
| | 1.003 (0.000) | 1.002 (0.000) |
| OPT | 45.53 (838.4) | 45.53 (838.4) |
| | 1.000 (0.000) | 1.000 (0.000) |

(a)

| Algo | without sweep operation | with sweep operation |
|---|---|---|
| propagation loss exponent $a = 2$ | | |
| MST | 12.24 (0.708) | 11.53 (0.645) |
| | 1.231 (0.005) | 1.165 (0.004) |
| BIP | 11.48 (0.576) | 10.81 (0.536) |
| | 1 154 (0 004) | 1 093 (0 004) |
| Algo B | 9.968 (0.547) | 9.926 (0.558) |
| | 1.000 (0.000) | 1.001 (0.000) |
| propagation loss exponent $a = 4$ | | |
| MST | 3.480 (0.723) | 3.352 (0.690) |
| | 1.073 (0.002) | 1.040 (0.002) |
| BIP | 3.397 (0.719) | 3.284 (0.679) |
| | 1.046 (0.001) | 1.018 (0.001) |
| Algo B | 3.254 (0.698) | 3.243 (0.700) |
| | 1.001 (0.000) | 1.004 (0.000) |

(b)

Fig. 3. Absolute and normalized broadcast tree power (and their variances): (a) $N = 10$ nodes, (b) $N = 100$ nodes.

about 4.1% (with the sweep operation).

The performance of the 3 algorithms MST, BIP, and Algorithm B for $N = 100$ nodes is shown in Fig. 3(b). It is not feasible to evaluate OPT in this case. Again, BIP outperforms MST, and Algorithm B outperforms BIP. Without the sweep, the Algorithm B trees consume about 15% and 23% less power than the BIP and MST trees, respectively. The sweep yields about 6% power reduction for both MST and BIP, while the power reduction for Algorithm B is only about 0.4%. With the sweep, the Algorithm B trees consume about 9% and 16% less power than the BIP and MST trees, respectively.

In summary, with or without the sweep operation, BIP outperforms MST, and Algorithm B outperforms BIP. Because the sweep operation is beneficial for most heuristic algorithms, we will apply it to all algorithms considered in this paper. Finally, we observe that the variances (of the absolute tree power values) when $N = 10$ are much greater than the variances when $N = 100$. For example, when $a = 2$ and with the sweep, the MST trees have variances 13.41 and 0.645 when $N = 10$ and $N = 100$, respectively.

The better performance under Algorithm B can be explained as follows. Recall that the goal is to construct trees that minimize the target cost (4). In MST (or BIP), $d_{ij}$ (or $d_{ij} - P_i$) is minimized at each step, but neither of these quantities match the target cost (4). In contrast, at each step, Algorithm B aims to minimize the target cost (4) directly. Thus, it should perform better. Algorithm B is also more complex because it is based on a "global" measure, i.e., the global tree cost (4) is evaluated at each step. In contrast, the MST algorithm and BIP are based on a "local" measure such as $d_{ij}$ or $d_{ij} - P_i$. Thus, they are less complex.

**Remark 4.** From Remark 1(e), the complexity of Algorithm B is $\mathrm{O}(N^3) \times (\mathrm{O}(N) + f(d))$, where $f(d)$ is the computational complexity of evaluating the power of a tree that spans $N$ nodes. For the case of omnidirectional antennas, it can be shown that $f(d) = O(N)$. Thus, the complexity of Algorithm B is $\mathrm{O}(N^3) \times (\mathrm{O}(N) + \mathrm{O}(N)) = \mathrm{O}(N^4)$. Recall that the computational complexity of MST and BIP is $\mathrm{O}(N^2)$ and $\mathrm{O}(N^3)$, respectively. Thus, among the 3 broadcasting algorithms considered here, the best performing algorithm is also the most complex one, and the weakest performing algorithm is also the simplest one.

So far we consider only the RF power and ignore the signal-processing cost. As shown in [1], it is straightforward to incorporate the impact of signal-processing power into the tree-construction algorithms. Let us revisit Fig. 1, where node $m$ transmits with the RF power $P_m = \max(p_{mi}, p_{mj}, p_{mk})$. The processing power can be incorporated into the algorithms as follows. First, let $p_n^T$ and $p_n^R$ be the transmission and reception processing power at node $n$, respectively. In Fig. 1, we have $p_m^R = 0$ (because the source node $m$ does not receive) and $p_i^T = p_j^T = p_k^T = 0$ (because the leaf nodes $i$, $j$, and $k$ do not transmit). Then the total power cost (RF power plus processing power) used by the tree in Fig. 1 is $P^* = P_m + p_m^T + p_i^R + p_j^R + p_k^R$. For a more general network, the total power cost is defined by

$$d^*(T) = d(T) + \sum_n (p_n^T + p_n^R)$$

where $d(T)$ is the RF power cost defined by (4). Note that $p_n^R = 0$ if $n$ is the source node, and $p_n^T = 0$ if $n$ is a leaf node.

BIP can now be extended to include the processing power as follows [1]. At each step of the extended BIP, we add the link $(i, j)$ into the current tree if $P_{ij}^* - P_i^*$ is minimized, where $P_i^*$ is the total power at node $i$ of the current tree, and $P_{ij}^*$ is the total power after the link $(i, j)$ is added into the current tree. Algorithm B is applicable to any cost functions. We now apply it to the cost $d^*(T)$ above. The performance of BIP and Algorithm B for different combinations of processing power values is shown in Fig. 4 for $N = 100$, where we let $p_n^T = p^T$ and $p_n^R = p^R$ for all nodes $n$ that have nonzero processing power. As noted previously, the sweep operation is also used here. As expected, the total tree power increases with the addition of the processing cost. For example, when $a = 2$, the absolute tree power under Algorithm B increases from 9.926 (Fig. 3(b), $p^T = p^R = 0$) to

| $p^T$ | $P^R$ | BIP | Algo B |
|---|---|---|---|
| propagation loss exponent $a = 2$ | | | |
| 0.01 | 0.01 | 12.37 (0.517)<br>1.100 (0.004) | 11.28 (0.646)<br>1.001 (0.000) |
| 0.1 | 0.1 | 24.88 (1.028)<br>1.126 (0.003) | 22.14 (1.229)<br>1.000 (0.000) |
| 0.01 | 0.1 | 21.28 (0.517)<br>1.054 (0.001) | 20.22 (0.639)<br>1.000 (0.000) |
| 0.1 | 0.01 | 15.97 (1.028)<br>1.218 (0.009) | 13.18 (1.419)<br>1.000 (0.000) |
| propagation loss exponent $a = 4$ | | | |
| 0.01 | 0.01 | 4.926 (0.668)<br>1.014 (0.000) | 4.872 (0.680)<br>1.003 (0.000) |
| 0.1 | 0.1 | 18.48 (0.675)<br>1.021 (0.000) | 18.11 (0.717)<br>1.001 (0.000) |
| 0.01 | 0.1 | 13.84 (0.668)<br>1.005 (0.000) | 13.78 (0.680)<br>1.001 (0.000) |
| 0.1 | 0.01 | 9.568 (0.675)<br>1.044 (0.001) | 9.180 (0.701)<br>1.001 (0.000) |

Fig. 4. Absolute and normalized broadcast tree power (and their variances) with nonzero processing power $p^T$ and $p^R$ ($N = 100$).

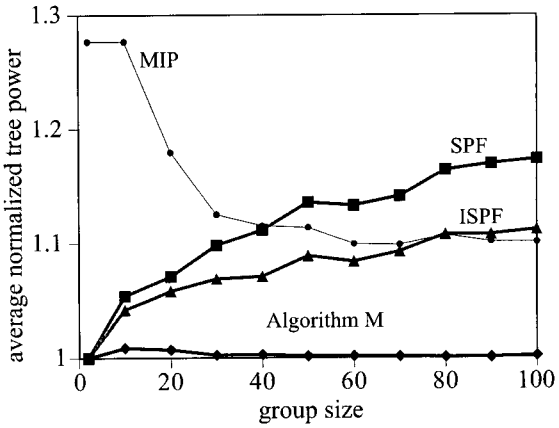



Fig. 5. Performance of multicast algorithms ($N = 100, a = 2$).

11.28 (Fig. 4, $p^T = p^R = 0.01$). Again, Algorithm B outperforms BIP in all cases. For the rest of this paper, we consider only the RF power.

*2) Multicasting*: In Section IV-A.1 we present algorithms for finding energy-efficient broadcast trees with the use of omnidirectional antennas. The power of a given tree is the cost (4). We now wish to construct multicast trees that include the nodes of a multicast group $S$. The previous broadcast algorithms can be modified to yield multicast algorithms. A simple method is to prune broadcast trees (i.e., to eliminate any link that does not lead to a group member) to produce multicast trees. For example, we can prune the broadcast BIP trees to yield multicast trees called the multicast incremental power (MIP) trees [14]. As seen later (Fig. 5), the pruning method does not work well, unless the multicast groups are large.

A better method, which is used in [12], [15], and [16] for heuristic multicast algorithms, is to incorporate the shortest paths into the tree construction. First, let $F_{ij}$ be the shortest path between nodes $i$ and $j$. A generalization of the MST algorithm, which is based on the well-known algorithm of [15], for constructing multicast trees works as follows. The initial tree includes only the source node. At each step, one group member is added into the tree according to the following rule. Let $i$ be a node inside the tree and $j$ be a group member outside the tree. We then add the path $F_{ij}$ into the tree if $d(F_{ij})$ is minimized. The algorithm stops when the tree contains all the group members. This algorithm, called the shortest path first (SPF), is proposed in [12] for multicasting in wireless networks.

The next algorithm, which is called the incremental shortest path first (ISPF) and has some features of both BIP and SPF, works as follows. Again, the initial tree includes only the source node. Let $i$ be a node inside the tree and $j$ be a group member outside the tree. At each step, we add the path $F_{ij}$ into the tree if $d(F_{ij}) - P_i$ is minimized (recall that $d_{ij} - P_i$ is minimized in BIP). The algorithm stops when the tree contains all the group members. Another algorithm is Algorithm M. At step $k$, Algorithm M forms a new tree $T_k$ by joining a path $F_{ij}^n$ to the tree $T_{k-1}$ such that $T_k$ includes at least one new group member. Then the tree $T_k$ is chosen if the cost $d(T_k)$ is minimized. Remark 3(c) describes Algorithm M in detail.

We now compare the performance of the 4 multicast algorithms: MIP, SPF, ISPF, and Algorithm M. We evaluate the performance for different group sizes, which vary from 2 to $N$. Group size of 2 corresponds to the unicast case where a source is to reach a single destination. Group size of $N$ corresponds to the broadcast case where a source is to reach all $N - 1$ destinations. Here, the performance is the average of normalized tree power values over many random networks. Specifically, for each multicast group size, we compute the average normalized tree power for an algorithm as follows. First, we generate 100 random networks $W_1, \cdots, W_{100}$. Each network has $N$ nodes that are located randomly in the same $5 \times 5$ square region. Consider a group size $|S|$. We then generate 100 random multicast groups $G_1^S, \cdots, G_{100}^S$ (group $G_i^S$ is generated for network $W_i$). Each group is required to have the same size $|S|$, i.e., $|G_i^S| = |S|$. Next, we run the 4 multicast algorithms for each network-group pair $(W_i, G_i^S)$. For each algorithm, we compute the normalized power values for each pair $(W_i, G_i^S)$, and then compute the average of these values over the 100 networks.

Let $N = 100$. In Fig. 5, we plot the average normalized power as function of group size $|S| = 2, 10, 20, \cdots, 90, 100$. When $|S| = 2$, the 3 algorithms other than MIP produce identical optimal results because they all find the same shortest paths. Fig. 5 shows that ISPF outperforms SPF, and Algorithm M outperforms ISPF for all considered multicast group sizes. As expected, MIP (the pruning algorithm) performs worst for smaller multicast groups (with 40 group members or less). However, MIP outperforms both SPF and ISPF for larger multicast groups (with 80 group members or more). It is important to note that when the multicast group includes all the nodes (i.e., the broadcast case), the SPF algorithm produces the MSTs, i.e., the SPF algorithm is a generalization of the MST algorithm [12], [15]. However, as evidenced from Fig. 5, ISPF algorithm is not a generalization of BIP. MIP is a generalization of BIP. Another generalization of BIP called the minimum incremental path first (MIPF) is proposed in [12]. ISPF is a slight variation of MIPF.

Using the technique of [15], both SPF and ISPF have computational complexity $O(N^3)$, where $N$ is the number of nodes. A

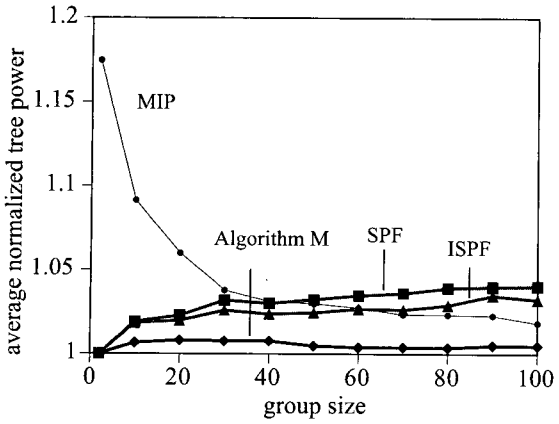



Fig. 6. Performance of multicast algorithms ($N = 100, a = 4$).

faster implementation of SPF is given in [17], which has complexity $O(N^2)$.The complexity of MIP is $O(N^3)$ [14]. From Remark 3(c), for a multicast group $S$, the complexity of Algorithm M is bounded by $O(N^3) \times (O(N) + f(d)) \times |S|$, where $f(d)$ is the computational complexity of evaluating the power of a tree that spans $N$ nodes. For the case of omnidirectional antennas, it can be shown that $f(d) = O(N)$. Thus, the complexity of Algorithm M is bounded by $O(N^4) \times |S|$.

Simulation results shown in Fig. 5 are for the propagation loss exponent $a = 2$. Fig. 6 shows simulation results for $a = 4$. The relative performance among the algorithms is similar to Fig. 5. However, the performance difference among the algorithms is now reduced. For the rest of this paper, we focus on the case $a = 2$.

### B. Broadcasting and Multicasting with Directional Antennas

In Section IV-A, we present tree-construction algorithms for energy-aware wireless networks that use ominidirectional antennas (see (2)–(4)). We now turn our attention to networks that use directional antennas. Refer to Fig. 1, where node $m$ transmits to nodes $i$, $j$, and $k$, using an omnidirectional antenna. The resulting power is given by (2). Suppose now that the same node $m$ uses a directional antenna to transmit to nodes $i$, $j$, and $k$ as shown in Fig. 7, where $A$ is the angle (beamwidth) that covers the nodes $i$, $j$, and $k$. Then the RF transmission power of node $m$ is

$$P_m(A) = \frac{A}{360} \max(p_{mi}, p_{mj}, p_{mk}). \tag{6}$$

In (6), we use an idealized model so that all of the transmitted energy is concentrated uniformly in the chosen beamwidth [1]. Within a multicast tree, to conserve the transmission energy, we assume that each transmitting node chooses the minimum beamwidth $A$ to cover its children, such that $0 < A_{\min} \leq A \leq A_{\max} \leq 360$ [1]. In this paper, we set $A_{\max} = 360$. For networks equipped with ominidirectional antennas, the beamwidth $A$ is always 360 degrees, i.e., $A_{\min} = A_{\max} = 360$. Thus, with proper design, networks equipped with directional antennas use less RF energy than those equipped with omnidirectional antennas. For a more general tree $T$ (such as the tree in Fig. 2), with the use of directional antennas, the total tree power (which is considered as a tree cost) is the sum of the terms of the form (6):

$$d(T) = \sum_{m \in T} P_m(A_m) \tag{7}$$

where $A_m$ is the antenna beamwidth at node $m$, and $P_m(A_m)$ is the transmission power at node $m$, which has the form (6). Note that $P_m(A_m) = 0$ if $m$ is a leaf node.

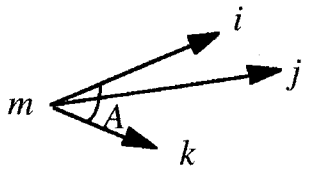



Fig. 7. Using a directional antenna, node $m$ transmits with power $P_m(A) = (A/360) \max(p_{mi}, p_{mj}, p_{mk})$.

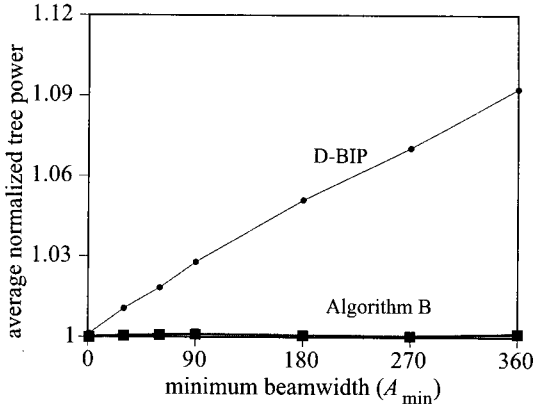



Fig. 8. Performance of broadcast algorithms ($N = 100, a = 2$).

A number of heuristic tree-construction algorithms that exploit directional antennas are proposed in [1]: Directional BIP (D-BIP) and directional MIP (D-MIP) for broadcasting and multicasting, respectively. D-MIP trees are D-BIP trees that are pruned. Recall that Algorithms B and M presented in Section III are for constructing broadcast and multicast trees with an arbitrary cost function $d(\cdot)$. We now use the cost function $d(T)$ defined by (7) in these general algorithms to yield directional-antenna versions of Algorithms B and M. In summary, we have 2 algorithms for constructing broadcast trees (D-BIP and Algorithm B applied to the cost (7)) and 2 algorithms for constructing multicast trees (D-MIP and Algorithm M applied to the cost (7)). The performance of these algorithms is shown in Fig. 8 (for broadcasting) and Fig. 9 (for multicasting). Clearly, Algorithms B and M outperform D-BIP and D-MIP, respectively, i.e., trees under Algorithms B and M typically use less power than trees under D-BIP and D-MIP.

The complexity of D-BIP and D-MIP is bounded by $O(N^3 \log N)$ [1]. From Remark 1(e), the complexity of Algorithm B is bounded by $O(N^3) \times (O(N) + f(d))$, where $f(d)$ is the computational complexity of evaluating the power of a tree that spans $N$ nodes. For the case of directional antennas, it can be shown that $f(d) \leq O(N \log N)$ [1]. Thus, the complexity of Algorithm B is bounded by $O(N^4 \log N)$. From Remark 3(c), for a multicast group $S$, the complexity of Algorithm M is bounded by $O(N^3) \times (O(N) + f(d)) \times |S|$, where

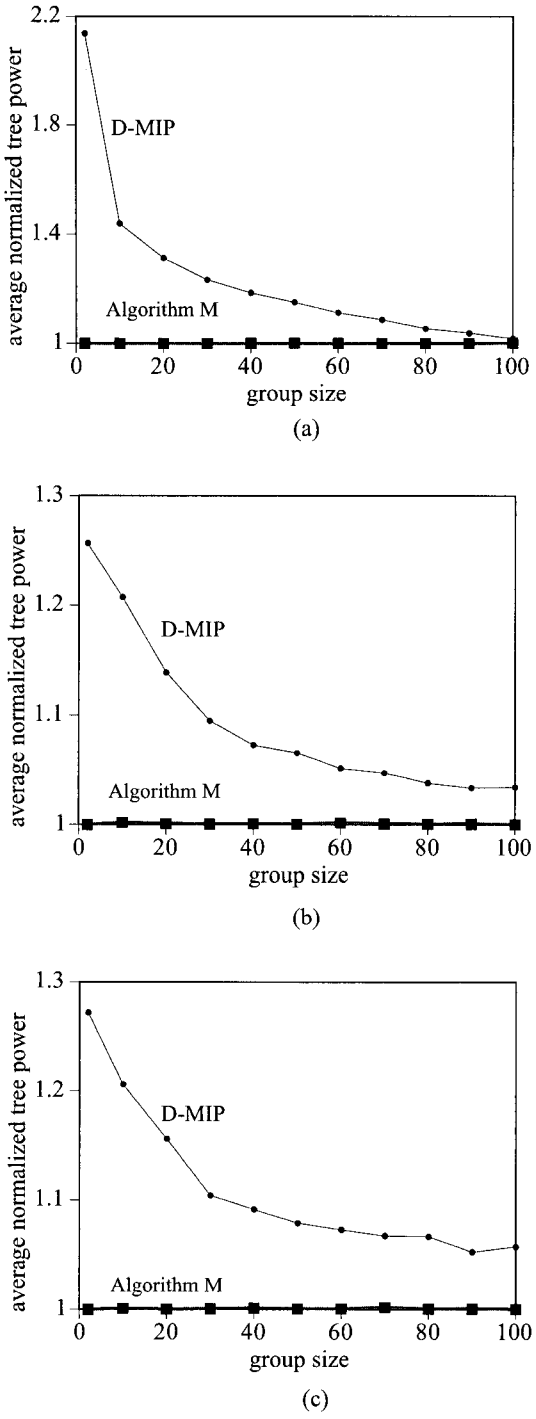



Fig. 9. Performance of multicast algorithms ($N = 100, a = 2$): (a) $A_{\min} = 30$, (b) $A_{\min} = 90$, (c) $A_{\min} = 180$.

$f(d) \leq \mathrm{O}(N \log N)$. Thus, the complexity of Algorithm M is bounded by $\mathrm{O}(N^4 \log N) \times |S|$.

### C. Multicasting with Energy Limitations

We now consider algorithms for *energy-limitation* operation, i.e., each network node is equipped with a fixed quantity of energy that cannot be renewed during network operation. Node energy is reduced after a transmission to support a multicast session. After supporting many sessions, a node dies (i.e., it is no longer able to transmit or receive messages) when its energy falls below a low level. Let $E_i(0)$ be the initial energy allocated to node $i$, and let $E_i(t)$ be its residual energy at time $t$. When node $i$ transmits with power $P$ for the time duration $D$ to support a multicast session, it expends the amount of energy $PD$. We have $E_i(t_2) \leq E_i(t_1)$ when $t_1 < t_2$. Node $i$ dies at time $t$ when $E_i(t) \approx 0$, and no further communication to or from node $i$ is possible.

Here, an appropriate performance measure is the total traffic volume the network can deliver [1]. Assume that the wireless network has $N$ nodes, which are randomly located in a region with dimensions $5 \times 5$. We simulate multicast session requests that arrive into the network according to exponential interarrival times with unit mean. Session durations are also exponentially distributed with unit mean. Each session forms a multicast group, which consists of a source and a number of destinations. The group size is uniformly distributed between 2 and $N$, and a source is chosen randomly among the group members. The total network traffic delivered to all reached destinations for $X$ multicast sessions is

$$B_X = \sum_{s=1}^{X} b_s \tag{8}$$

where $b_s$ is the traffic delivered to reached destinations in session $s$, which is defined to be the product of the session duration and the number of reached destinations of the session [1]. We assume that sufficient transceivers and frequency resources are available to support all session requests. We then run the simulation until no additional traffic can be delivered.

Our goal is to construct multicast trees to support session requests such that the total delivered traffic $B_X$ is maximized for a given value of initial energy at each network node. Heuristic suboptimal algorithms are needed because the optimal algorithms are complex. For the energy-limitation operation, the incremental power $d_{ij} - P_i$ used in the original MIP is modified to become [1]

$$(d_{ij} - P_i)\frac{E_i(0)}{E_i(t)} \tag{9}$$

where $P_i$ is the transmitted power at node $i$, and $d_{ij} = p_{ij} = r_{ij}^a$. Thus, using the energy-limitation version of MIP, we add the link $(i, j)$ into the tree at each step if the incremental cost (9) is minimized.

Note that the cost (4) (as well as the cost (7)), which is the total power of a tree $T$, can be written as

$$d(T) = \sum_{i \in T} P_i \tag{10}$$

where $P_i$ is the transmitted power at node $i$, and $P_i = 0$ if $i$ is a leaf node of $T$. For the energy-limitation operation, we now define a new cost function for a tree $T$

$$d(T) = \sum_{i \in T} P_i \frac{E_i(0)}{E_i(t)} \tag{11}$$

which is derived from (10) by multiplying the summands by $E_i(0)/E_i(t)$. Because any cost functions can be used in Algorithm M, so can the cost function (11). Thus, for the energy-limitation operation, the cost (11) is now used in Algorithm M.

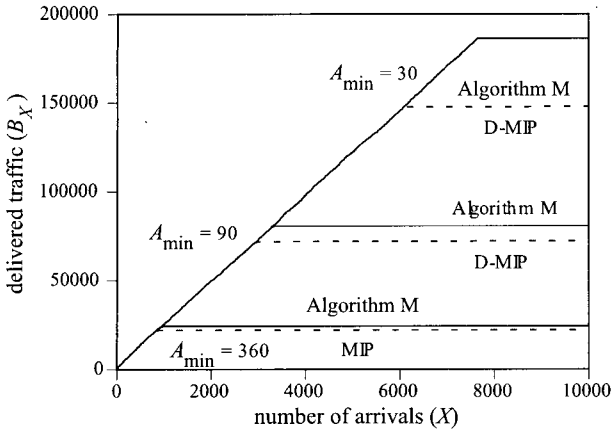



Fig. 10. Delivered traffic under D-MIP and Algorithm M ($N = 50, a = 2$).

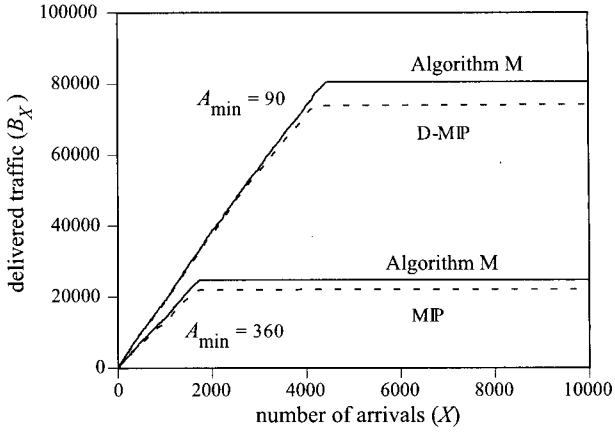



Fig. 11. Delivered traffic under D-MIP and Algorithm M ($N = 50, a = 2$, 2 transceivers, 4 frequencies).

The performance of D-MIP and Algorithm M is shown in Fig. 10 (D-MIP becomes MIP when $A_{\min} = 360$). Here, the network has $N = 50$ nodes, the number of arrivals is $X = 10{,}000$, and each network node is allocated the same initial energy $E_i(0) = 200$. The performance is shown for omnidirectional antennas ($A_{\min} = 360$) and directional antennas ($A_{\min} = 30$ and 90). As expected, Algorithm M outperforms D-MIP. Specifically, the delivered traffic under Algorithm M is 11% ($A_{\min} = 360$), 12% ($A_{\min} = 90$), and 26% ($A_{\min} = 30$) more than the delivered traffic under D-MIP. Algorithm M also outperforms MIP in terms of network lifetime (i.e., the time when the 1st node in the network dies due to energy depletion). In particular, when $A_{\min} = 360$, the network lifetime is 828.2 and 638.5 under Algorithm M and MIP, respectively. Thus, Algorithm M extends the network lifetime by about 30%.

*1) Impact of finite transceiver and frequency resources*: Recall that trees are used to support multicast sessions. Each participating node of a session needs a transceiver and a suitable frequency. When the transceiver-frequency resource is not available, the affected nodes are blocked. So far we assume the availability of an unlimited number of transceivers and frequencies, which are always available for each session request. The performance under such assumption is shown in Fig. 10.

We now make the more realistic assumption that the number of transceivers and frequencies is *limited*. Following [1], we use the following greedy method for dealing with the limitation of resources. Suppose that a new session request arrives. To support this new session, we consider only nodes that have available transceivers. We then use an algorithm (such as MIP or Algorithm M) to construct an initial multicast tree $T_{init}$ with the assumption of unlimited number of frequencies. Next, we attempt to assign the finite number of frequencies to the nodes of $T_{init}$. If a frequency $g$ is used by node $i$ for transmitting to node $j$, then we say that $g$ is the transmitting frequency of node $i$, and $g$ is the receiving frequency of node $j$. Suppose now that node $n$ wants to choose a noninterfering frequency $f$ to transmit. The frequency $f$ is noninterfering if (a) there is no any other node that is inside the beamwidth of $n$ and whose receiving frequency is $f$, and (b) if $n$ is also a receiving node, its receiving frequency is different from $f$, i.e., the same frequency cannot be used for transmission and reception by a node at the same time.

We use the following greedy procedure to assign noninterfering frequencies to transmitting nodes (non-transmitting nodes or leaf nodes do not need transmitting frequencies). First, we try to assign a noninterfering frequency to the source of the initial tree $T_{init}$. If this fails, the multicast session is blocked. If a noninterfering frequency is successfully assigned to the source, then we apply the frequency-assignment procedure to its transmitting descendants (as defined by $T_{init}$). At each assignment, we choose the lowest-numbered available noninterfering frequency. When no such frequency is available for a node, the frequency assignment fails at this node. Then this node and all of its descendants are pruned from the initial tree $T_{init}$.

To implement this frequency-assignment procedure, we construct a list of nodes. At each step of the procedure, we try to assign a noninterfering frequency to a node in this list. Initially, the list contains only the source node of $T_{init}$. As described in the following, the size of this list can change after each step. After a node is assigned a noninterfering frequency, this node is deleted from the list, and then all of its transmitting children are added to the list. When a node cannot be assigned a noninterfering frequency, it is also deleted from the list. Thus, after one step, exactly one node is permanently deleted from the list, i.e., this node is never reconsidered at later steps. In our implementation, at each step, we try to assign a noninterfering frequency to the lowest-numbered node from the list. Note that this frequency assignment is greedy and suboptimal, because there is no backtracking to revisit a node that could not be assigned a noninterfering frequency at a previous step, or to revise a previous frequency assignment. The frequency-assignment procedure ends when the list is empty.

The result is a tree (which is a subtree of the initial tree $T_{init}$) such that frequencies and transceivers are assigned to its nodes. Transceivers and frequencies are dedicated to a particular session (multicast tree) throughout its duration. Upon completion of a session, these resources are immediately released and become available for future sessions. Let us return to the model of Fig. 10. However, we now let the number of transceivers at each node be 2, and let the number of frequencies available to the network be 4. Fig. 11 compares the delivered traffic of D-MIP and Algorithm M. Again, Algorithm M outperforms D-

MIP, i.e., Algorithm M delivers about 9% ($A_{\min} = 90$) and 12% ($A_{\min} = 360$) more traffic than D-MIP.

## V. ALGORITHMS WITH LOWER COMPLEXITY

From Section IV, the general Algorithms B and M outperform some other algorithms at the cost of higher computational complexity. In particular, the complexity of Algorithm B is $O(N^4)$, while that of MST and BIP is $O(N^2)$ and $O(N^3)$, respectively (Remark 4). We will now show that our general algorithms can be adapted to yield 2 algorithms that have lower complexity but underperform only slightly. One algorithm is a distributed version of Algorithm B, the other is a centralized one. Here we reconsider the problem posed in Section IV-A.1, i.e., we wish to find energy-efficient broadcast trees with the use of omnidirectional antennas. These lower complexity versions can also be used to produce multicast trees (see Remark 7 later). In this section, we show the effectiveness of these lower complexity algorithms in connection with the cost (4) used in wireless networks. However, these algorithms are also general and applicable to any cost functions.

### A. Algorithm B1

Our basic idea is to divide the $N$ nodes into smaller subsets, each subset has a local source. We then apply Algorithm B to these subsets to form subtrees, which are then joined to form the final global tree. Before presenting the algorithm, we need some definitions. Let $T$ be a tree that spans $N$ nodes and has source $s$. Each node $i$ of this tree is assigned a *level* $L_i$ as follows. The level of the source node is 0, i.e., $L_s = 0$. If $i$ is a node of level $L_i$, then all of its children have level $L_i + 1$, and its parent has level $L_i - 1$ (if $i \neq s$). Thus, all children of the source $s$ have level 1, and all of its grandchildren have level 2, and so on. Note that $0 \leq L_i \leq N - 1$.

**Algorithm B1:** This algorithm is used to construct a broadcast tree that spans a network of $N$ nodes and is rooted at source $s$. Let $H$ be an integer such that $1 \leq H \leq N$. The algorithm has 2 phases. In phase 1, a planar Euclidean MST that spans the entire network of $N$ nodes is constructed and is rooted at the global source $s$. For each node $i$ of this MST, let $L_i$ be its level. A node $n$ becomes a *local* source when $L_n$ is a multiple of $H$. For each local source $n$, let $Z_n(H) = \{i : L_n \leq L_i \leq L_n + H\}$. Note that $n \in Z_n(H)$. In phase 2, we apply Algorithm B to the nodes of $Z_n(H)$ with $n$ as the (local) source node. The result is a subtree that is rooted at $n$ and spans all the nodes in $Z_n(H)$. From our construction, when $n \neq s$, the local source $n$ is a leaf node of another subtree that spans some subset $Z_m(H), m \neq n$. Thus, all these subtrees collectively form a single global tree that is connected and is loop-free.

As an example, consider a network of $N = 20$ nodes. The MST constructed from this network is shown in Fig. 12(a). Here the global source node is $s = 13$. Let $H = 3$. Consider node 17 of level $L_{17} = 6$, which is a multiple of $H = 3$. Thus, 17 is a local source, and $Z_{17}(3) = \{i : 6 \leq L_i \leq 9\}$. Because $L_{18} = 8$, we have $18 \in Z_{17}(3)$. It can be shown that nodes 2 and 16 also belong to $Z_{17}(3)$. Thus, $Z_{17}(3) = \{2, 16, 17, 18\}$. Similarly, it can be shown that nodes 5, 10, 13, 14, and 16 are also local sources, and their subsets $Z_n(3)$ can be easily determined. These local sources are shown in Fig. 12(a) by larger black dots. We then run Algorithm B on each of these subsets (e.g., applying Algorithm B to the subset $Z_{17}(3)$ with 17 as the source node) to yield the subtrees shown in Fig. 12(b). The final global tree is the union of these subtrees (Fig. 12(b)). Under Algorithm B1, the total tree power is 9.936 when $H = 3$. With the lower $H = 2$, the power increases to 10.05. With the higher $H = 4$, the power decreases to 8.633. Further, if we set $H \geq 8$, the power becomes 8.326, which is the same as the power produced by Algorithm B (Fig. 12(c)). Note that the MST is produced when $H = 1$. Thus, at one extreme ($H = 1$), Algorithm B1 produces the MST, while at the other extreme ($H \geq 8$), Algorithm B1 becomes Algorithm B. Thus, the performance and complexity of Algorithm B1 are somewhere between those of the MST algorithm and Algorithm B. For comparison, the BIP tree for the same network is shown in Fig. 12(d). Note that, for simple illustration, the sweep operation is not applied to the trees in this particular example. In summary, Algorithm B1 outperforms both the MST and BIP when $H \geq 2$ in this example.

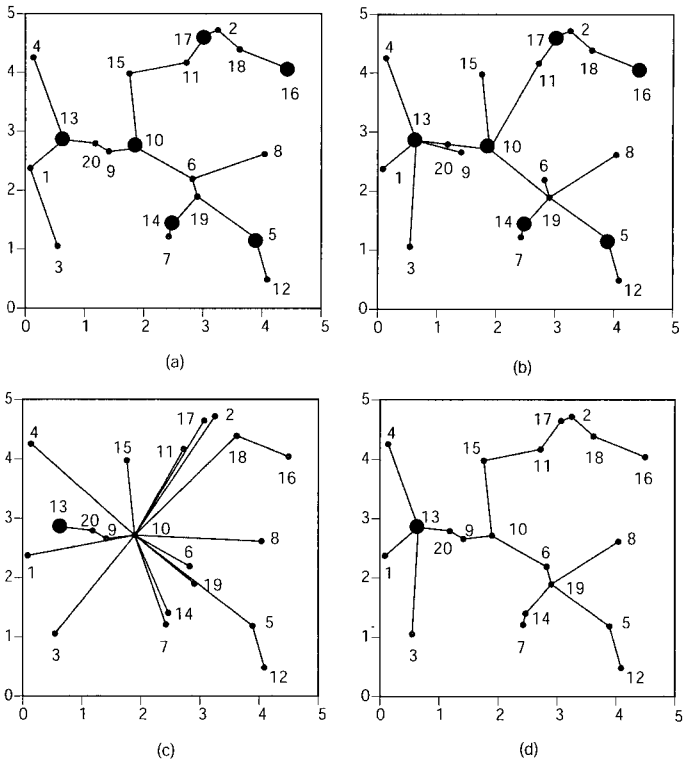



Fig. 12. Trees under different algorithms: (a) MST (power = 12.28), (b) Algorithm B1 with $H = 3$ (power = 9.936), (c) Algorithm B (power = 8.326), (d) BIP (power = 10.1).

**Remark 5.** (a) Because $L_s = 0$, which is a multiple of $H$, the global source is also a local source. If $n$ is both a local source and a leaf node of the MST, then $Z_n(H) = \{n\}$.

(b) Recall that $1 \leq H \leq N$. Suppose that $H = 1$, then $L_i$ is a multiple $H$ for all $i$. Then all the nodes of the MST are local sources, i.e., the number of local sources is $N$ when $H = 1$. Suppose that $H = N$, then $L_i$ cannot be a multiple of $H$ when $i \neq s$, because $1 \leq L_i \leq N - 1$ when $i \neq s$. Thus, the global source $s$ is the only local source when $H = N$ (or when $H$ is greater than the maximum level of the MST). In general, it can

be shown that the number local sources increases (decreases) when $H$ decreases (increases). Thus, we can use $H$ to control the number of local sources. Also, smaller (larger) $H$ yields smaller (larger) $Z_n(H)$.

(c) One of the reasons why a planar Euclidean MST is chosen to generate the subsets $Z_n(H)$ is the well-known fact that each non-source node $n$ of the MST has at most 5 children (the source node $s$ has at most 6 children). From this fact, we have

$$|Z_s(1)| \leq 7 \text{ and } |Z_n(1)| \leq 6 \tag{12}$$

for $n \neq s$. Note that the upper bounds in (12) are constants independent of $N$. The constant bounds may not hold for other trees, e.g., BIP can produce a star tree, in which $N-1$ nodes are children of the source node, i.e., $|Z_s(1)| = N$. From (12), it can be shown that for any given $H$, there exists a constant $E$ such that $|Z_n(H)| \leq E$ for all $n$ and $N$. Note that $E$ depends only on $H$ and not on $n$ or $N$. This property is used later to determine the complexity of Algorithm B1.

(d) Let $T$ now be the MST constructed in phase 1 of Algorithm B1. So far a single $H$ is used to divide the $N$ nodes of $T$ into the subsets $Z_n(H)$. However, different subsets can be constructed using different $H$'s. For example, consider a subset $Z_n(H)$ with $H > 1$. This subset is spanned by a smaller MST that is a subtree of $T$ and is rooted at $n$. Let $H_1 < H$ and $M = |Z_n(H)|$. We can then consider $n$ as the new global source, and use $H_1$ to divide the $M$ nodes of the subtree into $L$ smaller subsets $Z_{n_1}(H_1), Z_{n_2}(H_1), \cdots, Z_{n_L}(H_1)$, where $n_1, n_2, \cdots, n_L$ are new local sources (for some $L > 1$). Thus, any larger subset $Z_n(H)$ can be further divided into smaller subsets to which Algorithm B can be applied. In summary, we can use multiple values $H_j$ to divide the $N$ nodes of the MST into subsets $Z_{n_i}(H_j)$.

(e) Algorithm B1 has 2 phases. In phase 1, the MST is used to produce the subsets $Z_n(H)$, then in phase 2, Algorithm B is applied to these subsets to form subtrees. However, Algorithm B1 can be extended to allow other algorithms to be used in either phases, e.g., BIP can be used in phase 2 (the MST is used in phase 1 as before), or BIP can be used in phase 1 (and Algorithm B is used in phase 2). Also, different algorithms can be applied to different subsets. Some subsets can choose not to participate in phase 2 (e.g., due to lack of computational resources) and use the MST subtrees already computed in phase 1, while other subsets (e.g., with enough computational resources) can run more expensive algorithms (e.g., the optimal algorithm OPT) to produce better subtrees.

*1) Computational complexity*: When Algorithm B is applied to the $N$ nodes, it has computational complexity $O(N^4)$. Under Algorithm B1, the $N$ nodes are divided into smaller subsets to which Algorithm B is applied. In phase 1 of Algorithm B1, the MST is constructed and then the subsets $Z_n(H)$ are determined. This phase can be done in time $O(N^2)$. Let us now study the computational complexity of Algorithm B1 when $N$ is large. First, suppose that $H$ is a small constant that is independent of $N$ (e.g., $H \leq 5$). Then, by Remark 5(c), we have $|Z_n(H)|^4 \leq K$, for some constant $K$ independent of $N$. When Algorithm B is applied to each subset $Z_n(H)$, it has complexity $O(|Z_n(H)|^4) \leq O(K)$, which is also a constant independent of $N$. Because $H$ is a small constant, by Remark 5(b), the number of subsets is large, i.e., $O(N)$. When Algorithm B is applied to these $O(N)$ subsets, the resulting complexity is $O(N)O(K) = O(N)$. Thus, the complexity of Algorithm B1 is $O(N^2) + O(N) = O(N^2)$ when $H$ is a small constant. Next, suppose that $H$ is large. Then, by Remark 5(b), the number of the subsets is small, i.e., $O(1)$, and $|Z_n(H)| = O(N)$ for some local source $n$. Then the complexity of Algorithm B1 is $O(N^2) + O(1)O(N^4) = O(N^4)$. In summary, we show that the computational complexity of Algorithm B1 is between $O(N^2)$ (when $H$ is small) and $O(N^4)$ (when $H$ is large) for large $N$.

*2) Distributed implementation*: A centralized algorithm (such as BIP or Algorithm B) can be implemented as follows. In one method, a designated node gathers the full network information, then computes the tree, and finally forwards the tree to all other nodes. In another method, every node gathers the full network information. Then each node independently computes the desired tree, i.e., there is no need to forward the computed tree to other nodes. Thus, to implement a centralized algorithm, at least one node must know the full network information. In the following, we show that Algorithm B1 can be considered as a distributed algorithm because each node needs to know information about only other nodes that are $H$ levels away (i.e., only partial network information is needed).

First, in phase 1 of Algorithm B1, a distributed algorithm is used to construct the MST [4, p. 391]. Then each node of the MST determines its level as follows. If it is the (global) source node, its level is 0. Each node learns its level from its parent's level, i.e., its level is its parent's level plus 1. Further, a node becomes a local source if its level is a multiple of $H$. We now assume that each node knows information about its descendant nodes that are within $H$ levels from it. That is, each node $n$ only needs to know information about every node $i$ that is a descendant of $n$ and $L_i \leq L_n + H$, i.e., node $n$ knows only partial network information. We have $L_n < L_i$ because $i$ is a descendant of $n$. Thus, each node $n$ knows all the nodes in $\{i : L_n \leq L_i \leq L_n + H\}$, which is $Z_n(H)$ if $n$ is also a local source node. In summary, each local source node $n$ knows information about the nodes in the subset $Z_n(H)$. It then runs Algorithm B on $Z_n(H)$ to compute a subtree that is then forwarded to all other nodes in $Z_n(H)$. Note that each local source can compute its own subtree concurrently and independently from other local sources. Finally, the resulting subtrees are automatically joined at appropriate local sources to produce the final global tree that is connected and loop-free.

*3) Performance evaluation*: Algorithm B1 is a lower complexity and distributed version of Algorithm B (for small $H$). Algorithm B2, to be discussed later, is also a lower complexity but centralized version of Algorithm B. The performance of Algorithm B1 for $N = 100$ is shown in the 2nd column of Fig. 13(a) with $H$ ranging from 2 to 30. For comparison, we also show the performance of the MST algorithm, BIP, and Algorithm B. As before, the best is still Algorithm B and the worst is still the MST algorithm. Note that Algorithm B1 is closest to the MST algorithm when $H = 2$, and closest to Algorithm B when $H = 30$ (see also Fig. 12). Further, Algorithm B1 outperforms BIP when $H \geq 5$. Finally, we observe (but do not show in Fig. 13(a)) that Algorithm B1 and Algorithm B perform identically when $H \geq 40$.

### B. Algorithm B2

Under Algorithm B, a tree is constructed from scratch, i.e., the tree initially consists of only the source node, then other nodes are added into the tree at subsequent steps. The construction stops when the tree contains all $N$ nodes. We now present an alternative algorithm called Algorithm B2, in which we select an initial tree (e.g., MST), and then we attempt to improve this initial tree to yield a new tree of lower cost. The basic idea is as follows. At each step of the algorithm, we always refer to a "benchmark tree" (which is initially set equal to the initial tree). At each step, we transform the benchmark tree into another tree called a "test tree". Then the test tree and the benchmark tree are compared. If the test tree's cost is higher than the benchmark tree's cost, then the test tree is discarded. If the test tree's cost is lower than the benchmark tree's cost, then the test tree becomes the new benchmark tree (the old benchmark tree is then discarded). Thus, the new benchmark tree is no worse than the old benchmark tree at each step. Then the above procedure is repeated at the next step, and so on. The algorithm stops after a specified number of steps is reached. The result is the final (benchmark) tree that is no worse than the initial tree. We need the following remarks to describe the algorithm more precisely.

**Remark 6.** (a) Recall that, for a given tree, each non-source node has a unique parent (the source node has no parent). A tree rooted at source $s$ is specified by a vector $(A_i, i \neq s)$, where $A_i$ denotes the parent of node $i$, i.e., if $k = A_i$, then node $k$ is the parent of node $i$. Thus, by changing the parents of some nodes of a tree, we can transform it into another tree.

(b) Let $T_b$ be a tree specified by a parent vector $(A_i, i \neq s)$. We can use the following rule to transform $T_b$ into another tree. Let $i$ and $k$ be 2 different nodes of $T_b$ such that $i \neq s$ and $k$ is not a descendant of $i$. Let $j$ be the parent of $i$, i.e., $j = A_i$. Next, we set $A_i = k$, i.e., node $k$ becomes the new parent of $i$, and $j$ is no longer the parent of $i$. This parent-switching method is also used in [8] and [10]. In effect, node $k$ now adjusts its power to reach node $i$. Due to the broadcast wireless medium, this power also reaches any node $m$ such that $p_{km} \leq p_{ki}$. Thus, we also set $A_m = k$ for any such node $m$, provided that $k$ is not a descendant of $m$. In summary, we have a simple scheme, in which node $k$ becomes the new parent of node $i$ (and possibly also the parent of some other nodes $m$). The result is a new tree $T$, which is derived from $T_b$ and has no loop because we assume that previously $k$ is a descendant of neither $i$ nor $m$ (as determined from $T_b$). Clearly, this new tree is desirable only if cost($T$) < cost($T_b$). It can be shown that the parent-switching scheme and computing the cost (power) of the resulting new tree can be done in time $O(N)$ for a wireless network of $N$ nodes.

(c) Let $k$ be a node of the tree $T_b$ above, $1 \leq k \leq N$. We now use the scheme in Remark 6(b) above to assign $k$ as the new parent of some other nodes. Denote $k(i)$ as the $i$-th node closest to $k$, e.g., $k(1)$ is the closest node to node $k$, and $k(N-1)$ is the farthest node from node $k$. First, we assign $A_{k(1)} = k$ if $k$ is currently not a descendant of $k(1)$. Next, we assign $A_{k(2)} = k$ if $k$ is currently not a descendant of $k(2)$, and so on. Finally, we assign $A_{k(N-1)} = k$ if $k$ is currently not a descendant of $k(N-1)$. Thus, there are at most $N - 1 = O(N)$ assignments for each $k$. Because $1 \leq k \leq N$, there are $O(N)$ such $k$'s. Then the total number of assignments is $O(N) \times O(N) = O(N^2)$. In summary, we have a simple procedure (to be used in the following Algorithm B2), in which a tree ($T_b$) is transformed into another tree ($T$). There are $O(N^2)$ such procedures.

**Algorithm B2:** Let $T_{init}$ be a tree spanning $N$ nodes. Our goal is to transform $T_{init}$ into a new tree that is no worse than $T_{init}$ (in many cases, the new tree is a better tree). At each step of the algorithm, we refer to a benchmark tree $T_b$ (initially, $T_b = T_{init}$). Algorithm B2 has $O(N^2)$ steps. Each step implements the procedure described in Remark 6(c), where the benchmark tree $T_b$ is transformed into a new tree $T$. If $d(T_b) \leq d(T)$, then the new tree $T$ is discarded. If $d(T_b) > d(T)$, then we set $T_b = T$, i.e., when the new tree has lower cost, the old benchmark is replaced by the new tree (the old benchmark tree is then discarded). Thus, a better tree is found at a step where the benchmark tree is updated. The algorithm stops after $O(N^2)$ steps, and the result is the final benchmark tree $T_b$ with $d(T_b) \leq d(T_{init})$. Because each step can be done in time $O(N)$ by Remark 6(b), the overall complexity of Algorithm B2 is $O(N^2) \times O(N) = O(N^3)$.

*1) Performance evaluation*: We can view Algorithm B2 as a system whose input is a tree ($T_{init}$). The output is another tree ($T_b$) whose cost is no greater than the input tree cost. Thus, Algorithm B2 can be used to improve the performance of trees. As seen in the following numerical illustration, the level of the improvement depends on the types of input trees. Certainly, the optimal trees cannot be improved by Algorithm B2 or by any other algorithms. The performance of Algorithm B2 for the input trees constructed by the MST algorithm, BIP, Algorithm B, and Algorithm B1 (for various values of $H$) is given in the 3rd column of Fig. 13(a) for $N = 100$. In particular, from the 1st row, when the input trees are MSTs (whose absolute power is 11.53), the absolute power under Algorithm B2 is 10.14, yielding an improved performance of about 13.7%. But, from the 3rd row, when the input trees are constructed by Algorithm B, the absolute power before and after the application of Algorithm B2 is 9.926 and 9.818, respectively, i.e., the performance improvement is only about 1.1%. As expected, the improvement of Algorithm B after the application of Algorithm B2 is small because both algorithms are similar in the sense that they aim either to minimize or to reduce the target cost directly. In Algorithm B2, the target cost is evaluated as in Algorithm B. Thus, Algorithm B2 (of complexity $O(N^3)$) is closely related to Algorithm B, and is considered as a lower complexity version of Algorithm B (of complexity $O(N^4)$).

As seen in the 2nd column of Fig. 13(a), before Algorithm B2 is applied, the performance difference among the algorithms is greater, i.e., the normalized power varies from 1.008 (Algorithm B) to 1.172 (MST). In contrast, after Algorithm B2 is applied (see the 3rd column), the performance difference is smaller, i.e., the normalized power varies from 1.019 (Algorithm B) to 1.053 (MST). Note also that, with the application of Algorithm B2, the improved MST (with absolute power 10.14) outperforms the original BIP (with absolute power 10.81) by about 6.6%.

Let $T$ be a tree constructed from scratch by some algorithm called Algorithm X. For example, Algorithm X can be the MST algorithm, BIP, Algorithm B, or Algorithm B1. We can then apply Algorithm B2 to the tree $T$ to yield the combined algorithm

| Algo | Without Algo B2 | With Algo B2 |
|---|---|---|
| MST | 11.53 (0.645)<br>1.172 (0.004) | 10.14 (0.527)<br>1.053 (0.001) |
| BIP | 10.81 (0.536)<br>1.100 (0.004) | 9.983 (0.510)<br>1.037 (0.001) |
| Algo B | 9.926 (0.558)<br>1.008 (0.000) | 9.818 (0.534)<br>1.019 (0.001) |
| Algo B1<br>$H$=2 | 11.20 (0.641)<br>1.139 (0.004) | 10.04 (0.477)<br>1.043 (0.001) |
| Algo B1<br>$H$=3 | 10.94 (0.557)<br>1.112 (0.003) | 9.966 (0.546)<br>1.035 (0.001) |
| Algo B1<br>$H$=4 | 10.82 (0.582)<br>1.100 (0.003) | 9.981 (0.509)<br>1.037 (0.001) |
| Algo B1<br>$H$=5 | 10.67 (0.580)<br>1.084 (0.003) | 9.976 (0.533)<br>1.036 (0.001) |
| Algo B1<br>$H$=10 | 10.41 (0.646)<br>1.057 (0.002) | 9.927 (0.504)<br>1.031 (0.001) |
| Algo B1<br>$H$=20 | 10.17 (0.580)<br>1.033 (0.001) | 9.895 (0.500)<br>1.028 (0.001) |
| Algo B1<br>$H$=30 | 9.964 (0.578)<br>1.012 (0.000) | 9.821 (0.527)<br>1.020 (0.001) |

(a)

| Algo | Without Algo B2 | With Algo B2 |
|---|---|---|
| MST | 11.04 (0.083)<br>1.161 (0.001) | 9.874 (0.096)<br>1.056 (0.001) |
| BIP | 10.30 (0.063)<br>1.083 (0.001) | 9.587 (0.057)<br>1.025 (0.000) |
| Algo B | 9.515 (0.046)<br>1.000 (0.000) | 9.402 (0.044)<br>1.005 (0.000) |
| Algo B1<br>$H$=2 | 10.70 (0.077)<br>1.125 (0.001) | 9.753 (0.072)<br>1.043 (0.001) |
| Algo B1<br>$H$=3 | 10.52 (0.071)<br>1.106 (0.001) | 9.695 (0.067)<br>1.036 (0.000) |
| Algo B1<br>$H$=4 | 10.38 (0.076)<br>1.092 (0.001) | 9.665 (0.072)<br>1.033 (0.000) |
| Algo B1<br>$H$=5 | 10.31 (0.071)<br>1.084 (0.001) | 9.633 (0.052)<br>1.030 (0.000) |
| Algo B1<br>$H$=10 | 10.08 (0.062)<br>1.060 (0.000) | 9.590 (0.059)<br>1.025 (0.000) |
| Algo B1<br>$H$=20 | 9.886 (0.063)<br>1.039 (0.000) | 9.550 (0.056)<br>1.021 (0.000) |
| Algo B1<br>$H$=30 | 9.798 (0.060)<br>1.030 (0.000) | 9.533 (0.053)<br>1.019 (0.000) |

(b)

Fig. 13. Absolute and normalized broadcast tree power (and their variances), $a = 2$: (a) $N = 100$ nodes, (b) $N = 500$ nodes.

(Algorithm X, Algorithm B2). Let $O(g_1(N))$ and $O(g_2(N))$ be the computational complexity of Algorithm X and the combined algorithm, respectively. Because the complexity of Algorithm B2 is $O(N^3)$, we have $O(g_2(N)) = O(g_1(N)) + O(N^3) = \max\{O(g_1(N)), O(N^3)\}$. Recall from Section V-A.1 that the complexity of Algorithm B1 is between $O(N^2)$ (when $H$ is small) and $O(N^4)$ (when $H$ is large). When Algorithm X is the MST algorithm, BIP, or Algorithm B1 with small $H$, we have $O(g_1(N)) \leq O(N^3)$, i.e., $O(g_2(N)) = O(N^3)$. When Algorithm X is Algorithm B or Algorithm B1 with large $H$, we have $O(g_2(N)) = O(N^4)$. Thus, as seen in the 3rd column of Fig. 13(a), we have many combined algorithms of complexity $O(N^3)$, which underperform the original Algorithm B of complexity $O(N^4)$ only slightly. For example, the combination of BIP and Algorithm B2 (with absolute power 9.983) underperforms the original Algorithm B (with absolute power 9.926) by about 0.57%. The combination of Algorithm B1 with $H = 3$ (with absolute power 9.966) and Algorithm B2 underperforms the original Algorithm B by about 0.4%.

The performance results for larger networks of $N = 500$ nodes are shown in Fig. 13(b). Again, all algorithms receive the benefit from the application of Algorithm B2. Also, better algorithms such as Algorithm B and Algorithm B1 (with larger $H$) receives less improvement from Algorithm B2 than other algorithms such as MST and BIP. Further, the performance difference among the algorithms before applying Algorithm B2 is noticeably greater than the performance difference after applying Algorithm B2.

**Remark 7.** Algorithms B1 and B2 are originally developed for broadcasting. However, they can also be used for multicasting as follows. Let $T$ be any multicast tree (e.g., SPF or ISPF tree). First, recall that the MST (a broadcast tree) is used in phase 1 of Algorithm B1. Suppose that the MST is replaced by the multicast tree $T$, then Algorithm B1 will produce a multicast tree. Next, recall that any tree can be used as input to Algorithm B2. Suppose that the input to Algorithm B2 is the multicast tree $T$, then the output is another multicast tree that is no worse than $T$.

## VI. SUMMARY

We present Algorithm B for constructing broadcast trees, and then we modify it to yield Algorithm M for multicasting. Both algorithms are general because they are meant for arbitrary nonnegative cost functions. Thus, they are ready to serve as baseline algorithms for any existing cost functions as well as any new cost functions that may arise in future applications. As illustrative examples, we apply these general algorithms to the problem of broadcast and multicast routing in energy-aware wireless networks, which recently receives great attention. We show the performance of these algorithms for the use of omnidirectional and directional antennas. We also address the impact of constraints on resources such as energy, transceivers, and frequencies. These general algorithms improve on some existing algorithms (such as MST, BIP, MIP, SPF, and D-MIP).

A disadvantage of these general algorithms is their high computational complexity. In particular, for networks of $N$ nodes, the complexity of Algorithm B is $O(N^4)$, while that of MST and BIP is $O(N^2)$ and $O(N^3)$, respectively. Another disadvantage is that they are centralized algorithms, requiring the full network information for tree construction. Thus, we adapt these baseline algorithms to yield 2 lower complexity algorithms. The first is Algorithm B1, under which the entire network is divided into smaller subsets whose sizes are controlled by the param-

eter $H, 1 \leq H \leq N$. The performance and complexity of Algorithm B1 are somewhere between those of the MST and Algorithm B, depending on whether $H$ is small or large. Algorithm B1 has distributed implementation when $H$ is small, i.e., only partial network information is needed for tree construction.

Algorithm B2 is another lower complexity algorithm that has complexity $O(N^3)$ for a given input tree. Algorithm B2 is effective because when combining with MST, BIP, or some Algorithm B1 trees, it yields algorithms of complexity $O(N^3)$, which underperform the original Algorithm B of complexity $O(N^4)$ only slightly. Although both Algorithms B1 and B2 are lower complexity versions of Algorithm B, the former is a distributed algorithm and the latter is a centralized one.

## ACKNOWLEDGMENT

This work was supported by the Office of Naval Research.

## REFERENCES

[1] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Energy-aware wireless networking with directional antennas: The case of session-based broadcasting and multicasting," *IEEE Trans. Mobile Computing*, vol. 1, no. 3, pp. 176–191, July–Sept. 2002.

[2] G. D. Nguyen, "Construction of broadcast and multicast trees with arbitrary cost functions," in *Proc. Conf. Inform. Sci. Syst.*, Mar. 2002, pp. 1046–1051.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1993.

[4] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

[5] D. S. Hochbaum, *Approximation Algorithms for NP-Hard Problems*, Boston, MA: PWS Publishing Company, 1997.

[6] A. Ahluwalia, E. Modiano, and L. Shu, "On the complexity and distributed construction of energy-efficient broadcast trees in static ad hoc wireless networks," in *Proc. Conf. Inform. Sci. Syst.*, Mar. 2002, pp. 807–813.

[7] F. Bian, A. Goel, C. S. Raghavendra, and X. Li, "Energy-efficient broadcasting in wireless ad hoc networks: Lower bounds and algorithms," *J. Interconnection Networks*, vol. 3, no. 3/4, pp. 149–166, Sept./Dec. 2002.

[8] M. Cagalj, J.-P. Hubaux, and C. Enz, "Minimum-energy broadcast in all-wireless networks: NP-completeness, and distribution issues," in *Proc. MOBICOM 2002*, Sept. 2002, pp. 172–182.

[9] A. E. F. Clementi, P. Crescenzi, P. Penna, P. Rossi, and P. Vocca, "On the complexity of computing minimum energy consumption broadcast subgraphs," in *Proc. Symp. Theoretical Aspects Computer Sci.*, 2001, pp. 121–132.

[10] F. Li and I. Nikolaidis, "On minimum-energy broadcasting in all-wireless networks," in *Proc. IEEE Local Computer Networks*, Nov. 2001, pp. 193–202.

[11] W. Liang, "Constructing minimum-energy broadcast trees in wireless ad-hoc networks," in *Proc. MOBIHOC 2002*, June 2002, pp. 112–122.

[12] P.-J. Wan, G. Calinescu, and C.-W. Yi, "Minimum-power multicast routing in static ad hoc wireless networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, pp. 507–514, June 2004.

[13] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder, "Minimum-energy broadcast in static ad hoc wireless networks," in *Proc. IEEE INFOCOM 2001*, Apr. 2001, pp. 1162-1171.

[14] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. IEEE INFOCOM 2000*, Mar. 2000, pp. 585–594.

[15] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 1981.

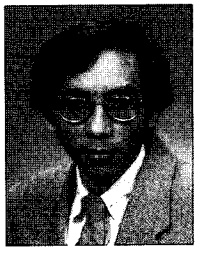
[16] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[17] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Inform. Processing Lett.*, vol. 27, pp. 125–128, Mar. 1988.

[18] A. Ephremides, "Energy concerns in wireless networks," *IEEE Wireless Commun.*, vol. 9, no. 4, pp. 48–59, Aug. 2002.

[19] A. Ephremides, "Ad-hoc networks: Not an ad-hoc field any more," *Wiley J. Wireless Commun. Mobile Computing*, vol. 2, no. 5, pp. 441–448, Aug. 2002.

[20] I. Stojmenovic, *Handbook of Wireless Networks and Mobile Computing*, New York: John Wiley & Son, Inc., 2002.



**Gam D. Nguyen** received the Ph.D. in electrical engineering from the University of Maryland, College Park, MD in 1990. He has been at the Naval Research Laboratory, Washington, DC, since 1991. His research interests include communication systems and networks.