Energy-aware Broadcasting in Wireless Networks

Ioannis Papadimitriou, Leonidas Georgiadis

Energy-aware Broadcasting in Wireless Networks

Ioannis Papadimitriou, Leonidas Georgiadis

***Abstract*—In this paper we address the problem of broadcasting in wireless networks, so that the power consumed by any node is as small as possible. This approach is motivated by the fact that nodes in such networks often use batteries and, hence, it is important to conserve energy individually, so that they remain operational for a long time. We formulate the problem as a *lexicographic* node power optimization one. The problem is in general NP-complete. We provide an optimal algorithm which runs in polynomial time in certain cases. We also provide a heuristic algorithm whose performance relative to the optimal one is fairly satisfactory. We next show that these algorithms can also be used to solve the problem of broadcasting so that the remaining battery lifetime of any node is as large as possible. Finally, we discuss the issues of implementing the above algorithms distributively, as well as their multicast extensions.**

***Index Terms*— Wireless Networks, Energy Conservation, Directed Spanning Tree, Lexicographic Optimization.**

I. Introduction

Most wireless devices today are portable and operate on batteries with finite amount of energy. Hence, it becomes imperative to take the issue of power management into account when designing a wireless network [1]. In this paper we address the problem of energy-aware broadcasting in wireless networks. In a wired environment, broadcasting is a well understood problem and polynomial algorithms exist for its solution [2], [3]. A common approach is to associate a cost with each link and determine a spanning tree whose sum of link costs is minimal. However, as indicated in [4], broadcasting in a wireless environment where omnidirectional antennas are used, must take into account the fact that a node's transmission can reach multiple neighbors at the same time. Hence, the power needed to reach a node's set of neighbors is the *maximum* of the powers needed to reach each of the neighbors separately. Since the efficient management of battery power is sought, the objective then is to determine a set of retransmitting nodes and their corresponding transmission powers such that a measure of consumed node power is optimized. In [4] the optimization objective was considered to be the sum of consumed node powers needed to convey the information from a given source to all destination nodes. The problem is NP-complete and heuristic algorithms were proposed and studied in [4], [5], [6]. In this paper we address the broadcast problem in wireless networks with the objective of ensuring that no node consumes excessive power, or no node's battery lifetime is reduced significantly. This is a reasonable objective in an environment where each node has its own batteries and it is important to keep every node operational for as long as possible. Minimizing the sum of node powers consumption in the network does not necessarily guarantee that each retransmitting node will consume a small amount of power.

Unlike much of previous work, where the wireless network is usually modeled as an undirected graph, the graphs we consider are directed. Our model does not necessarily imply the absence of bidirectional links between two nodes (although unidirectional links may exist in a real network [7]). It is rather used as a more realistic modelling of the fact that the power needed for communication between two nodes may not be the same in both directions due to noise or other signal propagation phenomena, and the heterogeneity in transmission hardware of nodes in the network. We first formulate the problem as a *min-max* node power optimization one, where maximum consumed node power is sought to be minimized. Next we look at the stronger optimization criterion where, provided that we succeed in minimizing the i^{th} maximum node power, we also seek to minimize the $(i+1)^{th}$ maximum. This objective is captured by the stronger *lexicographic* node power optimization problem (see Section II-B). The general problem is NP-complete. We present an optimal algorithm which is polynomial in certain cases. The algorithm runs in reasonable time for moderate size random networks but, as expected, its running time quickly deteriorates for larger networks. To deal with networks of larger sizes, we present a heuristic that is based on the steps of the optimal algorithm and avoids the most computing intensive operations. Numerical results show that the proposed heuristic has good running times and provides satisfactory performance relative to the optimal algorithm.

We also address the issue of lexicographic optimization under more general link cost functions. A special case of these functions can be used when the desirable performance metric is the node remaining lifetime. In this case it may be of interest to implement broadcasting in a manner that maximizes the minimum remaining node lifetime (max-min criterion), or maximizes lexicographically the remaining node lifetimes. To elucidate the difference between the two optimization criteria, consider that broadcasting from a given source r takes place until one of the nodes in the network runs out of batteries. Under both max-min and lexicographic criteria, the first time t where a node's lifetime becomes zero (i.e., its battery is exhausted) is maximized. However, under the lexicographic criterion, the number of nodes whose lifetime is zero at time t, is also minimized. Moreover, if only one node's battery is exhausted, then the minimum remaining lifetime of the rest of the nodes is maximized. Besides these two optimization criteria there are several others that can be proposed, e.g. maximization of the sum of node remaining lifetimes, maximization of convex combination of them, etc. The specific criterion to be used depends on the network requirements. An interesting issue is to determine and

I. Papadimitriou and L. Georgiadis are with Electrical and Computer Engineering Dept., Aristotle University of Thessaloniki, Thessaloniki, GREECE. E-mails: ipapad@egnatia.ee.auth.gr , leonid@eng.auth.gr .

I. Papadimitriou was fully supported for this work by the Public Benefit Foundation "ALEXANDER S. ONASSIS", Athens, GREECE.

classify the network environments for which each of these optimization criteria is appropriate (see [8] for one such comparison). While we leave this issue as a subject for further study, we note that in applications where the max-min optimization of lifetime is important, the use of the lexicographic criterion provides solutions with additional desirable properties.

The objective of maximizing the minimum battery lifetime was considered in [9]. In the latter work in effect the reverse problem to ours was considered. That is, each node needs to transfer information to a specific gateway. The broadcast nature of each node's transmission was not considered. Instead, a node was able to transmit to each of its neighbors with varying rate and the consumed node power for each "neighbor-to-neighbor" transmission was proportional to the transmission rate. The Multipoint relaying algorithm, proposed in [10] and used in [11], has also some relevance to our work. An efficient technique is proposed for flooding in wireless networks with limited information, so that the number of nodes that retransmit the information (relay nodes) is minimized. The latter setup can be considered as a special case of ours, by assuming that all nodes need unit power to reach their neighbors (see Section III).

The algorithms we develop assume knowledge of network topology and the powers needed by each node to reach its neighbors. Hence, they can be used in networks with infrequent topological changes and low mobility, such as computer terminals in a university campus or within a company. The issue of centralized versus distributed computation does not have a clear-cut answer [1]. As will be discussed in Section VI, in our setup at least partial optimization can be made in a distributed fashion by implementing distributively a basic algorithmic subroutine. In any case, our algorithms can be directly applied in network environments where at least partial information of network topology is proactively maintained at each node, as in OLSR [11] and ZRP [12]. Another important problem is energy-aware multicasting. The optimal algorithms presented in this paper can be used for the latter problem provided that a basic algorithmic subroutine is designed for multicasting. A simple algorithm for such a subroutine exists, but we believe that further improvements can be made.

The rest of the paper is organized as follows. Section II provides formal definitions and formulation of the min-max and lexicographic node power optimization problems. In Section III we indicate that the latter problem is NP-complete and provide an optimal algorithm which is polynomial in certain cases. We also provide a heuristic. In Section IV we consider generalized link cost functions and prove that the proposed algorithms can also be used to solve the broadcast problem so that the remaining battery lifetime of any node is as large as possible. Numerical results are presented in Section V. In Section VI we discuss the issues of implementing our algorithms in a distributed fashion and using them in the case of multicast communication. Finally, Section VII summarizes the conclusions of our study. All proofs of the lemmas are given in the Appendix.

II. Definitions and Problem Formulation

Consider a directed graph $G = (N, L)$, where N is the set of nodes and L is the set of directed links. For a node $i \in N$ we denote by $L^G_{out}(i)$ ($L^G_{in}(i)$) the set of links outgoing from (incoming to) i. A node j such that link (i, j) belongs to L is called a one-hop neighbor of i or simply a neighbor of i. A node j is a two-hop neighbor of i if j is not a neighbor of i and for some $k \in N$, $(i, k) \in L$ and $(k, j) \in L$. We denote by $N^G_1(i)$ the set of (one-hop) neighbors and by $N^G_2(i)$ the set of two-hop neighbors of node i.

Given a root node $r \in N$, an r-rooted tree $T = (N, L^T)$ spanning G (*spanning* tree for short) is a subgraph of G having the following properties: a) there is a directed path from node r to every other node of G using only the links in L^T, and b) T has $|N| - 1$ links, where $|N|$ is the cardinality of the set N. It follows from the definition of T that: a) for every node $n \neq r$ there is exactly one link in the set L^T terminating at n, b) there is no link in L^T terminating at node r, and c) there are nodes in T that have no outgoing links in L^T. The latter nodes are called leaf nodes of T. Hence, for a leaf node i we have $L^T_{out}(i) = \oslash$.

A. Wireless Communication Model

We model the wireless network as a directed graph $G = (N, L)$. N is the set of nodes in the network. If node j can successfully receive information transmitted by node i, then link (i, j) belongs to the set L of links in G. The power needed for a successful transmission over link $l = (i, j)$ is denoted by $c_l > 0$ and is also referred to as the link cost. Note that we allow for the possibility of asymmetric channels, i.e., it is possible that $c_{(i,j)} \neq c_{(j,i)}$. Each node is equipped with an omnidirectional antenna. Hence, the following useful property holds:

Property 1: *If node i transmits with power p_i, it can reach any node j for which $c_{(i,j)} \leq p_i$.*

Note that during transmission, battery power is also consumed by the receiver of information. For simplicity in the discussion that follows we assume that the receive power is zero. In Section IV we indicate how to incorporate it into the model.

Suppose that node r needs to broadcast information to all other nodes in the network. In this case, we have to determine a set of retransmitting nodes and their corresponding transmission powers, so that eventually all nodes receive the information. A way to achieve this, which will be useful in the sequel, is as follows. We define an r-rooted spanning tree $T = (N, L^T)$ with the following interpretation:

1) Node r transmits with power $p^T_r = \max_{l \in L^T_{out}(r)}\{c_l\}$.

2) Any node n of T receiving the information, retransmits with power $p^T_n = \max_{l \in L^T_{out}(n)}\{c_l\}$, where $\max_{l \in \oslash}\{c_l\} = 0$. We refer to p^T_n as the power induced on node n by tree T.

3) Step 2 is repeated until all non leaf nodes retransmit the information exactly once.

Note that, depending on the link costs, the same broadcast transmissions can be effected by more than one spanning trees. Consider for example the network in Fig. 1 with root node A. The spanning trees T_1 and T_2, with links {(A,B),(B,C),(B,D)} and {(A,B),(A,C),(B,D)} respectively, have the same leaf nodes. Hence, they both specify the same nodes for retransmission (node B in this case). Under link power set I, the node powers induced are the same for both trees, $p^{T_1}_A = p^{T_2}_A = 2$, $p^{T_1}_B = p^{T_2}_B = 4$, $p^{T_1}_C = p^{T_1}_D = p^{T_2}_C = p^{T_2}_D = 0$. Under link power set II, we have $p^{T_1}_A = 2$, $p^{T_1}_B = 6$, $p^{T_1}_C = p^{T_1}_D = 0$, and $p^{T_2}_A = 4$, $p^{T_2}_B = 3$, $p^{T_2}_C = p^{T_2}_D = 0$, and, hence, the node powers induced by the two trees are different.

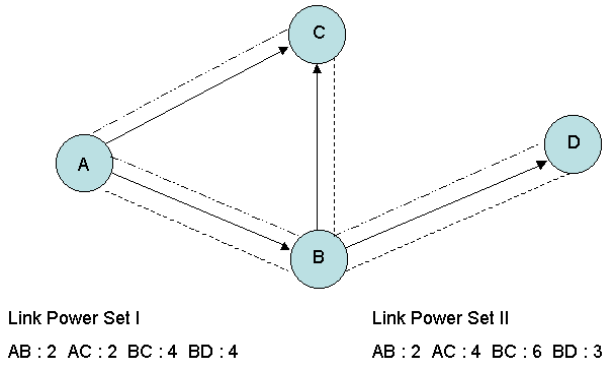

Fig. 1. Example of wireless broadcasting

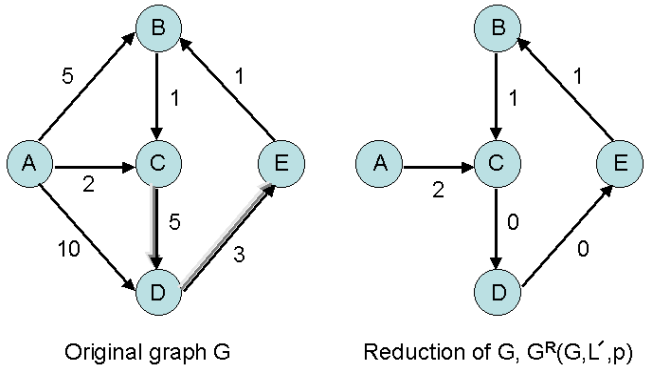

Fig. 2. Example of graph reduction

B. Optimal Broadcast Trees

We are interested in finding a spanning tree for which the maximum induced node power is as small as possible. This requirement is captured by the *min-max* node power optimization problem defined as follows:

Problem 1: (Min-Max Node Power Optimization) *Find a spanning tree* $\overline{T}$ *such that, for any spanning tree* T *of* G*, it holds* $\max_{n\in N}\{p_n^{\overline{T}}\} \le \max_{n\in N}\{p_n^T\}$.

Given a spanning tree T, let $\mathbf{P}^T = (p_n^T)_{n\in N}$ be the vector of induced node powers. A solution to Problem 1 minimizes the maximum consumed node power but does not specify how to treat nodes that consume power smaller than the maximum. As discussed earlier, it may be desirable to also keep the consumed power of the latter nodes as small as possible, so that no node in the network consumes excessive power. This latter requirement is captured by the stronger *lexicographic* node power optimization problem (referred to as min-max problem in [3]) stated below, after the following definition:

Definition 1: *Given an* n*-dimensional real vector* $\mathbf{v}=(v_1, v_2, ..., v_n)$*, define by* $\widehat{\mathbf{v}}$ *the* n*-dimensional vector whose coordinates are those of* $\mathbf{v}$ *arranged in non-increasing order, i.e.,* $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, ..., \widehat{v}_n) = (v_{i_1}, v_{i_2}, ..., v_{i_n})$*, where* $v_{i_1} \ge v_{i_2} \ge ... \ge v_{i_n}$*. Vector* $\mathbf{v}$ *is called lexicographically smaller than or equal to vector* $\mathbf{u}$*, if either* $\widehat{\mathbf{v}} = \widehat{\mathbf{u}}$ *or there exists a number* l*,* $1 \le l \le n$*, such that* $\widehat{v}_i = \widehat{u}_i$ *for* $1 \le i \le l-1$ *and* $\widehat{v}_l < \widehat{u}_l$*. We write* $\mathbf{v} \preceq_{lex} \mathbf{u}$ *and, if in addition* $\widehat{\mathbf{v}} \ne \widehat{\mathbf{u}}$*,* $\mathbf{v} \prec_{lex} \mathbf{u}$*. (For example, the vector* $(3, 4, 8)$ *is lexicographically smaller than* $(5, 8, 2)$*).*

Problem 2: (Lexicographic Node Power Optimization) *Find a spanning tree* T^* *such that, for any spanning tree* T *of* G*, it holds* $\mathbf{P}^{T^*} \preceq_{lex} \mathbf{P}^T$.

While our main interest is in providing algorithms for Problem 2, it will be seen in the sequel that Problem 1 is instrumental in the development of such algorithms. In the sequel, we will need the following notation. Let $\widetilde{p}_i^T$ be the i^{th} *distinct* maximal node power induced by spanning tree T, $\widetilde{S}_i^T$ the set of nodes that have to transmit with power $\widetilde{p}_i^T$, and $\widetilde{k}_i^T = |\widetilde{S}_i^T|$. That is, $\widetilde{p}_{i-1}^T > \widetilde{p}_i^T$, $i = 2, ..., \widetilde{I}^T$, and $p_n^T = \widetilde{p}_i^T$, for all $n \in \widetilde{S}_i^T$. According to this definition, $\widetilde{p}_{\widetilde{I}^T}^T = 0$ and $\widetilde{S}_{\widetilde{I}^T}^T$ is the set of leaf nodes of T. For example, if $\mathbf{P}^T = (4, 8, 8, 0, 3, 0, 0)$, then $\widetilde{p}_1^T = 8$, $\widetilde{S}_1^T = \{2, 3\}$, $\widetilde{p}_2^T = 4$, $\widetilde{S}_2^T = \{1\}$, $\widetilde{p}_3^T = 3$, $\widetilde{S}_3^T = \{5\}$, $\widetilde{p}_4^T = 0$, $\widetilde{S}_4^T = \{4, 6, 7\}$. It follows from the definition that any lexicographically optimal (with respect to node powers) tree T^* has the following property (to avoid special cases, we define $\widetilde{p}_0^T = \infty$ and $\widetilde{k}_0^T = 0$):

Lemma 1: *For any spanning tree* T *of* G*, if for some* $l < \widetilde{I}^{T^*}$ *it holds* $\widetilde{p}_i^T = \widetilde{p}_i^{T^*}$ *and* $\widetilde{k}_i^T = \widetilde{k}_i^{T^*}$ *for all* i *such that* $0 \le i \le l$*, then* $\widetilde{I}^T > l$*. Moreover, either* $\widetilde{p}_{l+1}^{T^*} < \widetilde{p}_{l+1}^T$*, or* $\widetilde{p}_{l+1}^{T^*} = \widetilde{p}_{l+1}^T$ *and* $\widetilde{k}_{l+1}^{T^*} \le \widetilde{k}_{l+1}^T$.

It follows that all lexicographically optimal trees have the same $\widetilde{I}^{T^*}$, $\widetilde{p}_i^{T^*}$, and $\widetilde{k}_i^{T^*}$, that is, if T^* is any lexicographically optimal tree, then $\widetilde{I}^{T^*} = \widetilde{I}^*$, $\widetilde{p}_i^{T^*} = \widetilde{p}_i^*$ and $\widetilde{k}_i^{T^*} = \widetilde{k}_i^*$, for $1 \le i \le \widetilde{I}^*$. Finally, for the analysis in the next sections, we need to define the following transformation of a graph $G(N, L)$:

Definition 2: *Let* $L' \subseteq L$ *and* $p \ge 0$ *be given. Eliminate from* G *all links in* $L - L'$ *with power larger than or equal to* p*. Let* L^R *be the set of remaining links after the elimination. Modify the link powers in* L^R *as follows:* $c_l^R = 0$*, if* $l \in L'$*, and* $c_l^R = c_l$*, otherwise. The resulting graph is called the "reduction" of* G *and is denoted by* $G^R(G, L', p)$.

Fig. 2 shows an example of graph reduction, given that L' ={(C,D),(D,E)} and $p = 3$ in the original graph G. This example also shows the importance of Problem 2. Assume that the root node is A and consider the spanning tree $\overline{T}$ of G with links {(A,B),(A,C),(C,D),(D,E)}. Tree $\overline{T}$ solves Problem 1 and induces a vector of node powers, $(p_A^{\overline{T}}, p_B^{\overline{T}}, p_C^{\overline{T}}, p_D^{\overline{T}}, p_E^{\overline{T}}) = (5, 0, 5, 3, 0)$, which is lexicographically larger than that induced by the spanning tree T^* with links {(A,C),(C,D),(D,E),(E,B)}, $(p_A^{T^*}, p_B^{T^*}, p_C^{T^*}, p_D^{T^*}, p_E^{T^*}) = (2, 0, 5, 3, 1)$.

III. Optimization Algorithms

To avoid cluttering the discussion, in the following we make the assumption that there is an r-rooted spanning tree. There are well known algorithms that test for the existence of such a tree in a given network (see e.g. [13]). Let us first address Problem 1. We have

$$\max_{n\in N}\{p_n^T\} = \max_{n\in N}\{\max_{l\in L_{out}^T(n)}\{c_l\}\} = \max_{l\in L^T}\{c_l\}. \qquad (1)$$

Hence, finding the spanning tree that minimizes the maximum induced node power is equivalent to finding the tree that minimizes the maximum link power. For the latter problem, known

also as *bottleneck* optimization problem, polynomial time algorithms exist [14], [15], [16]. Note that since any lexicographically optimal tree T^* solves Problem 1, the value of any solution to Problem 1 is $\widetilde{p}_1^{T^*} = \widetilde{p}_1^*$.

We now address Problem 2. A special case of this problem is NP-complete. To see this, assume that all link costs of G are equal, that is, $c_l = c > 0$, for any $l \in L$. Then, since each node has power either c or 0, it follows from the definition of the lexicographically optimal point that the solution to the problem is a spanning tree T^* with the following property; the number of non zero elements of the induced vector of node powers $\mathbf{P}^{T^*}$ (i.e., the number of non leaf nodes of T^*) is minimized. Hence, finding T^* is equivalent to finding the smallest set of nodes that will serve as relays to pass the information from the root node r to all other network nodes. This set is referred to as the optimal MultiPoint Relay (MPR) set in [10]. Consider now a graph G consisting of a node $r \in N$, the set $N_1^G(r)$ (one-hop neighbors of r), the set $N_2^G(r)$ (two-hop neighbors of r) and with bidirectional links. It is proved in [10] that for this graph, finding an optimal MPR set for node r is NP-complete.

According to the discussion above, an efficient algorithm for Problem 2 is very unlikely to exist. In the following, we first provide an optimal polynomial time algorithm under the condition that different nodes need different powers to reach their one-hop neighbors (see Condition 1 below for the exact statement). Next we will extend this algorithm to provide an optimal solution for the general case. In the latter case, as expected, the worst case running time of the algorithm is exponential in the size of the network. As will be seen in Section V, the derived algorithm has reasonable running times for moderate size random networks, where it can be employed to test the efficiency of proposed heuristics. Moreover, as will be discussed in Section III-C, this algorithm can be used as a basis for various heuristic algorithms that can be employed in larger networks.

A. Optimal Algorithm for a Special Case

We start with the optimal polynomial time algorithm. Assume that the following condition holds in the network:

Condition 1: *The powers of links outgoing from different nodes are different. That is, $c_{(i,n_1)} \neq c_{(j,n_2)}$, for all $i, j \in N$, $i \neq j$, $n_1 \in N_1^G(i)$, $n_2 \in N_1^G(j)$.*

Note that we allow for the possibility that links outgoing from the same node have equal power. As discussed above, $\widetilde{p}_1^*$ can be obtained in polynomial time for any network. According to Condition 1, there is a single node that needs power $\widetilde{p}_1^*$ to reach some of its neighbors and can therefore be identified. In general, assume that we know $\widetilde{p}_i^*$ for $1 \leq i \leq m < \widetilde{I}^* - 1$, which, according to Condition 1, implies that we know $\widetilde{S}_i^*$, $1 \leq i \leq m$. Note that each $\widetilde{S}_i^*$ contains a single node. We are interested in finding $\widetilde{p}_{m+1}^*$. Let $\widetilde{L}_i$ be the set of links whose power is less than or equal to $\widetilde{p}_i^*$ and emanate from the node in $\widetilde{S}_i^*$. Define also $\widetilde{L}^m = \cup_{i=1}^m \widetilde{L}_i$ and $\widetilde{S}^m = \cup_{i=1}^m \widetilde{S}_i^*$. The search for $\widetilde{p}_{m+1}^*$ is based on the following lemma:

Lemma 2: *Consider the solution to Problem 1 on the reduction of G, graph $G^{m+1} = G^R(G, \widetilde{L}^m, \widetilde{p}_m^*)$. The value of this solution is $\widetilde{p}_{m+1}^*$ and the min-max optimal tree $\overline{T}^{m+1}$ has $\widetilde{p}_i^{\overline{T}^{m+1}} = \widetilde{p}_i^*$ for $1 \leq i \leq m+1$.*

Algorithm 1:
1. $m = 0, G^m = G, \widetilde{L}^m = \oslash, \widetilde{p}_m^* = \infty$.
2. Form the reduction of G^m, $G^{m+1} = G^R(G^m, \widetilde{L}^m, \widetilde{p}_m^*)$.
3. Solve the min-max optimization problem on G^{m+1} and let $\overline{T}^{m+1}$ be this solution. The maximum induced node power of $\overline{T}^{m+1}$ in G^{m+1} is $\widetilde{p}_{m+1}^*$.
4. If $\widetilde{p}_{m+1}^* = 0$, then Return $\overline{T}^{m+1}$ (T^* is equal to $\overline{T}^{m+1}$).
5. Else, identify the node in G^{m+1} that has at least one outgoing link with power $\widetilde{p}_{m+1}^*$. Let $\widetilde{L}_{m+1}$ be the set of links whose power is less than or equal to $\widetilde{p}_{m+1}^*$ and emanate from this node.
6. $\widetilde{L}^{m+1} = \widetilde{L}^m \cup \widetilde{L}_{m+1}$, $m = m+1$. Go to step 2.

Fig. 3. Optimal Algorithm for a Special Case

It follows from Lemma 2 that, if we know $\widetilde{p}_i^*$ for $1 \leq i \leq m < \widetilde{I}^* - 1$, we can find $\widetilde{p}_{m+1}^*$ by solving Problem 1 on G^{m+1}. At the same time, this solution provides a tree $\overline{T}^{m+1}$ for which we have $p_n^{\overline{T}^{m+1}} = p_n^{T^*}$ for $n \in \widetilde{S}^m$. If we find that $\widetilde{p}_{m+1}^* = 0$, then we know that $\overline{T}^{m+1} = T^*$. Otherwise, we can repeat the procedure to find $\widetilde{p}_{m+2}^*$. Also note that, as follows from the definitions, we have $G^{m+1} = G^R(G^m, \widetilde{L}^m, \widetilde{p}_m^*)$. Hence, we have the algorithm of Fig. 3 for finding T^*, under Condition 1.

Complexity of Algorithm 1: Problem 1 at step 3 of the algorithm can be solved in $O(|N| \log |N| + |L|)$ time (see e.g. [15]). The other steps take time $O(|L|)$ for one iteration and, since all steps of Algorithm 1 are invoked at most $|N|$ times, its worst case running time is $O(|N|^2 \log |N| + |N||L|)$.

B. Optimal Algorithm for the General Case

Let us now remove Condition 1. According to (1), we can still solve Problem 1 and find the value $\widetilde{p}_1^*$. Since in general there may be many nodes that can reach others with power $\widetilde{p}_1^*$, we must now identify a set of nodes that will transmit with power $\widetilde{p}_1^*$ when a lexicographically optimal tree T^* is implemented. Note that there may be multiple optimal trees and, therefore, multiple optimal node sets. However, according to Lemma 1, the number of nodes in all these sets will be the same, i.e., $\widetilde{k}_1^*$. The search for $\widetilde{k}_1^*$ and the candidate optimal node sets is based on the two lemmas that follow. We denote by $\mathbb{T}^*$ the set of all lexicographically optimal trees. Lemma 3 part 1 provides a method for testing whether a given node set is *not* equal to $\widetilde{S}_1^{T^*}$ for any $T^* \in \mathbb{T}^*$. Lemma 3 part 2 provides a method for finding an upper bound for $\widetilde{k}_1^*$. Lemma 4 provides a method for testing whether a given node set S, such that $|S| = \widetilde{k}_1^*$, is *not* equal to $\widetilde{S}_1^{T^*}$ for any $T^* \in \mathbb{T}^*$.

Lemma 3: *Let S be a set of nodes with the property that each node in S has at least one outgoing link with power $\widetilde{p}_1^*$. Let L_S be the set of links emanating from some node in S and having power less than or equal to $\widetilde{p}_1^*$, i.e., $L_S = \{l \in L : l \in L_{out}^G(n), n \in S, c_l \leq \widetilde{p}_1^*\}$. Consider the reduced graph $G^1 = G^R(G, L_S, \widetilde{p}_1^*)$.*

1) If there is no spanning tree of G^1, then $S \neq \widetilde{S}_1^{T^*}$ for all $T^* \in \mathbb{T}^*$.

2) Else, the value of the solution $\overline{T}^1$ to Problem 1 on G^1 is strictly smaller than $\widetilde{p}_1^*$, $\widetilde{p}_1^{\overline{T}^1} = \widetilde{p}_1^*$ and $\widetilde{k}_1^* \leq |S|$.

Lemma 4: *Let S_1, S_2 be sets of nodes with the property that each node in S_i, $i = 1, 2$, has at least one outgoing link with power $\widetilde{p}_1^*$. Let L_{S_i} be the set of links emanating from some node in S_i and having power less than or equal to $\widetilde{p}_1^*$. Consider the reductions of G, graphs $G_i^1 = G^R(G, L_{S_i}, \widetilde{p}_1^*)$, $i = 1, 2$. Assume that there is at least one spanning tree of G_i^1 and let v_i be the value of the solution to Problem 1 on G_i^1. If $|S_1| = |S_2| = \widetilde{k}_1^*$ and $v_1 < v_2 < \widetilde{p}_1^*$, then $S_2 \neq \widetilde{S}_1^{T^*}$ for all $T^* \in \mathbb{T}^*$.*

Let us now describe how we can search for candidate optimal node sets based on the previous lemmas, once we have obtained $\widetilde{p}_1^*$. We test successively sets with cardinality 1 first, then 2, 3 and so on, as follows. We pick a set S such that each node in S has at least one outgoing link with power $\widetilde{p}_1^*$ and generate the reduced graph $G^R(G, L_S, \widetilde{p}_1^*)$. If the latter has no spanning tree, then we know, from Lemma 3 part 1, that $S \neq \widetilde{S}_1^{T^*}$ for all $T^* \in \mathbb{T}^*$ and, therefore, S can be removed from further consideration. Else, from Lemma 3 part 2, we know that $\widetilde{k}_1^* \leq |S|$. Since we already know that there can be no optimal set with $|S| - 1$ nodes, we have $\widetilde{k}_1^* = |S|$. Hence, at this point we know $\widetilde{k}_1^*$. However, we still have to identify the candidate optimal node sets. For this, we proceed as follows. We examine the sets with cardinality $\widetilde{k}_1^*$ and keep only those whose value of the solution to Problem 1 on the reduced graph is the smallest. From Lemma 4, it follows that only the latter sets are candidates for being equal to $\widetilde{S}_1^{T^*}$ for some $T^* \in \mathbb{T}^*$. Moreover, the smallest value obtained should be equal to $\widetilde{p}_2^*$. Hence, at the end of this process we have the values of $\widetilde{p}_1^*$, $\widetilde{k}_1^*$, $\widetilde{p}_2^*$, and a set of candidate optimal node sets $\mathbb{S}_1 = \{S_{1,1}, ..., S_{1,s_1}\}$ that may be equal to $\widetilde{S}_1^{T^*}$ for some $T^* \in \mathbb{T}^*$.

Unfortunately, with the information at hand we do not have any way to further isolate one of the sets in $\mathbb{S}_1$ as belonging to one of the lexicographically optimal trees. However, the process described above can be generalized to allow further refinement of the search, that results in the iterative determination of sequences of candidate optimal node sets $(S_{1,m_1}, S_{2,m_2}, ..., S_{i,m_i})$ with the following property:

Property 2: *It is possible that $(S_{1,m_1}, S_{2,m_2}, ..., S_{i,m_i}) = (\widetilde{S}_1^{T^*}, \widetilde{S}_2^{T^*}, ..., \widetilde{S}_i^{T^*})$ for some $T^* \in \mathbb{T}^*$, $1 \leq i < \widetilde{I}^{T^*}$.*

The refinement process is based on direct generalization of Lemmas 3 and 4 and can be achieved as follows. At iteration i we have a *candidacy tree*, whose levels and nodes have the following interpretation:

1) Level 0 is associated with $\widetilde{p}_1^*$, obtained from the solution to Problem 1 on graph G. The root node at level 0 is associated with the empty set of nodes, $\oslash$.

2) Level j, $j = 1, ..., i$, is associated with the values $\widetilde{k}_j^*$, $\widetilde{p}_{j+1}^*$ already obtained. Each node m at level j is associated with a candidate set $S_{j,m}$. The cardinality of each of the sets at level j is $\widetilde{k}_j^*$; each node in $S_{j,m}$ will transmit with power $\widetilde{p}_j^*$ if indeed $S_{j,m} = \widetilde{S}_j^{T^*}$ for some $T^* \in \mathbb{T}^*$ and the broadcast transmissions induced by T^* are eventually chosen. We denote by $\mathbb{S}_j = \{S_{j,1}, ..., S_{j,s_j}\}$ the set of candidate optimal node sets at level j.

3) A path $(S_{1,m_1}, S_{2,m_2}, ..., S_{i,m_i})$ in the candidacy tree denotes a sequence of candidate sets that may satisfy Property 2. Leaf nodes are located only at the highest level.

Algorithm 2:

1. For a candidate sequence (path from the root to a node at level i) $\mathbf{S}^i = (S_{1,m_1}, S_{2,m_2}, ..., S_{i,m_i})$, let $L_{\mathbf{S}^i} = \cup_{j=1}^{i} L_{S_{j,m_j}}$ and $N_{\mathbf{S}^i} = \cup_{j=1}^{i} S_{j,m_j}$.

2. Test successively sets with cardinality 1 first, then 2, 3 and so on, as follows. Pick a set S such that each node in S has at least one outgoing link with power $\widetilde{p}_{i+1}^*$ and $S \subseteq N - N_{\mathbf{S}^i}$.

3. Generate the reduced graph $G^R(G, L_S \cup L_{\mathbf{S}^i}, \widetilde{p}_{i+1}^*)$. If the latter has no spanning tree, then S is removed from further consideration. Else, it is known that $\widetilde{k}_{i+1}^* = |S|$.

4. The set S corresponds to a node at level $i+1$ if the value v of the solution to Problem 1 on the reduced graph is the smallest among the sets with cardinality $\widetilde{k}_{i+1}^*$. In this case, the node corresponding to S at level $i+1$ is connected to the node corresponding to set S_{i,m_i} at level i.

5. At the end of this procedure, there may be leaves in the candidacy tree that are located at level lower than $i+1$. The path that starts from the root and ends at one of these leaves corresponds to a sequence of node sets that cannot be a candidate satisfying Property 2 and should, therefore, be excluded from further consideration. Hence, we repeatedly eliminate all leaves that are not at level $i+1$ of the candidacy tree, so that only leaves at level $i+1$ remain.

Fig. 4. Optimal Algorithm for the General Case

When we are at level i, $\widetilde{p}_{i+1}^*$ is known. If $\widetilde{p}_{i+1}^* = 0$, then any spanning tree corresponding to a candidate path up to level i is a lexicographically optimal tree. Else, level $i+1$ is created by adding and removing nodes in the candidacy tree as described in the algorithm of Fig. 4. Upon completion, Algorithm 2 provides all lexicographically optimal trees and, therefore, all lexicographically optimal (with respect to node powers) sets of broadcast transmissions. Note that the number of nodes in the candidacy tree does not necessarily increase as the algorithm proceeds, since at step 5 the tree may be "pruned".

Fig. 5 shows an example of candidacy tree for a graph G with root node A. After creation of level 3, we observe that there is a node at level 2 of the candidacy tree, associated with node C of G, which is a leaf node at level lower than 3 and can, therefore, be eliminated. Then, the node at level 1, associated with node A of G, becomes such a leaf and is also eliminated. At level 4, we find that $\widetilde{p}_5^* = 0$ and the algorithm stops. In this example there are two optimal trees, T_1^*,T_2^*, with links {(A,B),(A,F),(A,G),(B,C),(B,D),(C,E),(F,H),(G,I)} and {(A,B),(A,F),(A,G),(B,C),(B,D),(C,E),(F,H),(H,I)} respectively. T_1^* corresponds to the path B→C→{F,G}→A of the candidacy tree and T_2^* corresponds to the path B→C→{F,H}→A. The sets of node powers, corresponding to the trees, are easily derived by inspecting the candidacy tree. For example, the power induced on node H by T_2^* is found as follows. T_2^* corresponds to path B→C→{F,H}→A. For this path, node H is located at level 3 and, hence, its retransmission power should be $\widetilde{p}_3^* = 3$. The (lexicographically equal) node power vectors induced by the two trees are shown in Table I.

Algorithm 2 requires exponential number of computations in $|N|$ in the worst case. In a wireless network however, where nodes are placed at irregular locations, the cardinality of the set of nodes that have at least one outgoing link with power $\widetilde{p}_i^*$

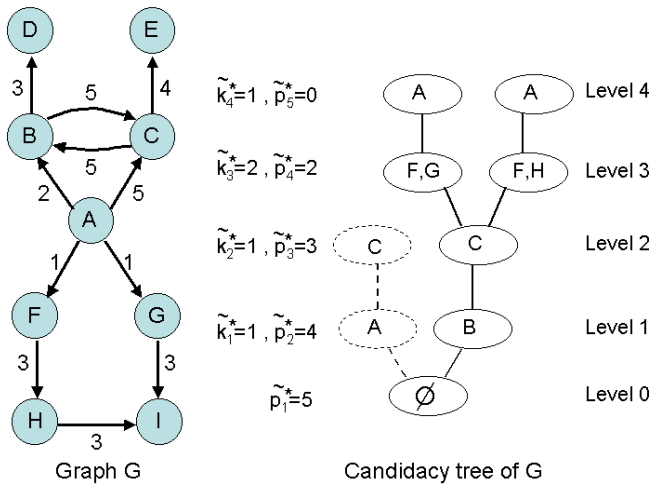

Fig. 5. Example of candidacy tree

TABLE I
NODE POWERS INDUCED BY THE OPTIMAL TREES

	A	B	C	D	E	F	G	H	I
T_1^*	2	5	4	0	0	3	3	0	0
T_2^*	2	5	4	0	0	3	0	3	0

may not be very large. Moreover, in such an environment, the condition for a branching to occur in the candidacy tree does not seem very likely. Note also that, even if a branching does occur, it is likely to be eliminated in the next few iterations, according to step 5 of the algorithm. Hence, for moderate size random networks, Algorithm 2 may not require many computations. However, as the simulation results show, its running time rapidly deteriorates as the number of nodes increases. In the next subsection we provide a heuristic that can be used for larger networks as well. As we will see, the various steps of Algorithm 2 are useful for the development of the heuristic.

C. Heuristic Algorithm

The main causes of explosion in the number of computations of Algorithm 2 are the following:
1) The cardinality of the set A_i, the set of nodes in the graph that have at least one outgoing link with power $\widetilde{p}_i^*$. In general, subsets of these nodes have to be tested to create the nodes of the candidacy tree at level i and, hence, the worst case number of computations required is proportional to $2^{|A_i|}$.
2) The branchings of the candidacy tree. A branching will occur if two or more of the node sets being tested result in the smallest value of the solution to Problem 1 on the corresponding reduced networks.

Simulation results show that branchings do not occur very frequently. On the other hand, as will be seen in Section V, $|A_i|$ does increase as the number of nodes within the same geographical area increases. Therefore, in order to reduce the computations required to obtain a good tree, we must develop efficient algorithms for selecting "good" subsets $\widetilde{S}_i^T$. Motivated by these observations, we modify Algorithm 2 as follows.

We solve Problem 1 on G. Let $p_1^H = \widetilde{p}_1^*$ be the value of this solution and A_1^H the set of nodes in G with at least one outgoing link with power p_1^H. According to Algorithm 2, we have to select a subset $S_1^H \subseteq A_1^H$ of minimal cardinality, such that the reduced graph $G^R(G, L_{S_1^H}, p_1^H)$, where $L_{S_1^H} = \{l \in L : l \in L_{out}^G(n), n \in S_1^H, c_l \leq p_1^H\}$, has at least one spanning tree, and solve Problem 1 on that graph.

To avoid excessive computations, we pick S_1^H as follows. We eliminate from G all links with power larger than p_1^H. Let G' be the resulting graph. For each node $n \in A_1^H$ we form the vector $\mathbf{v}_n = (c_l)_{l \in L_{out}^{G'}(n)}$. The maximal element of $\mathbf{v}_n$ is by definition of G' equal to p_1^H, but the rest of its elements may have any values smaller than or equal to p_1^H. Any node $n \in S_1^H$ will have to transmit with power p_1^H and, according to Property 1, will also transfer the required information to all nodes in $N_1^{G'}(n)$. Hence, it is preferable to include nodes in S_1^H with large values of elements in $\mathbf{v}_n$ and exclude those with small values. To achieve this, we arrange the nodes in A_1^H in non-decreasing lexicographic order of the vectors $\mathbf{v}_n$. Starting from the first node n in the lexicographic order, we test whether by eliminating from G' the outgoing links of n with power p_1^H, G' still has at least one spanning tree. If this is the case, we eliminate from G' all outgoing links of n with power equal to p_1^H and examine the next node in the lexicographic order. Else, we add node n to S_1^H and consider the next node in the lexicographic order. Note that, according to the selection procedure above, a node with lexicographically small vector $\mathbf{v}_n$ is not included in S_1^H, unless it is necessary to maintain network connectivity. Hence, S_1^H will generally contain nodes with lexicographically large vectors $\mathbf{v}_n$.

At the end of this procedure, the reduced graph $G^1 = G^R(G, L_{S_1^H}, p_1^H)$ is formed and Problem 1 is solved on that graph to obtain p_2^H. Note that p_2^H may not be equal to $\widetilde{p}_2^*$. All steps above are then applied to G^1, instead of G, and the process is repeated until $p_i^H = 0$ for some $i > 1$. The last spanning tree obtained is returned as the solution T^H of the algorithm. The node power vector $(p_n^{T^H})_{n \in N}$, induced by T^H in the original graph G, defines the set of broadcast transmissions of the proposed heuristic.

Note that in the previous algorithm, branchings in the candidacy tree are eliminated. In effect, we now have a single path at each step of the iteration. Some further optimization can be performed by observing that if a node $n \in A_1^H$ contains a link $l = (n, m)$ such that $c_l = p_1^H$, and l is the only incoming link to node m in graph G', then node n must transmit with power p_1^H and, therefore, is included in the set S_1^H. The final heuristic algorithm is presented in Fig. 6.

As an example, we apply the heuristic algorithm to graph G of Fig. 5. Since $p_1^H = 5$, we have that $A_1^H = \{\text{A,B,C}\}$ and $S_1^H = \{\text{B}\}$. After solving Problem 1 on the reduced graph $G^R(G, L_{\{\text{B}\}}, 5)$ we obtain $p_2^H = 4$, $A_2^H = S_2^H = \{\text{C}\}$ and, consecutively, $p_3^H = 3$, $A_3^H = \{\text{F,G,H}\}$, $S_3^H = \{\text{F,H}\}$, $p_4^H = 2$, $A_4^H = S_4^H = \{\text{A}\}$ and $p_5^H = 0$. The last spanning tree obtained is the solution T^H of the algorithm. For this particular graph, $T^H = T_2^*$, where T_2^* is one of the two lexicographically optimal trees of G found in the previous subsection.

Complexity of Algorithm 3: We will first provide the

Algorithm 3:
1. $i = 0, G^i = G$.
2. Let p^H_{i+1} be the value of the solution $\overline{T}^{i+1}$ to Problem 1 on G^i.
3. If $p^H_{i+1} = 0$, then Return $\overline{T}^{i+1}$ (T^H is equal to $\overline{T}^{i+1}$). Else, set $S^H_{i+1} = \oslash$ and identify A^H_{i+1}, the set of nodes with at least one outgoing link with power p^H_{i+1} in G^i.
4. Eliminate from G^i all links with power larger than p^H_{i+1} and let G' be the resulting graph.
5. For each node $n \in A^H_{i+1}$, if there is a link $l = (n, m)$ such that $c_l = p^H_{i+1}$, and l is the only incoming link to node m in graph G', then add n to S^H_{i+1}.
6. If the reduced graph $G^R(G^i, L_{S^H_{i+1}}, p^H_{i+1})$ has at least one spanning tree, then go to step 9. Else, for each node $n \in A^H_{i+1} - S^H_{i+1}$ form the vector $\mathbf{v}_n = (c_l)_{l \in L^{G'}_{out}(n)}$ and arrange these nodes in non-decreasing lexicographic order of the vectors $\mathbf{v}_n$.
7. Starting from the first node n in the lexicographic order, test whether by eliminating from G' the outgoing links of n with power p^H_{i+1}, G' still has at least one spanning tree.
8. If this is the case, eliminate from G' all outgoing links of n with power equal to p^H_{i+1} and examine the next node in the lexicographic order. Else, add node n to S^H_{i+1} and consider the next node in the lexicographic order.
9. $G^{i+1} = G^R(G^i, L_{S^H_{i+1}}, p^H_{i+1}), i = i + 1$. Go to step 2.

Fig. 6. Heuristic Algorithm

complexity for one iteration of the algorithm. Step 2 takes $O(|N| \log |N| + |L|)$ time. Steps 3-5 take $O(|L|)$ time. Step 6 requires the arrangement of at most $|N|$ nodes in non-decreasing lexicographic order of the vectors $\mathbf{v}_n$. We can assume that the coordinates of these vectors are already sorted in non-increasing order (this can be made during initialization in $O(\sum_{n \in N} x_n \log x_n) = O(|L| \log |L|)$ time, where x_n is the number of elements in $\mathbf{v}_n$). To evaluate the complexity of the above arrangement, let us assume that we implement bubble sort [13] - as will be seen, because of step 7, more efficient sorting does not improve worst-case complexity. Lexicographic comparison of vectors $\mathbf{v}_n$ and $\mathbf{v}_{n+1}$ takes time $O(\min\{x_n, x_{n+1}\})$ and a single *pass* of the vectors $\mathbf{v}_n$, $n \in N$, takes time $O(\sum_{n=1}^{|N|-1} \min\{x_n, x_{n+1}\})$, which is at most $O(\sum_{n=1}^{|N|} x_n) = O(|L|)$. Since at most $|N|$ passes of the vectors $\mathbf{v}_n$, $n \in N$, can occur, the complexity of the above arrangement with bubble sort is $O(|N||L|)$. A spanning tree can be obtained in $O(|L|)$ time [13] and, therefore, step 7 of the algorithm requires $O(|N||L|)$ time in the worst case. Hence, one iteration of steps 1-9 of the algorithm requires $O(|N||L|)$ time and, since at most $|N|$ such iterations may occur, the worst case running time of Algorithm 3 is $O(|N|^2|L|)$.

IV. More General Cost Functions

Assume that with each node $n \in N$ there is a cost function $f_n(p)$, which expresses the cost incurred at node n if it transmits with power p. We assume that $f_n(p)$ is strictly increasing in p and nonnegative. For reasons that will become apparent in the sequel, we allow for the possibility $f_n(0) > 0$. As in Section II-A, given an r-rooted spanning tree T, we define for any node $n \in N$, $\phi^T_n = \max_{l \in L^T_{out}(n)}\{f_n(c_l)\}$, if $L^T_{out}(n) \neq \oslash$, and $\phi^T_n = f_n(0)$, if $L^T_{out}(n) = \oslash$. The objective now is to find the spanning tree for which the vector $(\phi^T_n)_{n \in N}$ is lexicographically minimal. The case $f_n(p) = p$ for all $n \in N$ corresponds to the problem studied in Section II. In fact, if we use the quantities $f_n(c_l)$ as link cost functions, then the main difference between the current setup and the one examined up to now, is that the "power ϕ^T_n" of a leaf node n may be non zero in the general case. However, we will show that the same algorithms can be used in this case as well, by modifying $G(N, L)$ to a new network $\overline{G}(\overline{N}, \overline{L})$ as follows. $\overline{G}$ contains all nodes and links of G. For each node $n \in N$, a new node $\overline{n}$ is added to the set $\overline{N}$ of nodes in $\overline{G}$ and a new link $(n, \overline{n})$ is added to the set $\overline{L}$ of links in $\overline{G}$. The link powers in $\overline{L}$ are defined as $\overline{c}_{(n,m)} = f_n(c_{(n,m)})$, if $m \neq \overline{n}$, and $\overline{c}_{(n,\overline{n})} = f_n(0)$. The node powers for a tree $\overline{T}$ in $\overline{G}$ are defined as $\theta^{\overline{T}}_n = \max_{l \in \overline{L}^{\overline{T}}_{out}(n)}\{\overline{c}_l\}$, if $\overline{L}^{\overline{T}}_{out}(n) \neq \oslash$, and $\theta^{\overline{T}}_n = 0$, if $\overline{L}^{\overline{T}}_{out}(n) = \oslash$. Note that for all outgoing links of node n, the same function f_n is used to determine the corresponding generalized link costs. Since $f_n(p)$ is strictly increasing in p, Property 1 holds in network $\overline{G}$ as well (by replacing "power p" with "power $f_n(p)$") and the problem of minimizing lexicographically the vector $(\theta^{\overline{T}}_n)_{n \in \overline{N}}$ in $\overline{G}$ can be solved using the previously presented algorithms. According to the above definition, each spanning tree $\overline{T}$ of $\overline{G}$ corresponds to a spanning tree T of G and vice versa. Tree T is obtained from $\overline{T}$ by eliminating every link $(n, \overline{n}) \in \overline{L}$. Based on this observation, the following lemma holds:

Lemma 5: *Let $\overline{T}^*$ be a tree that minimizes lexicographically the vector $(\theta^{\overline{T}}_n)_{n \in \overline{N}}$ in $\overline{G}$. Then, the corresponding tree T^* in G, minimizes lexicographically the vector $(\phi^T_n)_{n \in N}$.*

Application I: Taking Into Account Node Receive Power Consumption.

Assume that node n consumes power q_n during the reception of information. Since all nodes in a broadcast tree T receive the information transmitted by the root node r, the total power consumed by node $n \neq r$ is now $p^T_n + q_n$. Therefore, if we are interested in minimizing lexicographically the total consumed node power, we can set $f_n(p) = p + q_n$ if $n \neq r$, and $f_r(p) = p$. Note that in this case we have $f_n(0) \neq 0$ if $n \neq r$.

Application II: Maximizing Lexicographically the Minimum Remaining Node Battery Lifetime.

Assume we know the duration of the broadcast transmission, t, and the energy E_n of the battery at node $n \in N$ at the beginning of the transmission, which we call "battery lifetime". Assume also for simplicity that the receive power q_n is zero. Then, if node n transmits with power p, at the end of the broadcast transmission, its remaining battery lifetime will be $E_n - pt$.

In an environment where spare batteries are not easily available, it is reasonable to attempt to maintain the remaining battery lifetime at each node as high as possible. Given an r-rooted spanning tree T, the broadcast transmissions defined by T result in remaining battery lifetime at node $n \in N$,

$$E^T_n = E_n - p^T_n t = E_n - \max_{l \in L^T_{out}(n)}\{c_l\}t$$
$$= -\max_{l \in L^T_{out}(n)}\{c_l t - E_n + E\} + E, \text{ where } E = \max_{n \in N}\{E_n\}.$$

An appropriate optimization criterion in this case is to attempt to maximize lexicographically the vector $(E_n^T)_{n\in N}$ or, equivalently, minimize lexicographically the vector $(F_n^T)_{n\in N}$, where $F_n^T = \max_{l\in L_{out}^T(n)}\{c_l t - E_n + E\}$. The latter problem is a special case of the one considered in this section, with $f_n(p) = pt - E_n + E$. This function is nonnegative by the definition of E and strictly increasing in p. Also, $f_n(0) = E - E_n$ which in general can be nonzero.

Application III: Taking Into Account Node Criticality.

The approach above can be generalized. In practice some nodes may be more critical than others for the operation of the network as a whole, for example keeping the root node alive is such a case. Hence, it is natural to associate different cost functions to different nodes, according to their remaining lifetime. More specifically the cost associated with node n, whose remaining lifetime is $\mathcal{E}_n$, is $g_n(\mathcal{E}_n)$, where g_n is strictly decreasing in $\mathcal{E}_n > 0$ and nonnegative. An appropriate optimization criterion in this case is to minimize lexicographically the vector $(g_n(E_n^T))_{n\in N} = (f_n(p_n^T))_{n\in N}$, where $f_n(p) = g_n(E_n - pt)$ is strictly increasing in p and nonnegative. By eliminating from the graph all links (n, j) such that $E_n - c_{(n,j)}t \leq 0$ (since transmission over these links exhausts node's n battery), we can ensure that $f_n(p)$ is well defined. Therefore, we can apply the previously developed methods to solve this generalized problem as well. If, as above, the maximization of node remaining lifetime is sought, we set $g_n(\mathcal{E}_n) = -\mathcal{E}_n + E$.

V. Numerical Results

In this section we compare numerically the performance of the three algorithms presented in Section III, namely: 1) the algorithm that solves Problem 1 ("Min-Max" algorithm for short), 2) the lexicographically optimal Algorithm 2 ("Lex-Opt" algorithm), and 3) the proposed heuristic Algorithm 3 ("Heuristic" algorithm).

We generate random networks with a specified number of nodes (20,40,...,120) as follows. We fix a rectangular grid of 100×100 points. A number of these points is randomly selected with uniform probability to represent the network nodes. One of the nodes is randomly chosen to be the origin of the broadcast information. The power needed for a successful transmission over link (i, j) depends on the distance $d_{(i,j)}$ between the two nodes and is given by $c_{(i,j)} = d_{(i,j)}^2$. For each network instance there is a constraint on the maximum transmitted power, $c_{\max}$, which is defined as the smallest value that guarantees the connectivity of the network, that is, the existence of a directed path from the root to every other node of the network. A link l belongs to the set L of links in the network if $c_l \leq c_{\max}$. The constraint $c_{\max}$ does not impose any limitation on the possible networks, since any links with power larger than $c_{\max}$ are removed from further consideration after the first iteration by any of the three algorithms we consider.

The main performance metric of interest is the vector of node powers provided by each algorithm. Since, however, this metric becomes too cumbersome to present for network sizes of interest, we introduce a closely related measure that indicates how far the obtained solution is from the optimal one. This is defined as follows. The coordinates of the node power vectors are first arranged in non increasing order. For a given network size, $|N|$, and for each individual network instance, let $(\widehat{p}_1^X, \widehat{p}_2^X, ..., \widehat{p}_{|N|}^X)$ be the vector of the arranged coordinates, obtained using algorithm X. Let l^X, $1 \leq l^X \leq |N|$, be the number for which it holds $\widehat{p}_i^X = \widehat{p}_i^{Lex-Opt}$ for all i, $1 \leq i \leq l^X$. Note that, since all algorithms solve Problem 1 in the first iteration, we have $\widehat{p}_1^X = \widehat{p}_1^{Lex-Opt}$ and thus l^X is well defined. We next define $R = \frac{l^X}{|N|}$. This metric provides a measure of how close each algorithm comes to providing the optimal (lexicographically smallest) vector of node powers. Obviously for any network, $0 < R \leq 1$, and when $R = 1$ the obtained solution is identical lexicographically to the one provided by the Lex-Opt algorithm.

Table II summarizes the performance of the Heuristic algorithm for various network sizes. Each row of the table represents the summary of results obtained from 100 randomly generated networks. The second column is the average of the obtained ratio R. In columns 3-6 the quantity $Q(R > p)$, $0 \leq p < 1$, corresponds to the percentage of network instances for which $l^{Heuristic} > p \cdot |N|$. We observe that the proposed heuristic generally provides fairly satisfactory results. For 40-node networks for example, it provides the optimal solution in 98% of the performed experiments. For 120-node networks, this percentage falls to 61%. However, the percentage of the experiments for which at least the first 30 (0.25×120) maximal node powers are the same as those obtained by the optimal algorithm, is 96%. Table III provides similar results for the Min-Max algorithm. We observe that the performance of this algorithm deteriorates rapidly as the network size increases. This is to be expected since it ensures only the minimization of the maximum consumed node power but does not specify how to treat nodes that consume power smaller than the maximum.

Table IV shows the average and standard deviation of the running times of the algorithms. The results were obtained using a Pentium4 PC, 1.7 GHz, with 256 MB RAM. As expected, the Min-Max algorithm has the shortest running times and Lex-Opt the largest. For network sizes no more than 80, the running time of Lex-Opt algorithm is reasonable, but rapidly deteriorates for larger networks. The Heuristic algorithm has satisfactory running times for all network sizes. It is also worth noting by observing the standard deviations, that the running times of the Heuristic are concentrated around their mean, while for the Lex-Opt they are widely dispersed, especially for larger networks. Combined with the results in Table II, we conclude that the Heuristic algorithm presents a good compromise between performance and running times.

The behavior of the Lex-Opt and Heuristic algorithms can be further elucidated by the results in Table V. Each row of the table provides the average over the 100 randomly generated network instances of the quantities

$$\overline{k} = \frac{\sum_{i=1}^{\tilde{I}^*-1}|A_i|}{\tilde{I}^*-1}, \qquad \overline{k}^* = \frac{\sum_{i=1}^{\tilde{I}^*-1}\widetilde{k}_i^*}{\tilde{I}^*-1}.$$

Note that, when $\widetilde{p}_{\tilde{I}^*}^* = 0$ is found, the Lex-Opt algorithm stops and no further computations are necessary to determine $A_{\tilde{I}^*}$ and $\widetilde{k}_{\tilde{I}^*}^*$. As the network size increases, the density of nodes (number of nodes per unit area) increases as well. This is indicated in Table V by the fact that $\overline{k}$ increases. However, $\overline{k}^*$ remains relatively small. Hence, in many cases the search per-

TABLE II

COMPARISON OF HEURISTIC ALGORITHM VS. LEX-OPT

$\|N\|$	R-Mean	$Q(R>0.25)$	$Q(R>0.5)$	$Q(R>0.75)$	$Q(R=1)$
20	0.9925	99%	99%	99%	99%
40	0.9898	100%	99%	98%	98%
60	0.9303	97%	93%	88%	88%
80	0.8901	95%	87%	81%	81%
100	0.8572	93%	84%	77%	77%
120	0.7694	96%	72%	61%	61%

TABLE III

COMPARISON OF MIN-MAX ALGORITHM VS. LEX-OPT

$\|N\|$	R-Mean	$Q(R>0.25)$	$Q(R>0.5)$	$Q(R>0.75)$	$Q(R=1)$
20	0.4510	70%	28%	17%	17%
40	0.2908	50%	10%	1%	1%
60	0.2212	37%	2%	0%	0%
80	0.2018	28%	1%	0%	0%
100	0.1995	30%	1%	0%	0%
120	0.1787	29%	0%	0%	0%

TABLE IV

RUNNING TIME PERFORMANCE OF THE ALGORITHMS (MEAN AND STANDARD DEVIATION IN SECONDS)

	Min-Max	Heuristic	Lex-Opt
$\|N\|$	Mean ; St.D.	Mean ; St.D.	Mean ; St.D.
20	0.004 ; 0.005	0.038 ; 0.013	0.047 ; 0.015
40	0.020 ; 0.009	0.290 ; 0.113	0.365 ; 0.134
60	0.051 ; 0.019	0.969 ; 0.307	1.441 ; 0.557
80	0.088 ; 0.031	2.163 ; 0.787	5.652 ; 9.368
100	0.146 ; 0.040	3.856 ; 1.120	20.649 ; 20.281
120	0.194 ; 0.061	5.563 ; 1.884	786.228 ; 2,436.465

TABLE V

MEAN OF $\overline{k}$ AND $\overline{k}^*$ FOR THE LEX-OPT ALGORITHM

$\|N\|$	$\overline{k}$ - Mean	$\overline{k}^*$ - Mean
20	2.155097	1.023994
40	2.939660	1.115641
60	4.207257	1.249700
80	5.855358	1.402980
100	7.925565	1.653131
120	10.533443	1.918101

formed by the Lex-Opt algorithm when trying to find the sets $\widetilde{S}_i^{T^*}$ does not take significant amount of time. However, the occasional existence of a pair $\{|A_i|,\widetilde{k}_i^*\}$ with values much larger than the corresponding averages is enough to increase dramatically the running time of the Lex-Opt algorithm. This explains the large standard deviation of the running time of the Lex-Opt algorithm, observed for large networks. Regarding the Heuristic algorithm, its running time is not very sensitive to an increase in $\overline{k}$, however, this increase makes it more difficult to determine the correct sets $\widetilde{S}_i^{T^H}$ and, thus, its performance deteriorates as the node density increases.

VI. EXTENSIONS - ISSUES FOR FURTHER STUDY

Distributed Implementation: The algorithms presented in this paper require knowledge of the network topology. Distributed implementation of the algorithms depends on the knowledge that each node has about the network topology in its neighborhood. If each node has knowledge of its one, two, ..., k-hop neighbors, then the proposed algorithms can be applied locally in a manner similar to the one proposed in [10], where all links were assumed to have the same power requirements. In general they can be applied in network environments where at least partial information of network topology is proactively maintained at each node, as in OLSR [11] and ZRP [12]. Of course, in this case optimization based on partial information does not guarantee global optimization as well, and further study is currently underway to evaluate the benefits of the approach. Let us consider next the case where each node knows only the powers needed to reach its one-hop neighbors. Instrumental to the algorithms proposed in this paper is the solution to Problem 1. As shown in [16], there is an optimal algorithm for this problem, which follows the same steps as Edmond's algorithm for finding a minimum-sum spanning tree in a directed graph [17], with the exception that the sum operation is replaced with the maximum operation. A distributed implementation of Edmond's algorithm has been proposed in [18]. The latter algorithm can be modified to implement in a distributed manner the algorithm for solving Problem 1 proposed in [16]. More substantial modifications are needed, however, to optimize the second, third, and so on, minimum. This effort would require power consumption which must be weighted against the power savings that will be obtained during the broadcast transmission. As an intermediate alternative, optimization of only a given number of successive minima may be performed.

Multicast Extensions: Multicasting of information from a source to a subset V of the nodes is an important mechanism. The optimal algorithms proposed in this paper can be applied in this case as well, provided that Problem 1 is replaced with the problem of finding a bottleneck tree connecting the source to all nodes in V. For an undirected graph, an algorithm for the bottleneck tree problem with $O(|L|)$ running time is proposed in [19]. For directed graphs, if node powers are integers that take values in $[0, C]$, then it is easy to construct an $O(|L| \log C)$ running time algorithm. Work is underway to provide improved-performance algorithms for the bottleneck tree problem, that can also be implemented distributively. In addition, we note that for the multicast problem new heuristics must be developed in order to take into account peculiarities that arise due to the fact that in general not all nodes participate in the multicast tree. Regarding multicasting, another important issue with potential significant energy savings is the development of algorithms with the same optimization criteria, when directional antennas are used [20].

VII. CONCLUSIONS

In this paper we addressed the problem of broadcasting in a wireless network in a manner that guarantees that no node in

the network consumes excessive power, or no node's battery lifetime is unnecessarily reduced. We formulated the problem as a lexicographic optimization problem and presented an optimal algorithm for it. The algorithm has exponential running time in the worst case. Certain interesting instances of the problem have polynomial worst case running time. We also presented a heuristic algorithm which eliminates the steps of the optimal one where computational explosions occur. Numerical results show that the optimal algorithm runs in reasonable time for moderate size networks, but its running time rapidly deteriorates as the network size increases. The heuristic algorithm has good running times and its performance, relative to the optimal one, is fairly satisfactory for all the network sizes considered. Hence, this algorithm presents a good compromise between running time and achieved performance.

REFERENCES

[1] A.J. Goldsmith and S.B. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *IEEE Wireless Commun.*, vol. 9, no. 4, pp. 8-27, Aug. 2002.

[2] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill, Computer Science Series, 1993.

[3] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, 1992.

[4] 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, pp. 585-594.

[5] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks," *MONET*, vol. 6, no. 3, pp. 251-263, June 2001.

[6] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "Resource management in energy-limited, bandwidth-limited, transceiver-limited wireless networks for session-based multicasting," *Computer Networks*, vol. 39, no. 2, pp. 113-131, June 2002.

[7] V. Ramasubramanian, R. Chandra, and D. Mossé, "Providing a bidirectional abstraction for unidirectional ad hoc networks," *in Proc. IEEE INFOCOM*, 2002.

[8] C.-K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *IEEE Commun. Mag.*, vol. 39, no. 6, pp. 138-147, June 2001.

[9] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *in Proc. IEEE INFOCOM*, 2000, pp. 22-31.

[10] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," *INRIA*, RR-3898, March 2000.

[11] P. Jacquet *et al.*, "Optimized link state routing protocol," *IETF Internet Draft*, draft-ietf-manet-olsr-07.txt, Dec. 2002 (Work in progress).

[12] Z.J. Haas, M.R. Pearlman, and P. Samar, "The zone routing protocol (ZRP) for ad hoc networks," *IETF Internet Draft*, draft-ietf-manet-zone-zrp-04.txt, July 2002 (Work in progress).

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

[14] P.M. Camerini, "The min-max spanning tree problem and some extensions," *Inform. Process. Lett.*, vol. 7, no. 1, pp. 10-14, Jan. 1978.

[15] H.N. Gabow and R.E. Tarjan, "Algorithms for two bottleneck optimization problems," *J. of Algorithms*, vol. 9, pp. 411-417, 1988.

[16] L. Georgiadis, "Arborescence optimization problems solvable by Edmond's algorithm," *J. of Theor. Comp. Science*, to be published. Available: *http://genesis.ee.auth.gr/SITE_AUTH_UNIVERSITY/SITE_TDIVISION/users/georgiadis/english/georgiadis.htm*

[17] J. Edmonds, "Optimum branchings," *J. Res. Nat. Bur. Stand.*, vol. 71b, pp. 233-240, 1967.

[18] P.A. Humblet, "A distributed algorithm for minimum weight directed spanning trees," *IEEE Trans. Commun.*, vol. 31, pp. 756-762, 1983.

[19] C.W. Duin and A. Volgenant, "The partial sum criterion for Steiner trees in graphs and shortest paths," *Europ. J. of Oper. Res.*, vol. 97, pp. 172-182, Feb. 1997.

[20] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "Energy-limited wireless networking with directional antennas: the case of session-based multicasting," *in Proc. IEEE INFOCOM*, 2002.

APPENDIX

We now give the proofs of the lemmas provided in the paper.

A. Proof of Lemma 2

Let T^* be a lexicographically optimal tree. Note that by definition the links in $L-\widetilde{L}^m$ with power larger than or equal to $\widetilde{p}^*_m$ do not belong to T^*. Hence, T^* is also a spanning tree of graph G^{m+1} and, therefore, a solution to the min-max optimization problem on G^{m+1} exists. Let c_l^{m+1} be the power of link l in graph G^{m+1}. By definition of G^{m+1}, we have $c_l^{m+1} = 0$ for all $l \in L_{out}^{G^{m+1}}(n)$, $n \in \widetilde{S}^m$. Hence, all nodes in T^* with power larger than or equal to $\widetilde{p}^*_m$ in G, have power 0 in G^{m+1}. Therefore, the maximum node power of T^*, when T^* is employed in graph G^{m+1}, is $\widetilde{p}^*_{m+1}$. The tree $\overline{T}^{m+1}$ in graph G^{m+1} induces node power at most $\widetilde{p}^*_{m+1}$ (otherwise it would not be a solution to Problem 1 on G^{m+1}). That is, if $q_n^{\overline{T}^{m+1}}$ is the power induced on node n by $\overline{T}^{m+1}$ in graph G^{m+1}, then

$$\max_{n\in N}\{q_n^{\overline{T}^{m+1}}\} \le \widetilde{p}^*_{m+1}. \tag{2}$$

Consider the node power vector $(p_n^{\overline{T}^{m+1}})_{n\in N}$ induced by $\overline{T}^{m+1}$ in the original graph G. The only nodes whose power may change are those in the set $\widetilde{S}^m$. By the definition of link powers c_l^{m+1}, it holds

$$p_n^{\overline{T}^{m+1}} \le \widetilde{p}^*_i,\ n \in \widetilde{S}^*_i,\ i = 1, ..., m, \text{ and} \tag{3}$$

$$p_n^{\overline{T}^{m+1}} = q_n^{\overline{T}^{m+1}},\ n \in N - \widetilde{S}^m. \tag{4}$$

Equation (4) and the fact that $q_n^{\overline{T}^{m+1}} = 0$ for $n \in \widetilde{S}^m$ imply

$$\max_{n\in N}\{q_n^{\overline{T}^{m+1}}\} = \max\{\max_{n\in N-\widetilde{S}^m}\{q_n^{\overline{T}^{m+1}}\}, \max_{n\in\widetilde{S}^m}\{q_n^{\overline{T}^{m+1}}\}\}$$
$$= \max_{n\in N-\widetilde{S}^m}\{p_n^{\overline{T}^{m+1}}\}.$$

From the latter equality and (2) we conclude that

$$\max_{n\in N-\widetilde{S}^m}\{p_n^{\overline{T}^{m+1}}\} \le \widetilde{p}^*_{m+1}. \tag{5}$$

It is easy to see now that, if at least one of the inequalities in (3), (5) is strict, then $\overline{T}^{m+1}$ is lexicographically smaller (with respect to node powers) than T^*. Therefore, all these inequalities are actually equalities

$$\widetilde{p}_i^{\overline{T}^{m+1}} = \widetilde{p}^*_i,\ i = 1, ..., m, \text{ and}$$
$$\widetilde{p}_{m+1}^{\overline{T}^{m+1}} = \max_{n\in N}\{q_n^{\overline{T}^{m+1}}\} = \max_{n\in N-\widetilde{S}^m}\{p_n^{\overline{T}^{m+1}}\} = \widetilde{p}^*_{m+1}. \quad \blacksquare$$

B. Proof of Lemma 3

1) Assuming that $S = \widetilde{S}_1^{T^*}$ for some $T^* \in \mathbb{T}^*$, we will reach a contradiction. From the definition of G^1 and $\widetilde{S}_1^{T^*}$, it follows that T^* is also a spanning tree of G^1. Therefore, there is at least one spanning tree of G^1, a contradiction.

2) Let $p_n^{\overline{T}^1}$, $q_n^{\overline{T}^1}$, $n \in N$, be the node powers induced by $\overline{T}^1$ in graphs G, G^1 respectively. From the definition of G^1, $q_n^{\overline{T}^1} = 0$ for $n \in S$, and, therefore, we have

$$\max_{n\in N}\{q_n^{\overline{T}^1}\} = \max\{\max_{n\in N-S}\{q_n^{\overline{T}^1}\}, \max_{n\in S}\{q_n^{\overline{T}^1}\}\}$$
$$= \max_{n\in N-S}\{p_n^{\overline{T}^1}\} < \widetilde{p}^*_1. \tag{6}$$

Since $\overline{T}^1$ uses only links in L_S as outgoing links from nodes in S, we have that

$$p_n^{\overline{T}^1} \leq \widetilde{p}_1^*,\ n \in S. \tag{7}$$

From definition of $\widetilde{p}_1^*$, (6) and (7), we have that $\widetilde{p}_1^{\overline{T}^1} = \widetilde{p}_1^*$ (otherwise $\overline{T}^1$ would be lexicographically smaller than T^*). From (6) we conclude that $\widetilde{S}_1^{\overline{T}^1} \subseteq S$ and from Lemma 1 $\widetilde{k}_1^* \leq \widetilde{k}_1^{\overline{T}^1} = |\widetilde{S}_1^{\overline{T}^1}| \leq |S|$. ■

C. Proof of Lemma 4

Assuming that $S_2 = \widetilde{S}_1^{T^*}$ for some $T^* \in \mathbb{T}^*$, we will reach a contradiction. Since T^* is also a spanning tree of G_2^1 and v_2 is the value of the solution to Problem 1 on G_2^1, we must have

$$v_2 \leq \widetilde{p}_2^*. \tag{8}$$

Consider now the optimal tree for Problem 1 on graph G_1^1, $\overline{T}_1^1$. Let $p_n^{\overline{T}_1^1}$, $q_n^{\overline{T}_1^1}$, $n \in N$, be the node powers induced by $\overline{T}_1^1$ in graphs G, G_1^1 respectively. From Lemma 3 part 2, we have that

$$\begin{aligned} \widetilde{p}_1^{\overline{T}_1^1} &= \widetilde{p}_1^*, \\ \widetilde{p}_2^{\overline{T}_1^1} &= \max_{n \in N - S_1} \{p_n^{\overline{T}_1^1}\} = \max_{n \in N} \{q_n^{\overline{T}_1^1}\} = v_1 < v_2. \end{aligned} \tag{9}$$

From (8) and (9), it follows that $\widetilde{p}_2^{\overline{T}_1^1} < \widetilde{p}_2^*$. The last inequality contradicts Lemma 1. ■

D. Proof of Lemma 5

Note that for any tree $\overline{T}$ in $\overline{G}$, each node $\overline{n}$ is a leaf node and, hence, $\theta_{\overline{n}}^{\overline{T}} = 0$. Therefore, since $\overline{T}^*$ minimizes lexicographically the vector $(\theta_n^{\overline{T}})_{n \in \overline{N}}$ in $\overline{G}$, we have for the vectors $(\theta_n^{\overline{T}})_{n \in N}$ (note that for these vectors we omit all nodes $\overline{n}$)

$$(\theta_n^{\overline{T}^*})_{n \in N} \preceq_{lex} (\theta_n^{\overline{T}})_{n \in N}. \tag{10}$$

Consider now the tree T in G corresponding to $\overline{T}$. If n has no outgoing links in T, then in $\overline{T}$ node n must have only one outgoing link $(n, \overline{n})$. Hence,

$$\theta_n^{\overline{T}} = \max_{l \in \overline{L}_{out}^{\overline{T}}(n)} \{\overline{c}_l\} = f_n(0) = \phi_n^T. \tag{11}$$

If n has at least one outgoing link in T, then

$$\begin{aligned} \theta_n^{\overline{T}} &= \max_{l \in \overline{L}_{out}^{\overline{T}}(n)} \{\overline{c}_l\} \\ &= \max\{\max_{l \in \overline{L}_{out}^{\overline{T}}(n), l \neq (n, \overline{n})} \{f_n(c_l)\}, f_n(0)\} \\ &= \max_{l \in L_{out}^T(n)} \{f_n(c_l)\} = \phi_n^T, \end{aligned} \tag{12}$$

where (12) is due to the fact that $f_n(c_l) > f_n(0)$, since $f_n(p)$ is strictly increasing in p. From (11) and (12), it follows that

$$\phi_n^T = \theta_n^{\overline{T}},\ \text{for each node } n \in N. \tag{13}$$

From (10) and (13), we conclude that

$$(\phi_n^{T^*})_{n \in N} = (\theta_n^{\overline{T}^*})_{n \in N} \preceq_{lex} (\theta_n^{\overline{T}})_{n \in N} = (\phi_n^T)_{n \in N}.$$ ■