CCIT Report #493

June 2004

Maximum-Lifetime Routing Algorithms for Wireless Networks

Ariel Orda and Ben-Ami Yassour
Department of Electrical Engineering
Technion-Israel Institute of Technology

June 30, 2004

Abstract

A major problem in wireless networks is how to route either broadcast, unicast or multicast traffic so as to maximize the *lifetime*, i.e., the time until the battery of a transmitting node drains out. Taking an algorithmic approach, we aim at finding solutions with provable performance bounds.

Focusing on the fundamental single-session problem, our solution approach is based on the employment of multi-topology routing schemes. Such a scheme consists of a *sequence of routing topologies*, which are employed sequentially, for some prescribed duration times.

First, we establish *optimal solutions* of polynomial complexity for the restricted cases of either single-topology schemes or unicast sessions. We then show that the general (multi-topology) cases of broadcast and multicast are NP-hard. Accordingly, we establish polynomial approximation schemes with *proven performance bounds*. We also derive a *novel heuristic scheme*, and demonstrate its efficiency by way of simulations.

We then consider an alternative, *single receiver* wireless environment. This change in the environment is shown to have major impacts on the complexity of the various problems, in both directions. Finally, we discuss the extension of our results to the case of multiple sessions.

1 Introduction

In recent years, stationary wireless networks were extensively studied due to their potential applications in the civil and military domains, in particular for the implementation of sensor networks. A sensor network is composed of numerous power constrained nodes, each equipped with processing, memory, short-range radio transmission and sensing capabilities. Scattered over a target environment, the nodes can monitor and collect useful information that is carried to some base-station by using the sensor nodes as relays [1], [2].

Since the amount of energy that can be stored on such nodes is limited, energy efficiency is a crucial aspect in the establishment of such networks. Thus, it is essential to develop protocols that optimize the overall energy utilization of the network, in order to maximize its capability to function for the longest possible time.

Accordingly, a wide variety of energy-efficiency problems have been addressed. One problem that received considerable attention was that of *minimum-energy broadcast*, i.e., finding a broadcast routing that minimizes the total energy consumed by all the nodes. In a wireless network, each node can adjust its transmission power. Transmitting at higher power levels enables more distant nodes to receive the transmission. Furthermore, broadcast transmissions can be received by all nodes within transmission radius. Hence, there is a trade-off between using higher power levels versus reaching more nodes. Two different variations of the energy-efficient broadcast problem have been studied. The first, termed as the *topology control problem*, is to assign each node a transmission power, such that each node in the network can reach any other node, i.e., any node can be a source of a broadcast tree. An optimal topology is then one that minimizes the total transmission power. This problem was shown to be NP-hard [3],[4], and approximate solutions were established [5],[6]. Another studied broadcast problem was the special case where only a single source is considered. This problem was also shown to be NP-hard [7],[8], and a greedy heuristic was proposed in [9]. That

heuristic was based on Prim's and Dijkstra's [10] minimum spanning tree algorithms, and in [11] it was proven that its approximation ratio is between $\frac{13}{6}$ and 12.

In [12], the problem of maximizing the network lifetime under broadcast was considered. Specifically, [12] defined two problem classes, namely "static" and "dynamic". The "static problem" is to find a (single) spanning tree rooted at a given source, such that the time until the first transmitting node consumes all its battery is maximized; i.e., the static problem considers a restriction where the routing scheme consists of a single spanning tree. In the "dynamic problem", on the other hand, any number of spanning trees can be used sequentially, in order to prolong the broadcast duration time. In [12], a polynomial time algorithm that finds an optimal solution to the static broadcast problem was established, while for the dynamic case several heuristics were proposed and evaluated by way of simulations.

In [13], the maximization of the lifetime of *multicast routing* was studied. Specifically, a time-quantized variant of the problem was considered, in which the time line was divided into slots and routing decisions were aligned with respect to these slots. For this model, it was shown that finding a multicast routing of maximum lifetime is NP-hard. Moreover, it was shown that finding a constant approximation is also NP-hard, and heuristic algorithms were considered and evaluated by way of simulations.

Energy-efficient protocols were considered for unicast routing as well, e.g., [14], [15], [16], [17]. In [14], a distributed protocol was proposed for finding a minimum-power routing. In [15], power-aware heuristics were considered to guide protocol design in networks that support certain types of collaboration. In [16], data-gathering wireless sensor networks were considered, and a bound on the lifetime of such networks was derived. In [17], a network environment was considered, which consisted of multiple source and destination nodes. Each source generated information at a given rate, and the goal was to identify corresponding paths from each source to any of the destinations, so as to maximize transmission lifetime. It was indicated that the problem could be defined as a linear program; however, the focus in [17] was on heuristic distributed routing protocols and their evaluation by way of simulations.

As stated, in this study we address the maximization of network lifetime for unicast, broadcast and multicast within a unified framework. Our focus is on the general (and harder) class of solutions that employ multiple routing topologies (i.e., "dynamic" solutions, per the terminology of [12]). However, we also consider the special ("static") case in which a single topology is allowed. We concentrate on lifetime maximization of a single session, and later discuss how our findings can be generalized to handle multiple sessions concurrently. Moreover, we focus on stationary topologies, being the typical case of sensor networks; yet, our results provide important insight for handling the mobile case. We also choose to focus on centralized solutions, due to several reasons. First, a centralized approach is often practical in stationary (or quasi-stationary) topologies. Second, centralized solutions provide an important design tool, as they provide bounds on what can be achieved with distributed solutions. Last, as it turns out, the problems are complex even under the centralized setting, yet our solution schemes provide important building blocks for future distributed implementations.

The main contributions of this paper can be summarized as follows.

1. For the case in which a single topology is allowed, we establish solutions that improve upon the previous solutions of [12].

2. For the unicast problem, we establish polynomial-time optimal solutions that are based on linear programming and max-flow algorithms.

3. We prove that the broadcast and multicast problems are NP-hard, and establish approximation algorithms with proven performance guarantees. For scenarios in which faster and simpler solutions are required, we establish a novel heuristic scheme for the broadcast case and evaluate it by way of simulations.

4. In addition, we investigate a different, "single-receiver" model, in which each transmission is heard by at most a single receiver. For this model, we show that the single-topology ("static") broadcast problem is NP-hard, while for the multi-topology ("dynamic") version we establish an efficient optimal solution. Compared with the standard (multiple-recipient) model, this model implies an interesting twist of difficulty of the single-topology versus the multi-topology broadcast problems: namely, for the standard model, the single-topology problem is

Model	***Unicast***	
	Single topology	*Multiple topologies*
(Both)	(Special case of multicast solution)	Computationally efficient optimal solution
Model	***Broadcast***	
	Single topology	*Multiple topologies*
Standard	(Special case of multicast solution)	a. NP-hardness proof b. Approximated solution c. Efficient heuristics
Single receiver	NP-hardness proof	Computationally efficient optimal solution
Model	***Multicast***	
	Single topology	*Multiple topologies*
Standard	Optimal solution with improved complexity	a. NP-hardness proof b. Approximated solution
Single receiver	NP-hardness proof	NP-hardness proof

Figure 1: Summary of contributions

computationally solvable and the multi-topology problem is NP-hard, while *the opposite* holds for the single-receiver model. Finally, we show that, in the single-receiver model, the single-and multi-topology multicast problems are both NP-hard.

5. We show how some of the results can be extended to the case of multiple sessions.

A summary of the results is depicted in figure 1.

The rest of the paper is organized as follows. In the next section, we formally define the standard model and the related problems. In Section 3, we consider the single-topology problems. In Section 4, we consider the multi-topology unicast problem. In Section 5, we consider the multi-topology broadcast and multicast problems. In Section 7, we consider the second, "single receiver", model. In Section 8, we consider the general case of multiple-sessions. Finally, conclusions are presented in Section 9.

2 Model and problem formulation

In this section we formally define the network model and the considered problems. We assume static locations of wireless nodes. The nodes communicate using omni-directional antennas, which means that, when a node is using enough power to transmit over a distance d, the transmission can be received by any other node that is not distant from the sender by more than d. In order for a sender to transmit to a receiver over a distance of d, the transmission power consumed by the sender is d^α, where α is the path loss factor (typically $2 \leq \alpha \leq 4$). Each node has a given set of legal transmission power levels at which it can transmit. Additionally, each node has a finite battery, and is able to transmit until it consumes all its energy.

In practice, reception of data consumes energy as well, however since that amount does depend on the transmission distance, it does not make the related problems essentially harder. In Appendix B, we explain how our model and solutions can be easily extended in order to accommodate such consumption of energy at the receiver. Hence, for conciseness, shall assume that no energy is consumed at the receiver.

There are two options to represent a stationary wireless network. The first option assumes that the nodes are positioned on the plane and the transmission power level between any two nodes is derived from the nodes locations on the plane, i.e., the required transmission power is d^α, where d is the distance between the nodes locations. On the other hand, in some cases the nodes are not positioned on a plane, moreover, obstacles may affect the wireless transmissions. A second, more general, option is then to allow the transmission power levels to take any positive values. We adopted the second option, as it is more general. Accordingly, our solution schemes do not assume that the nodes are positioned on a plane. On the other hand, when establishing that a certain problem is NP-hard, we do assume planar positions, hence getting a stronger intractability result. We proceed to describe the models that corresponds to these two options, starting with the more general one.

Definition 1 *A* stationary wireless network *is a weighted directed graph $G \equiv (V, E, b, p)$, where:*

1. *V is the set of nodes;*
2. *$E \subseteq V \times V$ is the set of directed edges, i.e., $(u, v) \in E$ if u is able to transmit to v;*
3. *for $v \in V$, $b(v)$ is the initial energy ("battery") of node v;*
4. *for $e = (u, v) \in E$, $p(e)$ is the power required for a node u to transmit to node v.*

Next, we formally define the model corresponds to geographic positioning of nodes on a plane.

Definition 2 *A* geographic stationary wireless network *is a tuple (V, b, p), where:*

1. *V is the set of nodes on the plane, i.e., $v_1 \in V$ corresponds to a location (x_1, y_1) on the plane;*
2. *for $v \in V$, $b(v)$ is the energy of node v;*
3. *for $v \in V$, $p(v)$ is the set of legal transmission power levels of node v.*

The power consumed by a node v_1 when transmitting to a node v_2, denoted as $p(v_1, v_2)$, is the minimal transmission power level $\hat{p}$ of v_1 for which
$\hat{p} \geq (\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2})^\alpha$, where α is the path loss factor.

For convenience, we represent a geographic static wireless network as a weighted directed graph $G \equiv (V, E, b, p)$, where E is the set of directed edges and $(v_1, v_2) \in E$ if v_1 has a transmission power level which enables it to reach v_2, i.e., $\max p(v_1) \geq (\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2})^\alpha$.

We address three different problems, namely *unicast*, *broadcast* and *multicast*. The *multicast* problem is to find a routing scheme that maximizes the time during which we can transmit from a source node to a set of target nodes. Such a scheme consists of some k trees, each spanning the set of target nodes, together with a duration time per tree. The idea is to activate the trees sequentially, each for its assigned duration time. When a tree is activated, multicast traffic flows from the source to the target along the tree nodes of the tree. Our goal is to find such trees and duration times, such that the sum of all duration times is maximized and the total energy consumed by each node is bounded by its initial energy. We proceed with a formal definition.

Definition 3 *Given are a directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a target set $N \subseteq V$. A* Multicast routing scheme *is a set of tree-duration time pairs $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$, $0 \leq i \leq k$, such that each h_i is a tree rooted at s that spans N, and the time durations $t_i \in \Re$ are such that:*

$$\forall u \in V, \sum_{i=1}^{k} t_i \cdot \max_{v \in V} \delta(h_i, (u, v)) \cdot p(u, v) \leq b(u) \tag{1}$$

where

$$\delta(h_i, e) = \begin{cases} 1 & e \in h_i \\ 0 & e \notin h_i \end{cases} \tag{2}$$

i.e., for each node, the total energy required by all the trees on which the node participates as a transmitter is bounded by the node's initial energy.

Definition 4 *The* lifetime of a multicast scheme *is the sum of tree duration times, i.e. $lifetime(\alpha) = \sum_{i=1}^{k} t_i$.*

Accordingly, our goal can be stated as follows.

Problem Maximum Lifetime Multicast (MLM): Given are a directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a target set $N \in V$. Find a multicast scheme α in G rooted at s that spans N, such that, for any other such multicast scheme β, $lifetime(\beta) \leq lifetime(\alpha)$.

Input: A directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a set $N \subseteq V$ of target nodes.

1 $D \leftarrow \{s\}$
2 $E' \leftarrow \{\}$
3 **foreach** $(u, v) \in E$, $w(u, v) \leftarrow \frac{b(u)}{p(u,v)}$
4 **while** $N \nsubseteq D$
5 select an edge $e = (u, v) \in E$ such that:
(i) $u \in D$;
(ii) $v \notin D$;
(iii) $\forall e' = (u', v') \in E$ such that $u' \in D$ and $v' \notin D$, $w(e) \geq w(e')$
6 $E' \leftarrow E' \bigcup \{e\}$
7 $D \leftarrow D \bigcup \{v\}$

Output: (V,E').

Frame 1: Algorithm GSM

The *unicast* and *broadcast* variants of the problem are the special cases for which $|N| = 1$ and $N \equiv V$, respectively; we term the corresponding problems as *Maximum Lifetime Unicast* (MLU) and *Maximum Lifetime Broadcast* (MLB). Note that, in the unicast case, the routing scheme consists of *paths*.

Additionally, we consider the special case in which the routing scheme is restricted to consist a single topology, i.e., we seek a single tree (or path, for unicast) of maximum lifetime. We term the corresponding multicast problem as *Minimum lifetime Single topology Multicast (MSM)*. We denote the unicast and broadcast problems as MSU and MSB correspondingly.

So far, we have considered a model in which a transmission is heard by all nodes within the transmission radius. In Section 7, we shall consider an alternative model, in which a transmission can be received by at most a single node.

3 Single Topology

In this section we investigate problems MSU, MSB and MSM, i.e., the problems of finding either a single path or a spanning tree that can be used for the longest period of time. In the following, we present greedy algorithms that solve Problem MSM in polynomial time. Since Problems MSU and MSB are special cases of Problem MSM, they can be solved by this algorithm as well.

In [12], an algorithm was presented that solved the single-topology broadcast problem (i.e., Problem MSB) within a time complexity of $O(((|E|)log(|V|)$. We now present an algorithm that solves this problem within a lower time complexity, namely, $O(|E| + |V|log(|V|))$. Our solution is based on a "bottleneck adaptation" of Prim's minimum spanning tree algorithm [10], as follows.

The algorithm's input is a directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a target set $N \in V$. We assign a weight w to each edge in G, namely, $w(u, v) = \frac{b(u)}{p(u,v)}$, and then execute the following variant of Prim's algorithm on G. A set of nodes D is initialized to include the source node s. Then, at each iteration, we select the *maximal edge* among all edges between a covered node and a non-covered node. We add that edge to the constructed tree and continue until all the nodes are covered. The formal algorithm, termed *Greedy Single-topology Multicast* (GSM), is specified in Frame 1.

We shall show that the spanning tree returned by Algorithm GSM is such that the weight of the minimal (i.e., bottleneck) edge is maximized. To that end, we need the following definition.

Definition 5 *Let $G = (V, E)$ be a directed graph. An edge e is a* bottleneck edge *of G if its weight is minimal in G, i.e. $w(e) = min_{e \in E} w(e)$. We denote $bw(G) = w(e)$.*

With this definition,we now show that the output of algorithm GSM maximizes the weight of the bottleneck edge.

Lemma 1 *Let T be the output of algorithm GSM, for a graph G and a source node s. Then:*

1. *T is a spanning tree of G rooted at s.*

Input: A directed weighted graph G, a source node and a target node $s, d \in V$ respectively.

1. $i \leftarrow 1$
2. $G' \leftarrow G$
3. Solve Problem MSU for G' and s, let p be the output path; if there is no such path, terminate.
4. Use p until some transmitting node consumes all its energy, let that time be t.
5. $G' \leftarrow G$ with the current residual energy.
6. $p_i \leftarrow p$; $t_i \leftarrow t$; $i \leftarrow i + 1$
7. Goto 3

Output: $\{(p_j, t_j) | 1 \leq j \leq i\}$.

Frame 2: Algorithm Greedy

2. For any other spanning tree T' of G rooted at s, $bw(T) \geq bw(T')$.

Proof: First, we show that the output of algorithm GSM is a tree rooted at s that spans N. This is done by induction on the nodes that are added to D, as follows. From the definition of the algorithm, it is immediate that, when a node is added to D, E' includes a directed path from s to that node. The algorithm halts only when D includes all the target nodes N. Hence, the output T includes a path from s to every target node.

We now show that T is such that the weight of the bottleneck edge is maximized. By way of contradiction, assume that there exists a tree T' rooted at s that spans N, such that $bw(T') > bw(T)$. Consider the stage in the algorithm where the first edge $e = (u, v)$ was added to E', such that $w(e) < bw(T')$. Let $p = (s = v_0, v_1, \ldots, v)$ be the path in T' from s to v. Let v_i be the first node in p, such that $v_i \notin D$. The edge $e' = (v_{i-1}, v_i)$ is in T', hence its weight is not less than that of the bottleneck edge of T'. Since we assumed that $w(e) < bw(T')$, we conclude that $w(e') > w(e)$. However, this contradicts the assumption that the edge e was chosen by Algorithm GSM: indeed, $v_{i-1} \in D, v \notin D$ and $w(e') > w(e)$, hence e' should have been selected instead of e. Therefore, at no stage the algorithm chooses an edge e such that $w(e) < bw(T')$. Thus, $bw(T) = min_{e \in E} w(e) \geq bw(T')$. ■

We now note that, by definition, the lifetime of a tree equals the weight of its bottleneck edge, i.e., $lifetime(T) \equiv min_{(u,v) \in T} \frac{b(u)}{p(u,v)} = min_{e \in E} w(e) = bw(T)$. Hence, by Lemma 1, Algorithm GSM finds a tree of maximum lifetime, i.e., solves Problem MSB.

The time complexity of algorithm MSU is the same as that of Prim's algorithm, which, using Fibonacci heaps, is $O(|E| + |V| log(|V|))$.

We note that the output T might contain leaf nodes that are not target nodes. Transmissions to such nodes are obviously unnecessary, and may be avoided. This can be done by performing a *post-sweep* algorithm, as specified in Appendix A, after which all leaf nodes are target nodes. We note, however, that while the post-sweep algorithm prevents unnecessary transmissions, it does not affect the lifetime of the tree.

As mentioned, Algorithm GSM solves the unicast and broadcast problems (MSU and MSB) as well.

4 Multi-topology Unicast

In this section we investigate Problem MLU, i.e., the problem of finding multi-topology routing schemes that maximize the lifetime of unicast. As shown in the previous section, finding single-topology solutions for the maximum lifetime unicast, broadcast and multicast problems can be obtained through a greedy approach. However, such a simple approach fails to solve the multi-topology versions of these problems. To illustrate the underlying difficulty, consider the application of Algorithm *Greedy*, specified in Frame 2 on Problem MLU.

We demonstrate the deficiency of such an algorithm through the example depicted in figure 2. The transmission power is indicated for each edge. The initial energies is indicated for each node. There are three possible paths, namely: $p_1 = S \rightarrow A \rightarrow D$, $p_2 = S \rightarrow B \rightarrow D$ and $p_3 = S \rightarrow C \rightarrow D$. Clearly, the optimal solution is to use p_1 and

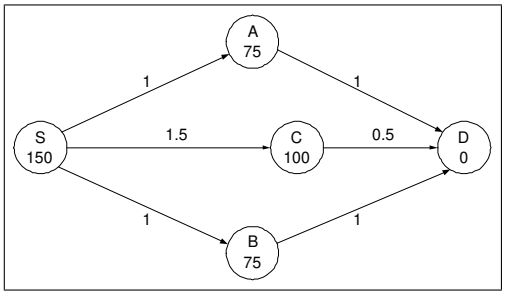


Figure 2: Inefficiency of greedy approach

p_3, each for 75 time units, yielding a maximum lifetime of 150 time units. The maximal time each path can be used is: $time(p_1) = 75$, $time(p_2) = 100$, $time(p_3) = 75$, hence, the proposed algorithm first chooses p_2 and uses it until the source node s consumes all its energy, after 100 time units. Therefore, the proposed algorithm achieves a solution that is 33.3% below the optimum. An alternative greedy algorithm might be considered, in which the shift to the next tree takes place when the currently used tree is not the optimal single-topology solution (at that point of time) anymore. However, this approach also fails in the above example. Indeed, the algorithm starts again with p_2 and uses it until the remaining time that node S can transmit to node B equals the time that node A can transmit to node D. Hence, the path p_2 is used for 25 time units. The maximal additional time possible is the remaining time that the source node S can transmit to nodes A and C. Since the remaining energy of node S is 112.5 and the required transmission power between nodes S and C, as well as between nodes S and A, is 1, that time is 112.5. Hence, the lifetime that can be achieved by this other greedy algorithm is $112.5 + 25 = 137.5$, which is 8.4% below the optimum. Clearly, more sophisticated approaches must be considered in order to optimally solve the multi-topology problems.

We recall that, Problem MLU, we need to find a set of paths such that the overall time we are able to transmit from the source node to the destination node over these paths is maximized. Our solution approach is based on describing the problem as a flow problem. However, in contrast to common flow formulations of network problems, e.g. [17], in which flow values represents traffic rates, we will use flows in order to represent time. That is, a path with a flow value of x will stand for using this path for a period of x time units. As we shall show, maximizing flow will ultimately maximize the total time, i.e., the lifetime. Energy constraints will be imposed as nodal flow constraints, while edge transmission powers will be translated into nodal constraints.

First, we consider the general version of Problem MLU, and show that it can be solved through linear programming; Later, where as in section 4.1, we consider a special case of problem MLU, in which each node has a single transmission power level, and show that this problem can be reduced to a standard maximum flow problem.

Specifically, we formulate problem MLU as a linear program on flows. For each node, the total time that each outgoing edge is used, multiplied by the transmission power corresponding to this edge, is bounded by the node's total energy. For each edge $e = (u, v) \in E$ we define a variable $x_{u,v} \equiv x_e$. The flow $x_{u,v}$ stands for the total time that (u, v) is used by the unicast scheme. Specifically, for a unicast scheme $\alpha = \{(p_1, t_1), (p_2, t_2), \ldots (p_k, t_k)\}$, let S_e be the set of paths that include the edge e, i.e., $S_e = \{i | e \in p_i\}$; then, $x_e = \sum_{i \in S_e} t_i$. In our solution scheme, we first solve a linear program, whose output is a network flow $\{x_e\}_{e \in E}$. Then, we employ *path decomposition* [18], in order to decompose $\{x_e\}_{e \in E}$ into path flows, out of which we compute the corresponding (optimal) unicast scheme. In frame 3 we specify *Procedure Maximal Time Flow* (*MTF*) that finds the maximal "time" flow. In frame 4 we present *Procedure Path Decomposition* (*PD*) that decomposes the flow into paths. Finally, in Frame 5, we present *Algorithm LPMLU*, which invokes Procedures MTF and PD in order to solve Problem MLU.

Referring to Frame 3, (4)-(7) are standard flow constraints. Specifically, (4) guarantees flow conservation at relay nodes, (5) guarantees that the outgoing flow from the source reaches the destination, (6) guarantees that no flow enters the source node nor leaves the destination, and (7) guarantees that the flow is positive. Finally, as we shall show, constraint (8) guarantees that the consumed nodal energy is bounded by its initial energy.

Next, in Frame 4, we describe Procedure Path Decomposition (PD), which calculates the unicast scheme using the output of the linear program. The basic idea is to iteratively find a path from the source to the destination, and add

Input: A directed weighted graph $G \equiv (V, E, b, p)$, and source and destination nodes $s, d \in V$, respectively.
Variables: edge flows, $\{x_{u,v} | \forall (u,v) \in E\}$.
Solve the following linear program:
maximize:

$$\sum_{v \in V} x_{s,v} \tag{3}$$

subject to:

$$\forall u \in V \backslash \{s,d\}, \sum_{v \in V} x_{u,v} = \sum_{v \in V} x_{v,u} \tag{4}$$

$$\sum_{v \in V} x_{s,v} = \sum_{v \in V} x_{v,d} \tag{5}$$

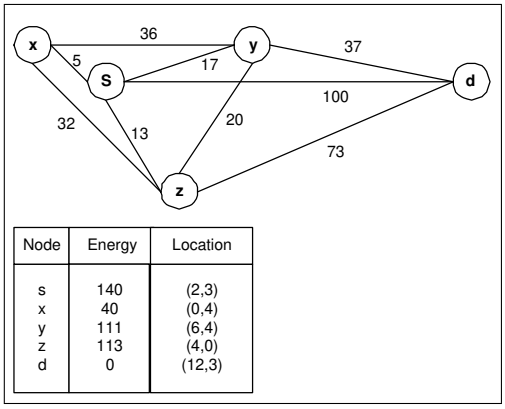
$$\forall u \in V, x_{u,s} = x_{d,u} = 0 \tag{6}$$

$$\forall u, v \in V, x_{u,v} \geq 0 \tag{7}$$

$$\forall u \in V, \sum_{v \in V} p((u,v)) \cdot x_{u,v} \leq b(u) \tag{8}$$

Output: $\{x_{i,j} | (i,j) \in E\}$

Frame 3: Procedure MTF



Node	Energy	Location
s	140	(2,3)
x	40	(0,4)
y	111	(6,4)
z	113	(4,0)
d	0	(12,3)

Figure 3: Multi-topology Unicast Example

it to the unicast scheme. The duration time that this path is used equals the minimal edge flow on that path. The procedure then reduces the flow value of each edge on that path by the flow value of the minimal edge. The algorithm continues until no path from the source to the destination can be found.

Finally, in Frame 5, we present Algorithm LPMLU, which solves Problem MLU.

Next, we demonstrate the operation of Algorithm LPMLU on a simple example. Consider the network in Fig. 3, which consists of 5 nodes, where s is the source and d is the destination. The locations of nodes on the plane and their values are specified in the table. The transmission power is indicated for each edge.

The output of Procedure MTF for this network is described in Fig. 4 by the numbers on the edges. Next, Procedure PD defines the paths and time durations. It chooses one of the paths from the source s to the destination d, i.e., one among $(s,x,z,y,d), (s,x,z,y,d), (s,z,y,d), (s,y,d), (s,d)$. Assume that it chooses the path (s,x,z,y,d). The minimal

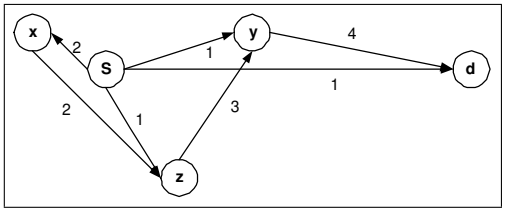


Figure 4: Output of Procedure MTF

Input: A directed weighted graph $G \equiv (V, E, b, p)$, source and destination nodes $s, d \in V$, respectively, and a flow on each edge, $\{x_{i,j} | (i,j) \in E\}$.

1. $i \leftarrow 1$
2. $\forall x_{u,v}, y_{u,v} \leftarrow x_{u,v}$
3. Create an auxiliary graph $G' \equiv (V, E')$ such that $e \equiv (u,v) \in E' \Leftrightarrow y_{u,v} > 0$
4. Find a path p from s to d in G', if none is found, terminate
5. $y_m \leftarrow min_{(u,v)\in p} y_{u,v}$
6. $p_i \leftarrow p$
7. $t_i \leftarrow y_m$
8. $\forall (u,v) \in p, y_{u,v} \leftarrow y_{u,v} - y_m$
9. $i \leftarrow i + 1$
10. Continue from step 3

Output: A unicast scheme $\alpha = \{(p_1, t_1), (p_2, t_2), \ldots (p_i, t_i)\}$.

Frame 4: Procedure PD

Input: A directed weighted graph $G \equiv (V, E, b, p)$, and source and destination nodes $s, d \in V$, respectively.

1. Execute procedure MTF on G, s, d. Let $\{x_{u,v}\}$ be the output of Procedure MTF.
2. Execute procedure PD on $\{x_{u,v}\}$. Let the unicast scheme $\alpha = \{(p_1, t_1), (p_2, t_2), \ldots (p_i, t_i)\}$ be its output.

Output: α.

Frame 5: Algorithm LPMLU

edge weight along this path is that of (s, x), namely, 2. Therefore, the time duration for path p_1 is set to 2. The last step of the first iteration is to reduce the value of each edge along p_1 by 2. During the next three iterations, the algorithm chooses the paths $(s, z, y, d), (s, y, d), (s, d)$, each with a time duration of 1. Hence, the unicast scheme is: $\alpha = \{((s, x, z, y, d), 2)((s, z, y, d), 1), ((s, y, d), 1), ((s, d), 1)\}$.

We proceed to prove that algorithm LPMLU solves problem MLU in polynomial time. First, we have to show that, when decomposing the flow into path flows, the total flow of all the paths remains unchanged. Additionally, we have to show that the decomposition process does not violate the energy constraints of the nodes.

Lemma 2 *Let $y^k_{u,v}$ be the value of edge (u, v) after the k'th iteration of Procedure PD. Then, the assignment of $y^k_{u,v}$ to the edges represents a legal flow from the source to the destination, i.e.,*

$$\forall u \in V \backslash \{s, d\}, \sum_{v \in V} y^k_{u,v} = \sum_{v \in V} y^k_{v,u} \tag{9}$$

$$\sum_{v \in V} y^k_{s,v} = \sum_{v \in V} y^k_{v,d}. \tag{10}$$

Proof: By induction on the iteration of Procedure PD. Before the first iteration, the conditions hold, as the input satisfies the flow constraints (4),(5) in the LP problem defined by Procedure MTF. After the $k+1$'st iteration, a path p is found from s to d with minimal edge value y_m. Then, for all nodes u in $V \backslash \{s, d\}$:

1. if $u \notin p$, then $\forall v \in V, y^k_{u,v} = y^{k+1}_{u,v}$ and $y^k_{v,u} = y^{k+1}_{v,u}$, therefore

$$\sum_{v \in V} y^{k+1}_{u,v} = \sum_{v \in V} y^k_{u,v} = \sum_{v \in V} y^k_{v,u} = \sum_{v \in V} y^{k+1}_{v,u}. \tag{11}$$

2. if $u \in p$, then

$$\sum_{v \in V} y_{u,v}^{k+1} = (\sum_{v \in V} y_{u,v}^{k}) - y_m = (\sum_{v \in V} y_{v,u}^{k}) - y_m = \sum_{v \in V} y_{v,u}^{k+1}. \tag{12}$$

Finally, for the nodes s and d:

$$\sum_{v \in V} y_{s,v}^{k+1} = (\sum_{v \in V} y_{s,v}^{k}) - y_m = (\sum_{v \in V} y_{v,d}^{k}) - y_m = \sum_{v \in V} y_{v,d}^{k+1}. \ \blacksquare \tag{13}$$

Next, we have to show, that, when Procedure PD terminates, the residual flow through the outgoing edges from the source is 0, which implies that the flow of all paths equals the original flow.

Lemma 3 *When Procedure PD terminates, for all nodes $v \in V$, $y_{s,v} = 0$.*

Proof: Assume that, in contradiction to the lemma, Procedure PD terminates, yet there is some flow value $y_{s,v} > 0$. By definition of Procedure PD, no path was found from s to d, since this is the halting condition of the algorithm. Consider a walk in the graph from the source node, using only edges with flow value $y_e > 0$. According to the assumption, $y_{s,v} > 0$, thus, there is a path from s to v. Continuing from v, it follows from Lemma 2 that each node we visit must have an outgoing edge. This process continues until we get to d, contradicting the assumption that there is no path, or else a cycle is created. In the later case, we reduce the minimal edge of the cycle out of all the cycle's edges. We start again from v, and note that (s, v) is not on a cycle since the in-degree of s is 0. Also, note that, after deleting a cycle, constraint (4) still holds, since deleting a cycle implies the subtraction of the same values from both sides of the equation. The number of cycles is bounded by the number of edges, hence, eventually, no more cycles exist and we must either get to d, contradicting the assumption that there is no path, or else to a node that does not satisfy constraint (4), contradicting Lemma 2. ■

We now show that the overall time during which an edge is used by all the paths is less or equal to the time it is used in the original flow, i.e., in the output of Procedure MTF. This would implies that the nodes' energy constraints, which are obeyed by the original flow, are not violated after Procedure PD is executed.

Lemma 4 *After each iteration r of Procedure PD:*

$$y_e^r = x_e - \sum_{i=1}^{r} \delta(p_i, e) \cdot t_i \tag{14}$$

Proof: We prove the claim by induction on the iteration number r. For $r = 0$, as $\sum_{i=1}^{r} \delta(p_i, e) \cdot t_i = 0$, we have $y_e^0 = x_e$, hence, $y_e^0 = x_e - \sum_{i=1}^{r} \delta(p_i, e) \cdot t_i$, as required. After iteration $r + 1$:

1. If $e \notin p_{r+1}$, then $\delta(p_{r+1}, e) = 0, y_e^{r+1} = y_e^r$, therefore,

$$x_e - \sum_{i=1}^{r+1} \delta(p_i, e) \cdot t_i = x_e - \sum_{i=1}^{r} \delta(p_i, e) \cdot t_i = y_e^r = y_e^{r+1}. \tag{15}$$

2. If $e \in p_{r+1}$ then by definition of Procedure PD $t_{r+1} \leq y_e^{r+1}$, therefore,

$$x_e - \sum_{i=1}^{r+1} \delta(p_i, e) \cdot t_i = x_e - \sum_{i=1}^{r} \delta(p_i, e) \cdot t_i - t_{r+1} = y_e^r - t_{r+1} = y_e^{r+1}. \ \blacksquare \tag{16}$$

For each edge e and iteration r, $y_e^r \geq 0$, hence by the above lemma we also conclude that the time that each edge is used is bounded by the value of the original flow on that edge, i.e. $\sum_{i=1}^{k} \delta(p_i, e) \cdot t_i \leq x_e$.

By Lemmas 2 - 4, we finally conclude that the solution returned by Algorithm LPMLU is a (legal) unicast scheme, that is, the nodes' energy constraints are obeyed. Moreover, since all of the original flow was converted into paths, the total time of all the paths equals the flow on the outgoing edges of the source. Formally:

Lemma 5 *The output of LPMLU is a DU α, and $lifetime(\alpha) = \sum_{v \in V} x_{s,v}$.*

Proof: In order to show that the output is a (legal) unicast scheme, we have to show that, for all nodes, the total energy used by the node in all the paths is bounded by its initial energy.

First, we have to show that, $\forall u \in V$, condition (1) holds.

$$\forall u \in V, \sum_{i=1}^{k} \sum_{v \in V} \delta(p_i, (u,v)) \cdot t_i \cdot p((u,v)) = \tag{17}$$

$$\sum_{v \in V} p((u,v)) \cdot \sum_{i=1}^{k} \delta(p_i, (u,v)) \cdot t_i. \tag{18}$$

Now, by Lemma 4,

$$\sum_{v \in V} p((u,v)) \cdot \sum_{i=1}^{k} \delta(p_i, (u,v)) \cdot t_i \leq \sum_{v \in V} p((u,v)) \cdot x_{v,u}, \tag{19}$$

and by constraint (8),

$$\sum_{v \in V} p((u,v)) \cdot x_{v,u} \leq b(v), \tag{20}$$

hence condition (1) holds. Second, we have to show that $lifetime(\alpha) = \sum_{v \in V} x_{s,v}$. By Lemma 3, we know that $\forall v \in V, y^k_{s,v} = 0$. By Lemma 4, we know that $y^k_e = x_e - \sum_{i=1}^{k} \delta(p_i, e) \cdot t_i$. Therefore,

$$\forall v \in V, x_{s,v} = \sum_{i=1}^{k} \delta(p_i, (s,v)) \cdot t_i \tag{21}$$

hence,

$$\sum_{v \in V} x_{s,v} = \sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (s,v)) \cdot t_i = \sum_{i=1}^{k} t_i \cdot \sum_{v \in V} \delta(p_i, (s,v)). \tag{22}$$

Since, in every path, the source s has exactly one outgoing edge, then, $\sum_{v \in V} \delta(p_i, (s,v)) = 1$. Therefore,

$$\sum_{i=1}^{k} t_i \cdot \sum_{v \in V} \delta(p_i, (s,v)) = \sum_{i=1}^{k} \cdot t_i = lifetime(\alpha) \tag{23}$$

thus concluding that $\sum_{v \in V} x_{s,v} = lifetime(\alpha)$. ■

Having shown that the solution of the algorithm is a (legal) DU, we now show that any (DU) solution of problem MLU must satisfy the constraints of the linear program defined by procedure MTF, namely, (4)-(8), hence it can be admitted as a solution by Algorithm LPMLU. In order to establish this, we have to show that, after applying the "inverse operation" of Procedure PD on a unicast scheme solution, the LP constraints (4)-(8) hold. We term this "inverse operation" as *paths superposition transformation* (T_p), formally defined as follows.

Definition 6 *The* Paths Superposition transformation *(T_p) of a unicast scheme $\alpha = \{(p_1, t_1), (p_2, t_2), ..., (p_k, t_k)\}$, is an assignment of the corresponding sum of path-time durations to each edge, i.e., $T_p(\alpha) = \{x_{u,v} = \sum_{i=1}^{k} \delta(p_i, (u,v)) \cdot t_i | u, v \in V\}$.*

Lemma 6 *Given a DU $\alpha = \{(p_1, t_1), (p_2, t_2), ..., (p_k, t_k)\}$, the constraints (4)-(8) of the linear program hold for $T_p(\alpha)$, and $\sum_{i=1}^{k} t_i = \sum_{v \in V} x_{s,v}$.*

Proof: First, we have to show that constraints (4)-(8) hold. Starting with constraint (4), we have that:

$$\forall u \in V \backslash \{s, d\}, \sum_{v \in V} x_{u,v} = \sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (u,v)) \cdot t_i \tag{24}$$

Since $u \neq s$ and $u \neq d$, then, if (u, w) is in path p, then there is some edge (v, u) in p. Therefore,

$$\sum_{i=1}^{k} \cdot t_i \sum_{v \in V} \delta(p_i, (u, v)) = \sum_{i=1}^{k} \cdot t_i \sum_{v \in V} \delta(p_i, (v, u)) = \tag{25}$$

$$\sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (v, u)) \cdot t_i = \sum_{v \in V} x_{v,u}, \tag{26}$$

hence establishing (4). Turning to (5), we have that:

$$\sum_{v \in V} x_{s,v} = \sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (s, v)) \cdot t_i = \tag{27}$$

$$\sum_{i=1}^{k} \cdot t_i \sum_{v \in V} \delta(p_i, (s, v)) = \sum_{i=1}^{k} \cdot t_i \sum_{v \in V} \delta(p_i, (v, d)) = \tag{28}$$

$$\sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (v, d)) \cdot t_i = \sum_{v \in V} x_{v,d} \tag{29}$$

hence establishing (5). Constraint (6) (namely, $\forall v \in V, x_{v,s} = x_{d,v} = 0$) holds, since paths start at s and terminate at d. Constraint (7) (namely, $\forall u, v \in V, x_{u,v} \geq 0$) holds, since time durations are positive. Finally, considering constraint (8), we have that:

$$\forall v \in V, \sum_{v \in V} p((u, v)) \cdot x_{u,v} = \sum_{v \in V} p((u, v)) \cdot \sum_{i=1}^{k} \delta(p_i, (u, v)) \cdot t_i = \sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (u, v)) \cdot t_i \cdot p((u, v)) \tag{30}$$

since α is a (legal) unicast scheme,

$$\sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (u, v)) \cdot t_i \cdot p((u, v)) \leq b(u), \tag{31}$$

hence establishing (8).

Finally, we show that $\sum_{i=1}^{k} t_i = \sum_{v \in V} x_{s,v}$. Indeed,

$$\sum_{v \in V} x_{s,v} = \sum_{v \in V} \sum_{i=1}^{k} \delta(p_i, (s, v)) \cdot t_i = \sum_{i=1}^{k} t_i \cdot \sum_{v \in V} \delta(p_i, (s, v)) = \sum_{i=1}^{k} t_i. \quad \blacksquare \tag{32}$$

We have shown that the output of the algorithm is a (legal) unicast scheme. Constraints (4)-(8) admit any (legal) unicast scheme solution. The objective of the linear program is to maximize the lifetime of the unicast scheme. Putting it all together, we can finally conclude that the output of Algorithm LPMLU is a unicast scheme of maximum lifetime.

Theorem 1 *Given are a directed weighted graph $G \equiv (V, E, b, p)$ and source and destination nodes $s, d \in V$, respectively. Then, Algorithm LPMLU solves Problem MLU.*

Proof: Let α be the output of Algorithm LPMLU with total time t. By Lemma 5, α is a legal solution. Assume that there is a solution β with total time t', such that $t' > t$, in contradiction to the theorem. By Lemma 6, the linear program constraints, namely, (4)-(8), hold for $T_p(\beta)$, moreover $T_p(\beta)$ yields a better (higher) value than $T_p(\alpha)$ for the linear program objective, which (in view of Procedure MTF), is a contradiction. ■

Finally, we address the time complexity of the algorithm.

Theorem 2 *The time complexity of LPMLU is polynomial.*

Proof: The number of equations in (4)-(8) is linear in the number of nodes, and the number of variables is linear in the number of edges. Therefore, the first part of the algorithm can be solved through any polynomial algorithm for

Input: A directed weighted graph $G \equiv (V, E, b, p)$, source and destination $s, d \in V$, respectively.

1. Construct an auxiliary graph $G' \equiv (V', E', c)$, $V' = V \bigcup \{v' | v \in V\}$, $E' = E \bigcup \{(v, v') | v \in V\}$. The capacity of an edge (v, v') is $c(v, v') = \frac{b(v)}{p(v)}$, and the capacity of any other edge e is infinite, i.e., $c(e) = \infty$.
2. Find a maximal flow from s to d in G'.
3. Execute Procedure PD on the flow obtained at step 2.

Output: $\{(p_i, t_i) | 1 \leq i \leq k\}$ (as provided by Procedure PD).

Frame 6: Algorithm NMF

linear programming [19]. The second part, namely, Procedure PD, is polynomial as well, since the complexity of each iteration is $O(|E|)$ and the number of iterations is also $O(|E|)$. ■

Thus, we have an optimal polynomial solution to problem MLU.

4.1 Single Transmission Power

Consider now the case where nodes do not support multiple transmission levels, i.e., each node has a single transmission level (not necessarily equal for all nodes). We present a more efficient algorithm for this case. Denote by p_u the (single) transmission power value of a node $u \in V$. Let $S = \{v |$ u can reach v with transmission power $p_u\} = \{v | distance(u, v)^\alpha \leq p_u\}$. Then, the outgoing edges from node u are $A = \{(u, v) | v \in S\}$, and $p(u, v) = p_u$.

Algorithm *Node-constrained Max Flow* (*NMF*), specified in the following, solves this problem using a reduction to the Max Flow problem [18]. Again, the idea is to find a "maximal flow of time". However, unlike the general case, now when a node participates in a path, the amount of energy it consumes does not depend on the node it transmits to. Thus, we can bound the energy usage of each node by a single edge with time value $\frac{b(v)}{p(v)}$. We define $p \colon V \rightarrow \Re$ as: $\forall v \in V, p(v) = max_{(u,v) \in E} p(u, v)$. Note that, so far, p was defined only on the edges. This observation leads us to Algorithm NMF, as specified in Frame 6.

Next, we prove that Algorithm NMF solves Problem MLU for the case of single nodal transmission levels. Namely, we show that the output of the algorithm is a legal unicast scheme, with maximum lifetime.

Theorem 3 *Given are a directed weighted graph $G \equiv (V, E, b, p)$, with single transmission levels, and source and destination nodes $s, d \in V$, respectively. then, Algorithm NMF solves problem MLU.*

Proof: First, we show that the output of Algorithm NMF, α, is a (legal) DU. If an edge (u, v) is such that $(u, v) \in p \in \alpha$, then $(u, u') \in p$, due to the way that the graph was constructed. Therefore,

$$\forall u \in V, \sum_{i=1}^{k} \sum_{v \in V} \delta(p_i, (u, v)) \cdot t_i \cdot p((u, v)) = \tag{33}$$

$$\sum_{i=1}^{k} \sum_{v \in V} \delta(p_i, (u, u')) \cdot t_i \cdot p(u)). \tag{34}$$

The flow is bounded by the edge capacities, hence,

$$\sum_{i=1}^{k} \sum_{v \in V} \delta(p_i, (u, u')) \cdot t_i \cdot p(u)) \leq c(u, u') = \frac{b(u)}{p(u)}. \tag{35}$$

Thus,

$$\forall u \in V, \sum_{i=1}^{k} \sum_{v \in V} \delta(p_i, (u, v)) \cdot t_i \cdot p((u, v)) \leq \frac{b(u)}{p(u)} \cdot p(u) = b(u). \tag{36}$$

Therefore, α is a DU. We now show that the lifetime of α is maximal. Assume, by way of contradiction, that there is another DU $\beta = \{(p_1, t_1), (p_2, t_2), ..., (p_k, t_k)\}$ in G, such that $lifetime(\alpha) < lifetime(\beta)$. We construct the auxiliary

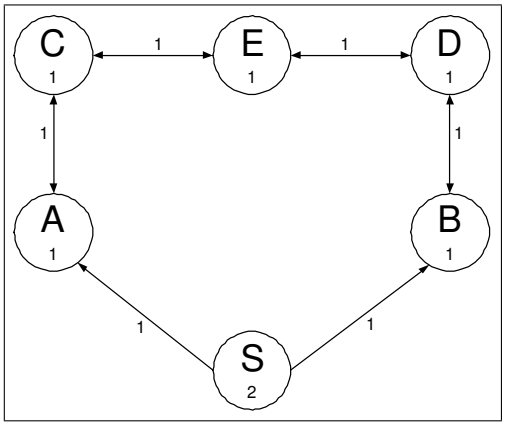


Figure 5: Problem MLB - difficulty example

graph G', as in Algorithm NMF. We then use β to define a flow on G' as follows. To each edge $e \in E$ we assign a flow value that is equal to the sum of duration time of the paths that includes e. Since β is a DU, it is clear that the flow over all the paths is bounded by the capacity of the edges, which is a contradiction to the correctness of the Max Flow solution that generated α. ■

Finally, the computation time complexity of Algorithm NMF is the same as that of the solution to the Maximum Flow problem with $2 * |V|$ nodes and $|V| + |E|$ edges.

5 Approximations for Multi-topology Broadcast and Multicast

5.1 NP-hardness of Problem MLB

In this section we consider Problem MLB, i.e., finding a set of spanning trees and a duration time per tree such that the energy constraints are satisfied and the sum of duration times is maximized. Before turning to this problem directly, let us consider an equivalent problem in directed non-wireless networks, known as the directed spanning tree packing problem, which is defined as follows. Given are a graph $G = (V, E)$, a source node s in V, and a non-negative capacity for each edge. A valid packing of trees is a set of trees and a bandwidth per tree, such that for each edge the sum of bandwidths of all the trees consisting that edge is bounded by its capacity. It is required to find a valid tree packing such that the sum of bandwidths over all the trees is maximized. An optimal solution can be found efficiently [20]. Most of the solution algorithms rely on the fact that, if the value of the maximum flow from the source to any node is at least F, then there is a tree packing with a value of F [21]. Unfortunately, in wireless networks, this property does not necessary hold, as shown in Figure 5. The transmission power is indicated for each edge, and the initial energy of the nodes are indicated for each node. We note that the value of the multi-topology unicast problem for each node is 2. Specifically, the unicast routing scheme from S to A consists of the path $S \rightarrow A$, with a duration time of 2; the routing scheme for B is symmetrical; the unicast routing scheme from S to C consists of the paths $S \rightarrow A \rightarrow C$ and $S \rightarrow B \rightarrow D \rightarrow E \rightarrow C$, each with a duration time of 1; the routing scheme for node D is symmetrical; the unicast routing scheme from S to E consists of the paths $S \rightarrow A \rightarrow C \rightarrow E$ and $S \rightarrow B \rightarrow D \rightarrow E$, each with a duration time of 1. Thus, for each node there is a unicast scheme with a lifetime of 2. However, we now show that there is no broadcast scheme with a value of 2.

Given any broadcast scheme α, let α_E be the trees of α in which node E transmits, and let $\alpha \backslash \alpha_E$ be the rest of the trees. E has a single energy unit, hence: *(1)* $lifetime(\alpha_E)$ *is bounded by 1.* $\alpha \backslash \alpha_E$ consists of at most two trees, namely, the trees in which the transmitting nodes are $\{S, A, B, C\}$ and $\{S, A, B, D\}$. Therefore: *(2) In any tree of* $\alpha \backslash \alpha_E$*, both* A *and* B *must transmit.* Each of A, B has a single energy unit, hence, by (2), we have that: *(3)* $lifetime(\alpha \backslash \alpha_E)$ *is bounded by 1.* But if $lifetime(\alpha) = 2$, then, by (1) and (3), we have that: *(4)* $lifetime(\alpha_E) = 1$, and: *(5)* $lifetime(\alpha \backslash \alpha_E) = 1$. By (2) and (4) we have that both A and B consume all their energy during the transmissions of $\alpha \backslash \alpha_E$. Since in any tree either A or B must transmit, we have that, $lifetime(\alpha_E) = 0$. Therefore,

Input: An instance of the satisfiability problem, i.e., a CNF form expression, φ, consisting of clauses $C_j, j = 1, 2, \ldots, m$ over variables $x_i, i = 1, 2, \ldots, n$.
Construct the auxiliary graph $G \equiv (V, E, b, p)$**:**

1. The set of nodes group V consists of the following set (i.e., $V = \cup_{i=1}^{3} V_i$):
   (a) $V_1 = \{s, T, F\}$, where s is the source.
   (b) For each variable x_i, there are 3 corresponding nodes $x_i, \overline{x_i}, a_i$,
   i.e., $V_2 = \cup_{i=1}^{n} \{x_i, \overline{x_i}, a_i\}$.
   (c) A node for each clause,
   i.e., $V_3 = \{C_j | 1 \leq j \leq m\}$.
2. The set of edges E consists of the following edge sets (i.e., $E = \cup_{i=1}^{5} E_i$):
   (a) Node s has outgoing edges to T and F,
   i.e., $E_1 = \{(s, T), (s, F)\}$.
   (b) Nodes T and F will have outgoing edges to each literal,
   i.e., $E_2 = \cup_{i=1}^{n} \{(T, x_i), (T, \overline{x_i}), (F, x_i), (F, \overline{x_i})\}$.
   (c) Each node x_i and $\overline{x_i}$, has an outgoing edges to each node a_i,
   i.e., $E_3 = \cup_{i=1}^{n} \{(x_i, a_i), (\overline{x_i}, a_i)\}$
   (d) Each node that corresponds to a literal has an outgoing edge to each node that corresponds to a clause containing that literal,
   i.e., $E_4 = \{(x_i, C_j) | 1 \leq j \leq m, 1 \leq i \leq n, x_i \in C_j\} \cup$
   $\{(\overline{x_i}, C_j) | 1 \leq j \leq m, 1 \leq i \leq n, \overline{x_i} \in C_j\}$.
   (e) Node F has an outgoing edge to each "clause node",
   i.e., $E_5 = \{(F, C_j) | 1 \leq j \leq m\}$.
3. For all edges e, $p(e) = 1$.
4. For the source s, $b(s) = 2$, and for any other node v, $b(v) = 1$.

Output: (V, E, b, p).

Frame 7: Reduction SAT → MLB

the total lifetime of any broadcast scheme is less then 2.

We turn to prove that Problem MLB is NP-hard. Actually, we shall show that it is NP-hard even if the problem is limited to the case in which all nodes have a single identical transmission power level or alternatively to the case in which the initial energy of all nodes is equal. We note that if all transmission power levels are equal and the initial energy of all the nodes is equal then the Maximum lifetime single-topology Broadcast tree is also an optimal multi-topology broadcast scheme, hence, in this case, the optimal solution can be efficiently calculated by Algorithm GSM defined in Section 3.

Theorem 4 *Problem MLB is NP-hard, even if one of the following conditions holds:*

1. *Every node has a single transmission power level, all transmission power levels are equal, and the initial energies of all nodes except one are equal.*
2. *The initial energy of all nodes is equal, and the transmission power levels of all nodes except one are equal.*

Before proving the above theorem, we define a reduction from the *Satisfiability problem (SAT)* to Problem MLB. An instance of SAT is a *Conjunctive Normal Form (CNF)* expression, i.e., a conjunction of clauses, where each clause is a disjunction of literals. A CNF expression is in SAT if it is satisfiable. The reduction SAT → MLB is specified in Frame 7.

We exemplify the construction of G for the expression $\varphi = ((\overline{x_1}) \wedge (\overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3})$ in Fig. 6.

Given an instance of Problem SAT, we apply reduction SAT → MLB and obtain an instance G of Problem MLB. We begin by proving the following claims, each referring to the graph G constructed in the reduction SAT → MLB.

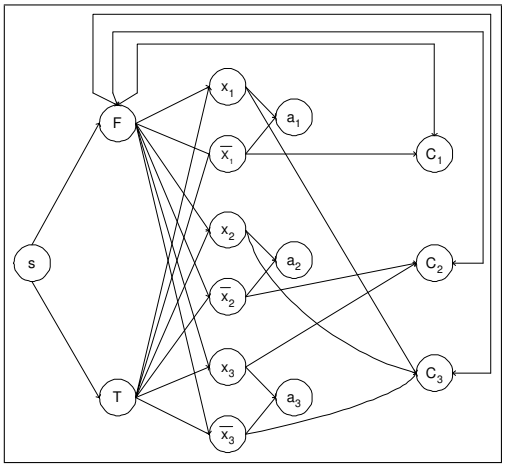


Figure 6: Example of auxiliary graph

Claim 1 *Each tree in G can be defined by the set of transmitting nodes.*

Proof: All edges in G have the same power and each node has a single transmission power level. ■

Claim 2 *The lifetime time of a broadcast scheme in G is bounded by 2.*

Proof: The source can transmit at most 2 time units. ■

Claim 3 *If there is a broadcast scheme α in G, such that $lifetime(\alpha) = 2$, then there is a tree, $h \in \alpha$, with time duration $t > 0$, in which T transmit and F does not transmit.*

Proof: Every spanning tree must include either T or F, and if F transmits in all trees, then the total time is bounded by 1. ■

Claim 4 *If there is a broadcast scheme, α in G, such that $lifetime(\alpha) = 2$, then, for each tree $h \in \alpha$ with time duration $t > 0$ and each $1 \leq i \leq n$, either x_i transmits or $\overline{x_i}$ transmits in h, but not both.*

Proof: If none of them transmits then there is no path to node a_i. If both transmit then, after using h for t time units, the residual energy of x_i and $\overline{x_i}$ is $1-t$ each, hence node a_i can receive transmissions for at most $2-2t$ additional time units, i.e., $lifetime(\alpha - h) \leq 2 - 2t$ (where $\alpha - h$ is a broadcast scheme that includes all trees of α except h). Therefore, $lifetime(\alpha) = lifetime(\alpha - h) + t \leq 2 - 2t + t = 2 - t$, contradicting the assumption that $lifetime(\alpha) = 2$. ■

We can now prove Theorem 4.

Proof(Theorem 4): We prove the theorem through a reduction from the satisfiability problem, namely Reduction SAT $\rightarrow$ MLB, in which there is an optimal broadcast scheme with lifetime of 2 time units, if and only if φ is satisfiable. Clearly, if there is an algorithm that solves Problem MLB in polynomial time, then we can solve the satisfiability problem in polynomial time.

We note that Reduction SAT $\rightarrow$ MLB can be calculated in polynomial time.

Suppose that some assignment τ satisfies φ. We have to show that there is an optimal broadcast scheme with lifetime of 2 time units in G. Consider the following broadcast scheme that consists of two trees, each fully defined by the set of transmitting nodes (in view of Claim 1). The first tree is defined by the following set of transmitting nodes: s, T and for all $1 \leq i \leq n$, it contains x_i if $\tau(x_i) = True$, otherwise it contains $\overline{x_i}$, i.e., $\{x_i | 1 \leq i \leq n, \tau(x_i) = True\} \cup \{\overline{x_i} | 1 \leq i \leq n, \tau(x_i) = False\}$. We show that the tree spans G by identifying a path on the tree to each of the nodes. Each node x_i ($\overline{x_i}$) $1 \leq i \leq n$, has the following path in the tree: $s \rightarrow T \rightarrow x_i$ ($s \rightarrow T \rightarrow \overline{x_i}$). Each node $a_i, 1 \leq i \leq n$, has the following path: if $\tau(x_i) = True$, then the path is $s \rightarrow T \rightarrow x_i \rightarrow a_i$, else it is $s \rightarrow T \rightarrow \overline{x_i} \rightarrow a_i$. τ satisfies φ, hence, for each clause $C_j, 1 \leq j \leq m$, there is some variable x_i, such that $x_i \in C_j$ and $\tau(x_i) = True$ or $\overline{x_i} \in C_j$ and $\tau(x_i) = False$, hence there is a path $s \rightarrow T \rightarrow x_i \rightarrow C_j$ in the former case, and a path $s \rightarrow T \rightarrow \overline{x_i} \rightarrow C_j$ in the

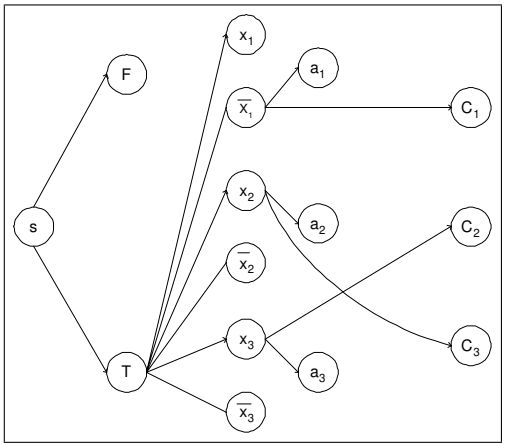


Figure 7: Example of Tree 1

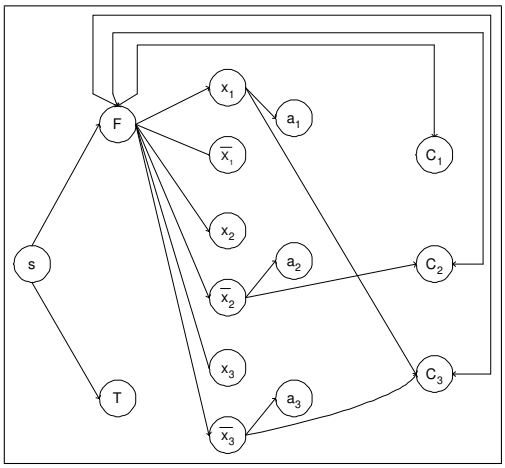


Figure 8: Example of Tree 2

latter. The second tree is defined by the following set of transmitting nodes: r, F and for all $1 \leq i \leq n$, it contains x_i if $\tau(x_i) = False$, otherwise it contains $\overline{x_i}$. Again, we show that the tree spans G by identifying a path on the tree to each of the nodes. Each node $x_i, \overline{x_i}$ $1 \leq i \leq n$, has the following path in the tree: $s \rightarrow F \rightarrow x_i, \overline{x_i}$. Each node $a_i, 1 \leq i \leq n$, has the following path: if $\tau(x_i) = False$, then the path is $s \rightarrow F \rightarrow x_i \rightarrow a_i$, else it is $s \rightarrow F \rightarrow \overline{x_i} \rightarrow a_i$. Each clause node $C_j, 1 \leq j \leq m$, has the following path: $s \rightarrow F \rightarrow C_j$. We set the time duration of each tree to be a single time unit. Since each node other than the source transmits in a single tree and the transmission power is 1, then the energy constraints hold. Hence, the broadcast scheme is legal and has a total duration time of 2, which by Claim 2, is the optimal value.

We illustrate the trees for the expression $\varphi = ((\overline{x_1}) \wedge (\overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3})$ in Figures 7 and 8, where τ assign True to x_2, x_3 and False to x_1.

Conversely, suppose there is a broadcast scheme α, such that $lifetime(\alpha) = 2$. By Claim 3, there exists a tree $h \in \alpha$, with time duration t, in which T transmits and F does not. Since h is a spanning tree and F does not transmit in h, then every clause node C_j must have a path of the form $s \rightarrow T \rightarrow z \rightarrow C_j$, where $z \in C_j$. By Claim 4, $\overline{z}$ does not transmit. Therefore, h implies an assignment τ that satisfies φ, as follows: $\tau(x_i) = True$ if x_i transmit in h and $False$ otherwise.

We note that we have reduced the satisfiability problem to an instance of Problem MLB in which each node has a single transmission power level, all transmission power levels are equal, and the initial energies as all nodes except one are equal. To complete the proof, we have to show that there is a reduction from an instance of Problem SAT to an instance of Problem MLB in which each node has a single transmission power level, the initial energies at all nodes are equal, and the transmission power levels of all nodes except one are equal.

Consider the reduction SAT $\rightarrow$ MLB, except that now $b(s) = 1$, and for each edge e outgoing from the source

Maximize:

$$\sum_{h \in \Upsilon} t_h \tag{37}$$

subject to:

$$\forall u \in V, \sum_{h \in \Upsilon} t_h \cdot \max_{(u,v) \in h} p(u,v) \leq b(u) \tag{38}$$

$$\forall h \in \Upsilon, t_h \geq 0 \tag{39}$$

Frame 8: Linear Program for Problem MLB

$p(e) = \frac{1}{2}$. Clearly, the required conditions are imposed and the reduction is still valid. ■

In some scenarios there might be a limitation on the number of trees that can be used in a broadcast scheme. We consider the simplest case, termed Problem MLB2t, in which at most two trees can be used, i.e. the problem is to find two spanning trees h_1, h_2 and time durations t_1, t_2 respectively, such that $t_1 + t_2$ is maximized. We show that Problem MLB2t is NP-hard, even if every node has a single transmission power level.

Theorem 5 *Problem MLB2t is NP-hard, even if one of the following conditions holds:*

1. *Every node has a single transmission power level, all transmission power levels are equal, and the initial energies of all nodes except one are equal.*
2. *The initial energy of all nodes is equal, and the (single) transmission power levels of all nodes except one are equal.*

Proof: Given a CNF form expression, φ, we construct the same auxiliary graph G as in the proof of Theorem 4. We need to show that there is an optimal broadcast scheme that consists of at most two trees with lifetime 2 if and only if φ is satisfiable. The proof is identical to that of Theorem 4, since if for a φ that is satisfiable we have shown how to construct a broadcast scheme that consisted of two trees with lifetime 2. ■

We note that Problem MLB2t has a trivial 2-approximation algorithm, namely, any algorithm that solves Problem MSB, e.g. the one presented in section **??**, provides such an approximation. We proceed to prove this claim.

Proposition 1 *Given is an instance of Problem MLB2t, G. Let α be an optimal solution and h a tree with maximal lifetime in G (i.e., a solution to Problem MSB for the same instance G). Then, $lifetime(h) \geq \frac{lifetime(\alpha)}{2}$.*

Proof: Let h_1, h_2 be the trees in α with durations t_1, t_2 respectively. We have $lifetime(h) > lifetime(h_1) \geq t_1$, $lifetime(h) > lifetime(h_2) \geq t_2$, hence $lifetime(h) > \frac{t_1+t_2}{2} = \frac{lifetime(\alpha)}{2}$ ■

5.2 Linear programming formulation

In this section we formulate Problem MLB as a linear program. This formulation will by used in the following sections to further analyze the problem. Given is an instance of Problem MLB, i.e., a directed weighted graph $G \equiv (V, E, b, p)$ and a source node $s \in V$. Let Υ denote the collection of all the directed spanning trees in G, rooted at s. Let t_h denote the time that a tree h is used. With these definitions, Problem MLB can be formulated as the linear program specified in Frame 8.

Constraint (38) guarantees that the total time that a node transmits over all the trees is bounded by the initial energy of that node. Note that the *maximum* operator in this constraint simply implies that, for each node u and for the given tree, the maximal transmission power $p(u, v)$ is selected over all edges (u, v) in that tree, that is, the expression $\max_{(u,v) \in h} p(u, v)$ is a constant. Constraint (39) guarantees that the tree weights are non-negative.

Since the number of variables is exponential, and in an attempt to find an approximation with proven performance bounds, we consider the dual problem, as specified in Frame 9. This problem can be described as follows: assign non-negative weights to the nodes such that for each wireless broadcast tree, the sum over all the nodes of the transmission

Minimize:

$$\sum_{u \in V} b(u) \cdot y_u \tag{40}$$

subject to:

$$\forall h \in \Upsilon, \sum_{u \in V} y_u \cdot \max_{(u,v) \in h} p(u,v) \geq 1 \tag{41}$$

$$\forall u \in V, y_u \geq 0 \tag{42}$$

Frame 9: Dual Linear Program for Problem MLB

Input: A constant B, a directed weighted graph $G \equiv (V, E, p)$ and a source node $s \in V$.
Output: The graph G, and node weights $y_u = \frac{1}{B}$.

Frame 10: Reduction R_1

energy multiplied by the node's weight is at least 1, and the sum of weights multiplied by the node is initial energy is minimized.

The number of constraints in the dual program is exponential, however this problem can be solved efficiently by the Ellipsoid Algorithm, provided there is a *separation oracle* [22]. The latter is an algorithm which, given an assignment of values to all variables, finds a constraint that is not satisfied or else verifies that there is no such constraint. A separation oracle for the dual linear program (Frame 9) is required to find a wireless broadcast tree, such that the sum, over all the nodes, of transmission energy multiplied by the node's weight is less than 1, or verify there is no such tree. In order to establish such an oracle, we can find the minimum weight tree and then check whether its weight is less than 1, resulting in the following problem.

Problem Separation for MLB dual program (SMLB): Given are a directed weighted graph $G \equiv (V, E, p)$, nodal weights y and a source node $s \in V$. Find a spanning tree h in G rooted at s, such that $\sum_{u \in V} y_u \cdot \max_{(u,v) \in h} p(u,v)$ is minimized.

5.3 NP-hardness of Problem GMLB

So far, we have considered Problem MLB, for which the transmission power function, p, can accept any non-negative values, and established that it is NP-hard. We turn to show that even in the special case of the Geographic model the corresponding problem, namely Problem GMLB, is NP-hard, hence establishing a stronger intractability result.

Specifically, we establish a reduction from the problem of Geometric Minimum Broadcast Cover (GMBC) [7], which is to find a broadcast tree for a wireless network, such that the total power consumed by all transmitting nodes is minimized. This problem was also shown to be NP-hard [7],[8], and is formally described as follows.

Problem GMBC: *Given are a directed weighted graph $G \equiv (V, E, b, p)$ representing a Geographic wireless network, a source node $s \in V$ and a constant B.*
Question: *Is there a spanning tree T in G rooted at s, such that $\sum_{u \in V} max_{(u,v) \in T} p(u,v) \leq B$?*

Consider the linear programming formulation of Problem MLB in Section 5.1. Clearly, this formulation describes Problem GMLB as well. We shall show that the separation problem of the dual program specified in Frame 9, is a generalization of Problem GMBC. We begin by showing that the separation problem, namely, Problem SMLB, is NP-hard.

Specifically, in Frame 10, we define a polynomial-time reduction, termed R_1, from Problem GMBC to Problem SMLB. For convenience, we employ a variant of Problem GMBC, in which the question is whether there is a tree with total transmission power of less than B (rather than less or equals B). A binary search can be employed to polynomially reduce the original problem to this variant, hence this variant is also NP-hard.

Clearly, the reduction can be calculated in polynomial time. We turn to prove that it is valid.

Lemma 7 *Reduction R_1 is a valid reduction from Problem GMBC to Problem SMLB.*

Proof: If the input is a valid instance of Problem GMBC, then there is a tree T with total transmission power that is less than B, i.e., $\sum_{u \in V} max_{(u,v) \in T} p(u,v) < B$, hence in the reduced separation problem instance for the tree T we have,

$$\sum_{u \in V} y_u \cdot \max_{(u,v) \in T} p(u,v) = \sum_{u \in V} \frac{1}{B} \cdot \max_{(u,v) \in T} p(u,v) < 1. \tag{43}$$

Conversely, if there is a tree T that violates the conditions of the separation problem, i.e., $\sum_{u \in V} y_u \cdot \max_v p(u,v) \in T < 1$, then by definition of the reduction, we have that

$$\sum_{u \in V} y_u \cdot \max_{(u,v) \in T} p(u,v) = \sum_{u \in V} \frac{1}{B} \cdot \max_{(u,v) \in T} p(u,v) \tag{44}$$

hence,

$$\sum_{u \in V} \cdot \max_v p(u,v) \in T < B \tag{45}$$

■

The relation of the computational complexity between a linear program and its corresponding separation problem is specified in the following theorem.

Theorem 6 [22] *For a family of rational polyhedra $P(n,T)$ whose input length is at least polynomial in n and $\log T$, there is a polynomial-time reduction of the linear programming problem over the family to the separation problem over the family, and conversely there is a polynomial-time reduction of the separation problem to the linear programming problem.*

Finally, we can establish that problem GMLB is NP-hard.

Theorem 7 *Problem GMLB is NP-hard.*

Proof: We present a polynomial-time reduction from Problem GMBC to Problem GMLB, as follows. Given an instance of Problem GMBC, reduce it to the separation problem of the dual problem, using reduction R_1. Now, employ the reduction implied by Theorem 6, hence getting a polynomial-time reduction from Problem GMBC to the dual program. Finally, this mean that there is a polynomial-time reduction from Problem GMBC to the primal program, i.e. to problem GMLB. We note that the above reduction preserves the geographic structure of the instance to Problem GMBC, hence it implies a legal instance to Problem GMLB. ■

5.4 Approximation algorithms for broadcast and multicast

In this section we establish approximation schemes for Problems MLB and MLM.

5.4.1 Approximation for Multi-topology Broadcast

We establish an approximation scheme for Problem MLB, using a reduction to the problem of minimizing the total power of a broadcast routing (Problem MBC) [7], which is defined as follows.

Problem MBC: Given are a directed weighted graph $G \equiv (V, E, b, p)$ representing a wireless network, a source node $s \in V$ and a constant B.

Question: Is there a spanning tree T in G rooted at s, such that $\sum_{u \in V} max_{(u,v) \in T} p(u,v) \leq B$?

We shall show that the reduction preserves the approximation factor, hence we can use any approximation algorithm for Problem MBC to find an approximated solution to problem MLB with the same approximation factor.

Consider the linear programming formulation of problem MLB, as specified in Frame 8, its dual program specified in Frame 9 and the separation problem for the dual linear program, namely Problem SMLB (Subsection 5.2).

Now, assume that there is an approximation algorithm for Problem MBC. In Frame 11, we define Algorithm SMLB, which employs it in order to find a solution to Problem SMLB with the same approximation factor. We now prove the correctness of Algorithm SMLB.

Given: Algorithm A, which finds an optimal solution of Problem MBC.
Input: A directed weighted graph $G \equiv (V, E, p, b)$, and a source $s \in V$.

1. Construct an auxiliary graph $G' \equiv (V, E, p')$, such that $p'(u, v) = y(u) \cdot p(u, v)$.
2. Execute algorithm A on G', let h be the output tree.

Output: h.

Frame 11: Algorithm Separation for problem MLB ($SMLB$)

Input: A directed weighted graph $G \equiv (V, E, p, b)$, and a source $s \in V$.

1. Execute a binary search on R in order to find the minimal value of R for which the dual linear program (Frame 9) is feasible. At each step, execute the Ellipsoid Algorithm[22] on the dual program, as follows:
   (a) Add the inequality $\sum_{u \in V} b(u) \cdot y_u \leq R$ to the set of constraints of the dual program.
   (b) Use the following separation oracle:
      i. if $\sum_{u \in V} b(u) \cdot y_u > R$, the program is not feasible;
      ii. execute Algorithm $SMLB$ to find the approximate minimum weight tree in the graph. Let h be the output spanning tree;
      iii. if $weight(h) < 1$, use h as a separation hyperplane;
      iv. else, the program is feasible.
   (c) Let $h_1, h_2, \ldots, h_k$ be the spanning trees found by Algorithm $SMLB$ as separating hyperplanes at step 1(b)ii.
2. Solve the primal linear program, by fixing to 0 all variables other than those corresponding to $h_i, 0 \leq i \leq k$ (where $h_1, h_2, \ldots, h_k$ are the spanning trees identified in the last execution of step 1(c)).

Output: The solution to the primal linear program.

Frame 12: Algorithm MLB Approximation ($MLBA$)

Lemma 8 *There is an α-approximation algorithm[1] for Problem SMLB if there is an α-approximation algorithm for Problem MBC.*

Proof: Assume there is an α-approximation algorithm, denoted as A, for Problem MBC and it is employed at step 2 of Algorithm SMLB. We denote the weight of a tree T, i.e., the sum of weights of its edges, as $w(T)$. By definition, $w(T) = \sum_{u \in V} y_u \cdot \max_{(u,v) \in T} p(u, v)$, whereas the weight of the same tree in G' is $w'(T) = \sum_{u \in V} \max_{(u,v) \in T} p'(u, v)$. By definition of G', $w(T) = w'(T)$, hence, if the output of Algorithm MBC for G' is an α-approximation, then so is the output of Algorithm SMLB. ■

Next, in Frame 12, we specify an approximation scheme for Problem MLB, termed Algorithm *MLB Approximation (MLBA)*. It is based on the Ellipsoid Algorithm [22] and uses Algorithm SMLB as an approximated separation oracle.

Theorem 8 *Algorithm MLBA is an α-approximation for Problem MLB, and has polynomial complexity.*

Proof: First, we note that the output of Algorithm SMLB is a feasible broadcast scheme, since it is a solution to the linear programming formulation of Problem MLB.

We turn to prove that the time complexity of the algorithm is polynomial. At step 1(b), a binary search is performed, and at each iteration the Ellipsoid Algorithm is executed. The latter is of polynomial time complexity, hence step 1(b) incurs polynomial complexity. The total number of separating hyperplanes found at step 1(b) while running the ellipsoid algorithm is polynomial, hence, at step 2, the algorithm solves a linear program with a polynomially bounded number of variables, thus the execution time is polynomial.

Finally, we show that the output is an α-approximation. Consider step 1(b), which solves the dual linear program. The algorithm seeks node weights, such that the target function is minimized and no tree violates the constraint (41). Since the separation oracle is an approximation scheme, it might not find the minimal tree, hence it might erroneously

[1] i.e., an algorithm that provides a solution that is at most within a factor of α away from the optimum.

conclude that the given weights imply a feasible solution. Denote by γ the value of the result of step 1(b). Clearly, the optimal solution of the dual problem is at least $\gamma - \epsilon$ (where ϵ depends on the precision of the algorithm). However, since the approximation ratio of the separation oracle is at most α, multiplying the node weights by α implies a feasible solution of weight $\alpha \cdot \gamma$. Therefore the optimal solution is no more than $\alpha \cdot \gamma$. By duality, the solution of the primal linear program calculated at step 2 of the algorithm is at least γ. Also, the optimal solution of both dual and primal programs is not more than $\alpha \cdot \gamma$, hence the algorithm guarantees an α approximation. ■

From the above theorem, we have the following corollary.

Corollary 1 *There is an α-approximation algorithm for Problem MLB if there is an α-approximation algorithm for Problem MBC.*

An approximation algorithm for Problem MBC was established in [23], with an approximation ratio of $O(log(|V|)$. Hence, we can conclude the following.

Corollary 2 *Algorithm MLBA in an $O(log(|V|)$-approximation to Problem MLB of polynomial complexity.*

5.4.2 Approximation for Multi-topology Multicast

So far, we have considered the multi-topology *broadcast* problem, namely, Problem MLB. We turn to consider the multicast case, i.e. Problem MLM. Clearly, Problem MLM is NP-hard, since Problem MLB is the special case in which $N \equiv V$.

Accordingly, we turn to consider approximation schemes for Problem MLM. We follow the same lines as for the approximation to Problem MLB, namely, consider the linear programming formulation of Problem MLM, and establish a reduction from the separation problem of the dual linear program; however, this time the reduction is to the Directed Steiner Tree problem.

The formulation of Problem MLM as a linear program is the same as the formulation of Program MLB, except that Υ denotes now the collection of all directed *Steiner trees* in G that span N and are rooted at s, rather than the collection of all directed *spanning* trees. As in the formulation of Program MLB, the number of variables is exponential, thus we consider the dual problem, which, again, is identical to that of Problem MLB, except the reinterpretation of Υ.

The separation problem is identical to Problem SMLB, except that here it is required to find a tree that *spans the target set* rather than a *spanning* tree. We denote the corresponding separation problem as SMLM.

Now, assume that there is an α-approximation algorithm for the Directed Steiner Tree problem. In Frame 11, we specify an algorithm that uses it in order to find an approximated solution to the separation problem, namely Problem SMLM, with the same approximation factor. The algorithm reduces Problem SMLM to the Directed Steiner Tree problem, by constructing an auxiliary graph. In this graph, each node is augmented by a set of nodes, one per each different power level at which the original node can transmit. The set of edges of the auxiliary graph consists of edges between each original node and the new nodes representing its transmission power levels. Each of these edges has a weight that equals to that power level. Additionally, we have edges from each "power level node" to nodes that are within transmission range (from the original node) with respect to this power level. The weight of these edges is 0. We illustrate the construction of the auxiliary graph on a simple example, which consists of a single transmitting node, u, which can transmit to three neighbors x, y and z, and the corresponding power levels are $p(u,x) \leq p(u,y) \leq p(u,z)$. The corresponding auxiliary graph is depicted in Figure 9. When u transmits at a power level that allows x to receive the transmission, no other node can receive that transmission, hence u_x consists of a single outgoing edge (u_x, x). However, when u transmits at a power level that allows z to receive the transmission, x and y are also within the transmission range, hence there are outgoing edges from u_z to all three nodes.

In Frame 13, we specify Algorithm SMLM, which reduces the separation problem, namely Problem SMLM, to the Directed Steiner Tree problem. We turn to prove the correctness of Algorithm SMLM.

Lemma 9 *There is an α-approximation for Problem SMLM if there is an α-approximation for the Directed Steiner Tree problem.*

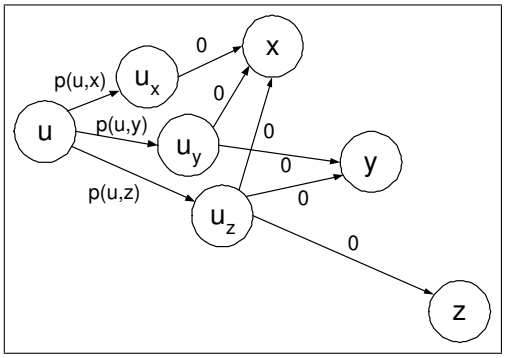


Figure 9: Multicast Reduction Auxiliary Graph

Given: Algorithm A, which finds an optimal solution to the Directed Steiner Tree problem.
Input: A directed weighted graph $G \equiv (V, E, b, p)$, nodal weights w, a target set N and a source $s \in V$.

1. Construct an auxiliary directed graph $G' \equiv (V', E', w')$, such that
   (a) $V' = V \bigcup \{u_v | (u, v) \in E\}$;
   (b) $E' = \{(u_v, x) | (u, x) \in E \wedge p(u, x) \leq p(u, v)\} \bigcup \{(u, u_v) | (u, v) \in E\}$;
   (c) $w' = p(u, v) \cdot w(u)$.
2. Execute Algorithm A on G', where N is the multicast group, let h' be the output tree.
3. $h \leftarrow \{(u, v) | (u_x, v) \in h'\}$

Output: h.

Frame 13: Algorithm MLM Approximation ($SMLM$)

Proof: Assume there is an α-approximation algorithm, denoted as Algorithm A, for the Directed Steiner Tree problem, and it is employed at step 2 of Algorithm SMLB. First, we show that the output h is a tree rooted at s that spans N. h', calculated at step 2, spans N in G', hence, for every node $u \in N$, there is a path p' from s to u in h'. By definition of G', $p' = (s, s_{x0}, v1, v1_{x1} \ldots vn, vn_{xn}, u)$. By step 3, the path $p = (s, v_1, \ldots v_n, u)$ is included in h. Next, we show that the weight of h, i.e., the sum of the weights of its edges, denoted as $w(h)$, is at most α times greater than the minimal weight of any tree in G that spans N. For any tree t in G rooted at s that spans N, consider a corresponding tree t' in G', defined as follows. For each edge $(u, v) \in t$, let x be the most distant node that u transmits to, i.e. $p(u, x) = \max_{(u,i) \in t} p(u, i)$, then t' include both (u, u_x) and (u_x, v). Clearly, t' is a tree that spans N in G'. By definition of G', $w(t') = \sum_{(u,v) \in t'} w'(u, v)$. All non-zero weight edges in t' are of the form (u, u_v), hence $w(t') = \sum_{(u,u_v) \in t'} w(u) \cdot p(u, v)$. By definition of t', $w(t') = \sum_{(u,v) \in t'} w(u) \cdot max_{(u,i) \in t} p(u, i)$. In t', there is at most a single outgoing edge for every node $u \in V$, hence,

$$w(t') = \sum_{u \in V} w(u) \cdot \max_{(u,i) \in t} p(u, i) = w(t). \tag{46}$$

In addition, we have, $w(h) = \sum_{u \in V} y_u \cdot \max_{(u,v) \in h} p(u, v)$. By definition of step 3, for each node u that has a maximal edge weight (u, v), h' contains the edge (u, u_v), hence

$$w(h) \leq \sum_{(u,u_v) \in h'} w(u) \cdot p(u, v) = w(h'). \tag{47}$$

By definition of Algorithm A, we have

$$w(t') \geq \alpha \cdot w(h'). \tag{48}$$

Finally, by (46)-(48), we have

$$w(t) = w(t') \geq \alpha \cdot w(h') \geq \alpha \cdot w(h). \tag{49}$$

Maximize:

$$\sum_{i=1}^{k} t_i \tag{50}$$

subject to:

$$\forall u \in V, \sum_{i=1}^{k} p^i(u) \cdot t_i \leq b(u) \tag{51}$$

$$\forall 0 \leq i \leq k, t_i \geq 0 \tag{52}$$

Frame 14: Linear Program for optimal duration time assignment

Hence Algorithm SMLM in an α-approximation. ■

Similarly to the broadcast case, we observe that it is possible to use Algorithm SMLM as an approximated separation oracle in the Ellipsoid algorithm for solving Problem MLM. We denote *Algorithm MLM Approximation (MLMA)* as the equivalent multicast version of Algorithm MLBA, in which the input is an instance of Problem MLM; at step 1(b), Algorithm *SMLM* is used to establish a separation oracle. By the same proof as for Theorem 8, we have the following.

Theorem 9 *Algorithm MLMA is an α-approximation for Problem MLM, and has polynomial complexity.*

In [24], an approximation algorithm for the minimum Directed Steiner Tree problem was established, which achieves an approximation ratio of $i(i-1)k^{\frac{1}{i}}$ and running time $O(n^i k^{2i})$, for n nodes, k terminals and any fixed $i > 1$. Thus, an $O(k^\epsilon)$ approximation ratio can be achieved in polynomial time for any fixed $\epsilon > 0$. Using this approximation to solve Problem MLM, we have $k = |N|$ and $n = |V| + \sum_{v \in V} |p(v)|$, hence we have the following corollary.

Corollary 3 *Problem MLM has an $O(|N|^\epsilon)$-approximation of polynomial complexity.*

6 Heuristic Solutions to Multi-topology Broadcast

For some scenarios, the approximation schemes established in the previous section may be too complex, hence simpler heuristics are called for. Thus, focusing on the broadcast problem (Problem MLB), in this section we explore a heuristic approach and evaluate it by way of simulations. Our heuristic scheme aims to improve a heuristic scheme established in [12]. The improvement is based on the NP-hardness analysis of Problem MLB (Section 5.1), which has indicated that the main difficulty lies in the selection of trees (rather then on the selection of duration times). Indeed, in the next subsection we show that finding an optimal duration time assignment for a given set of trees, such that the network lifetime is maximized, can be computed in polynomial time via linear programming. However, in some scenarios solving a linear program might still be prohibitively complex. Thus, we turn to consider heuristic solutions that do not rely on linear programming. In subsection 6.3, we establish fast heuristic algorithm for computing the duration time assignment for a given broadcast scheme. This algorithm is based on a procedure that finds duration time assignment for a pair of trees, and is specified in subsection 6.2. Finally, in subsection 6.4, we establish a novel heuristic algorithm for problem MLB, which is also based on the procedure that is established in subsection 6.2.

6.1 Optimal duration time assignment

In this section, we consider a solution to the following sub-problem. Assume the spanning trees are given, e.g., created by any given heuristics, and the problem is to assign a duration time per tree such that the lifetime is maximized. This subproblem can be solved through linear programming, as follows. Given are a directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a set of some k spanning trees rooted at s, $\{h_1, h_2, \ldots, h_k\}$. Let $p^i(u)\colon V \rightarrow \Re$ be the transmission power of node u in tree h_i, namely, $p^i(u) = max_{(u,v)\in h_i} p(u,v)$. Denote by $t_1, t_2, \ldots, t_k$ the respective tree duration times. The formulation of the linear program is specified in Frame 14.

Input: A directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a broadcast scheme $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$.
Solve the linear program (50)-(52) for G and $\{h_1, h_2, \ldots, h_k\}$, let $(t'_1, t'_2, \ldots, t'_k)$ be the resulting duration times.
Output: $\widehat{\alpha} = \{(h_1, t'_1), (h_2, t'_2), \ldots, (h_k, t'_k)\}$.

Frame 15: Algorithm Multi-topology Broadcast Time duration Optimization (MBTO)

Constraint (51) guarantees that the nodal energy limit is not violated, whereas constraint (52) guarantees that the time durations are positive. Based on the solution of the linear program (50)-(52), in Frame 15 we specify Algorithm *Multi-topology Broadcast Time duration Optimization* (*MBTO*), which aims at improving the performance of any given heuristic for solving Problem MLB.

Clearly, the assignment of time durations in $\widehat{\alpha}$ is optimal for the trees of α, hence its lifetime is no less than that of α. Therefore, we can compute the initial α through any given (heuristic) algorithm for Problem MLB, and improve its performance by applying Algorithm MBTO.

6.2 Optimal duration time assignment for a pair of trees

In section 6.1 we showed that finding an optimal time duration assignment for a given set of trees, such that the network lifetime is maximized, can be computed in polynomial time via linear programming. However, in some scenarios solving a linear program might still be prohibitively complex. Thus, we turn to consider heuristic solutions that do not rely on linear programming.

We begin by considering the problem of assigning optimal time durations to any given *pair* of trees. Clearly, this problem can be formulated as a linear program, as presented in Frame 14. However, we show that this specific linear program can be solved by a simple and fast algorithm. This solution requires that the program consists of just two variables, and that all the values in the linear program are non-negative. In our case, these conditions hold, since each variable represents the duration time of a single tree, and the initial energy, as well as the transmission powers, are all non-negative. We begin with the definition of such a linear program.

Definition 7 *A linear program (c,A,b) of the form:*

$$\begin{array}{ll} \textit{Maximize:} & cx \\ \textit{subject to:} & Ax \leq b \\ & x \geq 0 \end{array}$$

is Non-Negative 2-Dimensional *if: (1) $x \in \Re^2$ and (2) all values in A, b, c are non-negative.*

In the following, we can assume that $c_1 = c_2 = 1$ without loss of generality, since otherwise we may define $y_1 = c_1 \cdot x_1$ and $y_2 = c_2 \cdot x_2$, and define A' for y as follows: for each constraint i, such that $a_{i1} \cdot x_1 + a_{i2} \cdot x_2 \leq b_i$, let $a'_{i1} = \frac{a_{i1}}{c_1}$ and $a'_{i2} = \frac{a_{i2}}{c_2}$. Clearly, A' is non-negative, and the solution for $A'y \leq b$, implies a solution for $Ax \leq b$. Additionally, we can assume (w.l.o.g.) that, for each constraint i, $b_i > 0$ and either $a_{i1} > 0$ or $a_{i2} > 0$; since if $a_{i1} = a_{i2} = 0$ then we can ignore this constraint, while if $b_i = 0$ then $x = 0$ is the only solution to the linear program. Clearly, we can also ignore identical constraints, i.e., constraints that define the same hyperplane.

We begin by describing a non-negative 2-dimensional linear program, $P = (c, A, b)$, as a function of a single parameter, which is the ratio between the first component of the vector x, namely, x_1, and the objective of the linear program. We denote this ratio as r, i.e. $r = \frac{x_1}{x_1 + x_2}$. We have that

$$x_2 = x_1 \cdot \frac{1 - r}{r}. \tag{53}$$

x_1 and x_2 are non-negative, hence $x_1 \leq x_1 + x_2$, thus $r \in [0, 1]$. For a given r, the objective of the linear program is then

$$x_1 + x_2 = x_1 + x_1 \cdot \frac{1 - r}{r} = \frac{x_1}{r}, \tag{54}$$

hence we can define a corresponding linear program, denoted as $P_r \equiv (c_r, A_r, b)$,which consists of a single variable, x_1, where $c_r = \frac{1}{r}$ and $a_{r_{i1}} = a_{i1} + \frac{1-r}{r} \cdot a_{i2}$. We denote the solution of P_r for a fixed value of r as $T(r)$. The legal values of r are in $[0, 1]$, hence the solution of P, denoted as T^*, is

$$T^* = \max_{r \in [0,1]} T(r). \tag{55}$$

Next, for each constraint i, corresponding to the i'th row of A and the i'th component of b, we define a linear program, denoted as P_{r_i}, consisting of constraint i solely, i.e. $P_{r_i} = (\frac{1}{r}, a_{r_{i1}}, b_i)$. When we consider just the i'th constraint, we have that

$$x_1 \cdot a_{r_{i1}} = b_i \Rightarrow x_1 = \frac{b_i}{a_{i1} + \frac{1-r}{r} \cdot a_{i2}}. \tag{56}$$

The solution of P_{r_i}, denoted as $T_i(r)$, is then

$$T_i(r) = \frac{x_1}{r} = \frac{b_i}{r \cdot a_{i1} + (1-r) \cdot a_{i2}}. \tag{57}$$

Clearly, $T(r)$ equals the value implied by the most restricting constraint, hence

$$T(r) = \min_{1 \le i \le n} T_i(r). \tag{58}$$

Finally, we conclude that the solution of P is

$$T^* = \max_{r \in [0,1]} T(r) = \max_{r \in [0,1]} \min_{v \in V} \frac{b_i}{r \cdot a_{i1} + (1-r) \cdot a_{i2}}. \tag{59}$$

The functions $T_i(r)$ have some particular properties, specified by the following lemma.

Lemma 10 *For each i:*

1. *If $a_{i1} = a_{i2}$, then $T_i(r)$ is constant.*
2. *If $a_{i1} > a_{i2}$, then $T_i(r)$ is convex, and $T_i'(r) < 0$ for all $r \in [0, 1]$.*
3. *If $a_{i1} < a_{i2}$, then $T_i(r)$ is convex, and $T_i'(r) > 0$ for all $r \in [0, 1]$.*

Proof: In case 1, $T_i(r) = \frac{b_i}{a_{i2}}$, hence is constant. In cases 2, 3 we have

$$T_i(r) = \frac{b_i}{r \cdot (a_{i1} - a_{i2}) + a_{i2}}, \tag{60}$$

thus

$$T_i'(r) = -\frac{b_i \cdot (a_{i1} - a_{i2})}{(r \cdot (a_{i1} - a_{i2}) + a_{i2})^2} \tag{61}$$

$$T_i''(r) = \frac{2 \cdot b_i \cdot (a_{i1} - a_{i2})^2}{(r \cdot (a_{i1} - a_{i2}) + a_{i2})^3} \tag{62}$$

As mentioned, we ignore constraints for which $a_{i1} = a_{i2} = 0$ or $b = 0$. Thus $b_i > 0, a_{i1} > 0$ or $a_{i2} > 0, 0 \le r \le 1$, hence $(r \cdot (a_{i1} - a_{i2}) + a_{i2}) = (r \cdot a_{i1} + (1-r) \cdot a_{i2}) > 0$, thus $T_i''(r) > 0$, and we have that $T_i(r)$ is convex over $[0, 1]$.

By, (61), we have that $T_i'(r) < 0$ for $a_{i1} > a_{i2}$ and $T_i'(r) > 0$ otherwise, for all $r \in [0, 1]$. ■

Next, we show that any two constraints have at most a single intersection.

Lemma 11 *For any two constraints T_i and T_j, $i, j \in \{1, 2, \ldots, n\}$, $i \neq j$, If $(b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2})) = 0$, then there is no intersection point between T_i and T_j. In any other case there is a single intersection point between T_i and T_j, denoted as $R(i, j)$ and given by:*

$$R(i,j) = \frac{b_j \cdot a_{i2} - b_i \cdot a_{j2}}{b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2})}. \tag{63}$$

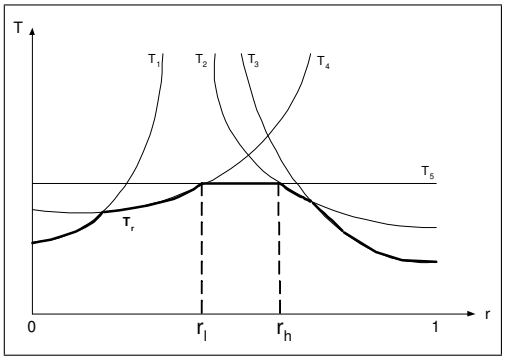


Figure 10: Example of function $T(r)$

Proof:

$$T_i(r) = T_j(r) \Rightarrow \frac{b_i}{r \cdot (a_{i1} - a_{i2}) + a_{i2}} = \frac{b_j}{r \cdot (a_{j1} - a_{j2}) + a_{j2}}. \tag{64}$$

It can be shown by simple algebra that

$$r \cdot (b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2})) = b_j \cdot a_{i2} - b_i \cdot a_{j2}. \tag{65}$$

Now, if $b_j \cdot a_{i2} - b_i \cdot a_{j2} = 0$ and $b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2}) = 0$, then the constraints are identical, i.e., for all r, $T_i(r) = T_j(r)$. Since we ignore identical constraints, this cannot be the case. Hence, if $b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2}) = 0$, then $b_j \cdot a_{i2} - b_i \cdot a_{j2} \neq 0$, hence T_i and $T_j(r)$ does not have any intersection points. In any other case, there is a single intersection point, given by

$$R(i,j) = r = \frac{b_j \cdot a_{i2} - b_i \cdot a_{j2}}{b_i \cdot (a_{j1} - a_{j2}) - b_j \cdot (a_{i1} - a_{i2})} \quad \blacksquare \tag{66}$$

We exemplify a typical form of $T(r)$ in Figure 10. In accordance with Lemma 10, each constraint either is constant, or has a positive derivative for each $r \in [0,1]$, or a negative derivative for each $r \in [0,1]$. Also, in accordance with Lemma 11, any two constraints have at most a single intersection. Note that $T(r)$ is monotonically increasing in $[0, r_l]$, constant in $[r_l, r_h]$, and monotonically decreasing in $[r_h, 1]$. Therefore, $T(r)$ accepts its maximum in $[r_l, r_h]$. We turn to show that there exist such r_l and r_h for any $T(r)$, starting with the their formal definition.

Definition 8 *Given are the functions $T_i(r)$, $1 \leq i \leq n$, and let $T(r) = \min_{1 \leq i \leq n} T_i(r)$. denote by r_l the maximal value of r, for which, for all $r \in (0, r_l)$ and $1 \leq i \leq n$, if $T_i(r) = T(r)$ then $T_i'(r) > 0$.*

Definition 9 *Given are the functions $T_i(r)$, $1 \leq i \leq n$, and let $T(r) = \min_{1 \leq i \leq n} T_i(r)$. denote by r_h the minimal value of r, for which, for all $r \in (r_h, 1)$ and $1 \leq i \leq n$, if $T_i(r) = T(r)$ then $T_i'(r) < 0$.*

We note that r_l and r_h are well defined, since in the extreme case where the definitions do not fit any value of r in $(0,1)$, we have $r_l = 0$ and $r_h = 1$. Next, we specify the properties of $T(r)$ with respect to r_l and r_h.

Lemma 12

1. *For all $r \in (r_l, 1]$ and for all $1 \leq j \leq n$, if $T_j(r) = T(r)$, then $T_j'(r) \leq 0$.*
2. *For all $r \in [0, r_h)$ and for all $1 \leq j \leq n$, if $T_j(r) = T(r)$, then $T_j'(r) \geq 0$.*
3. *for all $r \in [r_l, r_h]$ and for all $1 \leq j \leq n$, if $T_j(r) = T(r)$, then $T'(r) = 0$.*

Proof: We begin with the proof of 1. By way of contradiction, assume that it does not hold, and let $\tilde{r} \in (r_l, 1]$ be the minimal value for which $T_j(\tilde{r}) = T(\tilde{r})$ and $T_j'(\tilde{r}) > 0$ for some $1 \leq j \leq n$. By the minimality of $\tilde{r}$, $T(r)$ is non-increasing in $[r_l, \tilde{r}]$. But by Lemma 10 $T_j'(r) > 0$ for all $r \in [0,1]$, hence $T(r_l) \leq T_j(r_l) < T_j(\tilde{r}) = T(\tilde{r})$, which is a contradiction. The proof of 2 is symmetrical.

Input: Matrix A, and vector b.

$r \leftarrow 0$
$S \leftarrow \{k | 1 \leq k \leq n, T_k(0) = min_h T_h(0)\}$
select i such that $T_i(1) = min_{j \in S} T_j(1)$
while $r < 1$ and $T_i'(r) > 0$
    $r \leftarrow min\{1, min_{h:r<R(h,i)} R(h,i)\}$
    select j such that $r = R(j,i)$ and $T_j(1) = min_{h:r=R(h,i)} T_h(1)$
    $i \leftarrow j$

Output: $x_1 \leftarrow r \cdot T_i(r)$, $x_2 \leftarrow (1-r) \cdot T_i(r)$.

Frame 16: Algorithm Optimal Ratio Scan

For the proof of 3, consider any $r \in [r_l, r_h]$ and $1 \leq j \leq n$, such that $T_j(r) = T(r)$, by 1, we have that $T_j'(r) \leq 0$, by 2, we have that $T_j'(r) \geq 0$, hence $T_j'(r) = 0$. ■

We specify additional properties of $T(r)$ in the following lemma.

Lemma 13

1. *$T(r)$ is monotonically increasing in $[0, r_l]$.*
2. *$T(r)$ is constant in $[r_l, r_h]$.*
3. *$T(r)$ is monotonically decreasing in $[r_h, 1]$.*

Proof: By definition of r_l, $T(r)$ is monotonically increasing in $[0, r_l]$. By Lemma 12, for all $r \in [r_l, r_h]$, if $T_i(r) = T(r)$ then $T_i'(r) = 0$, hence by Lemma 10, T_i is constant. Therefore, $T(r)$ is constant in $[r_l, r_h]$. By definition of r_h, $T(r)$ is monotonically decreasing in $[r_h, 1]$. ■

From the above lemma we have the following corollary.

Corollary 4 *$T(r) = T^*$ if and only if $r \in [r_l, r_h]$*

Next, we specify additional conditions on r for which $T(r)$ accepts its maximal value.

Corollary 5 *For all $1 \leq i \leq n$ and $\hat{r} \in [0,1]$, if $T_i(\hat{r}) = T(\hat{r})$ and $T_i'(\hat{r}) > 0$ then $T(r) = T^*$ for some $r \in [\hat{r}, 1]$. Symmetrically, if $T_i(\hat{r}) = T(\hat{r})$ and $T_i'(\hat{r}) < 0$ then $T(r) = T^*$ for some $r \in [0, \hat{r}]$.*

Proof: If $T_i(\hat{r}) = T(\hat{r})$ and $T_i'(\hat{r}) > 0$, then $r_h \geq \hat{r}$. Since $T(r_h) = T^*$, the corollary follows. The proof of the second part is symmetrical. ■

Finally, in Frame 16 we specify Algorithm *Optimal Ratio Scan (ORS)*, that solves a Non-negative 2-dimensional linear program. Its correctness is established by the following theorem.

Theorem 10 *Given a non-negative 2-dimensional linear program $P = (c, A, b)$, where A is an $n \times 2$ matrix. Algorithm ORS solves P in $O(n^2)$ time.*

Proof: The algorithm scans $T(r)$ from $r = 0$ to $r = 1$, until it reaches r_l. By Corollary 4, this is indeed the maximal value of $T(r)$. We now show by induction that the scanning of $T(r)$ is correct. Specifically, we denote by r_k the value of r and by i_k the index of the constraint T_i at the beginning of the k'th iteration, and show that at the beginning the k'th iteration, the following conditions hold:

1. $T_{i_k}(r) = T(r)$,
2. $r_k \leq r_l$,
3. $T_{i_k}(1) = min_{h:T_h(r_k)=T(r_k)} T_h(1)$

   Additionally, before each iteration except for the last, the following condition holds:

Input: Matrix A, vector b, and an error size $\epsilon \cdot max_i, r|T_i'(r)|$.

1 $d \leftarrow 0$
1 $u \leftarrow 1$
2 **while** $u - d > \epsilon$
3 $\quad r \leftarrow \frac{u-d}{2}$
4 $\quad$ select i such that $T_i(r)$ is minimal
5 $\quad$ **if** $a_{i1} = a_{i2}$ **then quit**
6 $\quad$ **if** $a_{i1} < a_{i2}$ **then** $u \leftarrow r$ **else** $d \leftarrow r$

Output: $x_1 \leftarrow r \cdot T_i(r)$, $x_2 \leftarrow (1-r) \cdot T_i(r)$

Frame 17: Algorithm Optimal Ratio Binary Search (ORBS)

4. $T'_{i_k}(r_k) > 0$.

At the initialization, the inductive hypothesis holds, by definition. Now assume that conditions (1)-(4) hold at the beginning of that iteration. At step 6 of the k'th iteration, the algorithm finds the next intersection point of T_{i_k}, $r_{k+1} \in (r_k, 1]$, with any other constraint, i.e., $r_{k+1} = min\{1, min_{h:r_k<R(h,k)} R(h,k)\}$. Assume by way of contradiction that condition (1) does not hold at the beginning of the $k+1$'st iteration, i.e. $T_{i_k}(r_{k+1}) = T_{i_{k+1}}(r_{k+1}) > T(r_{k+1})$. Hence, there is some $m \neq i_k, 1 \leq m \leq n$, for which $T_m(r_{k+1}) = T(r_{k+1}) < T_{i_k}(r_{k+1})$. By the inductive hypothesis, we have that $T_{i_k}(r_k) = T(r_k)$, hence, the minimality of r_{k+1} implies that $T_m(r_k) = T_{i_k}(r_k)$. By Lemma 11, there is at most a single intersection between T_{i_k} and T_m, namely r_k, hence $T_m(1) < T_{i_k}(1)$, which is a contradiction to condition (3). Therefore, $T_{i_k}(r_{k+1}) = T_{i_{k+1}}(r_{k+1}) = T(r_{k+1})$, and condition (1) holds at the beginning of the $k+1$'st iteration.

Since $T_{i_k}(r_{k+1}) = T(r_{k+1})$ and $T'_{i_k}(r_{k+1}) > 0$, then, by Lemma 12, $r_{k+1} \leq r_l$, and condition (2) holds. Condition (3) holds by definition of step 6, and condition (4) holds by the "while" condition on line 4. This complete the proof of the induction. Clearly, by the "halt" condition on line 4, the algorithm will not terminate before $r \geq r_l$, hence, by condition (2), when the algorithm halts, we have that $r = r_l$. By Lemma 4, this is indeed the maximal value of $T(r_k)$.

We turn to consider the computation time of the algorithm. If constraint T_{i_k} was selected in the k'th iteration, i.e. $T_{i_k}(r_k) = T(r_k)$, and constraint j was selected in the $k+1$'st iteration, i.e. $T_{i_{k+1}}(r_{k+1}) = T(r_{k+1})$, then, by Lemma 11, for $r \in (r_{k+1}, 1]$, we have that $T(r) \leq T_{i_{k+1}}(r) < T_{i_k}(r)$, hence constraint T_{i_k} will not be selected in the following iterations. Therefore, each constraint can be selected at most once, hence, the number of iterations is bounded by n. In each iteration, the next constraint is found by scanning all constraints, which again is bounded by n. Therefore, the computation time of the algorithm is $O(n^2)$. ■

Next, we establish an alternative algorithm, termed *Optimal Ratio Binary Search (ORBS)*, which solves the above problem, but through a binary search. It is faster then Algorithm ORS, however it may incur an error whose size depends on the number of iterations. Algorithm ORBS is specified in Frame 17 and its correctness is established in the following theorem.

Theorem 11 *Given is a non-negative 2-dimensional linear program $P = (c, A, b)$, where A is an $n \times 2$ matrix. Algorithm ORS finds in $O(log(\frac{1}{\epsilon}))$ time a solution, whose value is at least $1 - \epsilon \cdot max_{1 \leq i \leq n} \{\frac{b_i \cdot |a_{i1} - a_{i2}|}{a_{i1}^2}, \frac{b_i \cdot |a_{i1} - a_{i2}|}{a_{i2}^2}\}$ times the optimum (i.e., maximum).*

Proof: By induction on the number of iterations. We show that, before each iteration, T has a maximum in $[d, u]$, i.e. for some $r \in [d, u]$, $T(r) = T^*$. By definition, the inductive hypothesis holds prior to the first iteration. Assume that, at the beginning of the k'th iteration, T has a maximum in $[d_k, u_k]$. In the k'th iteration, a constraint i for which $T_i(r) = T(r)$ and $r = \frac{u_k - d_k}{2}$ is considered. If $a_{i1} > a_{i2}$, then, by Lemma 10, $T_i'(r) < 0$, hence, by Corollary 5, T has a maximum in $[0, r]$. By step 6, $u_{k+1} \leftarrow r$, hence T has a maximum in $[d_{k+1}, u_{k+1}] = [d_k, r]$, and the inductive hypothesis is true at the beginning of the $k+1$'st iteration. If $a_{i1} < a_{i2}$, the inductive hypothesis holds by a symmetrical argument.

When the algorithm stops, either $a_{i1} = a_{i2}$, or else $u - d \leq \epsilon$. In the first case, by Lemma 10, $T_i'(r) = 0$, hence $r_l \leq r \leq r_h$ and by Corollary 4, $T(r) = T^*$. In the second case, as shown above, there exists some $r \in [d, u]$ for which $T(r) = T^*$. In this case, let the value of the objective function returned by the algorithm be $x = x_1 + x_2$. By definition

Input: A directed weighted graph $G \equiv (V, E, b, p)$ and two normalized broadcast scheme instances, $\alpha = \{(h'_1, t'_1), (h'_2, t'_2), \ldots, (h'_k, t'_k)\}$ and $\beta = \{(h''_1, t''_1), (h''_2, t''_2), \ldots, (h''_l, t''_l)\}$.

1 Construct the linear program $Ax \leq b$ as follows:
For each node i, which transmits in any of the trees, define a constraint
a. $a_{i1} = \sum_{j=1}^{k} t'_j \cdot p_i(h_j)$.
b. $a_{i2} = \sum_{j=1}^{k} t''_j \cdot p_i(h_j)$.
c. $b_i = b(i)$.
where $p_i(h_j)$ is the transmission power level of node i in tree h.
2 Calculate the optimal ratio r, by executing Algorithm ORS.

Output: $SRU(\alpha, \beta, r)$.

Frame 18: Procedure Optimal Union Ratio (OUR)

of the algorithm, either $T(d) = x$ or $T(u) = x$, but in both cases $T^* - x \leq \epsilon \cdot max_{1 \leq i \leq n, r \in [0,1]} |T'_i(r)|$. By (61), we have that the error of the algorithm is $T^* - x \leq \{\frac{b_i \cdot |a_{i1} - a_{i2}|}{a_{i1}^2}, \frac{b_i \cdot |a_{i1} - a_{i2}|}{a_{i2}^2}\}$, as required.

The algorithm performs a binary search in the region $[0, 1]$, until the search region is of size ϵ, hence, the number of iterations is bounded by $log_2(\frac{1}{\epsilon})$. At each iteration, the algorithm searches for a constraint with minimal value, thus the time to compute each iteration is $O(n)$. The overall time of the algorithm is then $O(n \cdot log(\frac{1}{\epsilon}))$. ■

6.3 Heuristic duration time assignment

Next, we employ the algorithms that find optimal time duration assignment for a pair of trees, in order to establish a heuristic algorithm for finding a duration time assignment for a given broadcast scheme. For define a *normalized broadcast scheme*. For each tree in the broadcast scheme, we maintain a *normalized* duration time, which is the ratio between the duration time of that tree and the sum of duration times of all the trees. More formally, a normalized broadcast scheme is defined as follows.

Definition 10 *A broadcast scheme, $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$, is said to be* normalized *if $\sum_{i=1}^{k} t_i = 1$.*

The lifetime of a normalized broadcast scheme is then defined as follows.

Definition 11 *Given are a directed weighted graph $G \equiv (V, E, b, p)$ and a* normalized *broadcast scheme $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots,$ The* lifetime *of the (*normalized*) α, denoted as $lifetime(\alpha)$, is the maximal time that α can be used until any of its nodes consumes all its energy, i.e. $lifetime(\alpha) \equiv min_{u \in V} \frac{b(u)}{\sum_i t_i \cdot p_u(h_i)}$, (where $p_u(h_i)$ is the transmission power level of node u in tree h_i).*

Next, we define the union of two normalized broadcast schemes.

Definition 12 *Given are two normalized broadcast schemes $\alpha = \{(h'_1, t'_1), (h'_2, t'_2), \ldots, (h'_k, t'_k)\}$ and $\beta = \{(h''_1, t''_1), (h''_2, t''_2), \ldots, (h''_l, t''_l)\}$ and some $r \in [0, 1]$. The* single ratio union *of α and β, denoted as $SRU(\alpha, \beta, r)$, is the normalized broadcast scheme $\gamma = \{(h'_1, r \cdot t'_1), (h'_2, r \cdot t'_2), \ldots, (h'_k, r \cdot t'_k)$, $(h''_1, (1-r) \cdot t''_1), (h''_2, (1-r) \cdot t''_2), \ldots, (h''_l, (1-r) \cdot t''_l)\}$.*

In Frame 18 we specify *Procedure Optimal Union Ratio (OUR)*, which employs Algorithm ORS to find the union of two normalized broadcast schemes, such that its lifetime is maximized. The time complexity of Procedure OUR is identical to that of Algorithm ORS. Alternatively, Procedure OUR can employ Algorithm ORBS, hence establishing a faster solution, at the expense of being sub-optimal.

Finally, in Frame 19 we specify a heuristic algorithm, termed Heuristic Duration time Assignment (HDA), which recalculates the tree duration times of any given broadcast scheme (e.g., provided by Algorithm HGB [12] or any other heuristic). Specifically, for each tree h in the given broadcast scheme, the algorithm calculates the normalized broadcast scheme, S', which includes all other trees. It then creates a new normalized broadcast scheme by uniting S' and h through Procedure OUR.

Input: A directed weighted graph $G \equiv (V, E, b, p)$ and a broadcast $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$.

1 $T \leftarrow \sum_{1 \leq i \leq k} t_i$
2 $S \leftarrow \{(h_i, \frac{t_i}{T}) | 1 \leq i \leq k\}$
3 $j \leftarrow 1$
4 **while** $j \leq k$
5 $\quad T' \leftarrow \sum_{1 \leq i \leq k, i \neq j} t_i$
6 $\quad S' \leftarrow \{(h_i, \frac{t_i}{T'}) | 1 \leq i \leq k, i \neq j\}$
7 $\quad S \leftarrow OUR(\text{G,S'}, (h_j, 1))$
8 $\quad j \leftarrow j + 1$

Output: $\{(h_i, t_i \cdot lifetime(S)) | (h_i, t_i) \in S\}$.

Frame 19: Heuristic Duration time Assignment (HDA)

The algorithm performs k iterations, each of which consists of an execution of Procedure OUR, hence, the total computation time is $O(k \cdot C_{OUR}))$, where C_{OUR} is the computation time of Procedure OUR.

6.4 Specification of heuristic scheme

In this section we establish a novel heuristic scheme which aims to improve a heuristic scheme established in [12]. The improvement is based on the NP-hardness analysis of Problem MLB (Section 5.1), which has indicated that the main difficulty lies in the selection of trees (rather then on the selection of duration times).

Refer to the heuristic algorithm proposed in [12] as Algorithm Heuristic Greedy Broadcast (HGB). Algorithm HGB constructs the broadcast scheme iteratively. At each iteration, a tree h with maximum lifetime t is selected and added to the broadcast scheme. The duration time corresponding to h is set to be the minimum of t and τ, where τ is some predefined bound. Then, h is used for that duration time, i.e., for each node, the algorithm reduces the amount of energy by the amount consumed at that node in h. The algorithm then continues to the next iteration with a network consisting of the residual energies. The algorithm stops when no further broadcast tree can be found. As the number of iterations may be exponential, in practice it should be limited by some predetermined limit.

We turn to describe our heuristic algorithm, termed *Heuristic Greedy Broadcast Optimal Ratio (HGBOR)*. Similarly to Algorithm HGB, the algorithm iteratively adds trees to the broadcast scheme. However, the way by which the duration time is selected at each iteration is different. Specifically, we maintain a *normalized* broadcast scheme (as defined in section 6.3). We then find the optimal ratio by which the new tree should be added to the normalized broadcast scheme, i.e. through Procedure OUR (section 6.3).

Before specifying Algorithm HGBOR, we have to define the way by which a new tree is chosen. At the first iteration, the tree with maximum lifetime is selected. Specifically, the tree is calculated by Algorithm GSM, which finds the maximum lifetime tree, as described in Section 4. Intuitively, in the next iterations, when adding a new tree to the broadcast scheme, we would like to select a new tree that uses resources of the network that are unused by the current broadcast scheme, e.g., use different transmitting nodes. We follow this intuition, through the following rule: use the current broadcast scheme (α) for as much time as possible (starting with the initial energy values), such that, after using α for that amount of time, the residual nodal energy still enables to identify a new broadcast tree (with non-zero duration time); then select that new tree. In Frame 20, we specify Procedure Find New Tree (FNT), which uses a binary search to implement this rule.

In order to bound the running time of Procedure FNT, its input includes a bound (L) on the number of iterations. In each iteration, it executes Algorithm GSM, hence the total computation time of Procedure FNT is $O(L \cdot |V| log |V|)$. As mentioned, the number of iterations must be bounded, hence the algorithm accepts a predefined bound on the number of trees, k, in the broadcast scheme, and executes k iterations.

Finally, in Frame 21, we specify *Algorithm Heuristic Greedy Broadcast Optimal Ratio (HGBOR)*, which uses Procedure FNT in order to find a solution to Problem MLB. We proceed with an informal description of the algorithm. In the first iteration, the algorithm calculates a maximum lifetime broadcast tree, h_1. Denote by α_i the solution at the end of the i'th iteration. Then, at the first iteration $\alpha_1 = \{(h_1, 1)\}$. At the second iteration, the algorithm selects

Input: A directed weighted graph $G \equiv (V, E, b, p)$, a source nodes s, a broadcast scheme $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$ and a bound L on the number of iterations.

$u \leftarrow lifetime(\alpha)$
$d \leftarrow 0$
while $L > 0$
    $r \leftarrow \frac{u-d}{2}$
    foreach u in V
        $b'(u) \leftarrow \sum_{i=1}^{k} r \cdot t_i \cdot \max_{(u,v)\in h} p(u, v)$
    $h' \leftarrow GSM((V, E, b', p), s)$
    if $h' = null$ **then**
        $u \leftarrow r$
    else
        $h \leftarrow h'$
        $d \leftarrow r$
    $L \leftarrow L - 1$

Output: h.

Frame 20: Procedure Find New Tree (FNT)

Input: A directed weighted graph $G \equiv (V, E, b, p)$, a source nodes s, a bound k on the number of trees, and an accuracy factor ϵ.

$h \leftarrow GSM(G, s)$
$\alpha = \{(h, 1)\}$
$\delta \leftarrow \epsilon$
while $\delta \geq \epsilon \wedge k > 0$
    $h \leftarrow FNT(G, s, \alpha)$
    $\alpha' \leftarrow OUR(G, s, \alpha, h)$
    $\delta \leftarrow lifetime(\alpha') - lifetime(\alpha)$
    $\alpha \leftarrow \alpha'$
    $k \leftarrow k - 1$

Output: α.

Frame 21: Algorithm Heuristic Greedy Broadcast Optimal Ratio (HGBOR)

a new tree. Specifically, it finds the time t such that, if α is used for t time units, then in the residual network there is a broadcast tree h_2, but if α_1 is used for more than $t + \epsilon$ time units, then in the residual network it is not possible to broadcast to all nodes. Having found h_2, the algorithm invokes Procedure OUR in order to add h_2 to α_1. Procedure OUR, in turn, finds the optimal ratio, r_2, and the new solution is then $\alpha_2 = \{(h_1, 1 - r_2), (h_2, r_2)\}$. Similarly, at the next iteration, h_3 and r_3 are calculated, and $\alpha_3 = \{(h_1, (1 - r_2)(1 - r_3)), (h_2, r_2(1 - r_3)), (h_3, r_3)\}$. Finally, the algorithm stops when either the lifetime improvement of the last iteration is less than some predefined ϵ or else after a predefined upper bound k on the number of iterations.

We turn to calculate the time complexity of the algorithm. The output α consists of at most k trees, hence the algorithm performs at most k iterations. At each iteration, both Procedure FNT and Algorithm OUR are executed, hence the total computation time is $O(k(L|V|log|V| + |V|log(\frac{1}{\epsilon})))$.

6.5 Simulation Results

We conducted simulation experiments to evaluate the new proposed heuristics. To recall, we have two classes of heuristics. The first class consists of heuristic algorithms that find broadcast schemes from scratch. This class includes Algorithm HGB [12], specified in Section 6.4, and Algorithm HGBOR, specified in Frame 21. The second class consists of heuristic algorithms that aim at improving the lifetime of broadcast schemes that are found by any other heuristic scheme; this is done by assigning different duration times. This class includes Algorithm MBTO, specified in Frame 15, which finds the optimal duration time assignment via linear programming; and Algorithm HDA, specified in Frame

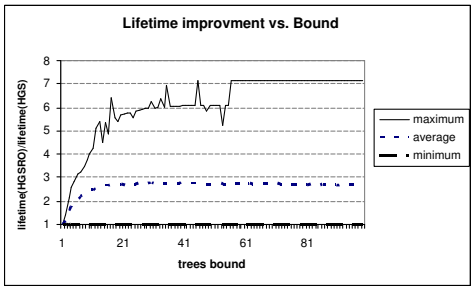


Figure 11: Lifetime improvement of HGBOR over HGB vs. Trees bound

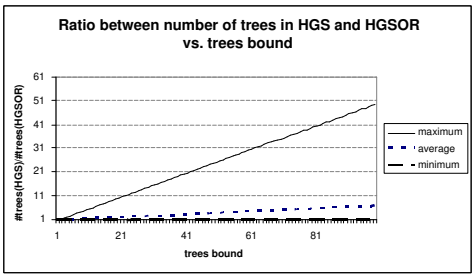


Figure 12: Number of trees in solution of HGBOR relative to HGB vs. Trees bound

19, which is faster but does not necessarily find optimal solution duration times.

The simulations were performed on 100 uniformly (randomly) positioned nodes on a 100×100 square. The initial energy was uniformly (randomly) selected in the range of $[10^4, 10^5]$. The path loss factor was assumed to be 2, hence, if the distance between a node u and a node v is d, then the power required by a node u to transmit to a node v is d^2.

We start by comparing Algorithm HGB with Algorithm HGBOR. An important factor in the performance of Algorithm HGB and HGBOR is the predefined bound k on the number of trees. Hence, we compared the results of the two heuristics for various values of k. Additionally, as mentioned, the definition of Algorithm HGB requires an additional predefined parameter, which is the bound τ on the duration time of each of the trees. The value of τ should be proportional to the number of trees, since $k \cdot \tau$ is an upper bound on the lifetime of the broadcast scheme. Hence, in order to determine τ, we calculated the maximum lifetime broadcast tree using Algorithm GSM. Denote by δ the lifetime of that tree divided by the bound on the number of trees. Clearly, it is not useful to execute Algorithm HGB with $\tau < \delta$. Hence, we executed Algorithm HGB with τ equal to each of the following values: δ, $\delta \cdot 2$, $\delta \cdot 4$, and selected the solution with the longest lifetime.

We compared between the two heuristic schemes, HGB and HGBOR, employing various bounds on the number of trees. Specifically, we executed the two algorithms on 100 network instances, and for each instance we executed each algorithm with tree bounds varying from 1 to 100. The simulation results are depicted in Figure 11. For each execution we calculated the improvement ratio of Algorithm HGBOR over Algorithm HGB, i.e. the lifetime of the output of Algorithm HGBOR divided by the lifetime of the output of Algorithm HGB. The x-axis represents the bound on the number of trees and the y-axis represents the improvement ratio. For each bound, we calculated the *average* improvement ratio over all network instances as well as the *minimal* and *maximal* improvement ratios. When the number of trees is bounded by 1, the output of both algorithms is the maximum lifetime broadcast tree, hence, as expected, the improvement ratio is always 1. However, as more trees are allowed, the lifetime of the solution returned by Algorithm HGBOR is significantly larger. When that bound is 15 or more, the average improvement over all instances is 2.7, the maximal improvement ratio is 7.1, and the minimal improvement ratio is 1; in *none* of the executions was the lifetime of the solution calculated by Algorithm HGBOR less than that of Algorithm HGB.

We also compared the number of trees in the broadcast scheme returned by each algorithm; the results are depicted in Figure 12. Specifically, for each execution, we calculated the ratio between the two numbers, i.e. the number of trees in the output of Algorithm HGBOR, divided by the number of trees in the output of Algorithm HGB. The x-axis

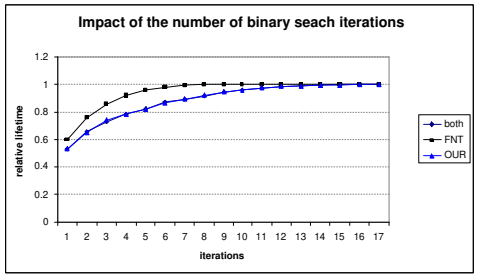


Figure 13: Lifetime of HGBOR vs. number of binary search iterations

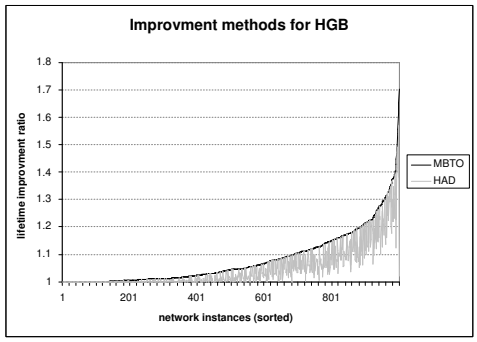


Figure 14: Duration time improvement for solutions of HGB

represents the bound on the number of trees and the y-axis represents the ratio between the number of trees. For each bound, we calculated the average ratio over all instances, as well as the minimal and maximal ratios. Clearly, when the number of trees is bounded by 1, the ratio is 1. However, as more trees are allowed, Algorithm HGB uses significantly more trees than Algorithm HGBOR, e.g., when the bound is 100, Algorithm HGB uses, in average, 6.7 times more trees than Algorithm HGBOR.

Having found that Algorithm HGBOR achieves better results than Algorithm HGB, we turned to evaluate the influence of various parameters on the solution. Specifically, Algorithm HGBOR uses two different binary searches. First, in step 5, Procedure FNT is invoked in order to find a new tree in each iteration. Second, in step 6, Algorithm OUR is invoked in order to find the optimal ratio for adding that tree to the current solution. We evaluated the dependence of the quality of the solution on the number of iterations in the binary searches. We generated 100 random network instances. For each instance we executed Algorithm HGBOR three times: in the first, we bounded just the number of iterations of Procedure FNT; in the second, we bounded the number of iterations of Procedure OUR; and in the third, we bounded both by the same value. In order to normalize the lifetime, we divided it by the lifetime of the solution returned when executing Algorithm HGBOR without any bounds on the number of iterations. We then calculated the average of the normalized values over all network instances. The results are depicted in Figure 13. It can be observed that increasing the number iterations in the search for a new tree improves the lifetime of the solution, hence supporting our intuition that suggested that we should find the most "orthogonal" tree. On the other hand, as expected, the lifetime of the solution is much more sensitive to the accuracy of the optimal ratio for adding the new tree. Finally, it can be observed that, when the bound is larger than 12, the optimal lifetime is achieved, hence practically, for these network instances, the number of iterations can be bounded by 12.

Next, we turn to evaluate the improvement methods, namely Algorithm MBTO and Algorithm HDA. We generated 1000 random network instances. For each instance, we executed Algorithm HGB, and then applied the improvement methods. We calculated the improvement ratio for each of the methods, and sorted the instances, by the improvement ratio of Algorithm MBTO. The results are depicted in Figure 14, where the x-axis represents the network instances and the y-axis represents the lifetime improvement ratio. In about 50% of the networks, an improvement of more than 5% can be achieved by reassigning duration times to the same trees. The maximal improvement is 70%, and the average optimal improvement is 8%. The average improvement of Algorithm HDA is 5%.

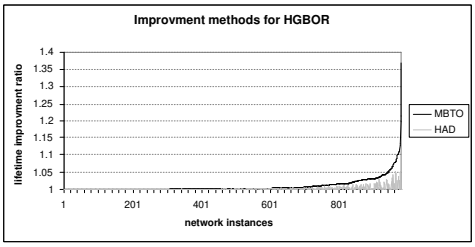


Figure 15: Duration time improvement for solutions of HGBOR

We then evaluated the improvement that can be achieved for the solution of Algorithm HGBOR. Again, we generated 100 random network instances. For each instance we executed Algorithm HGBOR, and then applied the improvement methods. Results are depicted in Figure 15, using the same representation as for Algorithm HGB. Only in 6% of the networks an improvement of more than 5% can be archived by reassigning duration times. The maximal improvement is 36%, and the average optimal improvement is 1%. The average improvement of Algorithm HDA is 0.1%. Clearly, the solution of Algorithm HGBOR is much closer to the optimal duration time assignment, hence in most cases the employing the improvement methods is useless.

From the simulations, it can be concluded that Algorithm HGBOR achieves better solutions than Algorithm HGB, in terms of both lifetime and size (i.e., number of trees). The computation time of Algorithm HGBOR is practically small, since the number of iterations for the binary searches of Algorithm HGBOR can be bounded by a small constant (i.e, about 12). Finally, the heuristic solutions can be improved by re-assigning new duration times to the same trees. However, while the lifetime improvement may be quite significant to the output of Algorithm HGB, it is much less significant to the output of Algorithm HGBOR. Therefore, for the scenario in which fast algorithms are required, Algorithm HGBOR can be used. However, for scenarios in which even the time complexity of Algorithm HGBOR is too great, then Algorithm HGB with Method HDA, as an improvement method, can be used as an alternative.

From the simulations, it can be concluded that Algorithm HGBOR achieves significantly better solutions than Algorithm HGB, in terms of both lifetime and size (i.e., number of trees); specifically, the lifetime improvement factor is 2.7 on the average, while the size improvement factor is 6.7. Also, we can conclude that the computation time of Algorithm HGBOR is practically small, since the number of iterations for the binary searches of Algorithm HGBOR can be bounded by a small constant (about 12). Finally, these and any other heuristic solutions can be improved by employing Algorithm HDA on their output, i.e., re-assigning new (optimal) duration times to the same trees. However, we have seen that, while the lifetime improvement of such re-assignment may be quite significant when applied to the output of Algorithm HGB, it is much less significant when applied to the output of Algorithm HGBOR. Therefore, in scenarios that cannot admit our approximation approach (presented in Section 5.4) due to limitations on the execution time, Algorithm HGBOR offers a good heuristic alternative, with relatively fast execution time and significantly improved lifetime performance over previous heuristics. Yet, when even faster solutions are called for, Algorithm HGB with the optimization of duration times via Algorithm HDA can be used as an alternative.

7 Single receiver model

In some scenarios we cannot assume that a transmission can be received by multiple nodes, e.g., because the antennae are directional. Clearly, the case of unicast is not affected by this change in the model, since in a unicast routing scheme there is one receiver per hop inherently. Hence, we focus on broadcast and multicast routing under this model, and for each we consider both the single-topology and multi-topology problems. We begin by formally defining the problems.

7.1 Problem Definitions

The single-topology multicast problem is to find a single tree that spans the target group, such that its lifetime is maximized. While in the multiple receiver model, the time during which a node u could transmit on a given tree h was defined by the outgoing edge with maximal weight, i.e. $\frac{b(u)}{\max_{(u,v)\in h} p(u,v)}$, in the single receiver model u consumes energy for each transmission, hence the time during which it can transmit is determined by the sum of outgoing edges, i.e. $\frac{b(u)}{\sum_{(u,v)\in h} p(u,v)}$.

We start with the definition of the multi-topology multicast problem for the single receiver model. We need to redefine the concept of "legal multicast schemes", since in the multiple receiver model, the power consumed by a node u in a given tree h is determined by the maximal weight of an outgoing edge from u in h, while in the single receiver model, u consumes energy for each transmission, hence the maximum operator is replaced with a summary operator. Therefore, a legal multicast scheme for the single receiver model is defined as follows.

Definition 13 *Given are a directed weighted graph $G \equiv (V, E, b, p)$, a source node $s \in V$ and a target set $N \subseteq V$. A* multicast scheme*, with respect to the single receiver model, is a set of tree-duration time pairs $\{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$, $0 \leq i \leq k$, such that each h_i is a tree rooted at s that spans N, and the time durations $t_i \in \Re$ are such that:*

$$\forall u \in V, \sum_{i=1}^{k} \sum_{v \in V} t_i \cdot \delta(h_i, (u, v)) \cdot p(u, v) \leq b(u) \tag{67}$$

where

$$\delta(h_i, e) = \begin{cases} 1 & e \in h_i \\ 0 & e \notin h_i \end{cases} \tag{68}$$

i.e., for each node, the total energy required by all the trees on which this node participates as a transmitter is bounded by the initial energy of that node.

Definition 14 *The* lifetime of a broadcast scheme *is the sum of tree duration times, i.e. $lifetime(\alpha) = \sum_{i=1}^{k} t_i$.*

Accordingly, the *Maximum Lifetime Multicast Single Receiver* (*MLMSR*) problem is to find a multicast schemes that maximizes the lifetime. The broadcast variant of the problem, denoted as *Maximum Lifetime Broadcast Single Receiver* (*MLBSR*) is a special case of the multicast problem for which $N \equiv V$.

Additionally, we consider the special case in which the routing scheme is restricted to consist a single topology, i.e., we seek a single tree of maximum lifetime. We term the corresponding multicast problem as *Maximum lifetime Single-topology Multicast Single Receiver (MSMSR)*. We denote the corresponding broadcast problem as MSBSR.

7.2 Broadcast

In this subsection we consider the broadcast problems for the new model. In the multiple-receiver model, we have shown that the single-topology broadcast problem can be solved efficiently while finding the optimal solution to the multi-topology problem is NP-hard. When turning to the single receiver model, there is an interesting twist of difficulty, namely, the single-topology problem is now NP-hard, while there is an optimal efficient solution to the multi-topology problem. In the following we establish these claims, starting with the single-topology case.

7.2.1 NP-hardness of the Single-Topology problem

We show that the single-topology broadcast problem, i.e. Problem MSBSR, is NP-hard, by showing that it is a generalization of the *Minimum Degree Spanning Tree* (*MDST*) problem, which is known to be NP-hard [**?**]. Specifically we will establish a reduction from the directed version of Problem MDST, denoted as DMDST, which is a generalization of Problem MDST to directed graphs, hence, it is also NP-hard.

Problem *DMDST* is defined as follows: Given a directed graph $G = (V, E)$ and a source node $s \in V$, find a spanning tree $T \in G$ rooted at s, such that the maximal degree of any node in T is minimized.

Maximize:

$$\sum_{h\in\Upsilon} t_h \tag{69}$$

subject to:

$$\forall u \in V, \sum_{h\in\Upsilon} t_h \cdot \sum_{(u,v)\in h} p(u,v) \leq b(u) \tag{70}$$

$$\forall h \in \Upsilon, t_h \geq 0. \tag{71}$$

Frame 22: Linear Program for Problem MLB for the single receiver model

Minimize:

$$\sum_{u\in V} b(u) \cdot y_u \tag{72}$$

subject to:

$$\forall h \in \Upsilon, \sum_{u\in V} y_u \cdot \sum_{(u,v)\in h} p(u,v) \geq 1 \tag{73}$$

$$\forall u \in V, y_u \geq 0. \tag{74}$$

Frame 23: Dual Program for Problem MLB for the single receiver model

Theorem 12 *Problem MSBSR is NP-hard, even if all nodes have the same initial energy and a single transmission power level.*

Proof: Problem DMDST can be reduced to Problem MSBSR as follows. Given are an instance of Problem *DMDST*, with $G = (V, E)$ and a source node s in V. Consider an instance of Problem MSBSR on the graph G and source node s, where the initial energy of each node is 1 and the transmission power of every edge is 1. We now show that the solution of Problem MSBSR for this instance is a minimum degree spanning tree in G. Given a spanning tree h in G rooted at s, with maximal degree k, $lifetime(h) = min_{u\in V} \frac{b(u)}{\sum_{(u,v)\in h} p(u,v)}$. Since $p \equiv 1$ and $b \equiv 1$, we have that $lifetime(h) = min_{u\in V} \frac{1}{degree(u)} = \frac{1}{k}$. Let the tree T be a solution to Problem MSBSR, and denote its maximal degree by k. Since it has maximal lifetime, it has minimum degree, hence solves Problem DMDST.

Since Problem DMDST is NP-hard, it implies that Problem MSBSR is NP-hard as well. Moreover, we reduced Problem DMDST to an instance of Problem MSBSR for which all nodes have the same initial energy and a single transmission power level, hence, the theorem follows. ■

7.2.2 Efficient optimal solution to the multi-topology problem

In this subsection we will show how the multi-topology problem, i.e. Problem MLBSR, can be solved (optimally and) efficiently. Given are an instance of Problem MLBSR, i.e. a directed weighted graph $G \equiv (V, E, b, p)$ and a source node $s \in V$. Let Υ denote the collection of all directed trees spanning G, rooted at s. Let t_h denote the time tree h is used.

Consider the linear programming formulation of problem MLBSR specified in Frame 22. Constraint (70) guarantees that the total time a node is transmitting over all the trees is bounded by that node's initial energy. Constraint (71) guarantees that the tree durations are non-negative.

Since the number of variables, namely $\{t_h\}$, is exponential, we consider the dual problem, specified in Frame 23. This problem captures the following problem: assign non-negative weights to the nodes such that, for each wireless broadcast tree, the sum over all nodes of transmission powers (to all out-bound neighbors in the tree), multiplied by the node's weight, is at least 1, and the sum over all nodes of weights multiplied by the node initial energy is minimized.

The separation oracle for the dual linear program is equivalent to the following problem: given nodal weights, find a wireless broadcast tree, such that the sum over all nodes of transmission power multiplied by the node's weight is less

than 1, or verify there is no such tree. In other words, the separation oracle algorithm should find a tree h such that $\sum_{u\in V} y_u \cdot \sum_{(u,v)\in h} p(u,v)$ is minimized. We have that $\sum_{u\in V} y_u \cdot \sum_{(u,v)\in h} p(u,v) = \sum_{u\in V} \sum_{(u,v)\in h} y_u \cdot p(u,v)$, hence, the required tree is a minimum spanning tree where the edge weights are defined by $w(u,v) = y_u \cdot p(u,v)$. Since the Minimum Spanning Tree problem can be solved efficiently, e.g. by the Prim-Dijkstra's algorithm [10], it implies that the separation problem can be solved efficiently as well. Finally, by Theorem 6 there is a polynomial-time reduction from the original problem to the separation problem. Specifically, consider Algorithm MLBA specified in Frame 12, modified such that, in step 1(b)ii, the minimum spanning tree is found through the Prim Dijkstra's algorithm.

7.3 Multicast

In this subsection we consider the multicast generalization. In subsection 7.2.1 we have shown that the single-topology broadcast problem is NP-hard. Clearly, the single-topology multicast problem is NP-hard as well, as it is a generalization of the broadcast case.

While we established that the multi-topology broadcast problem is polynomial, we turn to show that the multi-topology multicast problem is NP-hard.

Theorem 13 *Problem MLMSR is NP-hard.*

Proof: Consider the linear programming formulation (69)-(71) from subsection 7.2.2. In order to generalize this formulation to the multicast case, it is enough to redefine Υ as the set of trees in G rooted at the source node that span the target set. The separation problem for the dual program is as follows. Given nodal weights, find a wireless multicast tree that spans the target set, such that the sum over all the nodes of transmission energy multiplied by the node's weight is less than 1, or verify that there is no such tree. Clearly, when defining edge weights as in subsection 7.2.2, i.e., $w(u,v) = y_u \cdot p(u,v)$, the separation problem is to find a Steiner tree whose weight is less than 1, or verify that there is no such tree. The minimum Steiner tree problem can be reduced to this separation problem following the same scheme as in reduction R_1, specified in Subsection 5.3. Since the minimum Steiner tree problem is NP-hard, it follows that so is the separation problem. Finally, by Theorem 6, we conclude that Problem MLMSR is NP-hard as well. ■

8 Multiple Sessions

So far we have considered the lifetime maximization of a single session. In this section we discuss the case of multiple sessions, i.e., a set of predefined sessions is given and it is required to find a routing scheme that satisfies all sessions. Again, we consider the case in which each session is restricted to consist of a single topology, as well as the case in which each session may consist of multiple topologies. For each of these problems we consider both unicast, broadcast and multicast routing. We begin by formally defining the problems.

8.1 Problem Definitions

Given is a *set of sessions*. Considering the context of multicast, each session is specified by a source node, a target set and a duration time. It is required to find a routing scheme that complies with all the sessions while obeying the energy constraints, or verify that no such solution exists. Accordingly, we define the *Maximum Lifetime Multicast Multiple Sessions* (*MLMMS*) problem as follows.

Problem MLMMS: Given are a directed weighted graph $G \equiv (V, E, b, p)$ and a set of sessions $\{(s_i, N_i, t_i) | 1 \le i \le k\}$, where, for each session i, s_i is the *source*, N_i is the *target set* and t_i is the *required duration time*. Find a set of *multicast schemes* $\{\alpha^i | 1 \le i \le k\}$, such that, for each $i, 1 \le i \le k$, $\alpha^i = \{(h^i_1, t^i_1)(h^i_2, t^i_2) \dots (h^i_{k^i}, t^i_{k^i})\}$, where $h^i_j, 1 \le j \le k^i$,is a tree rooted as s_i that spans N_i, $\sum_{j=1}^{k^i} t^i_j = t_i$ and:

$$\forall u \in V, \sum_{i=1}^{k} \sum_{j=1}^{k^i} t^i_j \cdot \max_{v\in V} \delta(h^i_j, (u,v)) \cdot p(u,v) \le b(u) \tag{75}$$

where,

$$\delta(h,e) = \begin{cases} 1 & e \in h \\ 0 & e \notin h \end{cases} \tag{76}$$

or verify that there is no set of such routing schemes.

The *unicast* and *broadcast* variants of the problem are the special cases for which $|N| = 1$ and $N \equiv V$, respectively; we term the corresponding problems as *Maximum Lifetime Unicast Multiple Sessions* (*MLUMS*) and *Maximum Lifetime Broadcast Multiple Sessions* (*MLBMS*). Note that, in the unicast case, the routing scheme consists of *paths*.

Additionally, we consider the special case in which the routing schemes are restricted to consist of a single topology each. We term the corresponding multicast problem as *Maximum lifetime Single-topology Multicast Multiple Sessions (MSMMS)* and denote the corresponding unicast and broadcast problem as MSUMS and MSBMS, correspondingly.

8.2 Single-topology Problems

In this section we consider the single-topology problems. For the case of a single session we have shown in Section 3 that the unicast, broadcast and multicast problem can be solved efficiently. We proceed to show that, in the case of multiple sessions and a single topology, all problems, i.e., unicast, broadcast and multicast, are NP-hard. We start with the unicast case.

8.2.1 NP-hardness of the Unicast problem

We show that the single-topology unicast problem, i.e. Problem MSUMS, is NP-hard, through a reduction from the *Disjoint Connected Path (DCP)* problem, which is known to be NP-hard [25]. Specifically we shall establish a reduction from the directed version of Problem DCP, denoted as DDCP, which is a generalization of Problem DCP to directed graphs, hence it is also NP-hard.

Problem *DDCP* is defined as follows:
Given is a directed graph $G = (V, E)$ and a collection of disjoint vertex pairs $\{(s_i, d_i)|1 \leq i \leq k\}$.
Question: Does G contain k mutually vertex-disjoint paths, connecting s_i and d_i, for each i, $1 \leq i \leq k$?

Theorem 14 *Problem MSUMS is NP-hard, even if all nodes have the same initial energy and a single transmission power level.*

Proof: Problem DDCP can be reduced to Problem MSUMS as follows. Given is an instance β of Problem DDCP, with $G = (V, E)$, and a collection of disjoint vertex pairs $\{(s_i, d_i)|1 \leq i \leq k\}$. Consider an instance β' of Problem MSUMS on the graph G and the set of sessions $\{(s_i, d_i, 1)|1 \leq i \leq k\}$, where the initial energy of each node is 1 and the transmission power of every edge is 1. Clearly, if there is a solution to β, i.e., there is a set vertex-disjoint paths $\{p_i, |1 \leq i \leq k\}$, from s_i to d_i, then the same paths admits a solution to β', as each node can transmit during a single time unit, and each transmission consumes a single energy unit. Conversely, assume that there is a solution to β, i.e., there is a set paths $\{p_i, |1 \leq i \leq k\}$, from s_i to d_i. Each path is employed for a single time unit, and each node can transmit during at most a single time unit, hence, each node participating in at most a single path. Therefore, the paths are vertex-disjoint, hence there is a solution to β.

Since Problem DDCP is NP-hard, it implies that Problem MSUMS is NP-hard as well. Moreover, we reduced Problem DDCP to an instance of Problem MSUMS for which all nodes have the same initial energy and a single transmission power level, hence the theorem follows. ■

8.2.2 NP-hardness of the Broadcast problem

In this section we establish that the single-topology broadcast problem, i.e. Problem MSBMS, is NP-hard. Specifically, we show a reduction from the satisfiability problem (SAT), which is very similar to Reduction SAT → MLB presented in Section 5.1. Given an instance, φ, of SAT consider the same auxiliary graph as described in Reduction SAT →

Input: A directed weighted graph $G \equiv (V, E, b, p)$, and a set of session requirements $\{(s_i, d_i, t_i) | 1 \leq i \leq k\}$.
Variables: edge flows, $\{f_e^i | \forall i, 1 \leq i \leq k, \forall e \in E\}$.
Find values to all variables that satisfy or else verify that to such values exist:

$$\forall i \in \{1..k\}, \forall u \in V \backslash \{s_i, d_i\}, \sum_{v \in V} f_(^i u, v) = \sum_{v \in V} f_(^i v, u) \tag{77}$$

$$\forall i \in \{1..k\}, \sum_{v \in V} f_{s_i,v}^i = \sum_{v \in V} f_{v,d_i}^i = t_i \tag{78}$$

$$\forall i \in \{1..k\}, \forall u \in V, f_{u,s_i}^i = x_{d_i,u} = 0 \tag{79}$$

$$\forall i \in \{1..k\}, \forall u, v \in V, x_{u,v} \geq 0 \tag{80}$$

$$\forall u \in V, \sum_{v \in V} \sum_{i=1}^{k} p(u, v) \cdot f_{u,v}^i \leq b(u) \tag{81}$$

Output: $\{f_e^i | \forall i, 1 \leq i \leq k, \forall e \in E\}$ or "failure".

Frame 24: Procedure MTFMS

MLB and the set of sessions be $\{(s, V, 1), (s, V, 1)\}$. In Section 5.1 we showed that there is a (legal) broadcast scheme for both sessions if and only if φ is satisfiable. We then have the following corollary.

Corollary 6 *Problem MSBMS is NP-hard.*

Finally, the multicast case is NP-hard as well, as it is a generalization of the unicast and broadcast cases.

8.3 Multi-topology Problems

In this section we consider the multi-topology problems. For the case of a single session we have shown in Section 5 that the broadcast and multicast cases are NP-hard, hence they are NP-hard for the case of multiple sessions. For a single unicast session, we established in Section 4 a polynomial optimal solution. We now show a simple generalization of that algorithm (i.e., Algorithm LPMLU) to the case of multiple sessions. Specifically, we extend the linear program, specified in Frame 3, in order to accommodate multiple commodities, as follows. For each session i, we define a corresponding flow variable on each edge, i.e., to an edge e and a session i corresponds a variable f_e^i. The linear program specifies a flow for each session, such that the sum of all flows does not violate the nodal energy constraints. Then, we decompose the flow of each session into path flows (through Procedure PD), out of which we compute the corresponding (optimal) unicast scheme. Since the multi-commodity program remains to be polynomial solvable, it implies a polynomial-time optimal solution to the multi-session multi-topology unicast problem.

In Frame 24 we specify *Procedure Maximal Time Flow for Multiple Sessions* (*MTFMS*) that finds a flow for each of the sessions or else verifies that there is no solution to the given instance of Problem MSUMS. Specifically, (77)-(80) are standard flow constraints, constraint (77) guarantees flow conservation for each session at relay nodes, (78) guarantees that, for each session, the flow emanating from the source reaches the destination, (79) guarantees that, for each session, no flow enters the source node nor leaves the destination, and (80) guarantees that, the flow is positive. Finally, constraint (81) guarantees that the consumed nodal energy is bounded by its initial energy.

Finally, in Frame 25, we present Algorithm LPMLUMS, which solves Problem MLUMS. The proof of its correctness follows the same lines as for Algorithm LPMLU, and is thus omitted.

9 Conclusion

We have considered routing schemes for maximizing the lifetime of unicast, broadcast and multicast in wireless networks. We first focused on routing schemes that consist of a single topology, for which we established optimal solutions for unicast, broadcast and multicast, with time complexity of $O(|E| + |V|log|V|)$. These solutions improve upon previous known results. Next, we turned to consider multi-topology schemes. We started with the unicast

Input: A directed weighted graph $G \equiv (V, E, b, p)$, and a set of session requirements $\{(s_i, d_i, t_i) | 1 \leq i \leq k\}$.

1. Execute procedure MTFMS on the input.
   (a) If no "failure", let $\{f_e^i | 1 \leq i \leq k, e \in E\}$ be the output of Procedure MTF.
   (b) Else return "failure".
2. For each $i \in \{1..k\}$, $\alpha_i = PD(\{f_e^i | e \in E\})$.

Output: $\{\alpha_i | 1 \leq i \leq k\}$ or "failure".

Frame 25: Algorithm LPMLUMS

case, and established an optimal polynomial solution, which is based on linear programming. We proved that the broadcast and multicast problems are NP-hard, and established approximation algorithms with proven performance guarantees. The main significance of this finding is in establishing that such approximation schemes exist at all. For scenarios in which faster and simpler solutions are required, we established a novel heuristic scheme for the broadcast case, and evaluated it by way of simulations. Next, we investigated the single-receiver model, and showed that the single-topology broadcast problem is NP-hard, while for the multi-topology version we established an efficient optimal solution, which implies an interesting twist of difficulty compared with the standard model. Finally, we showed that, in the single-receiver model, the single-and multiple-topologies multicast problems are both NP-hard. We then showed how some of these results can be generalized to the case of multiple sessions.

Several issues merit further research. One immediate direction is to extend our approximation schemes in order to apply also for the (NP-hard cases) of the single-receiver model. Moreover, our finding that polynomial approximation schemes exist opens the way to a search after better approximations, in terms of both the approximation factor and the computation time. Another direction is to extend our heuristic approach in order to cope with other problems beyond that of multi-topology broadcast in the standard model. At a more general level, a main challenge is to derive distributed schemes out of the centralized solutions established in this study. Indeed, distributed implementations are often required, and our results imply upper bounds on their performance. Moreover, our findings provide some important building blocks for their design. Another challenging issue is the case of multiple sessions. While we have shown how some of our results easily extend beyond the case a single session, there are still many challenging problems. One open question is whether our approximation schemes can be extended to this case. Another important task is to extend our heuristics for the case of multiple sessions and evaluate their performance. In addition, while we have assumed that all sessions are known in advance, often an *online* approach is required [26], in which no such *a-priori* knowledge is available. Finally, this study has focused on stationary wireless networks, in which nodes do not change their locations. Obviously, when nodes are mobile, the corresponding routing problems are ever more challenging. Hence, future research should explore the impact of node mobility and establish efficient solutions in terms of lifetime maximization.

References

[1] A. Chandrakasan W.R. Heinzelman and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," Maui, Hawaii, January 2000.

[2] J. Heidemann D. Estrin, R. Govindan and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," Seattle, Washington, USA, August 1999, pp. 263–270.

[3] D. Krizanc L. Kirousis, E. Kranakis and A. Pelc, "Power consumption in packet radio networks," Berlin, Germany, February 1997, vol. 1200, pp. 363–374.

[4] P. Penna A. Clementi and R. Silvestri, "The power range assignment problem in radio networks on the plane," Lille, France, March 2002, vol. 1770, pp. 651–660.

[5] M. Marathe R. Ramanathan E. Lloyd, R. Liu and S. Ravi, "Algorithmic aspects of topology control problems for ad hoc networks," Lausanne, Switzerland, June 2002.

[6] T. Chu and I. Nikolaidis, "Energy efficient broadcast in mobile ad hoc networks," Toronto, Canada, September 2002, pp. 177–190.

[7] M. Cagalj, J.P. Hubaux, and C. Enz, "Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues," Atlanta, Georgia,USA, September 2002.

[8] W. Liang, "Constructing minimum-energy broadcast trees in wireless ad hoc networks," Lausanne, Switzerland, June 2002.

[9] G. Nguyen J. Wieselthier and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," Tel Aviv, Israel, March 2000, pp. 585–594.

[10] C.E. Leiserson T.H. Cormen and R.L. Rivest, *Introduction to Algorithms*, MIT press, Cambridge, 2nd edition, 1990.

[11] G. Calinescu P.-J. Wan and O. F. X.-Y. Li, "Minimum-energy broadcast routing in static ad hoc wireless networks," Anchorage, Alaska, April 2001.

[12] I. Kang and R. Poovendran, "Maximizing network lifetime of broadcasting over wireless stationary adhoc networks," Anchorage, Alaska, USA, May 2003.

[13] J. Kohonen P. Floren, P. Kaski and P. Orponen, "Multicast time maximization in energy constrained wireless networks," San Diego, CA, USA, September 2003.

[14] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1333–1344, August 1999.

[15] A. Chandrakasan W. Heinzelman and H. Balakrishnan, "Energy-efficient routing protocols for wireless microsensor networks," Maui, Hawaii, USA, January 2000.

[16] M. Bhardwaj and A. Chandrakasan, "Bounding the lifetime of sensor networks via optimal role assignments," New York, NY, USA, June 2002.

[17] J.H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," Tel Aviv, Israel, March 2000.

[18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Networks Flows*, Prentice-Hall, New Jersey, 1993.

[19] V. Chvatal, *Linear Programming*, Freeman and Company, New Jersey, 1983.

[20] Manu K. S. Gabow H. N., "Packing algorithms for arborescences (and spanning trees) in capacitated graphs," *Math. Program.*, vol. 82, pp. 83–109, 1998.

[21] J. EDMONDS, "Edge-disjoint branchings," *Combinatorial Algorithms, R. Rustin ed. Algorithmics Press,*, pp. 91–96, 1972.

[22] L. A.Wolsey G.L. Nemhauser, *Integer and Combinatorial Optimization*, A Wiley-Interscience Publications, New York, 1988.

[23] M. Cagalj and J.P. Hubaux, "Energy-efficient broadcasting in all-wireless networks," *to appear in ACM Mobile Networks and Applications (MONET), 2003.*

[24] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," San Francisco, California, January 1998, pp. 192–200.

[25] M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman, San Francisco, 1979.

[26] A. Borodin and R. El-Yaniv, *Online computation and competativ analysis*, Cambridge University Press, New York, NY, USA, 1998.

Input: A directed tree $T = (V, E)$, a source node $s \in V$ and a set of target nodes $N \subseteq V$.
Initialization: For every node $v \in V$, $parent(u) \leftarrow NULL$, $degree(u) \leftarrow 0$

1 **for** every edge $(u, v) \in E$
2 $\quad parent(u) \leftarrow v$
3 $\quad degree(u) \leftarrow degree(u) + 1$
4 **for** every node $v \in V$
5 $\quad$ **while** $degree(v) = 0$ and $v \notin N$
6 $\quad\quad E \leftarrow E \backslash \{(parent(v), v)\}$
7 $\quad\quad v \leftarrow parent(v)$
8 $\quad\quad degree(v) \leftarrow degree(v) - 1$

Output: $T' = (V, E)$.

Frame 26: Algorithm Post-Sweep

Appendix

A Post Sweep Algorithm

The output of Algorithm GSM presented in Section 3, might include leaf nodes that are not target nodes. Transmissions to such nodes are obviously unnecessary, and may be avoided. This can be done by performing the *post-sweep* algorithm specified in Frame 26, after which all leaf nodes are target nodes.

We proceed to prove the correctness of this algorithm.

Lemma 14 *Given is a tree T which is an output of Algorithm GSM, and the corresponding output of Algorithm Post-Sweep, T', i.e. $T' = Post - Sweep(T)$. Then T' is a tree rooted at s that spans the target set for which all leaf nodes are target nodes.*

Proof: In T, there is a path from s to every target node. An edge $e = (u, v)$ is removed only if $v \notin N$ and the out-degree of v is 0, hence, e cannot belong to a path from s to a node in N. Therefore, T' spans the set of target nodes.

Now, assume by way of contradiction that there is a node $u \notin N$ that is a leaf in T'. Clearly, when u was selected in step 4, it had an out-degree larger than 0. Consider the edge e, outgoing from node u, which was removed last. By step 5, node u should have been removed from the tree, which is a contradiction. ■

Next, we turn to consider the time complexity. Steps $1 - 3$ incur in $O(|E|)$. We note that $O(|E|) = O(|N|)$, since T is a tree. Any node can be removed at most once, hence, steps $6 - 8$ are performed at most N times. Therefore, the time complexity of the algorithm is $O(N)$.

B Supporting receiver energy consumption

So far we have assumed that no energy is consumed by a receiver. In this section we show how to overcome this additional complexity. We begin with a formal definition of the model and problems.

B.1 Definitions

We generalize the definitions from section 2 as follows. A *stationary wireless network* is represented by a weighted directed graph $G \equiv (V, E, b, p, r)$, where the only change with respect to the previous model is the function r. Specifically, for each node u in V, $r(u)$ is the energy consumed at node u when receiving a transmission. For convenience, we assume that, for the source node s, $r(s) = 0$.

We now turn to define the single-topology and multi-topology problems, starting again with the multicast case.

Definition 15 *Given are a directed weighted graph $G \equiv (V, E, b, p, r)$, a source node $s \in V$ and a target set $N \subseteq V$. A* Multicast scheme *is a set of tree-duration time pairs $\alpha = \{(h_1, t_1), (h_2, t_2), \ldots, (h_k, t_k)\}$, $0 \leq i \leq k$, such that each h_i is a tree rooted at s that spans N, and the time durations $t_i \in \Re$ are such that:*

$$\forall u \in V, \sum_{i=1}^{k} t_i \cdot ((\xi(h_i, u) \cdot r(u)) + \max_{v \in V} \delta(h_i, (u, v)) \cdot p(u, v)) \leq b(u) \tag{82}$$

where

$$\delta(h_i, e) = \begin{cases} 1 & e \in h_i \\ 0 & e \notin h_i \end{cases} \tag{83}$$

$$\xi(h_i, u) = \begin{cases} 1 & u \in h_i \\ 0 & u \notin h_i \end{cases} \tag{84}$$

i.e., for each node, the total energy required by all the trees on which the node participates as either a transmitter or a receiver is bounded by the node's (initial) energy.

The definition of the *lifetime* of a multicast scheme, is identical to its definition in the original model, i.e. the lifetime is equal to the sum of tree duration times. Accordingly, we can define the *Maximum Lifetime Multicast with energy consumption at the Receiver* (*MLMR*) as follows.

Problem MLMR: *Given are a directed weighted graph $G \equiv (V, E, b, p, r)$ a source node $s \in V$ and a target set $N \in V$. Find a multicast scheme α in G rooted at s that spans N, such that, for any other such multicast scheme β, $lifetime(\beta) \leq lifetime(\alpha)$.*

The *unicast* and *broadcast* variants of the problem are the special cases for which $|N| = 1$ and $N \equiv V$, respectively; we term the corresponding problems as *Maximum Lifetime Unicast with energy consumption at the Receiver* (*MLUR*) and *Maximum Lifetime Broadcast with energy consumption at the Receiver* (*MLBR*).

Additionally, we consider the special case in which the routing scheme is restricted to consist a single topology, i.e., we seek a single tree (or path, for unicast) of maximum lifetime. We term the corresponding multicast problem as *Maximum lifetime Single-topology Multicast with energy consumption at the Receiver (MSMR)*. We denote the unicast and broadcast problems as MSUR and MSBR correspondingly.

B.2 Single-topology Solutions

In this section we show how to modify the solutions for the single-topology problems, established in Section 3, in order to support non-negligible receiver energy. Specifically, we establish a solution for the multicast case, namely to problem MSMR, which implies solutions to the broadcast and unicast cases, namely Problems MSBR and MSUR. For Problem MSMR, consider Algorithm GSM specified in Frame 1, and modify step 3 as follows:

3 **foreach** $(u, v) \in E$, $w(u, v) \leftarrow \frac{b(u)}{r(u)+p(u,v)}$.

We refer to the modified algorithm as Algorithm GSMR. As with algorithm GSM, the Post-Sweep scheme specified in Frame 26 is applied to the output of Algorithm GSMR. We turn to prove that correctness of the Algorithm GSMR.

Theorem 15 *Algorithm GSMR solves Problem MSMR.*

Proof: By the correctness of Algorithm GSM (Lemma 1), the output T of Algorithm GSMR is a tree rooted at s that spans N, such that, for any other spanning tree T' of G rooted at s:

$$min_{(u,v) \in T} \frac{b(u)}{r(u) + p(u, v)} \geq min_{(u,v) \in T'} \frac{b(u)}{r(u) + p(u, v)}. \tag{85}$$

By way of contradiction, assume that there exists a tree T' rooted at s that spans N, such that $lifetime(T') > lifetime(T)$. Let u be some node in T for which the energy drains out first, i.e. $\frac{b(u)}{r(u)+max_{(u,v) \in T} p(u,v)}$ has minimum value. There are two cases:

1. If u is a leaf in T then $p(u,v)=0$ for all (u,v) in T. By the correctness of the Post-Sweep scheme (Lemma 14), u is a target node, hence u is a receiver in T'. Thus, $lifetime(T)=\frac{b(u)}{r(u)} \geq lifetime(T')$, which is a contradiction.

2. If u transmits in T, namely there is some edge $(u,v) \in T$, then by (85), there exists some edge $(x,y) \in T'$, such that $\frac{b(u)}{r(u)+p(u,v)} \geq \frac{b(x)}{r(x)+p(x,y)}$. Therefore,

$$lifetime(T) = \frac{b(u)}{r(u)+p(u,v)} \geq \frac{b(x)}{r(x)+p(x,y)} \geq lifetime(T'), \tag{86}$$

which is a contradiction. ■

B.3 Multi-topology Unicast

In this section we consider the multi-topology unicast problem, namely Problem MLUR. In section 4, we established Algorithm LPMLU, specified in Frame 5, for solving Problem MLU. Algorithm LPMLU consists of two parts: the first involves a solution of a linear program, while the second part consists of path decomposition. The energy constraints are imposed by constraint (8) in the first part. This constraint guarantees that, for each node, the energy consumed during transmissions of that node throughout all the paths, is bounded by its initial energy. In the new model, nodes consume energy both during reception and transmission; hence, in order for algorithm LPMLU to support the new model, i.e. solve Problem MLUR, the only change required is the replacement of constraint (8) with the following:

$$\forall u \in V, \sum_{v \in V} p(u,v) \cdot x_{u,v} + \sum_{v \in V} r(u) \cdot x_{v,u} \leq b(u). \tag{87}$$

The proof of the correctness of this variant of Algorithm LPMLU follows the same lines as for the original, and thus is omitted.

B.4 Multi-topology Broadcast and Multicast

In this section we describe how the approximation schemes for the broadcast and multicast problems, namely Algorithms MLBA and MLMA, can be modified in order to support the new model.

Algorithm MLBA is based on the linear programming formulation of Problem MLB, specified in Frame 8. In order to support consumption of energy at the receiver, constraint (38) should be replaced with the following:

$$\forall u \in V, \sum_{h \in \Upsilon} t_h \cdot (\xi(h,u) \cdot r(u)) + \max_{(u,v) \in h} p(u,v)) \leq b(u) \tag{88}$$

where

$$\xi(h,u) = \begin{cases} 1 & u \in h \\ 0 & u \notin h. \end{cases} \tag{89}$$

We note that the expression $(\xi(h,u) \cdot r(u) + \max_{(u,v) \in h} p(u,v))$ is constant for a given tree h and node u and is equal to the power consumed at node u in the tree h. In the dual program, constraint (41) should now be replaced with:

$$\forall h \in \Upsilon, \sum_{u \in V} y_u \cdot (\xi(h,u) \cdot r(u) + \max_{(u,v) \in h} p(u,v)) \geq 1. \tag{90}$$

Algorithm MLBA is based on finding an approximation algorithm to the separation problem of the dual program. With energy consumption at the receiver, that problem is as follows.

Problem Separation for the MLB Dual program with energy consumption at the Receiver (SMLBR): *Given are a directed weighted graph $G \equiv (V,E,p,r)$, nodal weights y and a source node $s \in V$. Find a spanning tree h in G rooted at s, such that $\sum_{u \in V} y_u \cdot (\xi(h,u) \cdot r(u) + \max_{(u,v) \in h} p(u,v))$ is minimized.*

Next, we show that Problem SMLBR can be reduced to Problem SMLB, i.e. to the separation of the dual linear program in the original model. The reduction is established through the construction of an auxiliary graph, in which

Given: An α-approximation, A for Problem SMLB.
Input: Given are a directed weighted graph $G \equiv (V, E, p, r)$, nodal weights y and a source node $s \in V$.

1. Construct the following auxiliary graph, $G' = (V', E', p')$:
   (a) $V' = V \cup \{u' | u \in V\}$.
   (b) $E' = E \cup \{(u', u) | u \in V\}$.
   (c) For all e in E, $p'(e) = p(e)$.
   (d) For all $e = (u, u')$ in E', $p'(e) = r(u)$.
2. Define y' as follows: for all u in V, $y'(u) = y'(u') = y(u)$.
3. $h' \leftarrow A(G', y')$.
4. $h \leftarrow \{(u, v) | (u, v') \in T'\}$.

Output: h'.

Frame 27: Reduction SMLBR $\rightarrow$ SMLB

each node u is replaced with two nodes u and u', and an edge (u, u'). The weight of that edge is the power consumed by u during reception. In Frame 27 we specify this reduction.

Next, we turn to prove the validity of the reduction.

Lemma 15 *The output T of reduction SMLBR $\rightarrow$ SMLB is an α-approximation.*

Proof: First, we show that h is a spanning tree. By the correctness of algorithm A, h' is a spanning tree in G', hence for every node u in V there is a path in G', $s \rightarrow u'_1 \rightarrow u_1 \ldots \rightarrow u'_k \rightarrow u_k = u$. Hence, by definition of T, $s \rightarrow u_1 \ldots \rightarrow u_k = u$ is a path in G. Therefore, T is a spanning tree in G rooted at s.

Second, by definition of h, and G', the weight of h, i.e $\sum_{u \in V} y_u \cdot (\xi(h, u) \cdot r(u) + \max_{(u,v) \in h} p(u, v))$, is the same as the weight of h'. Hence, the reduction preserves the approximation ratio. ■

Clearly, by the above lemma, we can establish an approximation scheme for Problem MLBR with the same approximation ratio as for Problem MLB. We note that the same reduction scheme can be used to establish an approximation scheme for Problem MLMR with the same approximation ratio as for Problem MLB.

B.5 Single receiver model

In this section we consider the single receiver model. Since we assume non-negligible receiver energy, we have that each node consumes energy during the reception of a transmission as well as during *each* transmission. As mentioned in Section 7, the unicast case is not affected by the change of model. The multicast case is intractable for both the single-topology and multi-topology problems. The broadcast case is intractable for the single-topology problem, hence it remains to be shown how to adjust the algorithm for the multi-topology broadcast problem. It is easy to see that this can be done through the same scheme described in Section B.4, thus the details are omitted.