Power Efficient Range Assignment in Ad-hoc Wireless Networks

E. Althaus G. Călinescu* I.I. Măndoiu† S. Prasad‡ N. Tchervenski* A. Zelikovsky‡

Abstract—**We study the problem of assigning transmission ranges to the nodes of ad hoc wireless networks so that to minimize power consumption while ensuring network connectivity. We give (1) an exact branch and cut algorithm based on a new integer linear program formulation solving instances with up to 35-40 nodes in 1 hour; (2) a proof that** MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENTS **is inapproximable within factor $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$ unless $P = NP$; (3) an improved analysis for two approximation algorithms recently proposed by Călinescu et al. (TCS'02), decreasing the best known approximation factor to $5/3+\epsilon$; (4) a comprehensive experimental study comparing new and previously proposed heuristics with the above exact and approximation algorithms.**

INTRODUCTION

Ad hoc wireless networks have received significant attention in recent years due to their potential applications in battlefield, emergency disaster relief, and other application scenarios (see, e.g., [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]). Unlike wired networks or cellular networks, no wired backbone infrastructure is installed in ad hoc wireless networks. A communication session is achieved either through single-hop transmission if the recipient is within the transmission range of the source node, or by relaying through intermediate nodes otherwise. When a transmission is made by a node it can be received by all nodes within its transmission range. This feature is extremely useful for energy-efficient multicast and broadcast communications.

For the purpose of energy conservation, each node can (possibly dynamically) adjust its transmitting power, based on the distance to the receiving node and the background noise. In the most commonly used power-attenuation model [11], the signal power falls as $\frac{1}{r^\kappa}$ where r is the distance from the transmitter antenna and κ is a coefficient dependent on the wireless environment, typically between 2 and 4. Under this model the power requirement for supporting a link from node u to node v separated by a distance r is given by

$$p(u,v) = \frac{r^{\kappa_{uv}}}{\chi_u \sigma_v} \qquad (1)$$

where $\chi_u > 0$ is the transmission efficiency of node u, $\sigma_v > 0$ is the signal detection sensitivity threshold of node v, and κ_{uv} is the signal attenuation exponent for the link from u to v. Typically it is assumed that all nodes have the same transmission efficiency and detection sensitivity coefficients, both normalized to 1, and that signal attenuation exponents κ_{uv} have a common value κ (see, e.g., [12]). With these assumptions, the power requirement for supporting a link between nodes u and v separated by a distance r becomes

$$p(u,v) = p(v,u) = r^\kappa \qquad (2)$$

Unless explicitly stated otherwise, in this paper we assume symmetric power requirements such as those given by (2).

Having every link established in both directions simplifies one-hop transmission protocols by allowing acknowledgement messages to be sent back for every packet (see, for example [13]). This motivates the study of the MIN-POWER SYMMETRIC CONNECTIVITY problem, where a link is established only if both nodes have transmission range at least as big as the distance between them, and the goal is to ensure that the network is connected [1], [14].

Formally, let V be a set representing network nodes. A *range assignment* is a function $r : V \to R_+$. We say that a *unidirectional link* from node u to node v is established under the range assignment r if $r(u) \geq p(u,v)$. Similarly, a *bidirectional link uv* is established under the range assignment r if $r(u) \geq p(u,v)$ and $r(v) \geq p(v,u)$. Let $B(r)$ denote the set of all bidirectional links established between pairs of nodes in V under the range assignment r.

MIN-POWER SYMMETRIC CONNECTIVITY: Given a set of nodes V and symmetric power requirements $p(u,v) = p(v,u)$, $u, v \in V$, find a range assignment $r : V \to R_+$ minimizing $\sum_{v\in V} r(v)$ subject to the constraint that the graph $(V, B(r))$ is connected.

Implicit in the work of Clementi, Penna, and Silvestri [4] is a proof that MIN-POWER SYMMETRIC CONNECTIVITY in E^2 is NP-Hard. Therefore, we study approximation algorithms and heuristics for the problem. In this paper we make the following contributions:

- We give an exact branch and cut algorithm based on a new integer linear program formulation for the problem (Section I). Experimental results show that the branch and cut algorithm solves instances with 25 nodes in less than one minute and instances with up to 35-40 nodes in 1 hour (Section III).
- We give an improved analysis for two approximation algorithms from [14], decreasing the best known approximation factor[1] to $5/3+\epsilon$ (Section II-A).

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany. E-mail: althaus@mpi-sb.mpg.de.

* Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: {calinesc,tchenic}@cs.iit.edu. Research of GC was partially performed while visiting the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

† Department of Computer Science & Engineering, UC San Diego, La Jolla, CA 92093. E-mail: mandoiu@cs.ucsd.edu. Partially supported by MARCO Gigascale Silicon Research Center.

‡ Department of Computer Science, Georgia State University, Atlanta, GA 30303. E-mail: {sprasad,alexz}@cs.gsu.edu. SP and AZ were partially supported by the State of Georgia's Yamacraw Initiative. AZ was also partially supported by NSF Grant CCR-9988331, Award No. MM2-3018 of MRDA and CRDF.

[1]The *approximation factor* of an algorithm A for a minimization problem is the supremum, over all possible instances I, of the ratio between the cost of the output of A when run on I and the cost of an optimal solution for I (the smaller the performance ratio, the better). We say that A is an *α-approximation algorithm* if its approximation ratio is at most α.

- We show that MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENTS is inapproximable within factor $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$ unless $P = NP$ (Section II-B).
- We present a comprehensive experimental study comparing new and previously proposed heuristics with the above exact and approximation algorithms. Experimental results show an average of 5-6% reduction in power consumption compared to the simple MST based solution (Section III).

For the important special case of nodes located on a line and monotonic symmetric power requirements, MIN-POWER SYMMETRIC CONNECTIVITY can be solved in polynomial time by dynamic programming; we omit the details for space reasons.

A. *Related Work*

Kirousis, Kranakis, Krizanc, and Pelc [5] give a minimum spanning tree (MST) based 2-approximation algorithm for MIN-POWER SYMMETRIC CONNECTIVITY (their algorithm is actually designed for the COMPLETE RANGE ASSIGNMENT problem discussed below). Călinescu et al. [14] push the approximation ratio below 2 by exploiting similarities between MIN-POWER SYMMETRIC CONNECTIVITY and the classic STEINER TREE problem. In particular, [14] gives a fully polynomial $1+\ln 2$ approximation scheme[2] based on [15] and a more practical $15/8$ approximation algorithm based on [16]. Blough et al. [1] give asymptotic bounds on the solution cost for random and so called (Δ, δ) Euclidean instances.

The objective of minimizing the total power has been more extensively addressed under the specific power requirements given by (2) and the related *asymmetric* connectivity model, in which unidirectional links give raise to a directed graph on V (see [12]). Four problems have been studied under this model.

1. ASYMMETRIC UNICAST, which requires establishing a minimum power directed path from a source s to a destination t. ASYMMETRIC UNICAST is easily solved in polynomial time by shortest-path algorithms. The related MIN-POWER SYMMETRIC UNICAST problem can also be solved efficiently by a shortest-path computation in an appropriately constructed graph [14].

2. ASYMMETRIC BROADCAST, which requires establishing a minimum power arborescence rooted at a given vertex s [8], [10]. Clementi et al. [3] prove that ASYMMETRIC BROADCAST is NP-Hard when the nodes are in E^2. The best known approximation algorithm for ASYMMETRIC BROADCAST [9], based on computing a minimum spanning tree, has performance ratio of at most 12 when the nodes are in E^2. Chu and Nikolaidis [2] give an experimental study of asymmetric broadcast algorithms for mobile ad hoc networks. As noted in [14], the related SYMMETRIC BROADCAST problem is identical to MIN-POWER SYMMETRIC CONNECTIVITY.

3. ASYMMETRIC MULTICAST, in which one is given a root s and a set of terminals T, and the goal is to establish a minimum-power branching rooted at s which reaches all vertices of T. As a generalization of ASYMMETRIC BROADCAST, ASYMMETRIC MULTICAST is also NP-Hard, and based on the work of [9], it is immediate that a minimum Steiner tree would give an approximation ratio of 12ρ, where ρ is the approximation for Steiner tree in graphs (the best result known at this moment, given in [17], is $\rho = 1 + \frac{1}{2}\ln 3 + \varepsilon$).

4. COMPLETE RANGE ASSIGNMENT, in which the objective is establishing a strongly connected subgraph of V. Kirousis, Kranakis, Krizanc, and Pelc [5] give an $O(n^4)$ dynamic programmig algorithm for the case of n colinear nodes and power requirements given by (2), prove that COMPLETE RANGE ASSIGNMENT in E^3 is NP-Hard, and give a 2-approximation algorithm based on the minimum spanning tree, As opposed to the ASYMMETRIC BROADCAST approximation of [9], the COMPLETE RANGE ASSIGNMENT approximation of [5] is valid in arbitrary graphs (that is, the distance between two points could be arbitrary, not necessarily Euclidean). Clementi, Penna, and Silvestri [4] give an elaborate reduction proving that COMPLETE RANGE ASSIGNMENT in E^2 is also NP-Hard. As shown in [14], the power required by for the asymmetric COMPLETE RANGE ASSIGNMENT can be as low as half the power required for MIN-POWER SYMMETRIC CONNECTIVITY.

[2]A *fully polynomial α approximation scheme* is a family of algorithms A_ε such that, for every $\varepsilon > 0$, A_ε (1) has performance ratio at most $\alpha + \varepsilon$, and (2) runs in time polynomial in the size of the instance and $1/\varepsilon$.

I. INTEGER LINEAR PROGRAM FORMULATION

In this section we give an integer linear program (ILP) formulation for MIN-POWER SYMMETRIC CONNECTIVITY and describe a branch and cut algorithm based on it. The results in Section III show that the algorithm is practical for instances with up to 35-40 nodes.

We begin by reformulating MIN-POWER SYMMETRIC CONNECTIVITY in graph theoretical terms. Let $G = (V, E, c)$ be an edge-weighted graph and uv denote the undirected edge between nodes u and v. The cost $c(uv)$ of an edge $uv \in E$ corresponds to the (symmetric) power requirement $p(u, v) = p(v, u)$. For a node $u \in V$ and a spanning tree T of G, let uu_T be the maximum cost edge incident to u in T, i.e., $uu_T \in T$ and $c(uu_T) \geq c(uv)$ for all $uv \in T$. The *power cost* of a spanning tree T is

$$p(T) = \sum_{u \in V} c(uu_T)$$

Since any connected graph contains a spanning tree, an equivalent formulation of MIN-POWER SYMMETRIC CONNECTIVITY is to ask for a spanning tree with minimum power-cost in the complete graph on V with edge costs given by $c(uv) = \|uv\|^\kappa$. Thus, MIN-POWER SYMMETRIC CONNECTIVITY can be reformulated as follows:

MINIMUM POWER-COST SPANNING TREE: Given a connected edge-weighted graph $G = (V, E, c)$, find a spanning tree T of G with minimum power-cost.

To formulate MINIMUM POWER-COST SPANNING TREE as a linear integer program we use two types of binary decision variables:

x_{uv}	for all $uv \in E$; x_{uv} is set to 1 if uv belongs to the selected spanning tree T and to 0 otherwise. We call these variables the *tree variables*.
$y_{\overline{uv}}$	for all $\overline{uv} \in \overline{E} := \{\overline{uv}, \overline{vu} \mid uv \in E\}$; $y_{\overline{uv}}$ is set to 1 if $u_T = v$ (i.e., if $uv \in T$ and $c(uv) \geq c(uw)$ for all $uw \in T$. We call these variables the *range variables*.

Note that there are $|E|$ tree variables and $|\overline{E}| = 2|E|$ range variables. Let ST be set of the incidence vectors of all spanning trees of G (viewed as subsets of E). Our ILP formulation is as follows.

$$
\begin{aligned}
\min \quad & \sum_{\overline{uv}\in\overline{E}} c(uv) y_{\overline{uv}} & & \\
\text{s.t.} \quad & \sum_{v\in V|\overline{uv}\in\overline{E}} y_{\overline{uv}} = 1, & \forall u \in V \quad & (3)\\
& x_{uv} \le \sum_{\overline{uw}\in\overline{E}|c(uw)\ge c(uv)} y_{\overline{uw}}, & \forall \overline{uv} \in \overline{E} \quad & (4)\\
& x \in \mathrm{conv}(ST) & & (5)\\
& x \in \{0,1\}^{|E|} & & \\
& y \in \{0,1\}^{|\overline{E}|} & &
\end{aligned}
$$

The constraints (3) enforce that we select exactly one range variable for every node $v \in V$, i.e., we properly define the range of each node. The constraints (4) enforce that an edge uv is included in the tree only if the range of each endpoint is at least the cost of the edge. The constraints (5) enforce that the tree variables indeed form a spanning tree. There are several well known linear descriptions for $x \in \mathrm{conv}(ST)$. We use the following, most famous formulation: $x \in \mathrm{conv}(ST) \Leftrightarrow x \ge 0, \sum_{e\in E} x_e = |V| - 1$ and $\sum_{e\in\gamma(S)} x_e \le |S| - 1$ for all $S \subseteq E$, where $\gamma(S)$ is the set of edges of E with both ends in S.

To solve the ILP we use branch and cut, i.e., we drop the integrality constraints and solve the corresponding LP relaxation. If the solution of the LP is integral, we found the optimal solution, otherwise we pick a variable with a fractional value and split the problem into two subproblems by setting the variable to 0 and 1 in the subproblems. We solve the subproblems recursively and disregard a subproblem if its LP bound is worse than the best known solution.

Since there are an exponential number of inequalities in this formulation of spanning trees, we can not solve the LP directly. Instead, we start with a small subset of these inequalities and algorithmically test whether the LP solution violates an inequality which is not in the current LP. If so, we add the inequality to the LP, otherwise we found the solution of the LP with the exponential number of inequalities. The inequalities added to the LP if needed are called *cutting planes*, algorithms that find violated cutting planes are called *separation algorithms*.

In our case, the initial LP consists of the constraints (3) and (4), the constraint $\sum_{e\in E} x_e = |V| - 1$, and the bound constraints, i.e., the constraints $0 \le x \le 1$ and $0 \le y \le 1$. The only constraints added on demand are the constraints $\sum_{e\in\gamma(S)} x_e \le |S| - 1$ for all $S \subseteq E$. A separation algorithm for these inequalities is due to Padberg and Wolsey [18].

The running time of a branch and cut algorithm can improved by tightening the LP relaxation, i.e., by finding additional inequalities which are valid for all integer points, but may be violated by solutions to the LP relaxation (Figure 1 shows an example). We use the following class of valid inequalities. Let $S \subset V$. For every $u \in S$ let $u_S \in V \setminus S$ so that $c(uu_s) \le c(uv)$ for all $v \in V \setminus S$. The inequality

$$\sum_{u\in S} \sum_{v\in V|c(uv)\ge c(uu_S)} y_{\overline{uv}} \ge 1 \qquad (6)$$

is valid for the problem above. We can argue as follows. There is at least one edge in the spanning tree T crossing the cut S. Let uv be such an edge and $u \in S$. Then $c(uv) \ge c(uu_S)$ and

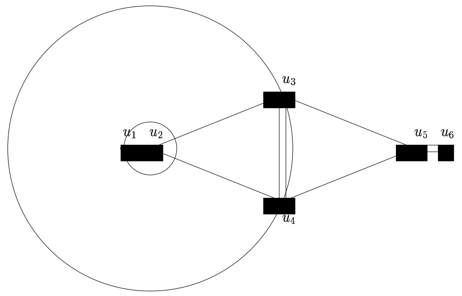


Fig. 1. Let $x_e = 1/2$ for all edges in the picture ($x_e = 1$, if there are two parallel edges). Let range variables $y_{\overline{u_2 v}}$ be equal to 1/2 for $v = u_1, u_3$, and to 0 otherwise. Then constraints of type (3) and (4), are satisfied, but the constraint (6) is violated for $S = \{u_1, u_2\}$.

the range of u is at least $c(uv)$. Thus $\sum_{v\in V|c(uv)\ge c(uu_S)} y_{\overline{uv}}$ is one and the inequality is valid.

Since we do not have a separation algorithm for these inequalities, we use the following heuristic to separate some of them. We chose an arbitrary node u. For every node $v \in V \setminus \{u\}$, we compute the minimal directed cut from u to v and from v to u, where the capacity of an edge xy is given by $\sum_{xw|c(xw)\ge c(xy)} y_{xw}$. For all computed cuts, we test whether the corresponding inequality is violated.

II. Approximation Algorithms

In this section we give an improved analysis for two approximation algorithms proposed in [14] for MIN-POWER SYMMETRIC CONNECTIVITY and show that MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENTS cannot be approximated within factor $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$ unless $P = NP$;

A. Improved Approximations for Symmetric Requirements

Recently, [14] proposed an algorithm with approximation ratio of $7/4 + \epsilon$ based on polynomial time approximation scheme for computing the minimum-weight spanning cactus in a 3-uniform hypergraph, and a more practical greedy algorithm with approximation ratio of 15/8. Below we improve the approximation factors of these two algorithms to $5/3 + \epsilon$ and 11/6, respectively.

The greedy algorithm in [14] (see Figure 2) iteratively improves the MST by inserting the best *fork*, i.e., pair of edges of G sharing a node. Note that the power cost of fork $K = \{e_1, e_2\}$ is $p(K) = 2\max\{c(e_1), c(e_1)\} + \min\{c(e_1), c(e_1)\}$. The edges of fork K added to the MST replace the highest cost edges in the two created cycles. As a result the power cost may decrease. The *gain* of fork K is defined by

$$gain(K) = 2mst(G) - 2mst(G/K) - p(K)$$

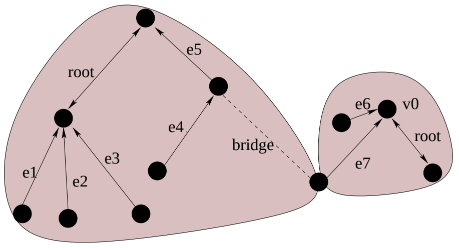
where $mst(G)$ the minimum cost of a spanning tree of G, G/K is the graph obtained from G by collapsing all nodes of K into a single node.

The proof of the approximation ratios in [14] is based on on the notion of *3-restricted decomposition* of a tree T, which is a partition of the edges of T into forks and individual edges. The power-cost of a 3-restricted decomposition Q, $p(Q)$, is the sum of power-costs of its elements, i.e., forks and individual edges. It is proved in [14] that, if there exists a 3-restricted decomposition Q with $P(Q) \le \rho p(T)$, then the greedy algorithm

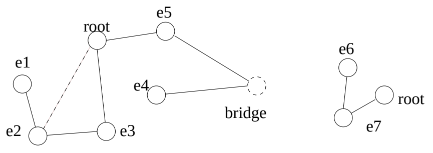
```
**Input:** Edge-weighted graph $G = (V, E, c)$
**Output:** Spanning tree of $G$

$T \leftarrow MST(G), H \leftarrow \emptyset$
**Repeat forever**
    Find a fork $K$ with the maximum $g = gain_T(K)$
    **If** $g \leq 0$ **then exit repeat**
    $H \leftarrow H \cup K, G \leftarrow G/K, T \leftarrow T/K$
**Output** $T \cup H$
```

Fig. 2. The greedy fork-contraction algorithm.



(a)



(b)

Fig. 3. (a) Partitioned tree T. Each vertex has a single outgoing arc denoting its maximum incident edge, double arcs are roots and dashed edges are bridges. (b) Consecutive line graphs for the components. Vertices represent edges of T and edges represent forks of T; "consecutive" edges are solid and "parity" edges are dashed.

in Figure 2 has approximation ratio $1 + \rho/2$ and there is PTAS with approximation ratio $\rho + \epsilon$. Below we prove that $\rho \leq 5/3$ improving the ratio $7/4$ from [14].

Theorem 1: For any tree T, there is a 3-restricted decomposition Q of T such that $p(Q) \leq \frac{5}{3}p(T)$.

Proof: The proof proceeds in three steps. First we partition the edges of T into disjoint components using structural information derived from power requirements. Then we construct a weighted subgraph of the line graph of each component, which we refer to as the "consecutive" line graph. Finally, we show that the consecutive line graph of each component has a matching exceeding a certain weight; the edges in these matchings correspond to the forks in the desired 3-restricted decomposition of T.

To describe how we partition the edges of T (see Figure 3(a)) we need to introduce some additional notations. Let $max(u)$ be the maximum edge of T incident to a vertex u.[3] For each vertex u, we direct the edge $max(u)$ away from u. An edge uv is called *root* if it is directed both ways (i.e., $max(u) = max(v) = uv$), and called *bridge* if it remains undirected (i.e., $max(u) \neq uv$ and $max(v) \neq uv$). In the power-cost of T, roots are counted twice (for both endpoints), bridges are not counted at all, and all other edges are counted exactly once. Thus, denoting by R the set of roots and by B the set of bridges, we have:

$$p(T) = c(T) + c(R) - c(B) \tag{7}$$

The edges of T are partitioned as follows. First, we start with the connected components of $T - B$; note that each such component contains exactly one root. Then we add each bridge b of B to one of the two adjacent components of $T - B$, such that each component gets at most one bridge. A bridge assignment with this property is obtained by selecting an arbitrary vertex v_0 and assigning to each component of $T - B$ not containing v_0 the unique adjacent bridge on the path to v_0. We denote by $\mathcal{D}$ the resulting partition.

[3]W.l.o.g., we assume that no two edges of T have the same cost.

A fork $(e_1 = uv, e_2 = u'v)$ is called *consecutive* if $c(e_1) < c(e_2)$ and there is no edge e$\in D$ incident to v such that $c(e_1) < c(e) < c(e_2)$. For each component $D \in \mathcal{D}$, the *consecutive line graph* L_D is defined as follows (see Figure 3(b)):

– vertices of L_D are the edges of D
– L_D has "consecutive" edges connecting each consecutive forks of D, and at most two "parity" edges connecting the root of D and the second most expensive non-root edge incident to each end of the root
– for every edge (e_1, e_2) of L_D, $w(e_1, e_2) = \min\{c(e_1), c(e_2)\}$

By construction, each edge of L_D corresponds to a fork of D. Therefore, each matching X of L_D corresponds to a 3-restricted decomposition of D (edges of X correspond to forks and isolated vertices correspond to isolated edges) which we denote Q_X. It is easy to see that $p(Q_X) = 2c(D) - w(X)$.

The theorem follows if, for each $D \in \mathcal{D}$, we find a matching X_D in L_D such that

$$w(X_D) \geq \frac{c(D) - c(r_D) + c(b_D)}{3} \tag{8}$$

where $c(D)$ is the total cost of the edges in D, r_D is the single root in D, and b_D is the single bridge in D, if one exists. Indeed,

$$\begin{aligned} p(\bigcup_{D\in\mathcal{D}} Q_{X_D}) &= \sum_{D\in\mathcal{D}} (2c(D) - w(X_D)) \\ &\leq \sum_{D\in\mathcal{D}} \left(\frac{5}{3}c(D) + \frac{1}{3}c(r_D) - \frac{1}{3}c(b_D)\right) \\ &= \frac{5}{3}c(T) + \frac{1}{3}c(R) - \frac{1}{3}c(B) \\ &\leq \frac{5}{3}p(T) \end{aligned}$$

where the last inequality comes from (7) and the fact that $c(T) \leq p(T)$ (see, e.g., [14]).

By Edmonds' theorem [19] it is sufficient to construct a fractional matching X_D satisfying (8). A *fractional matching* of L_D is an assignment of nonnegative fractions $x(e_1, e_2)$ to every edge $(e_1, e_2) \in L_D$ such that

(i) the sum of fractions assigned to the edges incident to a vertex e of L_D is at most 1, and
(ii) the sum of fractions assigned to all edges with both endpoints in a set of $2k + 1$ vertices of L_D is at most k.

The weight of a fractional matching X_D is given by

$$w(X_D) = \sum_{(e,e')\in E(D)} x(e, e')w(e, e')$$

We construct a fractional matching X_D by assigning $1/3$ to each consecutive edge (e_1, e_2) of L_D. This fractional matching satisfies (i) since each $e \in D$ is incident to at most 3 consecutive edges of L_D (if e is not the root r_D, then it participates into one consecutive edge of L_D as e_1, and into up to two edges as e_2; the root participates as the heaviest end in up to two edges).

Condition (ii) follows from the fact that consecutive edges form a tree. Since every vertex e of L_D except the root participates in exactly one consecutive fork (e_1, e_2) as e_1, we get that the weight of X'_D is equal to $(c(D) - c(r_D))/3$.

If D has no bridge then (8) follows. Otherwise we modify X_D such that the weight increases by $c(b_D)/3$ as follows. Let $P = (b_D = e_0, f_0, e_1, f_1, ..., e_k, f_k, e_{k+1} = r_D)$ be the unique path of consecutive edges of L_D, where $f_i = (e_i, e_{i+1})$, $i = 1, \ldots, k$ are edges of L_D corresponding to consecutive forks in D. We add 1/3 to $x(f_i)$, $i = 0, 2, 4, \ldots$, and substract 1/3 from $x(f_i)$, $i = 1, 3, \ldots$. Since both b_D and r_D participate in at most two consecutive forks, the above change leads to a feasible fractional matching (the sum of fractions assigned to the edges incident to each intermediate vertex of P remains the same). If k is even then the total weight of X_D increases by at least $c(b_D)/3$ since $w(f_{2l-1}) = c(e_{2l-1}) < c(e_{2l}) = w(f_{2l})$, $l = 1, \ldots, k/2$ and we are done.

If k is odd we add back 1/3 to $x(f_k)$ to guarantee increasing of $w(X_D)$ by at least $c(b_D)/3$. If e_k has degree 2 in L_D then we are done, since the sum of all fractions assigned to the edges incident to e_k equals to 1. Otherwise, e_k has degree 3 and we need to further modify X_D in order to make it a feasible fractional matching. Let v be the common vertex of e_k and r_D. Since $f_k = (e_k, e_{k+1} = r_D)$ is a consecutive fork, e_k is the most expensive non-root edge of D incident to v. Let e be the second most expensive non-root edge of D incident to v. Since e and e_k form a consecutive fork, L_D contains the (e, e_k). Recall that L_D also contains a parity edge (e, r_D). We modify X_D as follows:

- If $e_{k-1} \neq e$ (i.e., e_{k-1} is not adjacent to the root), then we substract 1/3 from $x(e, e_k)$ and set $x(e, r_D)$ to 1/3.
- If $e_{k-1} = e$ (i.e., e_{k-1} is adjacent to the root), then we substract 1/3 from $x(f_{k-1})$ and set $x(e = e_{k-1}, r_D)$ to 1/3.

In both cases, the resulting sums of fractions assigned to the edges incident each of e_k and r_D are equal to 1, and hence X_D satisfies (i). Condition (ii) is valid since edges with non-zero weight in X_D continue to form a tree. ■

B. Hardness of Approximation for Asymmetric Requirements

The following theorem shows that handling asymmetric power requirements is intrinsically more difficult than handling symmetric requirements, e.g., under asymmetric power requirements the minimum total power cannot be approximated within any constant.

Theorem 2: MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENTS cannot be approximated in polynomial time within factor $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$ unles $P = NP$.

Proof: We will prove the theorem by reduction from the set cover problem which is known to be non-approximable within factor $(1-\epsilon)\ln|V|$ for any $\epsilon > 0$ unles $P = NP$ [20].

For any instance of the set cover problem, we construct an instance of the problem of MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENTS such that approximating the second problem within a factor F implies approximating the first problem within the same factor.

Let $S_1, S_2, \ldots, S_k$ be subsets of a ground set $X = \{x_1, x_2, \ldots, x_n$. The set cover problem asks for the minimum number of subsets covering the entire set X. We construct an instance of Min-Power Symmetric Connectivity in which each

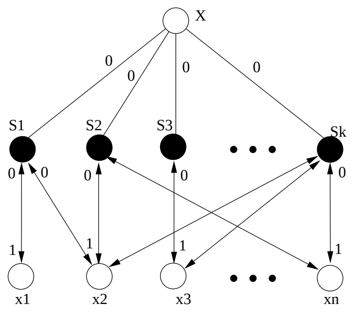


Fig. 4. Solid dots coorrespond to subsets and empty dots correspond to elements of the set X in the instance of the set cover problem. Each bididrected arc shows that an elemnt belongs to a subset.

point cooresponds to either a set S_i or an element $x_j \in X$ or is an auxiliary point X (see Figure 4). We set the power reqirements as follows: if $x_j \in S_i$, then $p(x_j, S_i) = 0$ and $p(S_i, x_j) = 1$. Also $p(S_i, X) = p(X, S_i) = 0$. All other power requirements are infinity. It is easy to see that the minimum total power equals the minimum number of sets S_i covering X. ■

Remark. Theorem 2 relies on using non-uniform signal attenuation exponents κ_{uv}; we leave open the approximability status of MIN-POWER SYMMETRIC CONNECTIVITY with uniform exponents.

III. EXPERIMENTAL STUDY

We have implemented the exact branch and cut algorithm described in Section I (OPT), the greedy fork-contraction algorithm of [14] (GFC), and three new heuristics:

- A simple edge-switching (ES) heuristic that starts from the MST, and repeatedly replaces a tree edge with a non-tree edge re-establishing connectivity. At every step, the algorithm chooses the pair of edges that results in the largest reduction in power cost; the process is repeated as long as improvement is still possible. We simulated a distributed implementation of the algorithm in which only non-tree edges that connect nodes within 10 tree-hops from each other are considered for switching.
- A heuristic performing both edge and fork switching (EFS). At every step the algorithm chooses the edge or the fork whose addition to the tree leads to the largest reduction in power cost. Unlike GFC, forks are not contracted, which means that an edge in an added fork can later be removed from the tree by other edge or fork switches.
- A Kruskal-like heuristic (KR) that starts with isolated nodes and iteratively adds an edge connecting two different components with *minimum increase* in power cost. A similar heuristic (called incremental search) was studied by Chu and Nikolaidis for computing low-power ASYMMETRIC BROADCAST trees in a mobile environment [2].

We included in our comparison faster versions of OPT and GFC, OPT-D and GFC-D, which speed-up the computation by working on the Delaunay graph defined by the nodes instead of the complete graph. We also implemented a faster version of EFS, EFS-D, in which only forks consisting of Delaunay edges (but still all non-tree edges) are considered as switching candidates.

All algorithms were implemented in C++, including the branch and bound algorithm whose implementation is built on SCIL [21]. The heuristics were compiled using `gpp` with `-O2`

n	OPT	OPT-D	ES	EFS	EFS-D	KR	GFC	GFC-D
10	4.01	3.66	3.81	4.00	3.94	0.49	1.39	1.19
15	4.77	4.26	4.48	4.70	4.51	1.72	1.56	0.48
20	5.84	5.17	5.46	5.75	5.47	2.54	2.01	1.40
25	5.63	4.72	4.78	5.53	5.12	2.19	1.56	0.72
30	5.46	4.90	4.87	5.36	5.03	1.77	1.65	0.24
35	5.68	5.11	5.04	5.60	5.40	2.13	1.93	0.96
40	5.41*	4.82	5.01	5.51	5.25	1.82	1.37	0.26
45	—	5.37	5.13	5.77	5.47	2.17	2.22	0.67
50	—	5.36	5.55	5.90	5.62	2.45	2.03	0.33
55	—	6.09	5.61	6.54	6.21	2.65	2.60	1.19
60	—	5.46*	5.25	6.06	5.73	2.31	2.15	0.50
65	—	—	5.01	5.80	5.56	2.30	1.65	0.38
70	—	—	5.12	6.01	5.60	2.41	1.94	0.24
75	—	—	5.10	5.78	5.50	2.46	1.69	0.48
80	—	—	5.14	6.03	5.77	2.88	2.00	0.64
85	—	—	4.73	5.69	5.37	2.52	1.82	0.39
90	—	—	5.42	6.30	6.01	2.84	2.18	0.38
95	—	—	5.29	6.08	5.81	2.35	1.73	0.19
100	—	—	5.45	6.25	6.09	2.56	2.30	0.99

TABLE I

PERCENT IMPROVEMENT OVER THE MST.

n	OPT	OPT-D	ES	EFS	EFS-D	KR	GFC	GFC-D
10	0.67	0.10	0.00	0.00	0.00	0.00	0.00	0.00
15	5.68	0.43	0.00	0.02	0.00	0.00	0.00	0.00
20	22.2	1.19	0.00	0.10	0.00	0.00	0.00	0.00
25	58.9	3.46	0.00	0.26	0.00	0.00	0.00	0.00
30	201	6.49	0.00	0.61	0.00	0.00	0.00	0.00
35	712	11.2	0.00	1.16	0.02	0.01	0.00	0.00
40	4725*	52.1	0.00	2.13	0.03	0.01	0.00	0.00
45	—	109	0.00	3.71	0.05	0.00	0.03	0.03
50	—	181	0.02	5.50	0.05	0.00	0.02	0.02
55	—	653	0.05	9.03	0.05	0.00	0.03	0.03
60	—	573*	0.05	12.48	0.06	0.00	0.05	0.05
65	—	—	0.05	17.9	0.09	0.04	0.03	0.03
70	—	—	0.03	25.5	0.10	0.04	0.01	0.01
75	—	—	0.02	33.4	0.09	0.02	0.00	0.00
80	—	—	0.05	44.9	0.12	0.00	0.00	0.00
85	—	—	0.06	55.0	0.16	0.00	0.00	0.00
90	—	—	0.09	75.5	0.21	0.00	0.00	0.00
95	—	—	0.11	101	0.26	0.00	0.05	0.05
100	—	—	0.14	123	0.32	0.00	0.05	0.05

TABLE II

RUNTIME (IN CPU SECONDS) FOR THE COMPARED ALGORITHMS.

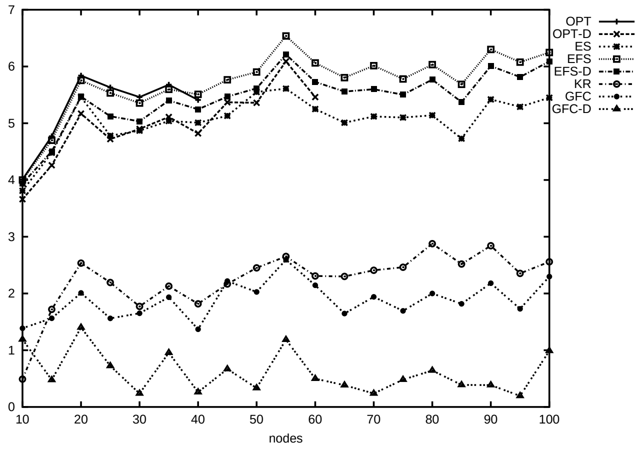


Fig. 5. Average percent improvement over MST for the implemented algorithms.

optimization, and run on an AMD Duron 600MHz PC. The experiments were run on randomly generated testcases. For each instance size n between 10 and 100, in increments of 5, 50 different instances were generated by chosing n points uniformly at random from a grid of size $10,000 \times 10,000$.

In Table I and Figure 5 we report the *percent improvement over MST*, i.e., $100 \times \frac{p(MST)-p(Algo)}{p(MST)}$ for the compared algorithms. Running times are reported in Table II. We report averages over 50 instances of each size; averages marked with an astersik do not include two intances not solved within one day. The results show that OPT has practical running time up to 35 nodes, and produces an average improvement over MST of 5-6%. The Delaunay version of OPT has practical runtime up to 60 nodes, but gives slightly worse solutions.

The provably good GFC algorithm, its faster Delaunay version, GFC-D, as well as the natural Kruskal-like heuristic KR are all very fast, but give less than half of the optimum improvement. In contrast, EFS, EFS-D, and even the distributed ES heuristic, come on the average within a fraction of a percent of the optimal improvement with very well scaling runtime.

REFERENCES

[1] D. Blough, M. Leoncini, G. Resta, and P. Santi, "On the symmetric range assignment problem in wireless ad hoc networks," in *2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*. Kluwer Academic Publishers, 2002, pp. 71–82.

[2] T. Chu and I. Nikolaidis, "Energy efficient broadcast in mobile ad hoc networks," in *Proc. AD-HOC NetwOrks and Wireless*, 2002.

[3] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca, "On the complexity of computing minimum energy consumption broadcast subgraphs," in *Symposium on Theoretical Aspects of Computer Science*, 2001, pp. 121–131. [Online]. Available: citeseer.nj.nec.com/442218.html

[4] A. Clementi, P. Penna, and R. Silvestri, "On the power assignment problem in radio networks," *Electronic Colloquium on Computational Complexity (ECCC)*, no. 054, 2000. [Online]. Available: citeseer.nj.nec.com/clementi00power.html

[5] L. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc, "Power consumption in packet radio networks," *Theoretical Computer Science*, vol. 243, pp. 289–305, 2000.

[6] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. Ravi, "Algorithmic aspects of topology control problems for ad hoc networks," in *Proc. ACM MobiHoc*, 2002, pp. 123–134.

[7] R. Ramanathan and R. Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proc. IEEE INFOCOM*, 2000, pp. 404–413. [Online]. Available: citeseer.nj.nec.com/ramanathan00topology.html

[8] S. Singh, C. Raghavendra, and J. Stepanek, "Power-aware broadcasting in mobile ad hoc networks," in *Proceedings of IEEE PIMRC*, 1999.

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

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

[11] T. Rappaport, *Wireless Communications: Principles and Practices*. Prentice Hall, 1996.

[12] A. Clementi, G. Huiban, P. Penna, G. Rossi, and Y. Verhoeven, "Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks," in *Proc. 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE)*, 2002.

[13] A. Tanembaum, *Computer Networks (3rd edition)*. Prentice Hall, 1996.

[14] G. Călinescu, I. Măndoiu, and A. Zelikovsky, "Symmetric connectivity with minimum power consumption in radio networks," in *2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*. Kluwer Academic Publishers, 2002, pp. 119–130.

[15] A. Zelikovsky, "Better approximation bounds for the network and Euclidean Steiner tree problems," Department of Computer Science, University of Virginia, Tech. Rep. CS-96-06, 1996.

[16] ——, "An 11/6-approximation algorithm for the network Steiner problem," *Algorithmica*, vol. 9, pp. 463–470, 1993.

[17] G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *Proceedings of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms*, 2000, pp. 770–779.

[18] M. Padberg and L. Wolsey, "Trees and cuts," *Anals of Discrete Mathematics*, vol. 17, pp. 511–517, 1983.

[19] L. Lovász and M. Plummer, *Matching theory*. Amsterdam–New York: North-Holland, 1986.

[20] U. Feige, "A treshold of $\ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, pp. 634–652, 1998.

[21] "SCIL–Symbolic Constraints for Integer Linear programming," www.mpi-sb.mpg.de/SCIL.