POWER OPTIMIZATION IN AD HOC WIRELESS NETWORK TOPOLOGY CONTROL WITH BICONNECTIVITY REQUIREMENTS

RENATO E. N. MORAES AND CELSO C. RIBEIRO

Abstract. We consider the problem of assigning transmission powers to the nodes of an ad hoc wireless network, so as that the total power consumed is minimized and the resulting network is biconnected, i.e., there are at least two node-disjoint paths between any pair of nodes. Biconnected communication graphs are important to ensure fault tolerance, since ad hoc networks are used in critical application domains where failures are likely to occur. A mixed integer programming formulation of the problem can be exactly solved to optimality by a commercial solver only for moderately sized problems. We recall a mixed integer programming formulation that can be exactly solved to optimality by a commercial solver only for very moderately sized problems. We propose a quick greedy algorithm and a GRASP with path-relinking heuristic for solving real-life sized problems. Computational experiments involving practical issues such as energy consumption and interference have been performed and reported for problems with up to 800 nodes, illustrating the effectiveness and the efficiency of the new algorithms. Both the greedy algorithm and the GRASP heuristic outperformed the best heuristic in the literature for very large problem sizes.

1. Introduction

Ad hoc networks consist of a collection of transceivers, in which a packet may have to traverse multiple consecutive wireless links to reach its destination. They have become increasingly common due to their large number of applications. They face a variety of constraints that do not appear in wired networks. Nodes in a wireless network are typically battery-powered, and it is expensive and sometimes even infeasible to recharge the device. We focus on radio power consumption, since radios tend to be the major source of power dissipation in wireless networks (Xing et al., 2007).

There are also increasing fault-tolerance requirements, due to the evolving critical applications and to the large number of failures that may result from mobility, fading, or obstructions. A connected graph is usually assumed as the minimum connectivity requirement by the algorithms running in different layers of the network. However, if there is only one path between a pair of nodes, the failure of a single node or link between them will result in a disconnected graph. Topologies with alternative disjoint paths between any pair of nodes are often required (Marina and Das, 2001).

Date: February 3rd, 2011.

Key words and phrases. Wireless networks, ad hoc networks, topology control, biconnectivity, fault tolerance, energy consumption optimization, GRASP, path-relinking, mixed integer programming.

Ad hoc networks can be represented by a set V of transceivers (nodes) numbered from 0 to $|V|-1$, together with their locations. A transmission power p_u is assigned to each node $u \in V$. For each ordered pair (u, v) of transceivers, with $u, v \in V$, there is a non-negative arc weight $e(u, v)$ such that a signal transmitted by transceiver u can be received at node v if and only if $p_u \geq e(u, v)$. Each node can adjust its transmitting power, based on the distances to the receiving nodes and on the background noise. In the most common power attenuation model (Rappaport, 2001), the signal power falls with $1/d^\varepsilon$, where d is the distance from the transmitter and ε is the path loss exponent (typically between 2 and 4). The power requirement at node u for supporting transmission through a link from u to v is given by $e(u, v) = d_{uv}^\varepsilon \cdot q_v$, where d_{uv} is the Euclidean distance between the transmitter u and the receiver v, and q_v is the receiver's power threshold for signal detection (usually normalized to 1).

This model holds only for free-space environments with non-obstructed lines of sight. In practice, power requirement values for two nodes u and v may be asymmetric, because of batteries with different power levels, heterogeneous nodes, and different ambient noise levels in the two regions. Therefore, in the general *asymmetric input* version of the problem there may be pairs of transceivers $u, v \in V$ for which $e(u, v) \neq e(v, u)$.

In a *bidirectional topology*, communication between nodes u and v is enabled whenever $p_u \geq e(u, v)$ and $p_v \geq e(v, u)$. The edge $[u, v]$ is used as a communication link to enforce biconnectedness if v is within the transmission range of u and vice-versa. The *transmission graph* associated with a power assignment $p = \{p_u : u \in V\}$ is defined as the undirected graph $G(p) = (V, B(p))$, where $B(p) = \{[u, v] : u \in V, v \in V, p_u \geq e(u, v), p_v \geq e(v, u)\}$.

Given the node set V and arc weights $e(u, v)$ for any $u, v \in V$, the *bidirectional biconnected minimum power consumption problem* consists in finding an optimal assignment of transmission powers p_u to every node $u \in V$, such that the total power consumption $\sum_{u \in V} p_u$ is minimized and the resulting transmission graph is biconnected. It was proved to be NP-hard by Calinescu and Wan (2006), who also described a 4-approximation algorithm. Among other results, Lloyd et al. (2005) presented a $2(2 - 2/n)(2 + 1/n)$-approximation algorithm and computational results with experimental and more realistic networks. Taghi et al. (2007) obtained an $O(k)$-approximation algorithm for the general problem version for k-connectivity. For the particular case in which biconnectivity is sought, they implemented centralized and distributed algorithms for its solution and compared their experimental results.

The transmission power assignments obtained with approximation algorithms (Calinescu and Wan, 2006; Lloyd et al., 2005; Taghi et al., 2007) based on submodular flow algorithms (Frank and Tardos, 1989; Khuller and Raghavachari, 1996; Kortsarz and Nutov, 2000) have constant approximation ratios. However, they run in time $O(n^2 m)$ in networks with n vertices and m edges (Gabow, 1993). Furthermore, they have very complicated implementations and are not practical for wireless ad hoc networks (Calinescu and Wan, 2006). We propose in this paper new algorithms based on the approach of expanding an spanning tree. These new algorithms are compared with our implementation of the $O(n \log n)$ MST-aug algorithm originally presented by Calinescu and Wan (2006) which also produces a biconnected graph by augmenting a spanning tree and and also achieves a constant approximation ratio.

A mixed integer programming formulation to solve problems in moderately sized networks is recalled in Section 2. A GRASP with path-relinking heuristic to approximately solve large problem instances is proposed in Section 3. Computational experiments are presented and numerical results involving practical issues such as energy consumption and interference are discussed in Section 4. Concluding remarks are made in the last section.

2. Integer Programming Model by Incremental Powers

Moraes et al. (2009) proposed and compared three integer programming formulations for the four variants of the k-connected minimum power consumption problem, regarding the topologies of the input graph (symmetric or asymmetric) and of the solution (unidirectional or bidirectional). We recall below the best mixed integer programming multicommodity flow formulation for the biconnected minimum power consumption problem for the variant with asymmetric input graphs and bidirectional solutions.

To formulate the k-connected minimum power consumption problem, we first define a set C of commodities. Raghavan (1995) has shown, in the context of the network design problem with connectivity requirements (Magnanti and Raghavan, 2005), that a more compact model can be formulated using a k-connected undirected requirement graph $G^k = (V, E^k)$ with a minimum number $|E^k| = \lceil k|V|/2 \rceil$ of edges (Harary, 1962) built as follows:

- If k is even, there is an edge $[i, j]$ in E^k for $i, j \in V$ whenever $(i - j) \mod |V| \leq k/2$.
- If k is odd and $|V|$ is even, first build graph G^{k-1}. Next, obtain E^k from E^{k-1} by incorporating edges $[i, i + |V|/2]$ for $i = 0, \dots, |V|/2$.
- Otherwise, build graph G^{k-1} and obtain E^k from E^{k-1} by adding incorporating edges $[0, (|V| - 1)/2)]$, $[0, (|V| + 1)/2]$, and $[i, i + (|V| + 1)/2]$ for $i = 1, \dots, (|V| - 1)/2$.

The set C of commodities is built as follows. Let $[i, j]$ be any edge in E^k, create k commodities between nodes i and j with an unit demand, arbitrarily choosing any of them as the origin and the other as the destination. This procedure entails a multicommodity flow model for the k-connected minimum power consumption problem with bidirectional topology using $\lceil |V|/2 \rceil$ commodities.

For each commodity $c \in C$, let $o(c)$ be its origin and $d(c)$ its destination. For any node $i \in V$ and any commodity $c \in C$, let $D_c(i) = -k$ if $i = o(c)$, $D_c(i) = +k$ if $i = d(c)$, $D_c(i) = 0$ otherwise. Since we are solving the biconnected minimum power consumption problem, the value of k is fixed to 2. The discrete variable f^c_{ij} and the continuous variable p_i represent, respectively, the flow of commodity c through arc (i, j) and the power assignment to node i. The binary variable f^c_{ij} is equal to one if arc (i, j) is used by commodity c for communication from node i to j, zero otherwise.

Let $P_i = [p^1_i, \dots, p^{\phi(i)}_i]$ be a finite list of efficient increasing power levels that can be assigned to node $i \in V$, where p^1_i is the minimum power assignment such that transmissions from node i reach at least one node in $V \setminus \{i\}$ and $p^{\ell+1}_i > p^\ell_i$, for any $\ell = 1, \dots, \phi(i) - 1$. By efficient, we mean that for any power assignment $p_i \in [p^\ell_i, p^{\ell+1}_i), \ell = 1, \dots, \phi(i) - 1$, the nodes reachable from i are the same reachable with the power level $p_i = p^\ell_i$, while taking $p_i = p^{\ell+1}_i$ reaches at least one additional node. For ease and completeness of notation, we assume that $p^0_i = 0$. Furthermore, we denote by $T^\ell_i \neq \emptyset$ the set of new nodes reachable from node i when its power assignment increases from $p^{\ell-1}_i$ to p^ℓ_i, for any $\ell = 1, \dots, \phi(i)$. We also define the increment list $Q_i = [q_i^1, \ldots, q_i^{\phi(i)}]$ such that $q_i^1 = p_i^1$ and $q_i^\ell = p_i^\ell - p_i^{\ell-1}$ for any $\ell = 2, \ldots, \phi(i)$; see Figure 1 for an example. The binary variable x_i^ℓ takes the value one if there is a node $j \in T_i^\ell$ such that (i,j) is used for communication from i to j, zero otherwise.

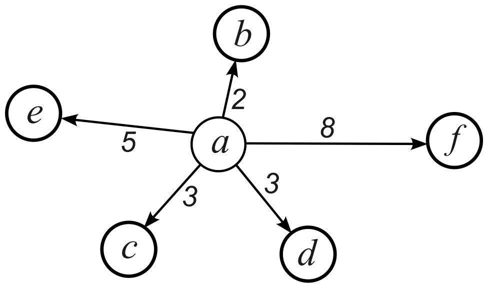


FIGURE 1. Example: $P_a = [2,3,5,8]$, $Q_a = [2,1,2,3]$ and $T_a^1 = \{b\}$, $T_a^2 = \{c,d\}$, $T_a^3 = \{e\}$, $T_a^4 = \{f\}$.

Since the transmission graph $G(p)$ is required to be biconnected, each node must be able to communicate with at least two other nodes. Therefore, we denote by $p_i^{\bar{\ell}}$ the minimum power level such that transmissions from node i reach at least two nodes in $V \setminus \{i\}$.

The mixed integer program defined by the objective function (1) and constraints (2)-(8) below is a valid formulation for the asymmetric input with bidirectional topology version of the biconnected minimum power consumption problem:

$$\min \sum_{i \in V} \sum_{\ell=1}^{\phi(i)} q_i^\ell \cdot x_i^\ell \tag{1}$$

subject to:

$$\sum_{j \in V} f_{ji}^c - \sum_{l \in V} f_{il}^c = D_c(i), \qquad \forall c \in C, \forall i \in V \tag{2}$$

$$\sum_{j \in V} f_{ij}^c \leq 1, \qquad \forall c \in C, \forall i \in V : i \neq o(c), i \neq d(c) \tag{3}$$

$$x_i^\ell \geq f_{ij}^c + f_{ji}^c, \qquad \forall i \in V, \forall c \in C, \forall j \in T_i^\ell, \ell = 1, \ldots, \phi(i). \tag{4}$$

$$x_i^{\ell+1} \leq x_i^\ell, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i) - 1 \tag{5}$$

$$x_i^\ell = 1, \qquad \forall i \in V, \ell = 1, \ldots, \bar{\ell}(i) \tag{6}$$

$$f_{ij}^c \in \{0,1\}, \qquad \forall i,j \in V, \forall c \in C \tag{7}$$

$$x_i^\ell \in \{0,1\}, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i). \tag{8}$$

Constraints (2) are the flow conservation equations. Inequalities (3) ensure node-disjointness. Inequalities (4) state that x_i^ℓ must be set to one if there is a node $j \in T_i^\ell$ such that arc (i,j) or (j,i) is used for communication from node i to j (or from node j to i) by commodity c. Constraints (5) enforce $x_i^{\ell+1}$ to be equal to zero if the previous increment was not used, i.e. if $x_i^\ell = 0$. Constraints (6) set to one the power increments that are necessary to reach at least the two closest nodes to each node i. Constraints (7) and (8) express the integrality requirements.

3. GRASP with Path-relinking Heuristic

A greedy randomized adaptive search procedure (GRASP) (Feo and Resende, 1995; Resende and Ribeiro, 2003b) is a multi-start process. Each of its iterations consists of two phases: construction and local search. The construction phase builds a feasible solution. The local search phase investigates its neighborhood until a local minimum is found. The best overall solution is returned. The best overall solution is kept as the result (Resende and Ribeiro; 2010; 2003a; 2005)). In spite of its simplicity and ease of implementation, GRASP is a very effective metaheuristic and produces the best known solutions for many problems (Festa and Resende (2002; 2009a;b)).

In the construction phase, a feasible solution is iteratively constructed, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list L with respect to a real-valued greedy function $g(.)$. This function measures the benefit of selecting each element. In a purely greedy implementation, the top candidate is always selected. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL).

It is almost always beneficial to apply local search as an attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution within its neighborhood. It terminates when no better solution is found in the neighborhood.

In the remainder of this section, we customize a GRASP with path-relinking heuristic for the asymmetric input with bidirectional topology version of the biconnected minimum power consumption problem.

3.1. Construction Phase.

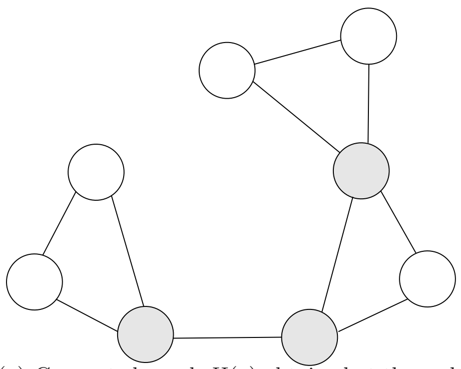
Solution algorithms for the bidirectional biconnected minimum power consumption problem have been developed by Calinescu and Wan (2006), Lloyd et al. (2005), and Taghi et al. (2007). They gave approximation bounds derived from algorithms based on that proposed by Frank and Tardos (1989), which has $O(n^2m)$ time complexity (Gabow, 1993). Calinescu and Wan (2006) also described a 4-approximation algorithm with $O(n \log n)$ time complexity, which produces a biconnected graph by augmenting a spanning tree.

The first stage of our construction phase builds a bidirectional connected graph one node at a time. Given the node set V and non-negative arc weights $e(u,v)$ for any $u, v \in V$, the algorithm sets $p_u = 0$ for all $u \in V$, and initializes a working graph $H(p) = (V', E(p))$ with $V' = \{r\}$ and $E(p) = \{[u,v] : u \in V', v \in V', p_u \geq e(u,v), p_v \geq e(v,u)\} = \emptyset$, where $r \in V$ is any randomly selected initial node. The greedy function that guides the construction is based on the wireless multicast property (Wieselthier et al., 2000): if p_u is the current power assignment to node u and there is a node v such that $e(u,v) > p_u$, then the incremental power required to set up communication from u to v is $e(u,v) - p_u$. Therefore, the greedy cost function is $g(u,v) = \max\{0, e(u,v) - p_u\} + \max\{0, e(v,u) - p_v\}$, for any $u, v \in V$. If $g(u,v) = 0$, then the bidirectional communication between u and v is already set up.

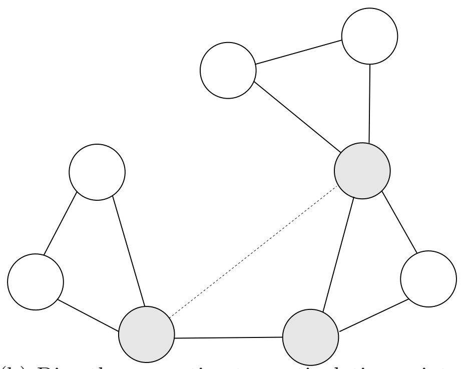
For every node $u \notin V'$, let $g(u) = \min_{v \in V'}\{g(u,v)\}$ be the minimum power increment to connect it to a node in V'. Let $\underline{g} = \min_{u \in V \setminus V'}\{g(u)\}$ and $\overline{g} =$

$\max_{u \in V \setminus V'}\{g(u)\}$ be, respectively, the minimum and maximum power increments over all candidate nodes (i.e., those not in the current solution). The restricted candidate list RCL is formed by all nodes $u \in V \setminus V'$ such that $g(u) \leq \underline{g} + \alpha(\overline{g} - \underline{g})$, with $0 \leq \alpha \leq 1$. The case $\alpha = 0$ corresponds to a pure greedy algorithm, while $\alpha = 1$ is equivalent to a random construction. A node u is randomly selected from RCL and inserted into V'. The power assignments of the nodes $u \in V \setminus V'$ and $v \in V'$ such that $g(u) = g(u, v)$ are increased by $\max\{0, e(u, v) - p_u\}$ and $\max\{0, e(v, u) - p_v\}$, respectively. Consequently, the bidirectional edge $[u, v]$ is inserted into $E(p)$. This stage finishes when $V' = V$, ensuring that a connected graph $H(p) = (V, E(p))$ is obtained.

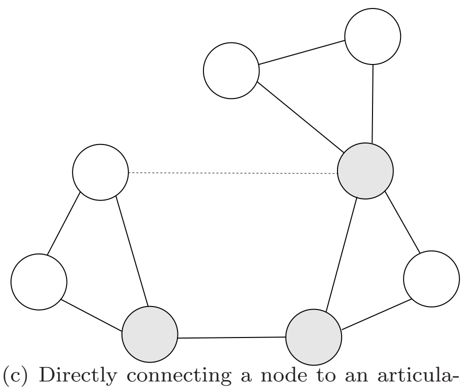
The second construction stage produces a biconnected graph $G(p) = (V, B(p))$, as illustrated in Figure 2. Basically, new edges connecting nodes that are not articulation points of the current solution are directly connected by the algorithm, progressively reducing the number of biconnected components until a biconnected graph is obtained. The power assignments are initialized with the values obtained in the first stage. Consequently, the edge set is initialized as $B(p) = E(p)$. Tarjan's algorithm (Tarjan, 1972) is used to compute the biconnected components and the



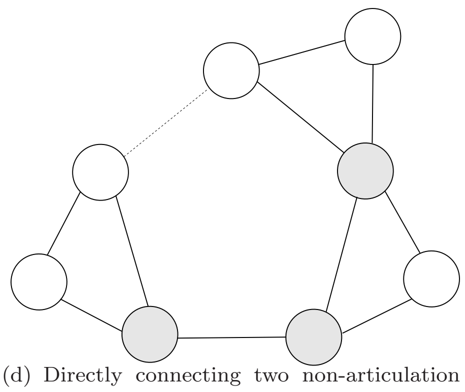
(a) Connected graph $H(p)$ obtained at the end of the first construction stage, with three articulation points (in shadow) and three biconnected components.



(b) Directly connecting two articulation points gives a weak contribution to biconnectivity, since both remain articulation points.



(c) Directly connecting a node to an articulation point in another biconnected component gives a weak contribution to biconnectivity, since the latter remains an articulation point.



(d) Directly connecting two non-articulation points from different biconnected components gives a strong contribution to biconnectivity: this principle is explored in the second stage.

FIGURE 2. Second stage of the construction phase.

articulation points of the current solution. A node $u \in V$ is an articulation point of $G(p)$ if it belongs to more than one of its biconnected components. For every node $u \in V$ that is not an articulation point of the current solution, let $g'(u) = \min_{v \in V}\{g(u,v) : u \neq v,$ node v is not an articulation point and does not belong to the same component as $u\}$ be the minimum power increment necessary to connect it to a node in a different biconnected component which is not an articulation point. Let $\underline{g}' = \min_{u \in V}\{g'(u) : u$ is not an articulation point$\}$ and $\overline{g}' = \max_{u \in V}\{g'(u) : u$ is not an articulation point$\}$ be, respectively, the minimum and maximum power increments over all nodes which are not articulation points.

The second stage restricted candidate list RCL$'$ is formed by all nodes $u \in V$ which are not articulation points and such that $g'(u) \leq \underline{g}' + \alpha(\overline{g}' - \underline{g}')$, with $0 \leq \alpha \leq 1$. A node u which is not an articulation point is randomly selected from RCL$'$, with $g'(u) = g(u,v)$ for some node v which is not an articulation point. The power assignments of nodes u and v are increased by $\max\{0, e(u,v) - p_u\}$ and $\max\{0, e(v,u) - p_v\}$, respectively. Consequently, the bidirectional edge $[u,v]$ is inserted into $B(p)$ and a new iteration resumes. Since linking any two biconnected components by an edge reduces their number at least by one, the algorithm stops when a biconnected graph is built.

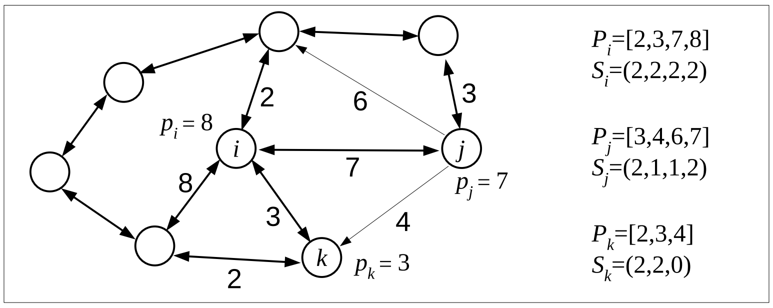
3.2. **Local Search Phase.** $P_i = [p_i^1, \ldots, p_i^{\phi(i)}]$ was defined in Section 2 as a list formed by the only efficient increasing power levels that can be assigned to each node $i \in V$. For a given power assignment $p = \{p_i : i \in V\}$, let $S_i = (s_i^1, \ldots, s_i^{\phi(i)})$ be a vector with components $s_i^\ell \in \{0,1,2\}$, for $\ell = 1, \ldots, \phi(i)$ and for each $i \in V$ such that (see Figure 3(a)):

- $s_i^\ell = 0$ if $p_i^\ell > p_i$ (node i operates with a power assignment smaller than p_i^ℓ);
- $s_i^\ell = 2$ if $p_i^\ell \leq p_i$ and there exist a node $j \in T_i^\ell$ and a level $k = 1, \ldots, \phi(j)$ such that $p_j \geq p_j^k$ and $i \in T_j^k$ (power level p_i^ℓ supports a bidirectional edge with node j); and
- $s_i^\ell = 1$ otherwise (power level p_i^ℓ is used, but only a unidirectional arc from i to j is established).

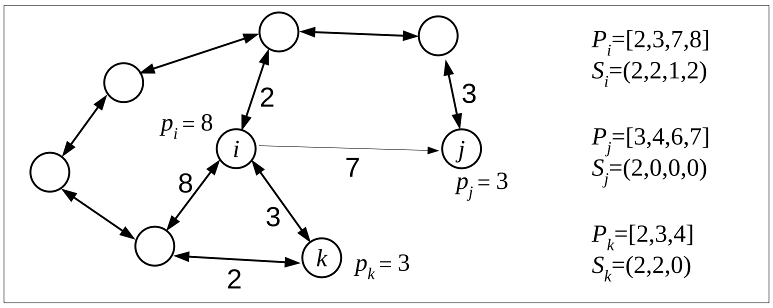
Local search and the definition of the neighborhoods make use of two basic operations for decreasing and increasing the power assignments. Applied to a node $i \in V$ (see Figure 3(b)), the first operation decreases its current power assignment $p_i = p_i^\ell$ (with $\ell \geq 2$) to $p_i = p_i^{\ell'}$, where ℓ' is the highest level which supports a bidirectional edge: $1 \leq \ell' < \ell$, $s_i^{\ell'} = 2$, and $s_i^{\ell''} = 1$ for all $\ell'' = \ell'+1, \ldots, \ell-1$. It removes the links (arcs and edges) between nodes i and j for all $j \in T_i^{\ell'+1} \cup \ldots \cup T_i^{\ell-1} \cup T_i^\ell$ and the total power assignment is decreased by $p_i^\ell - p_i^{\ell'}$.

Applied to a node $i \in V$ (see Figure 3(c)), the second operation increases its current power $p_i = p_i^\ell$ (with $\ell \leq \phi(i) - 1$) to $p_i = p_i^{\ell+1}$. If there exist a node $j \in T_i^{\ell+1}$ and a power level $k = 1, \ldots, \phi(j)$ such that $p_j \geq p_j^k$ and $i \in T_j^k$, then the objective function is increased by $p_i^{\ell+1} - p_i^\ell$. Otherwise, let $j \in T_i^{\ell+1}$ such that $p_j - p_j^{k(j)} = \min_{v \in T_i^{\ell+1}}\{p_v - p_v^{k(v)} : i \in T_v^{k(v)}$, for some $k(v) = 1, \ldots, \phi(v)\}$. In this case, the operation increases the current power p_j to $p_j^{k(j)}$ and the objective function is increased by $(p_i^{\ell+1} - p_i^\ell) + (p_j^{k(j)} - p_j)$. The power increase operation ensures the insertion of the bidirectional edge $[i,j]$.

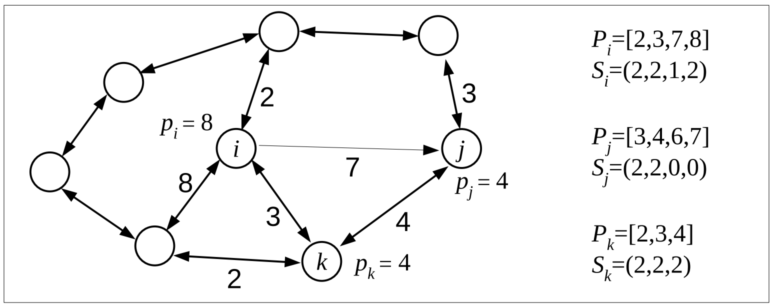
The local search phase explores the neighborhood of the current solution, attempting to reduce the total power consumption. A move starts (see Figure 3) by



(a) Current biconnected solution with $p_i + p_j + p_k = 18$.



(b) A power decrease in node j (from 7 to 3) generates an infeasible solution.



(c) A power increase in nodes j and k (from 3 to 4) restores feasibility and creates a better solution with $p_i + p_j + p_k = 16$.

FIGURE 3. Example of a complete local search move.

decreasing the power assignment of as many nodes as needed to break biconnectivity, followed by a sequence of as many power increases as necessary to restore biconnectivity applied only to nodes not affected by previous power decrease operations. Decrease operations are performed in non-increasing order of power decrease (i.e., start by largest power decrease). Increase operations are performed in non-decreasing order of power increase (i.e., start by smallest power increase). The first improving move is accepted and the search moves to the new neighbor. The procedure continues until no further improving moves exist.

Given the current solution $G(p) = (V, B(p))$, there are $O(|V|)$ possible power decrease operations and $O(|V|)$ possible power increase operations. Since the procedure to test feasibility runs in time $O(|V| + |B(p)|)$ with $|B(p)| = O(|V|^2)$, then the neighborhood of a single solution can be searched in time $O(|V|^4)$.

The number of power increase operations investigated may be reduced to speedup the local search. A candidate list is built, with its nodes sorted by the corresponding increase in the objective function (after the application of the decreasing operation). Whenever biconnectivity is destroyed by a power decrease operation, the biconnected components are computed and two acceleration schemes are implemented:

(1) the *reduced scheme* restricts the power increase operations to pairs of nodes belonging to the same biconnected components of the pair of nodes affected by the previous decrease; and
(2) the *extended scheme* considers power increase operations involving any pair of nodes from different biconnected components.

Three local search procedures are implemented, depending on the acceleration scheme used:

- *reduced local search* uses the reduced scheme;
- *extended local search* uses the extended scheme; and
- *mixed local search* first uses the reduced scheme until no further improving moves can be found, followed by the extended scheme.

3.3. **Path-Relinking.** The GRASP heuristic may be enhanced by path-relinking (Glover et al., 2000; Resende et al., 2010). Path-relinking is a very successful intensification strategy to explore trajectories connecting elite solutions obtained by the basic GRASP procedure. Path-relinking is usually carried out between two solutions: one is called the *initial solution*, while the other is the *guiding solution*. One or more paths in the solution space graph connecting these solutions are explored in the search for better solutions.

To hybridize path-relinking with the GRASP procedure, one usually makes use of an *elite set*, i.e. a diverse pool of high-quality solutions found during the search. The elite set starts empty and is limited in size by a number `Max_Elite`. Each locally optimal solution produced by a GRASP iteration is relinked with one or more solutions from the elite set. Each solution produced by path-relinking is a candidate for inclusion in the elite set, where it can replace an elite solution of worse value.

Given a pair of solutions $(p^{(1)}, p^{(2)})$, the algorithm starts by computing the set $\Delta(p^{(1)}, p^{(2)})$ of moves which should be applied to one of them (the initial solution) to reach the other (the guiding solution). One move from $\Delta(p^{(1)}, p^{(2)})$ still not performed is randomly selected to produce the next step in the path, until the guiding solution is attained. A move is defined as the difference between the power level of node i in the initial solution and the power level of node i in the guiding solution, for any $i \in V$. The randomized move selection strategy is very instrumental to obtain diversity along path-relinking, avoiding that infeasible solutions be often obtained.

Path-relinking is applied at every GRASP iteration using a backward strategy (Ribeiro et al., 2002), which usually outperforms other approaches (Resende et al., 2010). Suppose that path-relinking is be applied to a minimization problem between solutions $p^{(1)}$ and $p^{(2)}$ such that $f(p^{(1)}) \leq f(p^{(2)})$, where $f(\cdot)$ denotes the objective function. In backward path-relinking, the initial and guiding solutions are set to $p^{(1)}$ and $p^{(2)}$, respectively.

```
Require: Node set $V$, weights $e(u,v) : \forall u,v \in V$, $Max\_Iterations$ and $Seed$.
Ensure: Best known solution $p^*$
1:  $f^* \leftarrow \infty$;
2:  $Elite\_Set \leftarrow \emptyset$;
3:  for $iteration = 1,\ldots,Max\_Iterations$ do
4:    $p \leftarrow$ Greedy_Randomized_Construction($Seed$);
5:    repeat
6:      $p \leftarrow$ Reduced_Local_Search($p$);
7:      if no improvement then
8:        $p \leftarrow$ Extended_Local_Search($p$);
9:      end if
10:   until no improvement
11:   if $Elite\_Set \neq \emptyset$ then
12:     $p \leftarrow$ Path_Relinking($p$,Elite_Set);
13:   end if
14:   Update_EliteSet($p$,Elite_Set);
15:   if $f(p) < f^*$ then
16:     $p^* \leftarrow p$;
17:     $f^* \leftarrow f(p)$;
18:   end if
19: end for
20: return $p^*$;
```

Algorithm 1: Pseudo-code of the GRASP with path-relinking heuristic.

The algorithm GRASP with path-relinking makes use of an *elite set*, i.e. a diverse pool of high-quality solutions found during the search. The elite set starts empty and is limited in size. Each locally optimal solution produced by the local search procedure is relinked with one randomly selected solution from the elite set. Each solution produced by path-relinking is a candidate for inclusion in the elite set where it can replace an elite solution of worse value.

Algorithm 1 shows the pseudo-code of the GRASP with path-relinking heuristic using the mixed local search procedure for the biconnected minimum power consumption problem for asymmetric input graphs and bidirectional solutions. In line 1 the objective function value is initialized. The pool of elite solutions is initially empty (line 2). Each iteration of the loop in lines 3 to 19 finds a new solution to the problem, until the maximum number of iterations is reached. The procedure in line 4 finds a greedy randomized solution which is submitted to the local search procedure in lines 5 to 10. The mixed local search applies the reduced scheme until no improvement is made, followed by the application of the extended scheme. If a better solution is found with the extended scheme, then its neighborhood is explored again by reduced local search, until no improvement is made. Path-relinking is applied in line 12 and the elite set is updated in line 14. If the solution found by path-relinking improves upon the best previously found solution, then the best solution and its value are updated in lines 16 and 17, respectively. The best power assignment p^* is returned in line 20.

4. Computational Results

Computational experiments have been carried out on two classes of randomly generated asymmetric test problems with 10 to 800 nodes. For each problem size and type, 15 test instances have been generated.

- Euclidean instances: the nodes are uniformly distributed in the unit square grid. The weight of the arc between nodes $u, v \in V$ is $e(u, v) = F \cdot d^{\varepsilon}_{u,v}$, where $d_{u,v}$ is the Euclidean distance between nodes u and v, the loss exponent ε is set at 2, and $F \in [0.8, 1.2]$ is a random perturbation generated from a uniform distribution.
- Random instances: the weight $e(u, v)$ of the arc between nodes $u, v \in V$ is randomly generated in $(0, 1]$.

An Intel Core 2 Quad machine with a 2.40 GHz clock and 8 Gbytes of RAM memory running under GNU/Linux 2.6.24 was used in all experiments. CPLEX 11.0 was used as the integer programming solver.

4.1. **Optimal Solutions.** For each problem type and size $|V| = 10, 15, 20, 25, 30$, Table 1 shows the number of instances exactly solved to optimality by CPLEX in less than three hours, the average running time in seconds over the instances exactly solved, and the average relative linear relaxation gap in percent between the linear relaxation value and the optimal value. Since CPLEX did not solve all instances with $|V| = 30$ in three hours, the numbers in Table 1 are average results over all instances solved to optimality. These results show that the minimum power consumption problem is hard to solve, as already established by Moraes et al. (2009).

Table 1. Exact optimal solutions.

	Euclidean instances			Random instances		
$\|V\|$	solved	time (s)	gap (%)	solved	time (s)	gap (%)
10	15	0.47	7.51	15	0.48	5.98
15	15	7.55	10.34	15	6.99	10.83
20	15	66.61	8.10	15	117.36	10.87
25	15	298.53	7.71	15	872.44	13.48
30	12	1351.98	4.56	1	5559.86	13.55

The linear relaxation gaps are not small, which makes it difficult to the solver to find exact optimal solutions within the imposed time limits. Since the computation times increase very fast with $|V|$, CPLEX could not solve to optimality in three hours of computations even moderately-sized networks with 30 nodes. Random instances seem to be harder to solve than the Euclidean ones. The difficulties faced by CPLEX to solve large instances support the need for efficient heuristics, capable of finding good approximate solutions for large size problems in reasonable computation times.

4.2. **Heuristic Solutions.** The heuristics were coded in C++ and compiled with the GNU g++ compiler version 4.1, using the optimization flag -O2. We considered four GRASP variants using different local search procedures and path-relinking strategies, as proposed in Section 3:

(1) GRASP-R uses the reduced local search,
(2) GRASP-X uses the extended local search,
(3) GRASP-M uses the mixed local search, and
(4) GRASP-Mpr uses the mixed local search with path-relinking (see Algorithm 1).

We evaluate the effectiveness of the GRASP variants in terms of the tradeoffs between computation time and solution quality. Parameter α was set by using the reactive strategy described by Prais and Ribeiro (2000) with the probability distribution being updated after every 100 iterations. The size of the elite sets handled by path relinking is limited to five.

Table 2 illustrates the very small computation times (in seconds) observed for the GRASP heuristics on 15 instances with 25 nodes. We also notice that all GRASP heuristics found the optimal solutions for all such instances. They found the optimal solutions for the Euclidean instances in less than one second, but the random instances were harder to be solved and took much longer. Variant GRASP-Mpr was the fastest for all instances. The fact that the GRASP heuristics found the optimal solutions for small problems in a few seconds is a strong indication that they are robust and can be considered as good strategies to find approximate solutions for large problems that cannot be tackled by exact methods.

Table 2. GRASP average times in seconds for instances with 25 nodes.

Algorithm	Euclidean instances	Random instances
GRASP-R	0.04694	17.65470
GRASP-X	0.05840	10.45372
GRASP-M	0.14988	10.16597
GRASP-Mpr	0.02107	1.57930

For the instances with 100, 200, and 400 nodes, Table 3 displays the average objective values over five runs for one instance of each type as the running time limit increases from five to 625 seconds. All variants of the heuristic continue to improve their solutions as the time limit increases. Variant GRASP-Mpr found the best average solution values in most of the situations, as depicted in bold in Table 3. Figure 4 illustrates the behavior of each algorithm for one run and one instance with $|V| = 200$ nodes of each type as the running time increases up to 3125 seconds, showing that better locally optimal solutions are continuously found.

We also compared the four GRASP variants on two selected instances with $|V| = 25$ and $|V| = 100$ using the methodology proposed by Aiex et al. (2002; 2007). Two hundred independent runs have been performed for each algorithm and for each instance. Each run was terminated when a solution with value less than or equal to a given target was found. We use the optimal solution value as the target for $|V| = 25$, while for $|V| = 100$ the target is taken as a sub-optimal value chosen such that at least one run of the slowest variant could terminate in 15 minutes of computation time. The empirical probability distributions of the time observed to find a solution value less than or equal to the target are plotted in Figures 5 and 6. To plot the empirical distribution for each algorithm, we associate a probability $p_i = (i - \frac{1}{2})/200$ with the i-th smallest running time t_i and we plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 200$.

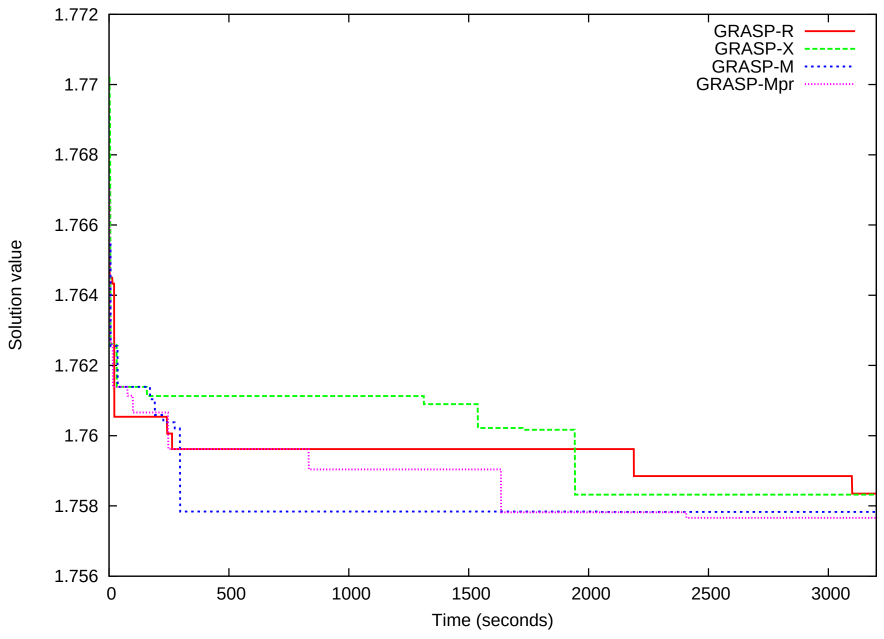
Figures 5(a) and 5(b) display the results for the two instances with $|V| = 25$, using their optimal value as targets. These figures show that, for all algorithms and both instances, the probability of finding a solution as good as the target in less than ten seconds is 100%. For these relatively easy targets, algorithm GRASP-R is more efficient than the others because the size of its neighborhood is the smallest, leading to a faster local search procedure. However, the small neighborhood sizes

Table 3. Average total power consumption for instances with 100, 200 and 400 nodes.

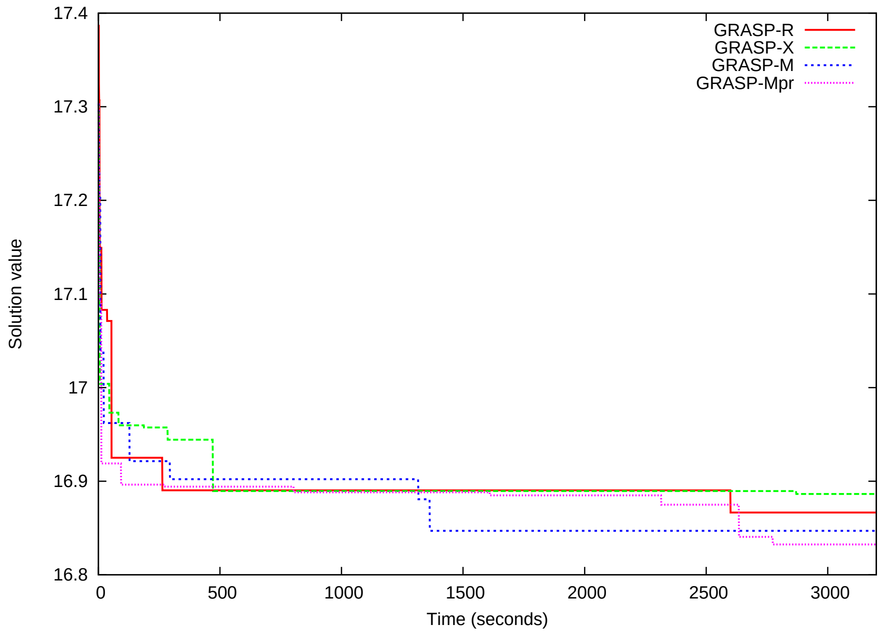
		Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
Euclidean instances	$\|V\| = 100$	25	**1.28578**	1.28905	1.28684	1.28708
		125	1.28552	1.28730	1.28520	**1.28441**
		625	1.28419	1.28556	1.28399	**1.28340**
		3125	**1.28303**	1.28431	**1.28303**	**1.28303**
	$\|V\| = 200$	Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
		25	1.76214	1.76272	1.76206	**1.76089**
		125	1.76072	1.76180	1.76081	**1.76014**
		625	1.75985	1.76047	**1.75893**	1.75921
		3125	1.75840	1.75853	1.75819	**1.75766**
	$\|V\| = 400$	Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
		25	2.85388	2.85282	**2.85239**	2.85240
		125	2.85053	2.84981	**2.84868**	2.84987
		625	2.84811	2.84715	**2.84736**	**2.84736**
		3125	2.84779	2.84682	2.84667	2.84653
Random instances	$\|V\| = 100$	Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
		25	11.12841	11.08156	11.08692	**11.05270**
		125	11.01791	10.94067	10.98430	**10.91835**
		625	10.96228	10.89523	10.92852	**10.87664**
		3125	10.92757	10.87511	10.86481	**10.86150**
	$\|V\| = 200$	Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
		25	17.11061	17.04288	17.03756	**17.00152**
		125	16.99810	16.96905	16.98448	**16.91619**
		625	**16.88499**	16.89468	16.88522	16.88639
		3125	16.86827	16.83667	16.83449	**16.81440**
	$\|V\| = 400$	Time (s)	GRASP-R	GRASP-X	GRASP-M	GRASP-Mpr
		25	24.83950	24.69378	24.69833	**24.67034**
		125	24.78642	24.63951	**24.65960**	24.66547
		625	24.70407	24.60203	**24.57762**	24.60415
		3125	24.63254	**24.54719**	24.57644	24.56174

become the main drawback of GRASP-R when the instances sizes grow and the targets become harder.

Figures 6(a) and 6(b) show that GRASP-Mpr (see Algorithm 1) becomes the fastest variant for $|V| = 100$ when harder target solution values are sought. The incomplete plots in the figures show that the target was not reached in 15 minutes for many runs for all but the fastest GRASP-Mpr variant. The combination of the two acceleration schemes in this variant gives more diversity to the local



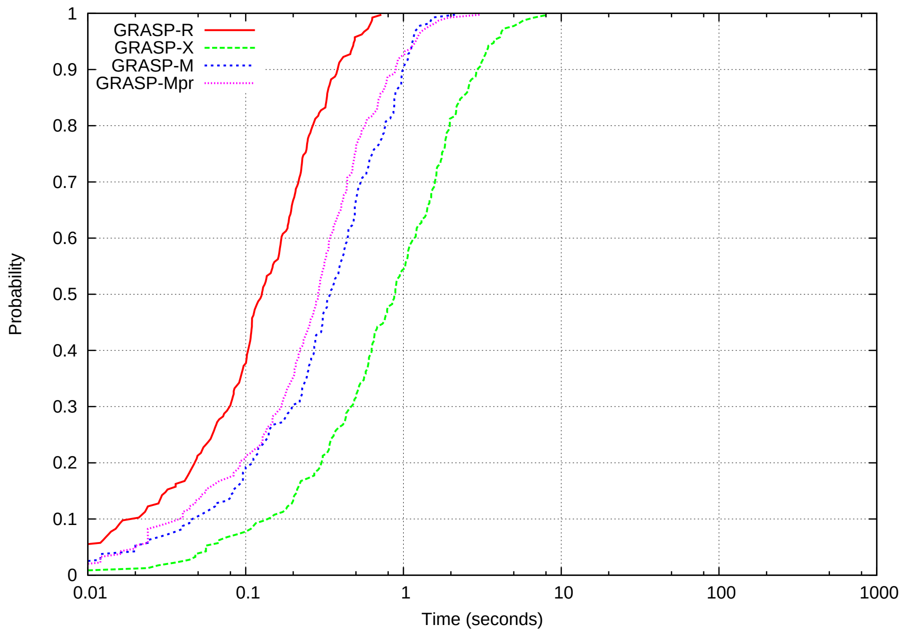
(a) Euclidean instance with $|V| = 200$



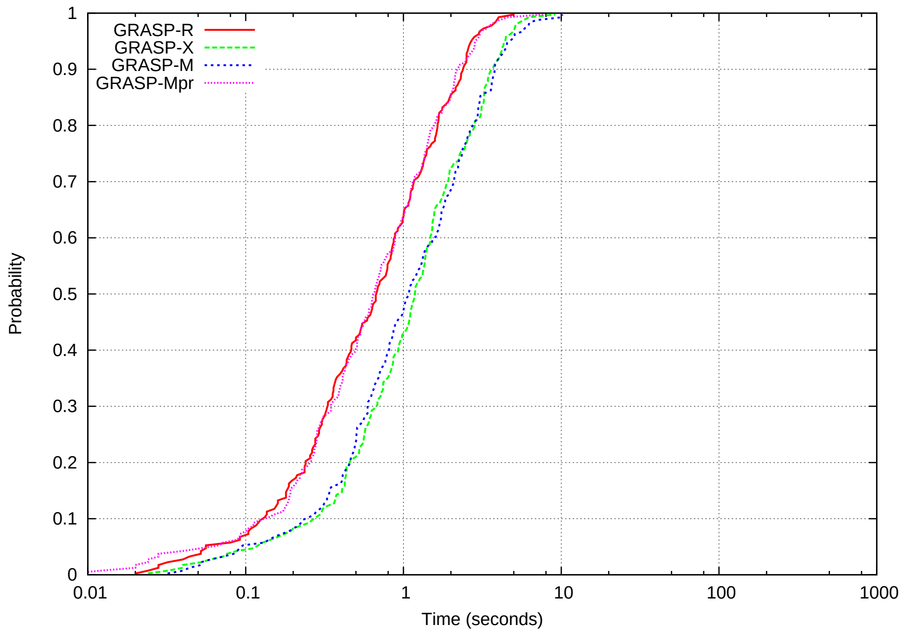
(b) Random instance with $|V| = 200$

FIGURE 4. Progressive improvement in solution values along the running time for different GRASP variants on instances with 200 nodes.

search, while path-relinking is used as an intensification strategy. Diversification and intensification improve the probability of finding good solutions in less time.



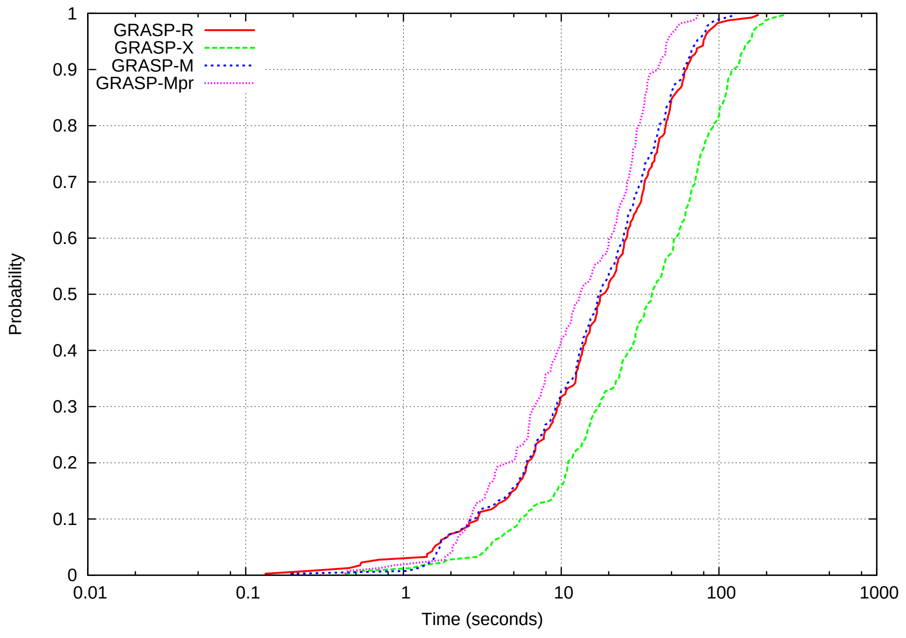
(a) Euclidean instance with $|V| = 25$



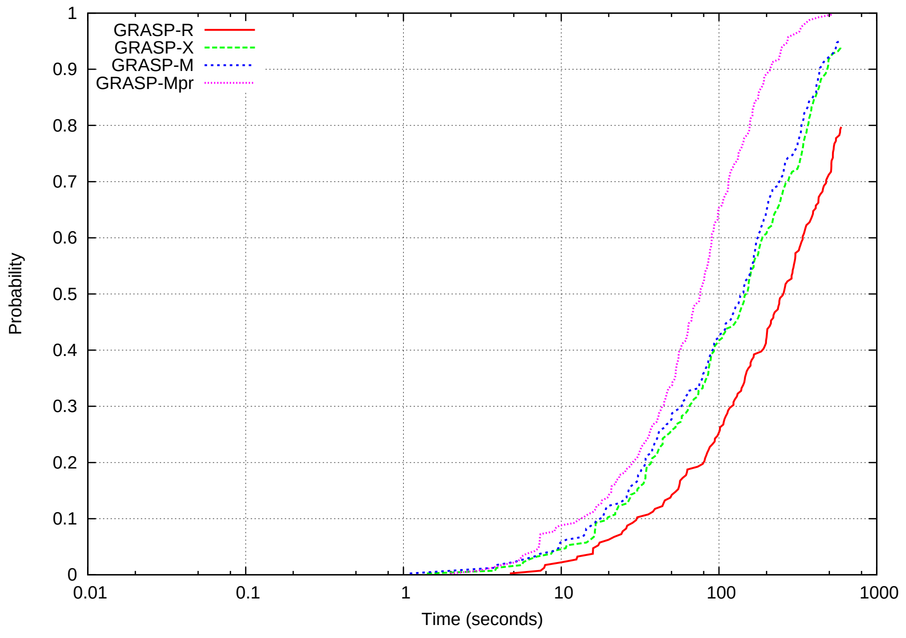
(b) Random instance with $|V| = 25$

FIGURE 5. Empirical distributions of the time to target-solution-value for different GRASP variants on instances with 25 nodes.

In the next experiment, we compare the MST-aug heuristic of Calinescu and Wan (2006) with the purely greedy implementation of the constructive algorithm presented in Section 3.1 and with the best heuristic GRASP-Mpr using a fixed



(a) Euclidean instance with $|V| = 100$



(b) Random instance with $|V| = 100$

Figure 6. Empirical distributions of the time to target-solution-value for different GRASP variants on instances with 100 nodes.

amount of time (ten minutes). Both algorithms MST-aug and the greedy heuristic found all solutions in less then one second of computation, even for instances with $|V| = 800$ nodes. Table 4 summarizes the average solution values over 15 instances

Table 4. Comparative average structural results for MST-aug, greedy and GRASP-Mpr on large problems: power consumption and node degrees.

Instance		Total power consumption					Average degree				
	$\|V\|$	MST--aug	Greedy	Impr. (%)	GRASP--Mpr	Impr. (%)	MST--aug	Greedy	Impr. (%)	GRASP--Mpr	Impr. (%)
Euclidean instances	25	2.02696	1.60533	20.80	1.37880	31.98	4.14	2.97	28.22	2.64	36.21
	50	2.04944	1.47477	28.04	1.27934	37.58	4.46	2.97	33.57	2.63	41.04
	100	2.15343	1.46062	32.17	1.32422	38.51	4.78	3.01	37.07	2.60	45.60
	200	2.84844	1.84594	35.19	1.76441	38.06	5.57	2.92	47.50	2.55	54.10
	400	4.72278	2.89544	38.69	2.82666	40.15	7.62	2.96	61.14	2.55	66.49
	800	8.44331	5.07814	39.86	4.99485	40.84	11.33	3.00	73,49	2.59	77.15
Random instances	25	12.84646	6.15077	52.12	5.46654	57.45	8.05	3.08	61.76	2.50	68.92
	50	24.20324	9.50228	60.74	8.35554	65.48	13.29	3.10	76.70	2.46	81.51
	100	44.97218	13.47090	70.05	11.87711	73.59	21.67	3.15	85.48	2.64	87.80
	200	85.08285	19.20624	77.43	17.16590	79.82	37.49	3.18	91.53	2.64	92.96
	400	158.26410	27.70172	82.50	25.05689	84.17	63.69	3.18	95.01	2.67	95.81
	800	293.63736	40.89464	86.07	37.34825	87.28	107.70	3.20	97.03	2.71	97.48

of each size. For each algorithm, we give the average node degree, the average total power consumption, and the improvements in percent obtained by the greedy and GRASP-Mpr heuristics with respect to the solution values provided by the existing algorithm MST-aug.

Heuristic MST-aug does not take into account the structure of biconnected components. The greedy heuristic systematically finds better solutions in all aspects, with its solutions being characterized by fewer bidirectional edges and smaller power assignments. In particular, it outperformed MST-aug for the Euclidean instances with reductions in power consumption ranging from 20.80% to 39.86%. For these instances, the network density grows with the number of nodes. As the density increases, the average distance between the nodes decreases and, consequently, so do the power requirements. Hence, the reduction in the average number of edges (or, equivalently, the reduction in the average node degree) (ranging from 28.22% to 73.49%) does not affect the power consumption by the same rate. For the random instances, however, there is no density variation and the reduction in the number of edges (ranging from 61.76% to 97.03%) directly impacts the reduction in power consumption (reductions ranging from 52.12% to 86.07%).

Table 4 also illustrates that GRASP-Mpr always improves the greedy solutions, being more effective with respect to the latter for the smaller instances. This is due to the fact that since the computation times given to GRASP-Mpr are fixed (ten minutes in this experiment), fewer GRASP iterations can be performed as the instance size grows. Better solutions can be obtained by GRASP-Mpr even for larger instances if more computation time is given, as already shown in Table 3.

This table also shows that the average node degree in the solutions produced by GRASP-Mpr range from 2.46 to 2.71, being much smaller than in the solutions found by MST-aug. Since the degree of any node must be at least 2 in a biconnected graph, these results indicate that the GRASP-Mpr solution values are very close to the best possible lower bounds and, consequently, to the exact optimal solutions. Furthermore, the average node degree range also shows that the greedy

TABLE 5. Comparative average structural results for MST-aug, greedy and GRASP-Mpr on large problems: interference measures.

Instance		Edge interference					Node interference				
	$\|V\|$	MST--aug	Greedy	Impr. (%)	GRASP--Mpr	Impr. (%)	MST--aug	Greedy	Impr. (%)	GRASP--Mpr	Impr. (%)
Euclidean instances	25	17.26	14.73	14.66	12.06	30.13	7,80	6,53	16,24	5,67	27,35
	50	24.73	18.46	25.35	14.13	42.86	9,47	7,53	20,42	6,60	30,28
	100	29.20	17.66	39.52	13.33	54.35	12,00	7,60	36,67	6,60	45,00
	200	40.73	16.46	59,59	15.33	62.36	15,27	8,40	44,98	8,13	46,72
	400	69.06	17.60	74.51	17.26	75.01	21,67	10,20	52,92	9,80	54,77
	800	136.13	23.46	82.77	23.06	83.06	33,33	12,67	62,00	11,80	64,60
Euclidean instances	25	37.33	21.53	42.33	19.00	49.10	17,67	11,20	36,60	10,07	43,02
	50	81.33	33.73	58.53	33.20	59.18	31,33	15,60	50,21	14,53	53,62
	100	169.73	55.06	67.56	52.73	68.93	56,33	21,87	61,18	20,20	64,14
	200	359.53	79.33	77.94	86.46	75,95	102,73	31,07	69,76	28,13	72,62
	400	729.13	127.20	82.55	125.33	82.81	185,07	42,00	77,31	37,93	79,50
	800	1484.93	194.20	86.92	196.86	86.74	330,80	58,00	82,47	53,27	83,90

and GRASP-Mpr algorithms obtain good solutions independently of the instance size, since, for all instance sizes, the average node degree values are very close to the values of the known optimal solutions (given by GRASP-Mpr for $|V| = 25$).

Given the transmission graph $G(p) = (V, B(p))$, we denote by $D(u) = \{w \in V : p(u) \geq e(u, w)\}$ the radio coverage area of node $u \in V$. According to Burkhart et al. (2004) and von Rickenbach et al. (2009), the interference $EI(G(p))$ is defined as the maximum *coverage* of a bidirectional edge in the transmission graph, i.e. $EI(G(p)) = \max_{[u,v]\in B(p)}\{Cov(u, v)\}$, where $Cov(u, v) = |\{w \in V : w \text{ is covered by } D(u)\} \cup \{w \in V : w \text{ is covered by } D(v)\}|$.

The interference of a node $u \in V$ is defined by von Rickenbach et al. (2005) as the number of nodes that potentially affect message reception at node u, i.e., $NI(u) = |\{w \in V \setminus \{u\} : u \in D(w)\}|$, where $D(w)$ once again denotes the radio coverage area of node $w \in V$. In other words, the interference of a node $u \in V$ represents the number of nodes covering u with their radio coverage areas, induced by the power assignments. The graph interference is defined as the maximum node interference over all nodes, i.e., $NI(G(p)) = \max_{u\in V}\{NI(u)\}$.

Table 5 shows interference results based on the above static models (Burkhart et al., 2004; von Rickenbach et al., 2009; 2005), defined independently of the network traffic. This table gives the average solution values over 15 instances of each size for the MST-aug heuristic of Calinescu and Wan (2006), for the purely greedy implementation of the constructive algorithm presented in Section 3.1, and for the best GRASP-Mpr heuristic using a fixed amount of time (ten minutes). For all algorithms, we give the edge interference $EI(G(p))$ and the node interference $NI(G(p))$ values. The reduction in percent obtained by the greedy algorithm and GRASP-Mpr with respect to algorithm MST-aug in terms of both interference measures are also reported. The results show that both the greedy algorithm and the GRASP-Mpr heuristic give smaller interference values than MST-aug while attempting to minimize the total power consumption.

We also observed that reductions in the interference measures are proportional to the reduction in the number of edges (see Table 4). From the definitions of the node and edge interference measures, we notice that a high number of arcs or edges is the major cause of interference. Therefore, algorithms leading to small number of arcs and edges are certainly appropriate choices for obtaining low interference values. We also noticed that unidirectional arcs increase interference, without contributing to connectedness. Therefore, solutions characterized by fewer unidirectional arcs are very useful to mitigate interference.

In the final experiment, we compare the average relative gap between the best feasible solution and the best lower bound. The values of the best feasible solutions are the optimal values for $|V| = 25$. For $|V| \geq 50$, the best feasible solutions are those obtained by GRASP-Mpr in ten minutes of computation time. The best lower bound is given by CPLEX applied to the linear relaxation of the formulation presented in Section 2, limited to three hours of computation.

Table 6 shows the average relative gaps for $|V| \in \{25, 50, 100, 200\}$. We observe that the relative gaps are very small for the Euclidean instances. Therefore, we may conclude that the solutions found by the GRASP-Mpr heuristic are very close to the optimal solutions for the Euclidean instances, which are closer to real-life applications. The experimental results also show that the GRASP-Mpr heuristic is very effective: it was able to approximately solve problems with up to 800 nodes, while an exact commercial solver such as CPLEX could not even find lower bounds to instances with more than 200 nodes.

Table 6. Average relative gaps between the value of the best feasible integer solution and the best lower bound.

Instances	$\|V\| = 25$	$\|V\| = 50$	$\|V\| = 100$	$\|V\| = 200$
Euclidean	4.41	5.26	4.22	3.25
Random	13.65	22.30	27.64	35.83

5. Concluding Remarks

We considered the problem of assigning transmission powers to the nodes of an ad hoc wireless network, so as that the total power consumption is minimized and the resulting network is biconnected, i.e., there are at least two node-disjoint paths between any pair of nodes. Biconnected communication graphs are important to ensure fault tolerance.

We recalled an integer programming formulation of the bidirectional topology version of the biconnected minimum power consumption problem. We showed that only very moderately-sized instances of this problem can be exactly solved to optimality by a state-of-the-art solver.

A very quick greedy algorithm and a GRASP with path-relinking heuristic were proposed to find good approximate solutions to real-life sized problems. Different implementation strategies have been considered and compared in the quest for algorithm efficiency and effectiveness. Computational experiments have been performed and reported for problems with up to 800 nodes.

A state-of-the-art integer programming solver such as CPLEX 11.0 provided optimal values for small problems and linear relaxation lower bounds for problems with up to 200 nodes. The GRASP with path-relinking heuristic was able to systematically find solutions that are very close to the optimal solutions and to the best lower bounds.

The greedy algorithm was able to find good solutions extremely fast. Both the greedy algorithm and the GRASP with path-relinking heuristic outperformed by far the best known heuristic in the literature for very large problem sizes with up to 800 nodes.

References

R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373, 2002.

R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, 1:355–366, 2007.

M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does topology control reduce interference? In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 9–19, Tokyo, 2004. doi: http://doi.acm.org/10.1145/989459.989462.

G. Calinescu and P.-J. Wan. Range assignment for biconnectivity and k-edge connectivity in wireless ad hoc networks. *Mobile Network and Applications*, 11: 121–128, 2006.

T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer, 2002.

P. Festa and M. G. C. Resende. An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24, 2009a.

P. Festa and M. G. C. Resende. An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009b.

A. Frank and E. Tardos. An application of submodular flows. *Linear Algebra and its Applications*, 114/115:329–348, 1989.

H. N. Gabow. A representation for crossing set families with applications to submodular flow problems. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete algorithms*, pages 202–211, Austin, 1993.

F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

F. Harary. The maximum connectivity of a graph. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 48, pages 1142–1146, 1962.

S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21:434–450, 1996.

G. Kortsarz and Z. Nutov. Approximating node connectivity problems via set covers. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 194–205, London, 2000. Springer-Verlag.

E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan, and S. S. Ravi. Algorithmic aspects of topology control problems for ad hoc networks. *Mobile Networks and Applications*, 10:19–34, 2005.

T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45:61–79, 2005.

M. K. Marina and S. R. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of the 9th International Conference on Network Protocols*, pages 14–23, Riverside, 2001.

R. E. N. Moraes, C. C. Ribeiro, and C. Duhamel. Optimal solutions for fault-tolerant topology control in wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 8:5970–5981, 2009.

M. Prais and C. C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176, 2000.

S. Raghavan. *Formulations and Algorithms for the netwotk design problems with connectivity requeriments*. PhD thesis, Massachussets Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, 1995.

T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, 2001.

M. G. C. Resende and C. C. Ribeiro. GRASP. In E. K. Burke and G. Kendall, editors, *Search Methodologies*. Springer, 2nd edition. To appear.

M. G. C. Resende and C. C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003a.

M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer, 2003b.

M. G. C. Resende and C. C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 29–63. Springer, Berlin, 2005.

M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 283–319. Springer, Berlin, 2nd edition, 2010.

M. G. C. Resende, C. C. Ribeiro, F. Glover, and R. Martí. Scatter search and path-relinking: Fundamentals, advances, and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 87–107. Springer, Berlin, 2010.

C. C. Ribeiro, E. Uchoa, and R. F. Werneck. A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–226, 2002.

H. M. Taghi, N. Immorlica, and V. S. Mirrokni. Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. *IEEE/ACM Transactions on Networking*, 15:1345–1358, 2007.

R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger. A robust interference model for wireless ad-hoc networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, page 239a, Denver, 2005.

P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Algorithmic models of interference in wireless ad hoc and sensor networks. *IEEE/ACM Transactions on Networking*, 17:172–185, 2009.

J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-eficient broadcast and multicast trees in wireless networks. In *Proceedings of the 9th Annual Joint Conference of the IEEE Computer and Communications*

Societies, volume 2, pages 585–594, Tel-Aviv, 2000.

G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless. Minimum power configuration for wireless communication in sensor networks. *ACM Transactions on Sensor Networks*, 3:200–233, 2007.

(Renato E. N. Moraes) Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria, 156, Niterói, RJ 24210-240 Brazil.
E-mail address: `rmoraes@ic.uff.br`

(Celso C. Ribeiro) Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria, 156, Niterói, RJ 24210-240 Brazil.
E-mail address: `celso@ic.uff.br`