Comput Optim Appl (2011) 49: 193–212
DOI 10.1007/s10589-009-9281-2

Minimum power multicasting in wireless networks under probabilistic node failures

János Barta · Valeria Leggieri · Roberto Montemanni · Paolo Nobili · Chefi Triki

Received: 12 February 2009 / Published online: 29 July 2009

Abstract In this paper we deal with a probabilistic extension of the minimum power multicast (MPM) problem for wireless networks. The deterministic MPM problem consists in assigning transmission powers to the nodes, so that a multihop connection can be established between a source and a given set of destination nodes and the total power required is minimized. We present an extension to the basic problem, where node failure probabilities for the transmission are explicitly considered. This model reflects the necessity of taking uncertainty into account in the availability of the hosts. The novelty of the probabilistic minimum power multicast (PMPM) problem treated in this paper consists in the minimization of the assigned transmission powers, imposing at the same time a global reliability level to the solution network. An integer linear programming formulation for the PMPM problem is presented. Furthermore, an exact algorithm based on an iterative row and column generation procedure, as well as a heuristic method are proposed. Computational experiments are finally presented.

Keywords Minimum power multicasting · Probabilistic mathematical models · Multihop networks

1 Introduction

A multihop wireless network is a collection of wireless devices that communicate without using any wired infrastructure. Even though each device has a limited trans-

J. Barta (✉) · R. Montemanni
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland
e-mail: janos.barta@supsi.ch

V. Leggieri · P. Nobili · C. Triki
Dipartimento di Matematica, Università del Salento, 73100 Lecce, Italy

mission range, global connectivity may be ensured by using multihop wireless links. Originally, the multihop networks were proposed for military applications in the battlefield. However, their employment has been extended to cover many fields such as emergency operations, healthcare, large events organization (sporting or conferences), city logistics, etc. The size of the adopted wireless networks depends mainly on the kind of the applications. It can reach hundreds of nodes, but currently it varies typically between about ten and fifty nodes (see also [20]). Because of these and other potential applications, the interest in multihop wireless networks has recently increased, generating several research challenges (see for example [1, 14]). Since the devices are usually characterized by limited resources (batteries), energy-aware communication becomes of crucial importance for the network functionality.

In this paper we focus on the problem of minimizing the power required to connect a source device to a set of hosts. This optimization problem, which is at present an intensive topic of study, is known as the minimum power multicast (MPM) problem. Indeed, significant effort is being done for modeling and solving the MPM and, even more intensively, its particular variant the minimum power broadcast (MPB) problem (see [6, 8, 16, 18]). A detailed review on exact and heuristic methods to solve both problems can be found in the recent surveys [10, 14].

All the above works assume a deterministic behavior of the transmitting devices. In reality it has to be expected that the terminals can be affected by temporary damage or permanent failure. Therefore it is reasonable to extend the deterministic case to a probabilistic formulation that takes into account the uncertain nature of node availability. It will be soon clear that this extension introduces an extra layer of complexity to the problem. However, the advances in heuristic and exact algorithms for the MPM, lead, in our opinion, to the opportunity of studying more realistic, although more complex, models of the problem, as the one we propose in this paper.

The novelty of our contribution consists in dealing with a probabilistic version of the MPM (PMPM) by explicitly considering a probabilistic availability of the hosts. We suppose that each device has a given probability of failure, due to temporary damage or battery depletion, and we impose that connectivity should be guaranteed with a given level of reliability. The resulting mathematical formulation lies into the class of probabilistic integer programming model for which we propose an original specialized exact solution approach.

It is trivial to see that PMPM is an NP-hard problem: the deterministic MPM problem, which is known to be NP-complete (see [4, 13]), is a special case of problem PMPM. More precisely, a PMPM instance reduces to an MPM problem if the probability of failure related to each node is set to zero (see also [3, 7, 21]).

The sequel of the paper is organized as follows. This Introduction will include the literature review. In Sect. 2 we give the problem definition. In Sect. 3 our probabilistic mathematical model for the PMPM is illustrated. In Sect. 4 our exact solution approach is described. Section 5 is dedicated to the discussion of the computational results, and finally in Sect. 6 our heuristic method is presented.

1.1 Related works

To the best of our knowledge in the literature there is no work that proposes mathematical formulations or exact methods to solve the PMPM problem. Moreover, even

its deterministic variant, i.e. MPM, has received less attention than what one may expect. Indeed, while many heuristic and approximation algorithms are so far available to solve the MPM (see the surveys of [10, 15] and the references therein), we are aware of only few contributions dealing with its exact solution. The first is [12], due to Leino, who proposed an integer formulation and sketched a general scheme based on a cutting plane algorithm for its solution. The second formulation, proposed by Guo and Yang in [9], is a flow-based mixed integer model. The most recent work is [11], due to Leggieri et al., who proposed a set covering-based formulation for the MPM problem, and presented two specialized algorithms for its solution.

In addition to these original MPM formulations some new models could be obtained, according to [10], by modifying existent formulations that were proposed for the MPB problem. Among the contributions available in this context we cite the multi-commodity flow model proposed by Yuan in [23].

Considering the probabilistic aspect of the PMPM problem, we can claim with certainty that the literature is extremely poor in the case of broadcasting and inexistent in the case of multicasting. Specifically, the only work dealing with power management in random settings is [19], due to Montemanni et al., and proposes three novel mixed integer formulations for a version of the MPB problem where a failure probability is associated with each node, but the objective is different from the one considered in the present work: it is to find a broadcasting structure such that each node is connected to the root and on each of the paths in the solution the reliability level is higher than a given threshold.

For the sake of completeness we shall note that there are other aspects of randomness that have been studied in the context of mobile networks but that do not fall into the framework of topology optimization. Examples of these aspects include the randomness related to the links failure due to mobility [5] and a statistical analysis for the broadcast problem under Byzantine failures [2].

2 Problem definition

A network of wireless devices can be modeled mathematically as a directed complete graph $G = (V, A)$, where the elements of the set V are the devices and those of A are all the possible connections between pairs of devices. We denote by n the cardinality of the set V and we suppose that $n \geq 3$. We select a node s to be the source of the communication and a subset R of V that contains all nodes that are supposed to be reached by the signal generated in s. Let r be the cardinality of set R.

Each node $i \in V$ can receive data from other nodes of the network and send data to any node in its transmission range. With each arc $(i, j) \in A$ we associate the minimum amount p_{ij} of power that must be assigned to node i in order to establish a direct communication with node j.

The MPM problem consists in defining the so called *range assignment* function ρ, which assigns to each node $i \in V$ a transmitting power $\rho(i)$

$$\rho : V \to \mathbb{R}^+, \qquad i \mapsto \rho(i)$$

minimizing the sum:

$$\sum_{i \in V} \rho(i),$$

so that it is ensured a connection between s and each destination $d \in R$. In establishing such a connection one can exploit the so-called wireless multicast advantage (WMA), a fundamental property of the wireless networks (see [22]). The WMA property simply consists in the following: since the devices are equipped with omnidirectional antennas, all the nodes that are within the transmission range of a transmitting node receive the signal. Therefore several nodes can be covered and reached at the same time using a single transmission.

The probabilistic aspect of our problem lies in the fact that each node $i \in V$ is available with a given probability q_i. We assume that the probabilities q_i are independent from each other and that for each node $i \in R \cup \{s\}$ it holds $q_i = 1$. Typically the value of q_i will depend on the characteristics of both the node and the area where it is deployed. For example a node i positioned in a dangerous region or in an impervious territory will be assigned a small value of q_i.

Whenever a node i with $q_i \neq 1$ is involved in the routing of a message between the source s and a destination, the correct reception of the communication depends on the availability of node i at the moment of the transmission. In order to ensure a certain quality of service, we impose in the model that the message is received by all the destinations with a given reliability.

As a consequence, the PMPM problem consists in defining a range assignment function in such a way that s is connected with all the destinations in R with a probability greater than a given reliability threshold $\alpha \in]0, 1[$. The objective function to be minimized is still $\sum_{i \in V} \rho(i)$, that is the total assigned transmission power, but in addition a reliability constraint is also imposed.

3 Mathematical formulation

Our approach consists in formulating the PMPM problem as an integer linear programming (ILP) model.

The powers p_{ij} (with $(i, j) \in A$) can be used to order the arcs outgoing from each node. Indeed, for each node $i \in V$, we sort all arcs $(i, j) \in A$ outgoing from i in a non-decreasing order with respect to the p_{ij} values. For each subset K of the set of nodes V, we denote by $b(i, K)$ the first arc of the ordering relative to i that is incoming in a node of K, so that we can introduce the subset B of A as follows:

$$B := \{b(i, V) \in A : \ i \in V\}.$$

B contains the arcs connecting each node with its closest neighbor. Furthermore, for every arc $(i, j) \in A \setminus B$ we denote by $a(i, j)$ the *ancestor* of arc (i, j), that is the arc that immediately precedes (i, j) in the ordering with respect to i.

Using the above notation, the incremental cost c_{ij} associated with each arc $(i, j) \in A$ is defined in the following way:

$$c_{ij} := \begin{cases} p_{ij} & \text{if } (i, j) \in B, \\ p_{ij} - p_{a(i,j)} & \text{otherwise.} \end{cases}$$

In order to formulate the PMPM problem, it is convenient to introduce binary variables y_{ij} associated with the arcs. In particular, for each arc $(i, j) \in A$ the variable y_{ij} has the following interpretation:

$$y_{ij} := \begin{cases} 1 & \text{if } p_{ij} \leq \rho(i), \\ 0 & \text{otherwise} \end{cases}$$

that is, $y_{ij} = 1$ if the node i is assigned enough power to reach at least node j. In this case we say that the arc (i, j) is *active*.

Since we want to minimize the power to be assigned to the nodes of the network in order to connect the source to the destinations, we notice that the range assignment function evaluated in each node i shall assume a value of either zero or exactly p_{ij} for some $j \in V$. Therefore, it is easy to verify that

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} = \min \sum_{i \in V} \rho(i).$$

In the sequel, we will formulate the objective function of the PMPM problem by means of the y variables and we will refer to the subgraph induced by the arcs (i, j) such that $y_{ij} = 1$ as a *solution* y.

In view of the WMA property, if an arc $(i, j) \in A \setminus B$ is active in the communication then its ancestor $a(i, j)$ is active too and this property can be modeled by the constraint:

$$y_{ij} \leq y_{a(i,j)} \quad \forall (i, j) \in A \setminus B.$$

Denoting by $\delta^+(S)$ the set of arcs $(i, j) \in A$ with $i \in S$ and $j \in V \setminus S$, it is possible to formulate the *deterministic* MPM problem with a directed cut based formulation as follows (see [18]):

$$\boxed{\mathcal{F}} \quad \min \sum_{(i,j)\in A} c_{ij} y_{ij}$$

$$\text{s.t. } y_{ij} \leq y_{a(i,j)} \quad \forall (i, j) \in A \setminus B, \tag{1}$$

$$\sum_{(i,j)\in\delta^+(S)} y_{ij} \geq 1 \quad \forall S \subseteq V : s \in S,\ (V \setminus S) \cap R \neq \emptyset, \tag{2}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A. \tag{3}$$

Constraints (1) guarantee the fulfillment of the WMA property while constraints (2) guarantee that there exists a path from s to any destination in any feasible solution.

Because of the WMA property, it is possible to substitute constraints (2) with constraints

$$\sum_{i \in S} y_{b(i,V \setminus S)} \geq 1 \quad \forall S \subseteq V : s \in S,\ (V \setminus S) \cap R \neq \emptyset. \tag{4}$$

Indeed, it is easy to observe that if variable $y_{ik} = 1$ for a given $i \in S$ and $k \in V \setminus S$, then, by constraints (3), for any other node $j \in V \setminus S$ such that $p_{ij} \leq p_{ik}$ it holds that $y_{ij} = 1$ and in particular $y_{b(i,V \setminus S)} = 1$. In other words, for any node $i \in S$ it suffices to consider only the arcs $b(i, V \setminus S)$ in order to express the connectivity requirement. On one side, substituting constraints (2) with constraints (4) does not modify the set of the integer feasible solutions of the problem, on the other if we consider the linear programming (LP) relaxations, using constraints (4) strengthens the LP relaxation of $\mathcal{F}$. Indeed, if we suppose that $\bar{y}$ is a feasible solution of the LP relaxation of $\mathcal{F}$ with constraints (4) instead of constraints (2), then $\bar{y}$ fulfills the inequalities $\sum_{(i,j) \in \delta^+(S)} \bar{y}_{ij} \geq \sum_{i \in S} \bar{y}_{b(i,V \setminus S)} \geq 1$ for all the subsets S of V such that $s \in S$ and $(V \setminus S) \cap R \neq \emptyset$.

In order to take into account the probabilistic nature of the problem PMPM we need to introduce additional notation to the formulation $\mathcal{F}$ of the deterministic problem MPM. Due to the fact that each node $i \notin R \cup \{s\}$ has a probability q_i of being available, in any realization (network scenario) only a given subset C_l of the devices can be considered to work. Since we assume that $q_i = 1$ for each $i \in R$, it follows that for each subset C_l it holds that $R \subseteq C_l$. We enumerate the family of all such subsets as the collection $\{C_l\}_{l \in \{1,\ldots,N\}}$, where $N := 2^{n-r-1}$. In the remainder of the paper we will refer to any of the N possible network scenarios as a *configuration*, denoted either by C_l or simply by its corresponding index l. It is easy to see that the probability of realization of a given configuration l is expressed as:

$$Q_l := \prod_{i \in C_l} q_i \cdot \prod_{i \notin C_l} (1 - q_i). \tag{5}$$

Configurations are at the basis of the model we propose to calculate the probability associated with a given topology. In order to calculate the reliability of a solution y its connectivity state on each configuration l has to be assessed. We say that solution y is *connective* on configuration l, if the active arcs outgoing from the nodes of C_l contain an arborescence rooted at s and spanning all the destinations $d \in R$. To express this feature we associate with each configuration $l \in \{1, \ldots, N\}$ a binary variable v_l with the following meaning:

$$v_l := \begin{cases} 0 & \text{if solution } y \text{ is connective on configuration } l, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

For a given configuration l and for each set $S \subseteq C_l$ containing the source s and such that $C_l \setminus S$ contains at least one destination, we must have that either solution y is not connective on configuration l (i.e. $v_l = 1$) or the cut constraint associated with S is satisfied by y. In other words, solution y should satisfy:

$$\sum_{i \in S} y_{b(i,C_l \setminus S)} + v_l \geq 1.$$

Hence, taking into account all the configurations and all subsets S, the PMPM model should require the fulfillment of the following connectivity constraints:

$$\sum_{i \in S} y_{b(i,C_l \setminus S)} + v_l \geq 1 \quad \forall l \in \{1, \ldots, N\},\ \forall S \subseteq C_l : s \in S,\ (C_l \setminus S) \cap R \neq \emptyset.$$

The number of connectivity constraints is very large. It can be verified that their number increases exponentially with n and amounts to:

$$L = 3^{n-1} \cdot \frac{2^r - 1}{3^r}.$$

However, it is interesting to observe that for a given n, as the number of destinations grows, L decreases exponentially from a maximum of 3^{n-2} (in the unicast case $r = 1$) to a minimum of $2^{n-1} - 1$ (in the broadcast case $r = n - 1$). The reason is that the number of configurations depends on the number of transition nodes (belonging to $V \setminus (R \cup \{s\})$) and it decreases when the set of destinations R is large.

The PMPM problem requires a solution with a reliability greater than or equal to a given threshold α. The reliability of a solution y can be defined as the value $\sum_{l \in J} Q_l$ where J is the set of all the configurations on which y is connective. In order to fulfill the reliability constraint, we should either require that $\sum_{l \in J} Q_l > \alpha$ or, considering the probability of the complementary event (the probability of realization of configurations on which y is not connective), we impose the constraint

$$\sum_{l=1}^{N} Q_l v_l \leq 1 - \alpha. \tag{7}$$

We remind that a variable v_l has to assume value 1 if configuration l is not connective.

Summarizing, we can formulate the PMPM problem as an integer linear programming problem as follows:

$$\boxed{\mathcal{P}} \quad \min \sum_{(i,j) \in A} c_{ij} y_{ij}$$

$$\text{s.t. } y_{ij} \leq y_{a(i,j)} \quad \forall (i, j) \in A \setminus B, \tag{8}$$

$$\sum_{i \in S} y_{b(i,C_l \setminus S)} + v_l \geq 1 \quad \forall l \in \{1, \ldots, N\},$$

$$\forall S \subseteq C_l : s \in S,\ (C_l \setminus S) \cap R \neq \emptyset, \tag{9}$$

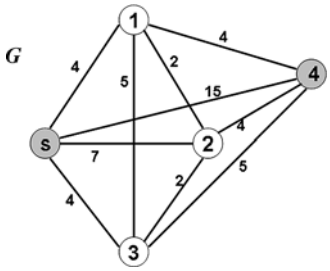
$$\sum_{l=1}^{N} Q_l v_l \leq 1 - \alpha, \tag{10}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A, \tag{11}$$

$$v_l \in \{0, 1\} \quad \forall l \in \{1, \ldots, N\}. \tag{12}$$

The combination of constraints (9) and constraint (10) guarantees the existence of a connection from s to all the destinations with at least reliability α.

Fig. 1 Example



Since the transmission range of the devices has no upper limit, formulation (8)–(12) admits always a feasible solution (e.g. the source transmits with a power sufficient to reach in a single hop all the destinations in R). Moreover, it is easy to notice that the optimal solution of the multicast problem in deterministic settings is a lower bound for our problem. We also notice that in each integer feasible solution of the problem, in view of the WMA, it holds that the variable $y_{b(s,V)}$ relative to the first arc outgoing from s assumes always value 1 and hence, we add to the formulation the constraint

$$y_{b(s,V)} = 1 \tag{13}$$

in order to strengthen the linear relaxation of our formulation.

Finally, we present an example for better clarifying the differences between formulations $\mathcal{F}$ and $\mathcal{P}$.

Example 1 Consider the graph in Fig. 1, suppose that $R = \{4\}$ and that $q_1 = 0.89$, $q_2 = 0.40$, $q_3 = 0.89$. Set the reliability threshold to $\alpha = 0.9$. The solution $y_{s1} = y_{s3} = y_{12} = y_{14} = 1$ with all the other variables equal to zero is optimal for the MPM problem and it has cost 8. All the possible configurations with their probability of realization are displayed in the following table:

l	1	2	3	4	5	6	7	8
C_l	$\{s, 1, 2, 3, 4\}$	$\{s, 1, 2, 4\}$	$\{s, 1, 3, 4\}$	$\{s, 2, 3, 4\}$	$\{s, 1, 4\}$	$\{s, 2, 4\}$	$\{s, 3, 4\}$	$\{s, 4\}$
Q_l	0.31684	0.03916	0.47526	0.03916	0.05874	0.00484	0.05874	0.00726

It is easy to see that this optimal solution does not satisfy the requirement (10) having reliability equal to 0.89 (the connective configurations are C_1, C_2, C_3 and C_5).

The optimal solution for the PMPM problem is $y_{s1} = y_{s3} = y_{12} = y_{14} = y_{32} = y_{34} = 1$ with all the other variables equal to zero. It has cost equal to 13 and reliability equal to 0.9879.

4 Exact solution approach

In this section we present an exact algorithm based on an iterative row and column generation (IRCG) procedure for solving the PMPM problem. The bottleneck of the

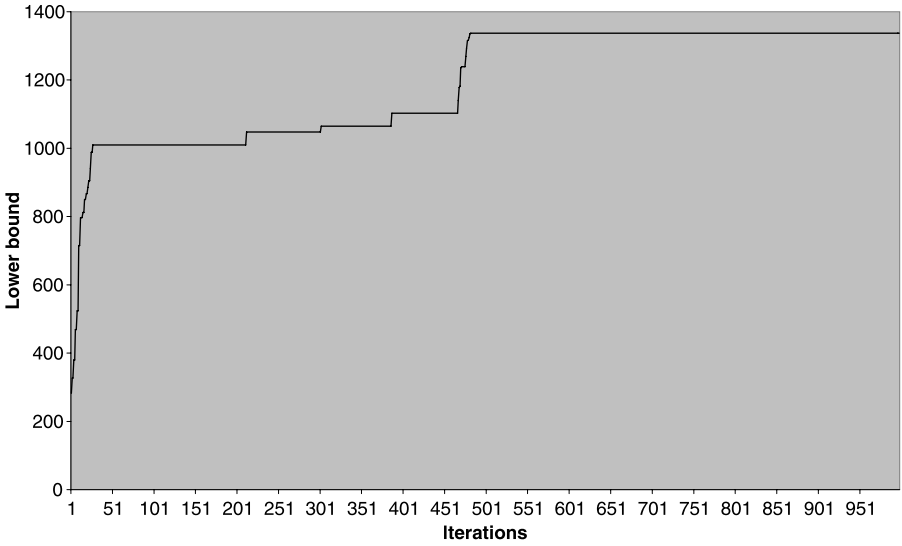


Fig. 2 Evolution of the lower bounds

problem formulation ($\mathcal{P}$) described in Sect. 3 is represented by the high number of variables v_l and of connectivity constraints (9). Preliminary tests showed that using an ILP solver only very small instances can be solved directly. However, it is reasonable to expect that many variables v_l and constraints (9) will not be relevant at optimality. Therefore a suitable approach is to generate the rows and the columns of the ILP in a dynamic way.

4.1 Algorithm description

The key idea is to consider a reduced ILP ($\mathcal{P}_k$) containing only k connectivity constraints (9) and only the variables v_l that appear in these constraints. Therefore, in ($\mathcal{P}_k$) also the reliability constraint (10) will contain only the variables v_l involved in the connectivity constraints.

Essentially algorithm IRCG generates a sequence of problems ($\mathcal{P}_k$) by adding a new constraint (9) (and possibly a v_l variable) at each iteration. It is easy to see that any feasible solution of a problem ($\mathcal{P}_{k+1}$) is also feasible for problem ($\mathcal{P}_k$), in other words ($\mathcal{P}_k$) can be regarded as a relaxation of ($\mathcal{P}_{k+1}$) and ultimately of ($\mathcal{P}$). Therefore the solution of any relaxed problem ($\mathcal{P}_k$) provides a lower bound to the original problem ($\mathcal{P}$). A typical evolution of the lower bounds is shown in Fig. 2. As a consequence, if (y, v) is the solution of a relaxed problem ($\mathcal{P}_k$) and a reliability computation shows that y reaches the required threshold α, it follows that there exists a setting of the variables v_l, $\forall l \in \{1, \ldots, N\}$, such that all constraints of problem ($\mathcal{P}$) can be satisfied. In other terms the obtained solution is feasible for the original problem ($\mathcal{P}$) and thus optimal.

Summarizing, the iterative algorithm IRCG is based on three main steps: the solution of the ILP ($\mathcal{P}_k$), the estimation of the reliability of the obtained solution y and

the insertion of a new connectivity constraint (9) in the ILP. How these operations are performed is explained in detail in the next sections and in the following sketch of the algorithm.

Estimation of the reliability

As mentioned before, calculating an estimate $\bar{\alpha}$ of the reliability of the current solution (y, v) is an essential step of the algorithm IRCG.

The reliability procedure implemented in algorithm IRCG tests the connectivity of the configurations defined by the sequence $\{C_l\}_{l\in\{1,\dots,N\}}$. More specifically, we start by setting $\bar{\alpha} := 0$ and each time a connective configuration C_l is found, $\bar{\alpha}$ is incremented by Q_l. In other words $\bar{\alpha}$ sums up the probabilities of the connective scenarios in the sequence $\{C_l\}_{l\in\{1,\dots,N\}}$.

It is easy to see that this procedure leads to the exact reliability of solution (y, v) if all configurations are tested, but due to the high number of configurations, the calculation of the exact reliability at each iteration would be very time consuming. Therefore the procedure is performed on a subset of the sequence $\{C_l\}_{l\in\{1,\dots,N\}}$, obtaining an estimation $\bar{\alpha}$ from below of the reliability. The exit criterion $\bar{\alpha} \geq \alpha$ simply states that if the estimated reliability $\bar{\alpha}$ reaches the threshold α, the solution is feasible and thus optimal.

Ordering of the configurations

The sequence in which the configurations are analyzed in order to detect violated constraints (9) is a crucial issue in the design of algorithm IRCG. Indeed, the quality of the added violated constraints and also of the estimated reliability strongly depends on the chosen sequence of the configurations.

The key idea behind the strategy we adopt is to process first the configurations with a high probability of realization. In this way the added cuts regulate the connectivity on the most probable configurations and, on the other side, the estimated reliability $\bar{\alpha}$ is likely to increase as fast as possible. In realistic applications it might be expected that the probabilities of the transition nodes are close to 1. According to (5), the configurations with many active nodes are usually weighted with high probabilities, therefore they should be tested first. Formally, if $|C_l|$ is the number of active nodes in configuration l, the set of the configurations is ordered in such a way that $|C_{l+1}| \leq |C_l|\ \forall l \in \{1, \dots, N-1\}$. In other words, the sequence begins with C_1, the scenario with all nodes available, followed by the configurations with one failure, and so on. We refer to this enumeration as an *ordering by increasing failure*.

Row and column generation

Due to the fact that problem $(\mathcal{P}_k)$ does not contain all connectivity constraints (9), the obtained solution (y, v) might violate some of them. Thus the following inconsistency might occur: a variable v_l assumes value 0, or it has not been yet generated, although the solution (y, v) is not connective on configuration l. As soon as the algorithm detects such a situation, a violated constraint (9) is identified and added to the

problem $(\mathcal{P}_k)$. In the case that the violated constraint contains a new variable v_l, also a new column is inserted in the ILP (see Step 4 of the outline of the algorithm). Then the reinforced ILP is solved again in order to obtain an improved solution. We notice that the opposite situation, i.e. $v_l = 1$ while (y, v) is connective on configuration l, does not cause violated constraints.

It would be possible to add more than one constraint (9) at each iteration, but in the economy of the algorithm we propose it is computationally more convenient not to add too many constraints each time (see also Sect. 4.2).

As it will be clear in Sect. 5, which is devoted to the computational results, in most of the cases only a small subset of the configurations is actively used by the algorithm. In other words, most variables v_l do not appear in the inserted connectivity constraints (9). Since most of the computational times are spent for the solution of the integer linear programs, it is not surprising that the speed up obtained by generating the variables dynamically is considerable. An evidence of this will be provided in Sect. 5.3.

Algorithm IRCG can now be described formally:

Step 0. Set $k := 0$, $l := 1$ and $\bar{\alpha} := 0$.
Step 1. Solve ILP $(\mathcal{P}_k)$. Let (y, v) be the solution of problem $(\mathcal{P}_k)$.
Step 2. Check configuration l: if $v_l = 1$ increment $l := l + 1$ and repeat Step 2. If $v_l = 0$ or v_l has not been yet generated then verify whether y is connective on configuration l. If y is connective on configuration l, increment $\bar{\alpha} := \bar{\alpha} + Q_l$ and go to Step 3, otherwise go to Step 4.
Step 3. Exit criterion: if $\bar{\alpha} \geq \alpha$, (y, v) is feasible. Stop. Otherwise set $l := l + 1$ and go to Step 2.
Step 4. Row and column generation: Add the violated constraint detected on configuration l to the ILP and, if v_l is a new variable, add also a column. Set $k := k + 1$, $l := 1$, $\bar{\alpha} := 0$ and go to Step 1.

It is worthwhile remarking that algorithm IRCG is exact, in the sense that it always provides an optimal solution of problem $(\mathcal{P})$. Indeed, if an optimal solution is reached, algorithm IRCG is able to recognize its feasibility. In the worse case this happens when all configurations $l \in \{1, \ldots, N\}$ have been tested and no new constraint can be added. This means that the current solution satisfies all constraints of the formulation $(\mathcal{P})$. We notice that this theoretical worse case never occurred during our computational tests.

Observe that in the proposed procedure the column generation activity is only carried out implicitly, alongside the row generation step. However, no true advantage would result by implementing a full-fledged pricing procedure. In fact, on the one hand the price of a variable v_l not appearing in any of the currently active constraints of type (9) is simply given by βQ_l, where $\beta \geq 0$ is the dual variable associated with constraint (10). It follows that at any step of the proposed procedure all the currently missing variables have positive price and should not be considered for addition. On the other hand, no real saving would be obtained by not generating (if not already present) the single variable v_l contained in a constraint of type (9) as soon as it is added.

4.2 Implementation details

During preliminary tests we observed that a high number of violated cuts is often introduced very early in the ILP. This has two drawbacks: firstly the exit criterion can not immediately recognize a feasible solution, because the algorithm is stuck on the first configurations. Secondly the rapid growth of the ILP has a strong slow-down effect on the algorithm.

For these reasons we adopted the strategy that, before adding a violated cut to the ILP, the estimation of the reliability is carried on for a given number l_m of additional configurations. Basically this strategy improves the quality of the estimate $\bar{\alpha}$ and slows down the growth of the ILP. In Sect. 5.2 we provide detailed information about the tuning of parameter l_m.

When an algorithm like that described in Sect. 4 is implemented, it is common practice (see for example [17]) to first generate some cuts from a linear relaxation of the problem. The aim of such a pre-processing procedure is to speed up the convergence of the method. Cuts generated on a linear relaxation are potentially the same that would have been generated on the original, fully integer, problem. On the other hand, solving a relaxation is likely to be less time consuming.

Therefore we first consider the problem $(\mathcal{P}_0')$ obtained from $(\mathcal{P}_0)$ by relaxing the integrity requirement on the variables y_{ij}. We start to solve the MIP (state variables v_l are still integer) and to heuristically generate constraints (9) searching for violated cuts of the form

$$\sum_{(i,j)\in A, i\in S, j\in C_l\setminus S} y_{ij} + v_l \geq 1 \tag{14}$$

and then strengthening the resulting inequality. We highlight the fact that with this procedure only a subset of violated cuts (9) can be detected.

Preliminary tests suggest that the use of the linear relaxation is useful to speed up the algorithm IRCG, especially on the largest networks.

5 Computational experiments

All the algorithms have been coded in ANSI C. Ilog Cplex 10.2 (http://www.ilog.com/products/cplex) has been used to handle and solve integer linear programs. The experiments reported in Sects. 5.2 and 5.3 have been run on a computer equipped with a Pentium 4 (2.8 GHz) processor with 512 MB of memory. All the results reported in Sect. 5.4 have been obtained on a computer equipped with an AMD Opteron 246 (2 GHz) dual processor and 2 GB of memory. The complete, detailed output of our tests can be found at http://www.idsia.ch/~roberto/PMPM09.pdf. Finally the tests reported in Sect. 6 have been carried out on a Pentium M (1.5 GHz) processor with 512 MB of memory.

5.1 Description of the test problems

No benchmark problem is available from the literature for the PMPM problem. Consequently we have generated a new benchmark set. We defined Euclidean instances

according to the following schema. A multicast problem is characterized by the following parameters:

- n number of devices in the network
- r number of destinations, i.e. devices that have to receive the messages originated at the root node s
- α reliability level required for the multicast structure
- $q_{\min}$ minimum value considered for the reliability probabilities associated with devices: $q_i \geq q_{\min}\ \forall i \in V \backslash \{s \cup R\}$
- $q_{\max}$ maximum value considered for the reliability probabilities associated with devices: $q_i \leq q_{\max}\ \forall i \in V \backslash \{s \cup R\}$

The coordinates of the nodes are chosen at random on a 100×100 square grid. Power p_{ij} required to send from node i to node j is obtained according to the relation $p_{ij} = (d_{ij})^{\gamma}$, where d_{ij} is the Euclidean distance between nodes i and j and the coefficient γ—which models path loss in the signal propagation model—is set to 2.

5.2 Tuning of parameter l_m

The maximum number of configurations l_m tested on a single solution y is defined as a function of the total number of configurations N and of the required reliability α. A suitable bound l_m keeps the number of added cuts low, but it also stops the testing of a solution when its feasibility can not be decided in a reasonable number of steps.

In Table 1 we consider two problem instances and we report the computation times required to solve them when different values of parameter l_m are considered. Notice that for each of the problems considered, the highest value reported for l_m coincides with the total number of configurations of the problem.

Table 1 suggests that values in the range [3000, 7000] for parameter l_m guarantee the fastest convergence. Additional tests, not reported here, confirmed that values in this range are indicated for all kind of problems, although it does not seem to be possible to further discriminate within this interval, since results are very instance-dependent. For the remainder of our tests we set $l_m = 5000$.

It is finally interesting to observe how, for the first problem considered in Table 1, there is a threshold for parameter l_m after which the optimal solution is identified almost immediately.

5.3 Importance of the dynamic insertion of variables v_l

The aim of the experiments reported in this section is to show the idea of incrementally (and dynamically) introducing variables v_l into problems $(\mathcal{P}_k)$ only when involved in some active constraint of type (9). Table 2 reports the computation times required to solve two instances when all the variables v_l are introduced at the beginning and when they are added dynamically only when required (which is the strategy we propose).

The benefit of the strategy we propose is clear from Table 2, where the introduction of all the variables v_l at the beginning even leads to no solution in the maximum computation time of 3600 s for the second problem considered (the corresponding entry of the table is marked with "–").

Table 1 Tuning of parameter l_m

n	r	α	$q_{\min}$	$q_{\max}$	l_m	Seconds
20	1	0.9	0.85	0.95	10	262.20
					100	206.61
					1000	186.49
					3000	186.42
					5000	0.08
					7000	0.08
					10000	0.08
					262144	0.08
25	3	0.9	0.85	0.95	10	320.61
					100	248.80
					1000	230.52
					3000	227.00
					5000	223.70
					7000	231.94
					10000	269.88
					2097152	888.02

Table 2 Insertion strategy of variables v_l

n	r	α	$q_{\min}$	$q_{\max}$	v_l strategy	Seconds
20	1	0.9	0.85	0.95	Static	14.09
					Dynamic	0.08
25	3	0.9	0.85	0.95	Static	–
					Dynamic	223.70

An important side effect of the dynamic strategy we adopt for the introduction of variables v_l is on the number of constraints of type (9) we introduce at each iteration of the algorithm IRCG. As explained in Sect. 4, we introduce only one cut during each iteration. We found that this is the most effective strategy because otherwise many variables v_l (corresponding to configurations, and in turns to constraints (9)) were quickly introduced into the sequence of problems $(\mathcal{P}_k)$, leading to a much slower algorithm.

5.4 Detailed results

For each problem considered, ten instances have been generated and IRCG has been run on them. Experimental results are summarized in Tables 3 to 6, that are organized as follows. The first five columns describe the problems, as defined in Sect. 5.1. The column *Solved* reports, for each problem considered, the number of instances solved to optimality within the allowed computation time of 3600 s out of the ten considered. Then we report, for each problem, average, minimum and maximum values, calculated over the instances solved to optimality, for the following three indicators. *Cuts*, which indicates the number of cuts (9) added to obtain the optimal solution;

Table 3 Performance of the IRCG algorithm when the number of devices n is varied (small/large sets of destination devices). Statistics over ten instances

n	r	α	$q_{\min}$	$q_{\max}$	Solved	Indicator	Avg.	Min	Max
15	5	0.9	0.85	0.95	9	Cuts	57.00	5.00	166.00
						Configs	28.78	5.00	45.00
						Seconds	15.22	0.02	37.34
20	5	0.9	0.85	0.95	6	Cuts	452.67	65.00	660.00
						Configs	192.83	13.00	640.00
						Seconds	1431.34	89.87	3443.99
25	5	0.9	0.85	0.95	1	Cuts	359.00	359.00	359.00
						Configs	580.00	580.00	580.00
						Seconds	389.34	389.34	389.34
25	20	0.9	0.85	0.95	10	Cuts	61.90	32.00	109.00
						Configs	4.30	2.00	7.00
						Seconds	111.63	26.27	301.54
30	25	0.9	0.85	0.95	10	Cuts	107.80	74.00	161.00
						Configs	6.60	2.00	16.00
						Seconds	602.44	166.34	1330.49
35	30	0.9	0.85	0.95	3	Cuts	112.00	93.00	131.00
						Configs	5.00	5.00	5.00
						Seconds	2597.00	1376.14	3406.53

Configs, which indicates the configurations (l) considered by the IRCG algorithm to prove optimality (see Sect. 4); *Seconds*, which contains the time (in seconds) required to solve problems to optimality. Notice that for each problem, there is a line for each of the indicators considered.

Tables are organized in such a way that only one, or at most two, of the problem-defining parameters are varied in each table. This choice should highlight how algorithm IRCG reacts at changes in a single parameter, while the others are blocked to reference settings.

In Table 3 we change the number of devices in the network (n). The table shows that algorithm IRCG is able to handle problems with up to 30 nodes within the given time limit. This value reduces to 20 when just a few destination devices are considered. A simple explanation exists for this phenomenon: according to Sect. 3, when the set of destinations is decreased, the number of possible configurations, and in turn the number of possible constraints, rapidly increases, making the problem more difficult. It is also interesting to observe that most of the difficulty seems to come from the increase of n itself, since (see the last three problems), the computation time rapidly increases even if the number of configurations considered and of cuts generated does not have such a rapid increase. However it has to be remarked that the number of activated cuts and configurations remains very small compared to L, the total number of constraints of type (10) and the total number of configurations N,

Table 4 Performance of the IRCG algorithm when the number of destinations r is varied. Statistics over ten instances

n	r	α	$q_{\min}$	$q_{\max}$	Solved	Indicator	Avg.	Min	Max
20	1	0.9	0.85	0.95	5	Cuts	109.60	2.00	491.00
						Configs	307.60	2.00	987.00
						Seconds	15.96	0.00	72.58
20	5	0.9	0.85	0.95	6	Cuts	452.67	65.00	660.00
						Configs	192.83	13.00	640.00
						Seconds	1431.34	89.87	3443.99
20	9	0.9	0.85	0.95	10	Cuts	177.10	32.00	425.00
						Configs	38.80	22.00	56.00
						Seconds	543.44	15.14	2642.79
20	14	0.9	0.85	0.95	10	Cuts	45.70	15.00	70.00
						Configs	5.80	4.00	8.00
						Seconds	73.71	3.75	199.31
20	19	0.9	0.85	0.95	10	Cuts	17.30	12.00	32.00
						Configs	1.00	1.00	1.00
						Seconds	9.39	1.97	30.46

Table 5 Performance of the IRCG algorithm when the reliability threshold α is varied. Statistics over ten instances.

n	r	α	$q_{\min}$	$q_{\max}$	Solved	Indicator	Avg.	Min	Max
20	5	0.95	0.85	0.95	5	Cuts	291.40	89.00	476.00
						Configs	436.80	113.00	611.00
						Seconds	1080.76	26.60	2781.83
20	5	0.9	0.85	0.95	6	Cuts	452.67	65.00	660.00
						Configs	192.83	13.00	640.00
						Seconds	1431.34	89.87	3443.99
20	5	0.85	0.85	0.95	9	Cuts	266.89	15.00	608.00
						Configs	69.56	1.00	309.00
						Seconds	597.71	0.53	2002.00

respectively (see Sect. 3). For instance, in the case of the problem with parameters $n = 20, r = 5, \alpha = 0.9, q_{\min} = 0.85$ and $q_{\max} = 0.95$, even considering the maximum values given in Table 3, less than 0.001% of the possible cuts and less than 4% of all configurations have been explicitly used to solve the instances. This result confirms that the cut generation strategy adopted, as well as the configurations ordering chosen are suitable.

Table 6 Performance of the IRCG algorithm when the range of possible probabilities associated with nodes ($q_{\min}$ and $q_{\max}$) is varied. Statistics over ten instances

n	r	α	$q_{\min}$	$q_{\max}$	Solved	Indicator	Avg.	Min	Max
20	5	0.9	0.88	0.93	5	Cuts	350.80	61.00	630.00
						Configs	147.40	29.00	297.00
						Seconds	659.73	81.12	1313.19
20	5	0.9	0.85	0.95	6	Cuts	452.67	65.00	660.00
						Configs	192.83	13.00	640.00
						Seconds	1431.34	89.87	3443.99
20	5	0.9	0.7	1	5	Cuts	322.40	170.00	458.00
						Configs	290.00	78.00	1070.00
						Seconds	887.04	55.53	2254.50

The number of destination devices is varied for the tests shown in Table 4. The results reported can be explained with the same arguments already used to justify the behavior of the IRCG algorithm in the tests of Table 3. Results in the table clearly show how the number of configurations considered, and the number of cuts generated rapidly decreases when the number of destination devices is increased.

In Table 5 we study how the indicators vary when the reliability threshold α is changed. As expected, the computational times and the number of generated configurations and cuts tend to grow as the required reliability α increases. The explanation is trivial: higher values for α lead to multicasting structures that are potentially very different from those obtained when reliability is not considered (what our algorithm IRCG in fact considers during the first iterations).

Table 6 is devoted to the study of how variations on the range of possible probabilities associated with nodes ($q_{\min}$ and $q_{\max}$) affect the performance of algorithm IRCG. The table suggests that the performance of the algorithm are almost independent on variations in the range of the probabilities assigned to the nodes, although the number of configurations considered seems to increase as the range of possible probabilities is enlarged.

It is finally worthwhile remarking the high variance that characterizes all the indicators reported in the tables. This suggests that the algorithm IRCG is in some sense not very robust, since its performance is strongly instance-dependent.

6 Heuristic algorithm

As seen in Sect. 5.4, the exact algorithm IRCG does not always succeed in finding a feasible solution in a given maximum computation time. In this case it is of interest to switch to a fast heuristic method that generates a feasible solution.

The heuristic algorithm (HPMPM) we propose is based on the idea that the reliability of any solution y obtained during the execution of the exact algorithm IRCG can be increased by adding suitable arcs to it. After having evaluated different arc insertion criteria, we implemented the following strategy in algorithm HPMPM.

Consider a solution y having an estimated reliability $\bar{\alpha} < \alpha$, calculated with the sequential procedure explained in Sect. 4, and a total cost c. The algorithm HPMPM calculates the estimate of the reliability $\bar{\alpha}'$ and the cost c' of a new solution y', obtained by adding a given arc (i, j) to the solution y. This procedure is performed for all arcs (i, j) not contained in y, but outgoing from an active node i of solution y. The insertion of arcs is carried on iteratively until the estimated reliability $\bar{\alpha}$ reaches the required threshold α.

It has to be remarked that, according to the WMA property, the insertion of an arc (i, j) implies also the insertion of the arcs covered by (i, j).

The key idea of the heuristic HPMPM is to choose the arc(i, j) that provides a maximum increase of reliability and at the same time a minimum increase of cost. Essentially the algorithm maximizes the ratio $\frac{\bar{\alpha}'-\bar{\alpha}}{c'-c+1}$.

In addition, algorithm HPMPM has been equipped with a useful post-processing phase, inspired by the procedure discussed in [22], which tries to eliminate certain arcs of the obtained solution in such a way that the reliability level does not fall below the threshold α. The novelty of our approach consists in performing the arc elimination by considering both the cost difference and the reliability difference. The arc elimination criterion is based substantially on the minimization of the ratio $\frac{\bar{\alpha}-\bar{\alpha}'}{c-c'+1}$.

When the maximum computation time is reached, the algorithm IRCG switches to the heuristic HPMPM and starts the arc insertion procedure on the last solution provided by the exact algorithm. Preliminary tests showed that this choice is usually the most convenient.

The heuristic algorithm HPMPM can be described formally as follows:

Step 0. $y =$ solution provided by algorithm IRCG, $\bar{\alpha} =$ reliability of y.
Step 1. $\forall$ (i, j) such that $y_{ij} = 0$, calculate the cost c' and the reliability $\bar{\alpha}'$ of the solution y' obtained by adding arc (i, j) to the support of y. Choose the arc (i, j) with the maximum ratio $\frac{\bar{\alpha}'-\bar{\alpha}}{c'-c+1}$ (if the maximum is 0 the shortest arc is chosen). Update $y := y'$ and $\bar{\alpha} := \bar{\alpha}'$.
Step 2. If $\bar{\alpha} < \alpha$ go to Step 1. Otherwise go to Step 3.
Step 3. $\forall(i, j)$ such that $y_{ij} = 1$, calculate the cost c' and the reliability $\bar{\alpha}'$ of the solution y' obtained by eliminating arc (i, j) from the support of solution y. Choose the arc (i, j) with the minimum ratio $\frac{\bar{\alpha}-\bar{\alpha}'}{c-c'+1}$.
Step 4. If $\bar{\alpha}' \geq \alpha$ update $y := y'$ and $\bar{\alpha} := \bar{\alpha}'$ and go to Step 3. Otherwise stop.

Table 7 contains some computational results obtained on problems with up to 32 nodes. For each problem three different random instances have been considered. Furthermore, in order to observe the performance of the heuristic HPMPM, the maximum computation time has been set to 10 s for the problems with $n = 12$ and $n = 15$, to 120 s for the problems with $n = 20$, $n = 30$ and for the first instance with $n = 32$ and finally to 7200 s for last two instances. The second and the third column of the table show the number of added and deleted arcs respectively. The most interesting results are contained in the columns four and five: the percentage gap between the heuristic upper bound and the best available lower bound and the percentage improvement provided by the post-processing phase. Column six indicates whether the lower bound used to compute the gap coincides with the optimum or not. Finally,

Table 7 Performance of the heuristic algorithm HPMPM

n	r	α	$q_{\min}$	$q_{\max}$	#Arcs(+)	#Arcs(−)	Gap (%)	Imp (%)	Optimal	Seconds
12	3	0.9	0.85	0.95	4	1	0.00	1.85	Yes	0.09
					8	4	13.40	7.73	Yes	0.21
					2	2	0.00	1.07	Yes	0.08
15	5	0.9	0.85	0.95	10	0	8.19	0.00	Yes	0.19
					1	2	4.23	14.16	Yes	0.17
					4	0	15.54	0.00	Yes	0.17
20	10	0.8	0.85	0.95	12	12	2.35	29.16	Yes	0.72
					4	4	0.00	4.72	Yes	0.70
					2	3	3.03	17.36	Yes	0.36
30	20	0.8	0.85	0.95	10	6	6.97	10.81	Yes	1.67
					5	5	0.00	5.25	Yes	2.29
					19	17	0.86	33.64	Yes	5.71
32	20	0.9	0.85	0.95	14	6	15.43	12.16	Yes	13.25
					12	1	41.29	0.01	No	15.60
					13	3	78.22	3.02	No	21.99

the last column shows the CPU-times (in seconds) required by the arc insertion and elimination procedures.

Table 7 shows that algorithm HPMPM succeeds in generating quickly a feasible solution. The number of inserted arcs is not very high, which means that the adopted selection strategy of the arcs is efficient. On the other hand the arc elimination phase is usually able to improve the last solution. It can be observed that the gap is moderate when the optimal value is available as a lower bound. This fact suggests that the large gaps of the last instances are mainly on the lower bound side. In addition, it can be expected that a more elaborated heuristic approach could lead to an improvement of the upper bounds. However this open issue definitely deserves further studies that we leave for future research.

7 Concluding remarks

A probabilistic variant of the minimum power multicast problem for wireless networks has been defined and studied in this paper. In the model proposed node failure probabilities for the transmission are explicitly considered. The problem has been formulated as an integer linear program, by means of connectivity constraints and a global reliability requirement. An exact algorithm, based on an iterative cut generation procedure, and an heuristic approach have been proposed. Experimental results suggest that problems of moderate size can be solved to optimality by the proposed exact algorithm and that larger instances can be efficiently handled by a suitable heuristic method.

References

1. Andrade, R., Lisser, A., Maculan, N., Plateau, G.: Telecommunication network capacity design for uncertain demand. Comput. Optim. Appl. **29**, 127–146 (2004)
2. Bhandari, V., Vaidya, N.H.: Reliable broadcast in wireless networks with probabilistic failures. In: IEEE Infocom Proceedings, pp. 715–723 (2007)
3. Bodlaender, H.L., Wolle, T.: A note on the complexity of network reliability problems. Technical Report UU-CS-2004-001, Utrecht University (2004)
4. Cagalj, M., Hubaux, J.P., Enz, C.: Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues. In: Proceedings of the Mobicom 2002 Conference, Atlanta, GA, September 23–28 (2002)
5. Chlamtac, I., Faragó, A.: A new approach to the design and analysis of peer-to-peer mobile networks. Wireless Netw. **5**(3), 149–156 (1999)
6. Das, A.K., Marks, R.J., El-Sharkawi, M., Arabshahi, P., Gray, A.: Minimum power broadcast trees for wireless networks: integer programming formulations. In: Proceedings of the IEEE Infocom 2003 Conference, San Francisco, CA, March 30–April 3 (2003)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability—A Guide to the Theory of NP-completeness. Freeman, New York (1979)
8. Guo, S., Yang, S.: Minimum-energy multicast routing in static wireless ad hoc networks. In: IEEE VTC Proceedings, pp. 3989–3993 (2004)
9. Guo, S., Yang, S.: A constraint formulation for minimum energy multicast routing in wireless multi-hop networks. Wireless Netw. J. **12**(1), 23–32 (2006)
10. Guo, S., Yang, S.: Energy-aware multicasting in wireless ad hoc networks: a survey and discussion. Comput. Commun. **30**, 2129–2148 (2007)
11. Leggieri, V., Nobili, P., Triki, C.: Minimum power multicasting problem in wireless networks. Math. Methods Oper. Res. **68**(2), 295–311 (2008)
12. Leino, J.: Optimal multicast routing in ad hoc networks. Technical Report (2002)
13. Liang, W.: Constructing minimum-energy broadcast trees in wireless ad hoc networks. In: Mobihoc Proceedings, pp. 112–122 (2002)
14. Min, M., Chinchuluun, A.: Optimization in wireless networks. In: Resende, M.G.C., Pardalos, P.M. (eds.) Handbook of Optimization in Telecommunication, pp. 891–915. Springer, Berlin (2006)
15. Min, M., Pardalos, P.M.: Energy efficient multicasting problem in wireless ad hoc networks. In: 10th WSEAS International Conference on Computing (2006)
16. Min, M., Prokopyev, O., Pardalos, P.M.: Optimal solutions to minimum total energy broadcasting problem in wireless ad hoc networks. J. Combin. Optim. **11**, 59–69 (2006)
17. Montemanni, R., Barta, J., Mastrolilli, M., Gambardella, L.M.: The robust traveling salesman problem with interval data. Transport. Sci. **41**(3), 366–381 (2007)
18. Montemanni, R., Gambardella, L.M.: Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. Comput. Oper. Res. **32**(11), 2891–2904 (2005)
19. Montemanni, R., Leggieri, V., Triki, C.: Mixed integer formulations for the probabilistic minimum energy broadcast problem in wireless networks. Eur. J. Oper. Res. **190**(2), 578–585 (2008)
20. Römer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wireless Commun. **11**(6), 54–61 (2004)
21. Rosenthal, A.: Computing the reliability of complex networks. SIAM J. Appl. Math. **32**, 384–393 (1977)
22. Wieselthier, J., Nguyen, G., Ephremides, A.: On the construction of energy-efficient broadcast and multicast trees in wireless networks. In: IEEE Infocom Proceedings, pp. 585–594 (2000)
23. Yuan, D.: Computing optimal or near-optimal trees for minimum-energy broadcasting in wireless networks. In: Proceeding of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, pp. 323–331 (2005)