Chapter 4

MIP formulations for a probabilistic Broadcasting Minimum Power problem

In this chapter, we consider a new variant of the Minimum Energy Broadcast (MEB) problem: the Probabilistic MEB (PMEB) [63]. As seen in chapter 2, the objective of the classic MEB problem is to assign transmission powers to the nodes of a wireless network is such a way that the total energy used in the transmission is minimized, while a connected broadcasting structure is guaranteed. In the new variant of the problem presented in section 4.1, we take into account a concept of reliability for the nodes with the goal of guaranteeing the broadcasting structure satisfying a chosen reliability level. Three mixed integer linear programming formulations for the new problem are presented in section 4.4, whereas efficient formulation-dependent methods for the solution of the different formulations are described in section 4.5. Computational results, aiming at ranking the proposed approaches, depending on the characteristics of the problems

under investigation, are proposed in section 4.6.

4.1 Introduction

We recall that in Ad-Hoc wireless networks, one terminal can communicate through wireless channels with another terminal using a single hop if the second terminal is within the transmission range of the first one, otherwise a multi-hop communication is required.

A crucial issue in this context consists in assigning a transmission power to each node in order to ensure connectivity of the network, while minimizing the total power expenditure over the network. We consider in this chapter the case of the Broadcast problem, in which a designated source terminal has to communicate with all the other nodes, and we assume to operate on a static network, i.e. distances among terminals are known in advance, together with the characteristics of the environment in which the terminals are operating. However, even if many contributions have been given to the deterministic models for the MEB problem none has considered nodes' reliability. The deterministic assumption represents a poor approximation of the reality; the terminals are, indeed, electronic devices that may be subject to a temporary damage or a permanent failure. This remark suggests the appropriateness of solving the problem as an optimization problem that takes into account the uncertain nature of nodes availability. This is a salient characteristic that makes the problem much more complex to solve than its classic, fully deterministic counterpart. To the best of our knowledge, no mathematical models explicitly incorporating the uncertain availability of the nodes have been proposed so far. We want to provide an original contribution in this direction. More specifically, we present three mixed integer linear programming formulations for a variant of the MEB

problem in which nodes failure is taken into account, and the optimal solution not only minimizes the total transmitting power over the network, but also guarantees a certain reliability level for the whole network, based on assumption about the reliability of the single terminals. The rationale is that in the reality one implicitly accepts that failures will happen in the devices and, therefore, the goal of the PMEB problem is to provide broadcasting structures robust enough to guarantee, in case of failure of some terminals, a reliable connectivity for the remaining terminals.

4.2 Related works

The Minimum Energy Broadcast (MEB) problem and its variants have already been the subject of many works. Both Cagalj *et al.* and Clementi *et al.* have shown its NP-hardness in [13] and [20], respectively. Althaus *et al.* have proposed a mixed integer linear programming model and have developed an exact approach, based on branch and bound, for its solution [1]. Alternative formulations have been suggested and solved to optimality by Das *et al.* in [25]. Montemanni *et al.* have proposed in [60] two mixed integer programming formulations together with a preprocessing rule and some valid inequalities [62]. Several heuristic methods have been also proposed in the literature. Wieselthier *et al.* have developed in [4] the well known BIP (Broadcast Incremental Power) algorithm. Metaheuristic approaches have been suggested by Marks *et al.* in [42] and by Das *et al.* in [23]. More recently, Lagrangian relaxation procedures have been proposed by Altinkemer *et al.* in [3] and by Yuan in [88]. Montemanni *et al.* have used the simulated annealing paradigm to find a near-optimal solution [61]. The cross-entropy metaheuristic has been also employed by Li *et al.* in [54] to define a new probabilistic approach called the RTO (Randomized Tree Op-

timization) algorithm. Another method has been proposed in [42] in which the initial solution is determined by means of a random tree generation within an evolutionary approach.

4.3 Network Model

The mathematical formulation of the MEB problem can be given considering the network as a directed complete graph $G = (V, A)$ where V represents the set of nodes corresponding to the terminals of the network and A is the set of arcs. As in chapter 2, a cost p_{ij} that corresponds to the power required to establish a link from node i to j is associated with each arc $(i, j) \in A$.

The MEB problem consists, therefore, in defining a range assignment r minimizing $\sum_{i \in V} r(i)$, subject to the constraints that a directed path exists from a source node s to all the other nodes in the network.

Another definition of the MEB problem can be given in terms of the optimal arborescence rooted at node s: for a node i and an arborescence T of G, let (i, i_T) be the maximum cost arc originated from i in T, i.e. $(i, i_T) \in T$ and $p_{ii_T} \geq p_{ij}$, for all $(i, j) \in T$. Due to the broadcasting property, the power cost of an arborescence T is then $c(T) = \sum_{i \in V} p_{ii_T}$. It is now easy to observe that an arborescence rooted at s is contained (not necessarily strictly contained) in any valid broadcasting structure. The MEB problem can, therefore, be described as the problem of finding the arborescence T with the minimum power cost $c(T)$.

In reality, some nodes of the network may fail due to technical problems or battery draining. This important aspect is neglected in the models

presented so far in the literature. We aim at starting to close this gap by presenting a model where a concept of reliability, connected with node failures, is taken into account.

In order to consider node failures, we associate with each node i of the network a value $q_i \in]0,1]$ representing the probability that node i will remain active (i.e. it will not fail) for the whole operating time of the network. The value of q_i has to be assigned by the decision makers, and reflects the reliability of each node. Typically it will depend on the physical characteristics of the area where each node is deployed. For example, in military applications a node i close to the enemy will have a high probability to be destroyed, and consequently a small value for q_i. Based on the same idea, a node i deployed in an impervious territory will have again a small value for q_i.

We can now formally define the Probabilistic Minimum Energy Broadcast (PMEB) problem as a MEB problem where a given minimum reliability level $\alpha \in]0,1]$ has to be achieved. Specifically, the reliability level of the paths from s to each other node i of the network will have to be at least α. A more formal definition of the PMEB problem will be given in the remainder of this section, after some important remarks.

The uncertain events characterizing our problem (i.e. node failures) are independent from each other, that is, if a node happens to fail, this does not affect the correct functioning of the other terminals of the network. It is also possible to observe that if nodes i and j have a probability values of functioning q_i and q_j respectively, then link (i,j) has a probability value of being available equal to the product $q_i q_j$. The same reasoning can be extended to paths: the probability of a multi-hop transmission path from node i to node j is equal to the product of the probabilities q_k associated

with the nodes involved in the path. In mathematical terms:

$$\mathcal{P}(P_{ij}) := \prod_{v \in P_{ij}} q_v,$$

where P_{ij} represents the path connecting i to j under investigation, and $\mathcal{P}$ is the probability function.

Finally, we would like to observe that, since $\mathcal{P}(P_{sj}) \leq q_s q_j$ for each $j \in V \setminus \{s\}$ (because s and j will be the extremes of each path from s to j), a feasible solution to the PMEB problem can exist if and only if $q_s q_j \geq \alpha$ for each $j \in V \setminus \{s\}$. We suppose again that there are no limits in the transmission power that can be assigned to the nodes, so that the arcs (s, j) are always elements of A.

4.4 Mixed integer linear programming formulations

For the PMEB problem, the decision variables are a set of continuous variables y representing the transmission power of each node, i.e. $y_i := r(i)$ for each $i \in V$, and a second set of integer variables z, that describe the optimal arborescence structure, and that are defined as follows:

$$z_{ij} := \begin{cases} 1 & \text{if } (i,j) \in T, \\ 0 & \text{otherwise,} \end{cases}$$

where T represents the arborescence connecting the source s with all the other nodes of the network.

4.4.1 F_1: Path-Based formulation

Let $\mathcal{U}$ represent the set of all infeasible paths originated in s. The generic element P of $\mathcal{U}$ verifies the condition that the product of the probabilities of the nodes involved in path P is less than the reliability level α, i.e.

$$\mathcal{U} := \{P : P \text{ is an } s-k \text{ path for } k \in V \setminus \{s\}, \text{ such that } \prod_{i \in P} q_i < \alpha\}. \quad (4.1)$$

Notice that the set $\mathcal{U}$ potentially has a huge cardinality and for this reason, it will be used in an implicit way in the method we propose, as described in the following sections.

The first MIP formulation F_1 that we propose for the PMEB is as follows:

$$\min \sum_{i \in V} y_i \quad (4.2)$$

$$s.t.$$

$$y_i \geq p_{ij} z_{ij} \qquad \forall\, (i,j) \in A \quad (4.3)$$

$$\sum_{\substack{(i,j) \in A, \\ i \in S,\, j \in V \setminus S}} z_{ij} \geq 1 \qquad \forall\, S \subset V,\ s \in S \quad (4.4)$$

$$\sum_{(i,j) \in P} z_{ij} \leq |P| - 1 \qquad \forall\, P \in \mathcal{U} \quad (4.5)$$

$$z_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \quad (4.6)$$

$$y_i \in I\!R^+ \qquad \forall\, i \in V. \quad (4.7)$$

Constraints (4.3) establish the relation between variables z and y. Constraints (4.4) represent the connectivity requirements: for each partition

(S, S^c) such that $s \in S$ and $S^c \neq \emptyset$, there must be at least an arc outgoing from S and incoming in S^c. Inequalities (4.5), ensure the reliability constraints across all the source-destination paths: if the source s and a destination t are connected by the path $P \in \mathcal{U}$, and, hence, by a path that do not respect the reliability level, then constraint (4.5) excludes the path P from any feasible solution. Finally, constraints (4.6) are the binary restrictions on the variables and constraints (4.7) define the domain definition for the continuous y variables.

Since the cardinality of $\mathcal{U}$ will be large already for small values of $|V|$, handling $\mathcal{U}$ efficiently becomes a critical issue. For this reason in our method for solving F_1, we will initially omit constraints (4.5), and we will generate them in a dynamic way only when they are violated. An analogous reasoning can be applied also to constraints (4.4), that are present again in a huge number. The procedure will be explained in details in section 4.5.1.

4.4.2 F_2: Cumulative Probability formulation

The idea behind our second PMEB model is to get rid of set $\mathcal{U}$ used in formulation F_1 and to introduce a new variable associated with each node k of the network expressing the probability value accumulated till that node along the arborescence. Such a variable can be defined as the product of the probability values of the nodes along the $s - k$ path. Instead, we will use here a continuous variable τ_k, for $k \in V$ equivalently defined as the sum of the logarithm of the probability values of the nodes along the $s - k$ path. The use of the logarithm will be clarified in a formal way in the next section 4.4.3.

It is worth noting that variables τ_k are, indeed, state variables since they depend on the values assumed by variables z, that are still present in this

formulation with the same meaning as in section 4.4.1. Also variables y have the same meaning as in 4.4.1.

The model F_2 can be, thus, formulated as follows:

$$\min \sum_{i \in V} y_i \tag{4.8}$$

$s.t.$

$$y_i \geq p_{ij} z_{ij} \qquad \forall\, (i,j) \in A \tag{4.9}$$

$$\sum_{\substack{(i,j) \in A, \\ i \in S,\, j \in V \setminus S}} z_{ij} \geq 1 \qquad \forall\, S \subset V,\ s \in S \tag{4.10}$$

$$\tau_i \leq \tau_j + \log q_i + M(1 - z_{ji}) \qquad \forall\, (i,j) \in A \tag{4.11}$$

$$\tau_s = \log q_s \tag{4.12}$$

$$\tau_i \geq \log \alpha \qquad \forall\, i \in V \tag{4.13}$$

$$\tau_i \leq 0 \qquad \forall\, i \in V \tag{4.14}$$

$$y_i \in I\!R^+ \qquad \forall\, i \in V \tag{4.15}$$

$$z_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A. \tag{4.16}$$

While most of the constraints are common with the model presented in subsection 4.4.1, some others are specific for the Cumulative Probability model and deserve some description. For each arc $(i,j) \in A$ constraint (4.11) updates, through a recursive process, the value of τ_i whenever node i is reached directly from node j. Clearly, such a constraint should be meaningful only if arc (i,j) belongs to the arborescence, otherwise it should become redundant. This is guaranteed by means of the term $M(1-z_{ji})$, that appears in the right hand side of the constraint. It dominates the inequality whenever $z_{ji} = 0$ for a big enough coefficient M (it suffices for M to take the value in (4.17)), and vanishes otherwise. Constraint (4.12) initializes

the recursive process by assigning $\log q_s$ to τ_s. The set of constraints (4.13) imposes the reliability requirement on each terminal of the network. Finally, constraints (4.14), (4.15) and (4.16) define variables domains. We notice that variables τ_i take nonpositive values since they are sums of logarithms of values belonging to the $]0, 1]$ interval.

This formulation uses, within constraints (4.11), a constant M whose value must be sufficiently big. In this specific context, it is possible to show that M can be set, for example, to

$$M := -(n+1) \min_{i \in V} \log q_i \tag{4.17}$$

in order to guarantee the reliability level satisfaction.

It is possible, however, to strengthen constraints (4.11) by defining a specific constant M for each node i, in such a way that constraints (4.11) become redundant when arcs (j, i) do not belong to the arborescence T. A choice for these constants, using constraints (4.13), can be the following:

$$M_i := -\log \alpha - \log q_i \qquad \forall i \in V. \tag{4.18}$$

By setting these constants to the previous values, constraints (4.11) of the Cumulative Probability formulation can be replaced by the constraints:

$$\tau_i \leq \tau_j + \log q_i + M_i(1 - z_{ji}) \qquad \forall\, (i, j) \in A. \tag{4.19}$$

This latter strengthened version of the constraints will be, thus, used in the formulation and for the experiments presented in section 4.6.

4.4.3 F_3: Multicommodity Flow formulation

The formulation presented in this section is based on a Multicommodity Flow model as described, for example, in [57]. It includes into the model an

explicit representation of all the paths connecting the source s to each node $d \in V$. For this goal, we do not use the spanning arborescence variables z and we introduce, for each node $d \in V$ and each arc $(i,j) \in A$, a new variable denoted by t_{ij}^d that takes value 1 if arc (i,j) is on the path from s to d, and 0 otherwise, in fact, it represents the value of the commodity d flowing through the arc (i,j). Variables y remains the same as before, and have the same meaning as in sections 4.4.1 and 4.4.2.

The model F_3 can be thus summarized as:

$$\min \sum_{i \in V} y_i \tag{4.20}$$

s.t.

$$y_i \geq p_{ij} t_{ij}^d \qquad \forall\,(i,j) \in A, \forall\, d \in V \setminus \{s\} \tag{4.21}$$

$$\sum_{j \in V \setminus \{s\}} t_{sj}^d = 1 \qquad \forall\, d \in V \setminus \{s\} \tag{4.22}$$

$$\sum_{i \in V \setminus \{d\}} t_{id}^d = 1 \qquad \forall\, d \in V \setminus \{s\} \tag{4.23}$$

$$\sum_{i \in V \setminus \{j\}} t_{ij}^d - \sum_{i \in V \setminus \{j\}} t_{ji}^d = 0 \qquad \forall\, d \in V \setminus \{s\},\ \forall\, j \in V \setminus \{s,d\} \tag{4.24}$$

$$q_d \prod_{i \in V} q_i^{\sum_{j \in V \setminus \{i\}} t_{ij}^d} \geq \alpha \qquad \forall\, d \in V \setminus \{s\} \tag{4.25}$$

$$t_{ij}^d \in \{0,1\} \qquad \forall\,(i,j) \in A,\ \forall\, d \in V \setminus \{s\} \tag{4.26}$$

$$y_i \in I\!R^+ \qquad \forall\, i \in V \tag{4.27}$$

The objective function (4.20) of this model remains unchanged with respect to the other formulations F_1 and F_2. Constraints (4.21) regulates the power emitted by node i based on the value of variables t. The sets of constraints (4.22)–(4.24) are the usual multicommodity flow equations

that guarantee, for each possible source-destination pair, the flow conservation on the source node, on the destination node, and on any intermediate node, respectively. We remark that $\sum_{j \in V \setminus \{i\}} t_{ij}^d = 1$ if node i is on the active path from s to d. Constraints (4.25) are the reliability requirements, and finally, constraints (4.26) and (4.27) are the limitations on the decision variables. The Multicommodity Flow formulation F_3 is a non-linear programming model because of the presence of the set of reliability constraints (4.25). Such constraints could be, however, linearized by making use of the logarithmic properties, as follows:

$$\begin{aligned} \log\left(q_d \prod_{i \in V} q_i^{\sum_{j \in V \setminus \{i\}} t_{ij}^d}\right) &= \log q_d + \sum_{i \in V} \log\left(q_i^{\sum_{j \in V \setminus \{i\}} t_{ij}^d}\right) \\ &= \log q_d + \sum_{i,j \in V,\, j \neq i} t_{ij}^d \log q_i. \end{aligned} \tag{4.28}$$

Constraints (4.25) can be, thus, replaced by the following linear constraints:

$$\sum_{i,j \in V,\, j \neq i} t_{ij}^d \log q_i + \log q_d \geq \log \alpha \qquad \forall d \in V \setminus \{s\} \tag{4.29}$$

These considerations above are the motivations on the use of the cumulation of the logarithms of the probability values for the nodes in formulation F_2 instead of the product of the probability values accumulated along the paths.

4.5 Algorithms for the MIP formulations

Here we present the methods for solving the different formulations presented in section 4.4.

4.5.1 Algorithm for F_1

The drawback of formulation F_1 is represented by the sets of constraints (4.4) and (4.5) that are in an intractable number, from a practical point of view. However, since only a small fraction of these constraints is saturated at optimality, we choose to solve the problem by means of an iterative approach. Namely, constraints (4.4) and (4.5) are initially not considered, and a subset of them will be inserted into the formulation only in case the current optimal solution violates them. This iterative mechanism will be repeated until a solution that respects all constraints (4.4) and (4.5) (both those explicitly added to the formulation and those implicitly checked) is found.

The procedure sketched above can be formalized by means of the following procedure:

Step 0: Let F_1' be formulation F_1 for problem $PMEB$ without constraints (4.4) and (4.5);

Step 1: Solve F_1', and let $(\overline{y}, \overline{z})$ be the optimal solution;

Step 2: If $\overline{z}$ violates a constraint ctr_4 of type (4.4) (the separation routine will be described later), then add ctr_4 to F' and go to Step 1;

Step 3: If $\overline{z}$ violates a constraint ctr_5 of type (4.5) (the separation routine will be described later), then add ctr_5 to F' and go to Step 1;

Step 4: $(\overline{y}, \overline{z})$ is the optimal solution of F_1 (and not only of F_1').

Notice that the procedure converges after a limited number of iterations since the number of inequalities (4.4) and (4.5) is, albeit significant, finite.

It is important to observe that a speed-up may be obtained by first considering the linear relaxation of F_1' in Step 2, and adding the corresponding violated constraints of type (4.4). In this way, many of the constraints might be added before considering the (more time consuming) integer program F_1'. In section 4.6 some results that confirm the correctness of this idea will be presented.

We however did not implement the speed-up since the computation times reported in section 4.6.3 indicate that the method based on F_1 is already the fastest one for some types of problems (without considering the linear relaxation first). On the other hand, the method is far from being the best one on problems with different characteristics.

Separation of inequalities (4.4) Once a solution $(\overline{y}, \overline{z})$ of F_1' is available, the presence of violated inequalities of type (4.4) of F_1 not inserted into F_1' can be easily detected. We use a set L containing all the nodes of the connected component of the source, that is for each node $i \in L$ the exists a directed path from the source to i using the arcs in which the values of the variables z are equal to 1. Two situations are possible at this point:

(i) if $|L| = |V|$, then no violated constraint of type (4.4) exists in the current solution $(\overline{y}, \overline{z})$;

(ii) if $|L| < |V|$, then a violated constraint of type (4.4) has been identified. Therefore, we can add the following violated inequality to F_1:

$$\sum_{i \in L, j \in V \setminus L} z_{ij} \geq 1.$$

Separation of inequalities (4.5) Once a solution $(\overline{y}, \overline{z})$ of F_1' is available, the presence of violated inequalities of type (4.5) of F_1 not inserted into

F_1' can be detected as follows. Since variables $\overline{z}$ define an arborescence (no violated constraint of type (4.4) exists because of the structure of the algorithm), it is enough to calculate, for each $k \in V \setminus \{s\}$, the following value:

$$R_{sk}^{\overline{z}} := \prod_{i \in P_{sk}^{\overline{z}}} q_i$$

where $P_{sk}^{\overline{z}}$ is the set of nodes encountered along the (unique) path from s to k on the arborescence defined by variables $\overline{z}$. In our current implementation of the algorithm, we visit the arborescence defined by variables $\overline{z}$, and as soon as we identify a path from s to k (with k possibly not a leaf) with $R_{sk}^{\overline{z}} < \alpha$, we add the constraint of type (4.5) corresponding to $P_{sk}^{\overline{z}}$ to model F_1'. After a constraint has been added, we do not stop the separation procedure, but we seek for other violated constraints, i.e. more than one constraint can be added at each invocation of the separation routine.

4.5.2 Algorithm for F_2

Similarly to what happens in the Path-Based formulation (see section 4.5.1), subtour elimination constraints (4.10) are in a very large number, too. Therefore, in order to solve the problem F_2, we need to run an iterative approach, starting with a relaxation of this formulation. The procedure we use, which is formally defined in the reminder of this section, is very similar to that described in section 4.5.1 for the Path-Based formulation. The main difference between the two solution approaches is that in this case we have only one set of critical inequalities to be added whenever violated (instead of the double set in case of the Path-Based formulation). The procedure can be formalized by means of the following procedure:

Step 0: Let F_2' be formulation F_2 for problem $PMEB$ without constraints

(4.10);

Step 1: Solve F_2', and let $(\overline{y}, \overline{z}, \overline{\tau})$ be the optimal solution;

Step 2: If $\overline{z}$ violates a constraint ctr_{10} of type (4.10) (the separation routine is analogous to that described in section 4.5.1 for the separation of inequalities (4.4)), then add ctr_{10} to F_2' and go to Step 1;

Step 3: $(\overline{y}, \overline{z}, \overline{\tau})$ is the optimal solution of F_2 (and not only of F_2').

Notice that the procedure converges after a limited number of iterations since inequalities (4.10) are in finite, although often huge, number.

An observation analogous to that reported in section 4.5.1 for the method based on formulation F_1 can be done here. In particular, a theoretical speed-up for the method might be obtained by considering first the linear relaxation of F_2' in step 3, for the generation of violated constraints (4.10). However, the results we will report in section 4.6, clearly indicate that this is not the case for the method based on formulation F_2.

4.5.3 Algorithm for F_3

The Multicommodity Flow formulation may have a large number of variables but it does not have critical constraints (like (4.4) and (4.5) in F_1 and (4.10) in F_2) that impose the development of a specific solution technique. Formulation F_3 can be, thus, directly solved by any mixed integer linear programming solver.

4.6 Experimental Results

This section presents the computational experience carried out with the exact methods described in section 4.5. Two different types of experiments will be discussed, covering the following aspects:

- how many constraints of type (4.4) and (4.5) (respectively (4.10)) are generated during the execution of the method based on formulation F_1 (respectively F_2);
- computation times of the three methods: we want to estimate the largest problem which is possible to solve with the methods we propose, and at the same time understand which is the most promising approach, depending on the characteristics of the problem under investigation.

First of all, we describe the characteristics of the benchmarks used for the experiments.

4.6.1 Benchmark description

No benchmark is available from the literature, being the problem treated here for the first time. We have, therefore, generated a set of random instances, trying to produce realistic scenarios.

The nodes have been chosen uniformly in a 5000×5000 grid and the probability that any of the nodes is functioning is assumed to be uniformly distributed in the interval [0.85, 0.95]. These values should be reasonable for real-life applications. Moreover, the value of the coefficient κ, which models signal propagation, has been set to 2.

The three methods described in section 4.5 have been implemented in C and the experiments have been carried out on an Intel Celeron 1.3 GHz / 256 MB machine. The callable library version of CPLEX 9.0 has been used as mixed integer programming solver. Ten random instances have been generated for each problem considered, and a maximum computation time of 3600 seconds has been allowed for each instance.

Table 4.1: Average number of constraints generated while solving the Path-Based formulation F_1 and the Cumulative Probability formulation F_2.

		F_1		F_2
$\|V\|$	α	(4.4)	(4.5)	(4.10)
10	0.50	2.75	0.50	0.00
10	0.60	7.75	4.10	0.00
10	0.70	28.50	39.60	0.00
10	0.80	94.00	46.60	0.00
15	0.50	11.50	0.60	0.00
15	0.60	42.75	44.60	0.00
20	0.50	17.00	23.00	0.00
20	0.60	43.75	82.40	0.00

4.6.2 Number of constraints added

In Table 4.1, we present, for a subset of the problems we will consider in section 4.6.3, the number of constraints (4.4) and (4.5) generated while solving the Path-Based formulation F_1 as described in section 4.5.1, and the number of constraints (4.10) generated while solving the Cumulative Probability formulation F_2 as described in section 4.5.2. In Table 4.1, we report, for each problem considered, the average number of constraints generated.

From Table 4.1, it can be observed how, during the solution of formulation F_1, a considerable number of constraints (4.4) and (4.5) are generated. Moreover, a weak correlation seems to exist among the number of constraints generated for the two families. This result suggests that a speed-up for the solution method described in section 4.5.1 may be obtained by considering the linear relaxation of F_1 for the generation of constraints (4.4) (as suggested in section 4.5.1).

Table 4.2: Computational results for the methods in section 4.5.

		Path–Based F_1			***Cumulative Probability*** F_2			***Multicommodity Flow*** F_3		
$\|V\|$	α	T (sec)	σ (sec)	OOT	T (sec)	σ (sec)	OOT	T (sec)	σ (sec)	OOT
10	0.50	**0.58**	0.71	-	4.67	10.92	-	1.56	1.54	-
10	0.60	**1.55**	2.14	-	3.11	5.71	-	2.26	2.08	-
10	0.70	41.38	52.21	-	14.67	18.71	-	**0.46**	0.70	-
10	0.80	309.84	536.32	-	54.55	51.43	-	**0.05**	0.04	-
15	0.50	**4.64**	3.02	-	65.68	91.15	-	58.10	28.78	-
15	0.60	237.11	540.40	-	459.35	593.40	-	**109.66**	80.38	-
15	0.70	2338.25	1558.93	5	2097.16	1580.71	4	**4.02**	5.56	-
15	0.80	-	-	10	2935.27	1330.43	8	**0.074**	0.01	-
20	0.50	**365.95**	626.78	-	2017.29	1630.16	5	2863.30	952.29	5
20	0.60	**2032.40**	1555.29	5	2710.02	1367.56	7	2267.56	1370.81	5
20	0.70	3364.93	964.74	9	3269.61	991.38	9	**93,72**	200.91	-
20	0.80	-	-	10	-	-	10	**0.21**	0.01	-
25	0.70	-	-	10	-	-	10	**949.33**	1240.36	1
25	0.80	-	-	10	-	-	10	**0.42**	0.02	-
30	0.70	-	-	10	-	-	10	**1809.67**	1791	5
30	0.80	-	-	10	-	-	10	**0.78**	0.05	-

Even more interesting is the situation for constraints (4.10), generated while solving formulation F_2: none of these constraints is generated during the experiments summarized in Table 4.1. The results suggest that considering the linear relaxation of F_2 first, to generate constraints (4.10) in the algorithm discussed in section 4.4.2, would not improve the overall

computation times of the method.

4.6.3 Computation times

Computational results for the algorithms discussed in section 4.5 are summarized in Table 4.2. For each method and for each problem considered we report the average T and standard deviation σ for the execution time (in seconds) and the number of instances not solved to optimality in the given time limit (OOT, out of time). When not all the problems are solved to optimality, only the instances solved to optimality concur to the computation of T and σ. Different values for the reliability threshold of the network α are finally considered. For each problem considered, the best value for T is in bold.

From the results reported in Table 4.2, the exact method based on the Path-Based formulation F_1 appears to be the most efficient approach for small networks (i.e. with at most 15 nodes) and for low values of the reliability threshold α. On the other hand, as the value of α increases, the approach based on the Multicommodity Flow formulation F_3 outperforms by far the other methods, reaching the point of becoming the only method able to solve many of the problems in the given time limit.

It is also interesting to observe how, for most of the problems, the average computational time required to solve the Multicommodity Flow model F_3 decreases as the value of α increases. When α increases, several paths are preliminarily discarded because the product of the probabilities associated with their nodes does not reach the threshold.

A final remark is about the potential speed-up for the method based on model F_1, achievable by considering the linear relaxation of the formulation

Table 4.3: Additional computational results for the Multicommodity Flow formulation F_3.

$\|V\|$	α	T	OOT
25	0.75	2.923	-
30	0.75	47.47	-
35	0.75	90.59	-
40	0.75	936.22	2
45	0.75	1810.50	3
50	0.75	2788.13	5

first while generating violated constraints (4.4). Even if such a speed-up is likely to exist (see section 4.6.2), it would definitely not close the gap between the performance of the methods based on F_1 and F_3 for the problems where the latter is the fastest method.

This attractive performance of the Multicommodity Flow model F_3 suggests to solve larger problems. Indeed, Table 4.3 summarizes the average computational times (and number of instances not solved to optimality) for test problems with up to 50 nodes by setting a constant value of 0.75 for α. The results show how both the computational times T and the number of instances not solved within the required amount of time OOT increase quite drastically as the number of nodes increases. This is related to the explosion in size of formulation F_3. Nevertheless, the method based on model F_3 remains the only one, among those considered, which is able to handle problems with up to 50 nodes in the given time.

4.7 Conclusions

In this chapter we have studied the Minimum Broadcast problem for Ad-Hoc wireless and sensor networks in probabilistic settings. The possible failure of any node in the network is considered explicitly within the mathematical representation of the problem, in order to provide more robust solutions with a given level of reliability. We proposed three different mixed integer linear programming formulations for the problem, and we developed an efficient solution approach for each of them.

Experimental results, aiming at understanding how the different methods perform, have finally been presented. These experiments, carried out on instances with up to 50 nodes, suggest that one method dominates the other two, when reasonable reliability levels are considered.