Journal of Applied Operational Research (2012) **4**(3), 110–124
© Tadbir Operational Research Group Ltd. All rights reserved. www.tadbir.ca

ISSN 1735-8523 (Print), ISSN 1927-0089 (Online)

Robust multicasting on stochastic wireless actuator networks: an algorithmic approach

Nihat Engin Toklu and Roberto Montemanni *

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Manno, Switzerland

Abstract. In this paper, we study the minimum power multicasting problem where the transmission power required for one network terminal to transmit to another is subject to uncertainty. We present our algorithmic approach which executes a classical mixed integer linear programming approach on the problem and then executes algorithmic procedures on the solution to improve its robustness. We then show our experimental procedures which test our approach on different levels of uncertainty and study the trade-off provided by the approach between solution cost and robustness. We also mathematically verify the correctness of the sampling method used in the experiments.

Keywords: wireless networks; minimum power multicasting; robust optimization; mathematical programming

Received September 2011. Accepted April 2012

Introduction

Minimum power multicasting problem is a problem faced for remotely commanding actuators which operate where the conditions are not human friendly. Another situation where this problem is faced is the establishment of communication in areas where the existing infrastructure is damaged by natural disasters (Leggieri *et al*., 2008). *Terminals* being the devices which are used for establishing the network, in this study, we consider the applications in which a source terminal has to do multicasting to a set of destination terminals. The destination terminals can either receive the message sent from the source terminal directly, or via intermediate terminals acting as routers (Rappaport, 1996). Also, since the messages are sent by multicasting (via the antennea attached to the terminals), several terminals can receive data from the same terminal if they are in the range of the transmitting terminal's coverage area. This is known as the wireless multicast advantage (Wieselthier *et al*., 2002). Wireless multicast advantage is illustrated in Figure 1.

* *Correspondence*: Roberto Montemanni, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, CH-6928 Manno, Switzerland. E-mail: roberto@idsia.ch

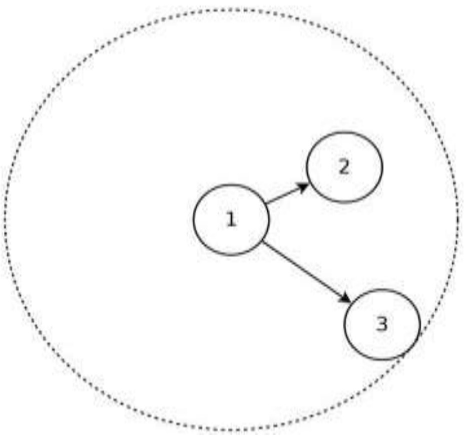

Fig. 1. The wireless multicast advantage. In this example, both terminals 2 and 3 are within the coverage area of terminal 1. Therefore, terminal 1 is multicasting to terminals 2 and 3.

Because the terminals depend on small batteries, it is important to find routing paths which depend on smaller coverage areas, minimizing the power usage on terminals, thus allowing the network to survive longer. This problem is known as minimum power multicasting problem (MPMP), where the goal is to find a coverage area for each terminal such that all destination terminals are reached from the source and the total transmission power is minimized (Leggieri, 2008). MPMP is illustrated in Figure 2.

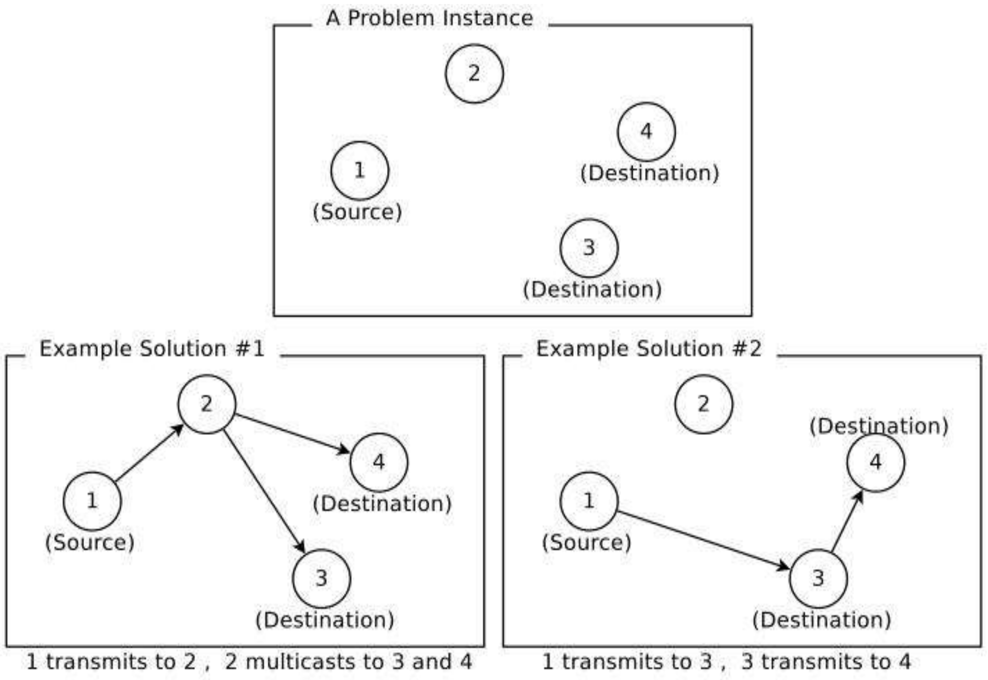

Fig. 2. An example instance for minimum power multicasting problem (MPMP), and two example solutions, which are obtained according to different decisions on coverage areas of terminals. The total transmission power, which is to be minimized, can vary significantly between such solutions.

In traditional optimization models, it is assumed that the data of the problem is known exactly. Because of unforeseen (or ignored) events, however, the actual data of the problem can differ and the optimum solution of the original model can turn out to be far from optimal, or even infeasible in reality (Ben-Tal and Nemirovski, 1998; Ben-Tal and Nemirovski, 2000). An early study on optimization against uncertainty can be found in (Soyster, 1973). Robust optimization (Bertsimas and Sim, 2003; Bertsimas and Sim, 2004; Babonneau *et al.*, 2010) is a methodology which assumes that there are data with uncertainty in the problem and those data with uncertainty reflect into the robust optimization model as intervals or sets, instead of exactly known numbers. An example of robust optimization applications in stochastic processes can be found in (Bertsimas *et al*., 2010).

In this paper, we study MPMP and we also consider that the data of the problem, i.e. the transmission power required for the network terminals to transmit to others (shortly, power requirement values), are subject to uncertainty. This uncertainty represents the fact that the power requirement values are affected by weather conditions and errors in the measurements of the distances between the network terminals. Engineers are in charge of estimating this uncertainty.

On the version of MPMP with uncertain power requirement values considered here, due to some technical reasons, it is not straightforward to apply the method described in (Bertsimas and Sim, 2003) and to represent the robust counterpart model as a mixed integer linear programming formulation. Therefore, in this study, we present our three-steps approach. The three-steps approach solves the problem by using a mixed integer linear programming formulation and constructs an early topology in the first step; then, based on the early topology, estimates the criticalities of the terminals in the second step; and reinforces the protections of the critical terminals against the uncertainty in the third step. Since step one is equivalent to a classical approach used in practice (see, for example, (Leggieri *et al*., 2008; Montemanni and Mahdabi, 2011)), the three-steps approach can be seen as an extension to the classical approach, which adds two steps implementable as polynomial time procedures, therefore does not change the complexity class of the overall method, and improves the robustness of solutions.

The structure of this paper is as follows: in section 2, the definition of the problem is made. In section 3, our three-steps approach is described. Section 4 presents our experimental results. Section 5 provides further formulations which can be used to validate the results found in the experiments. Conclusions are finally drawn in section 6.

Problem definition

Let us have a graph $G = (V, A)$ where V is the set of nodes and $A = \{(i, j) \mid i, j \in V\}$ is the set of arcs. Within the graph G, each node $i \in V$ represents a terminal of the network. Each arc $(i, j) \in A$ represents a transmission link between the terminals represented by the nodes i and j. To include terminal j within its coverage area (i.e. to be able to transmit to terminal j), the transmission power spent by terminal i has to be at least the power requirement value between terminals i and j. These power requirement values are denoted by $r_{ij} \mid i, j \in V$. The power requirement value for a terminal to reach itself is always set as 0 (i.e. $r_{ii} = 0\ \forall i \in V$). The objective of MPMP is to decide a transmission power ρ_i for each node $i \in V$ in such a way that a message produced by the source terminal $s \in V$ can reach (directly or via multi-hop communications) the destination terminals $D \subseteq (V \setminus \{s\})$, with the minimum total cost $\sum_{i\in V} \rho_i$. We say that terminal i reaches reaches terminal j when $\rho_i \geq r_{ij}$. Notice that it is not necessary to reach terminals which are not destinations (i.e., which are not in the set D). They can be however used as routers to decrease the total transmission power spent over the network.

In this study, we consider that the problem has a stochastic behavior, having uncertain r_{ij}. When the uncertainty is introduced into the power requirement values, r_{ij} are not numbers anymore, but they are intervals which represent possible power requirement values which can be encountered. Therefore, for each $(i, j) \in A$, r_{ij} becomes $[\underline{r}_{ij}; \overline{r}_{ij}]$. It is assumed that the actual power requirement values can be any value from their intervals according to a uniform probability distribution. A scenario being a realization where single power requirement values are picked from their intervals, we define robustness as having as many destination nodes reached in as many scenarios as possible. Therefore, we define the objective as a trade-off between the maximization of the scenarios in which the destination nodes are reached and the minimization of the total transmission power of the network topology returned.

The three-steps approach

In this section, we introduce our three-steps approach. At the first step, the transmission links are decided by mathematical programming, for which the formulations are widely available in the literature. Then, at the second step, it decides which terminals are critical. In the final step, *protection* being the additional power spent to make sure a connection is established, protections on the less critical nodes are decreased and the protections on the more critical nodes are increased. The basic idea is to gain additional power from less critical nodes and use that power to reinforce the critical parts of the network to have a more robust same-cost solution.

The behavior of the approach is configured by the decision maker, by using two parameters:

- $\alpha \in [0; 1]$, which is the reference protection ratio used for generating the basic structure of the network;
- $\beta \in [0; 1]$ which is the minimum protection ratio that a node should at least have.

The reference protection value α affects the solution cost (i.e. the total transmission power of the network). In more details, as α goes towards 1, the solution cost goes higher. Therefore, the parameter α is used as a setting the trade-off between cost and robustness. The β parameter, which is a number less than α, configures the behavior of the three-steps approach during the adjustment of the protections on the nodes: it ensures that while the protections of the critical nodes are increased, the less critical nodes still have some protection.

The rest of this section describes each section in more details.

Step one

The purpose of this step is to have an early network topology by deciding the transmission links between the terminals. In more details, in this step, we decide which terminals should transmit to which other terminals and which terminals should not transmit at all. The latter steps will take these decisions as basis and then the final transmission powers will be decided.

To have the early network topology, a mixed integer linear programming formulation is used. The formulation is based on an adaptation of the network flow model proposed in (Montemanni and Mahdabi, 2011).

This step depends on α, the reference protection ratio parameter which is given by the decision maker. By using this reference protection value, a special scenario is constructed where all power requirement values are set as $r^{\alpha}_{ij} = \underline{r}_{ij} + \alpha(\overline{r}_{ij} - \underline{r}_{ij})$ and the mathematical model is executed on top of this special scenario.

We now define an array v^i for each node i. Each destination node j of a node i is stored in v^i, where nodes are sorted in a non-decreasing order according to r^{α}_{ij}. The first element of v^i is always node i itself. To make the v^i concept clear, we provide an example in Figure 3. In addition, when we say k-th destination of node i, we refer to the node v^i_k.

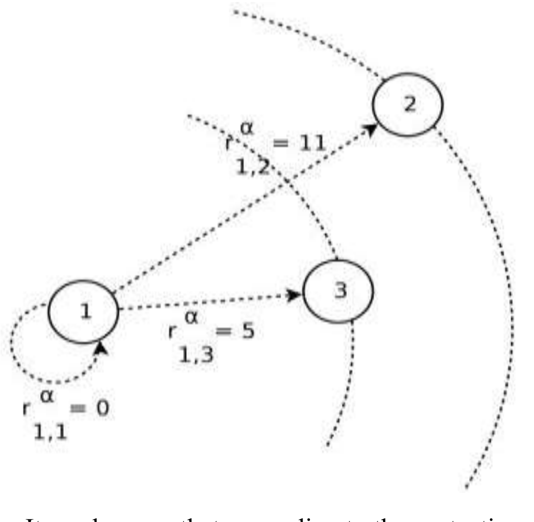

Fig. 3. A portion of an example instance. It can be seen that, according to the protection vaslue α, the power requirement of the arc (1, 2) is greater than the power requirement of the arc (1, 3) (i.e. $r^{\alpha}_{1,2} > r^{\alpha}_{1,3}$). Therefore, for node 1, we say $v^1 = \{1, 3, 2\}$.

For the model, we define x_{ik} as the binary decision variable which tells whether the node i should configure its transmission power enough to reach its k-th destination exactly. We also define y_{ij} as a variable which represents the flow from node i to node j. These variables are used to enforce the connectivity of the network (see (Montemanni and Mahdabi, 2011)). Using these variables, the model can be formulated as follows:

$$
\begin{array}{llrr}
\text{minimize} & \displaystyle\sum_{i\in V}\sum_{k=1}^{|V|} r^{\alpha}_{i,v^i_k} x_{ik} & & (1)\\
\text{subject to} & \displaystyle\sum_{k=1}^{|V|} x_{ik} = 1 & \forall i \in V & (2)\\
& \displaystyle(|V|-1)\sum_{k=l\,:\,v^i_l=j}^{|V|} x_{ik} \geq y_{ij} & \forall (i,j)\in A & (3)\\
& \displaystyle\sum_{j\in V} y_{ji} - \sum_{j\in V} y_{ij} = \begin{cases} -|D| & \text{if } i = s\\ 1 & \text{if } i \in D\\ 0 & \text{otherwise}\end{cases} & \forall i \in V & (4)\\
& 0 \leq y_{ij} \leq |V| & \forall (i,j)\in A & (5)\\
& x_{ik} \in \{0,1\} & \forall i \in V, \forall k \in \{1,2,\ldots,|V|\} & (6)
\end{array}
$$

Constraints (2) impose that only one x_{ik} has to be set as 1 for each node i, so that one node will not be able to have more than one transmission power level simultaneously assigned to it. Constraints (3) provide the connection between x variables and y variables. Constraints (4) impose that the network flow indicated by variables y must form an arborescence routed at s and reaching all the destination nodes in D. Constraints (5) and (6) specify the domains of y and x variables, respectively.

After having solved the model above, we have the basic structure for our network, given by the x_{ik} variables of the model. Also, an early decision about the transmission powers is made. We denote these transmission powers as $\hat{\rho}_i$, and we define them as follows:

$$\hat{\rho}_i := \sum_{k=1}^{|V|} r^{\alpha}_{i,v^i_k} x_{ik} \tag{7}$$

For each node $i \in V$ we also define λ_i as the most distant node reached by i while transmitting at power $\hat{\rho}_i$ (this information will be used in step three):

$$\lambda_i := j \text{ such that } r^{\alpha}_{ij} = \hat{\rho}_i \tag{8}$$

We finally define the total transmission power C of the topology found during this step as follows:

$$C := \sum_{i=1}^{|V|} \hat{\rho}_i \tag{9}$$

Notice that the final network topology returned after step three (see Section 3.3) will not exceed C.

An interesting feature of the three-steps approach is that, while it is based on the mathematical programming method presented above, the additional steps it introduces are actually independent from the formulations used. This means that the first step is a component which can be chosen by the decision maker or the implementer according to the factors like ease of implementation or the execution time required by the formulation. In addition to alternative mathematical model formulations, it could be possible to plug heuristics too into the first step as components. Such heuristic methods are widely available in the literature (Montemanni and Mahdabi, 2011; Montemanni *et al*., 2005; Wieselthier *et al*., 2002). Using a heuristic for the first step, or combining the second and third steps in another way with a heuristic and analyzing the effects on the results is left for future studies.

Step two

At the previous step, the basic structure of the network was decided and the early transmission powers $\hat{\rho}_i$ were prepared. At this step, the purpose is to estimate the criticalities of the nodes, so that the next step can adjust their protections against uncertainty. The criticality of a node *i* is denoted by I_i, which is a non-negative real number. I_i values mean higher criticalities, while $I_i = 0$ means that the criticality of node *i* is at minimum.

We now propose a scoring system where the idea is that when the probability to reach a destination terminal is low, the terminals carrying data to that destination should be considered critical. The estimation of criticalities can be done by recursively traversing from a source terminal, picking the most reliable paths leading to each destination terminal (i.e. for each destination node $d \in D$, picking the path which has the largest probability to reach *d*). Later, within each most reliable path *P* reaching a node $d \in D$, for each node $i \in P$, I_i is increased by 1 minus the probability of reaching the destination node *d* from the source *s* via the path *P*. The algorithmic listing of this procedure is shown in Algorithm 1.

Algorithm 1 The algorithm for estimating the criticality values

```
1: initialize $current\_path = \{\}$
2: initialize $most\_reliable\_path(i) = \{\}\ \ \forall i \in V$
3: initialize $most\_reliable\_path\_score(i) = 0\ \ \forall i \in V$
4: initialize $I_i = 0\ \ \forall i \in V$
5: call $find\_most\_reliable\_paths(s, 1)$
6: for all $d \in D$ do
7:    for all $i \in (most\_reliable\_path(d) \backslash \{d\})$ do
8:       $I_i \leftarrow I_i + |V| + (1 - most\_reliable\_path\_score(i))$
9:    end for
10: end for
11: procedure find_most_reliable_paths(i, score) is:
12: if $score > most\_reliable\_path\_score(i)$ then
13:    push $i$ into $current\_path$
14:    $most\_reliable\_path(i) \leftarrow current\_path$
15:    $most\_reliable\_path\_score(i) \leftarrow score$
16:    for all $j \in (V \backslash current\_path)$ do
17:       if $\hat{\rho}_i \geq \overline{r}_{ij}$ then
18:          $reaching \leftarrow 1$
19:       else if $\hat{\rho}_i \leq \underline{r}_{ij}$ then
20:          $reaching \leftarrow 0$
21:       else
22:          $reaching \leftarrow (\hat{\rho}_i - \underline{r}_{ij})/(\overline{r}_{ij} - \underline{r}_{ij})$
23:       end if
24:    end for
25:    call $find\_most\_reliable\_paths(j, score \cdot reaching)$
26:    pop $i$ from $current\_path$
27: end if
```

Although in experiments, the time consumed by Algorithm 1 was seen to be insignificant, it is classified as an exponential-time procedure. Therefore, we propose an alternative procedure, which does the estimations in polynomial time. For the polynomial-time procedure, we first introduce the concept of dependency. Assuming that the transmission powers are $\hat{\rho}_i$ and the power requirement values are $r^{\beta}_{ij} = \underline{r}_{ij} + \beta(\overline{r}_{ij} - \underline{r}_{ij})$, if node *j* receives data from node *i*, we say node *j depends* on node *i*. Also, if node *k* receives data from node *k*, we say node *k depends* on nodes *i* and *j*.

We now define the concept of critical dependency. If node *j* receives data from node *i*, but the connection (*i*, *j*) fails in the worst case scenario, we say that node *j critically depends* on node *i*. In this situation, we also say that node *i weakly reaches* node *j*. On the other hand, if the connection (*i*, *j*) is also established in the worst case scenario, then we say that node *j non-critically depends* on node *i* and that node *i strongly reaches* node *j*. The formulations of the queries for strongly or weakly reaching are as follows:

$$strongly\_reaches(i,j) := \begin{cases} 1 & \text{if } \hat{\rho}_i \geq \overline{r}_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$weakly\_reaches(i,j) := \begin{cases} 1 & \text{if } \hat{\rho}_i \geq r^{\beta}_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Now we define a function which returns a ratio indicating the quality of the connection between node *i* and node *j*. If node *j* is reached strongly by node *i*, then the quality is 1, which is the maximum quality. If the transmission power of node *i*, i.e. ρ_i is equal to the power requirement value imposed by the underlying scenario, i.e. r^{β}_{ij}, then we say the quality is at minimum, therefore, 0. According to the position of ρ_i in the interval $[r^{\beta}_{ij}; \overline{r}_{ij}]$, we assign a quality between 0 and 1. If node *i* can not reach node *j*, then there is no quality to assign. The calculation of reaching quality is formulated as follows:

$$quality(i,j) := \begin{cases} 1 & \text{if } \hat{\rho}_i \geq \overline{r}_{ij} \\ \dfrac{\left(\frac{\hat{\rho}_i - \underline{r}_{ij}}{\overline{r}_{ij} - \underline{r}_{ij}}\right) - \beta}{1-\beta} & \text{if } \underline{r}_{ij} \leq \hat{\rho}_i < \overline{r}_{ij} \end{cases} \tag{12}$$

Now we define the polynomial-time procedure which estimates the criticalities of the nodes. Similar to the procedure shown in Algorithm 1, the whole recursive criticality estimation process is started by calling the criticality estimation function with the starting node argument *s* (i.e. the source node). After visiting node *i*, for each non-visited reachable node *j*, the function calls itself with the starting node argument *j*. While visiting node *j* from node *i*, if the connection (*i*, *j*) is critical, the function adds node *i* to the stack of critical dependencies. Also, according to the reaching quality from *i* to *j*, the criticality values of each node in the stack are increased. In particular, as the reaching quality for the visited node goes lower, the increasing amounts for the criticality values of the nodes in the stack goes higher. If node *j* is a destination node, the criticalities are multiplied by a factor (chosen as |*V*| after some preliminary tests), to express the importance of reaching a destination. In more details, if node *i* weakly reaches node $i' \in D$ and if node *j* weakly reaches node $j' \notin D$; even if both nodes *i* and *j* are depended by the same number of nodes (one weakly reached node for each), node *i* should be classified as more critical, because the node it reaches is a destination node. Therefore, its criticality (and the criticalities of its dependencies) is multiplied by |*V*|. When a node is done being traversed, it is removed from the critical dependencies stack, if it was there. The function is summarized in Algorithm 2.

Algorithm 2 The polynomial-time algorithm for finding the criticalities

```
1: initialize $critical\_dependencies = \{\}$
2: initialize $I_i = 0 \quad \forall i \in V$
3: call $estimate\_criticalities(s, 1)$
4: procedure $estimate\_criticalities$(starting node $i$, quality $q$) is:
5: mark node $i$ as reached.
6: $score \leftarrow$ ( $|V|$ if $i \in D$; 1 otherwise )
7: $I_d \leftarrow I_d + score(1 - q) \quad \forall d \in critical\_dependencies$
8: for all $j \in V$ do
9:   if node $j$ was not marked as reached already then
10:    if $strongly\_reaches(i, j)$ then
11:      call $estimate\_criticalities(j, quality(i, j))$
12:    else if $weakly\_reaches(i, j)$ then
13:      push $i$ into $critical\_dependencies$
14:      call $estimate\_criticalities(j, quality(i, j))$
15:      pop $i$ from $critical\_dependencies$
16:    end if
17:  end if
18: end for
```

Since, the function *estimate_criticalities* marks each visited node and a marked node is not visited again, this function is executed for $|V|$ times. While visiting a node, the function scans for all other nodes to see which ones are reachable. Therefore, this whole recursive process ends in $O(|V|^2)$ time. This shows that executing this procedure is very fast in practice.

By using the polynomial-time estimation procedure, instead of the exponential-time one, we give up the guarantee that the criticalities will be estimated according to the most reliable paths. This is because the polynomial-time procedure visits each node once: it does not consider alternative paths and does not pick among them. According to our experiments, there was no significant difference between the results of these two procedures.

The result of this step is the I_i values, each indicating the criticality of a node. These criticality values serve as priority indicators for step three, during the adjustment of transmission powers.

Step three

The purpose of this step is to make the final decisions on the transmission powers. The idea of this step is that, according to the criticality values provided by the second step, the critical nodes should be protected more against the uncertainty, while the protections of the less critical nodes can be decreased. The additional constraints that the total transmission power over the network can not exceed the cost upper bound C found in step one.

This step is expressed by a linear programming formulation, where the transmission powers ρ_i are variables, while the criticality values I_i (defined in step two) and nodes λ_i (defined in step one) are constants. The formulation is as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i \in V} (I_i + 1)\rho_i & & (13) \\
\text{subject to} \quad & r^{\beta}_{i,v^i_k} \leq \rho_i \leq \overline{r}_{i\lambda_i} & \forall i \in V,\ k \in \{1, 2, ..., l : v^i_l = \lambda_i\} \quad & (14) \\
& \sum_{i \in V} \rho_i \leq C & & (15) \\
& \rho_i \in \mathbb{R} & \forall i \in V \quad & (16)
\end{aligned}$$

The objective (13) states that the protections of the nodes against the uncertainty should be maximized, with the priorities given to the critical nodes. The constraints (14) define the lower and upper bounds for the transmission powers of the nodes, lower bounds stating that all terminals reached by terminal i in the reference scenario should have at least the minimum protection imposed by β. The constraint (15) imposes that the total power consumption of the network can not exceed that of the solution retrieved in step one. The constraint (16) is a technical constraint imposing that all transmission powers have to be real values.

It is important to stress that executing step three corresponds to solve the linear program described before. This tasks can be accomplish in polynomial time (see (Karmarkar, 1984)), and therefore very quickly in practice.

Experimental results

The three-steps approach was implemented in C++, by using Gurobi 4.5 library for solving the mixed integer linear programming formulations. To evaluate our three-steps approach, we compared its solutions to the early solutions generated by the step one, which, by itself, is equivalent to a classical mathematical programming-based approach. In rest of this section, we will refer to using step one alone as taking the classical approach.

The comparison of the two approaches is done by solving an example instance by using both the classical approach and the three-steps approach, with the same α value. Being given the same α value, both approaches generate solutions with the same solution cost. Finally, these cost-wise equivalent solutions are evaluated in terms of robustness. The evaluation of a solution is done by testing it on 10000 randomly generated scenarios, where the power requirement values are randomly picked from their intervals according to uniform probability distribution. If, in a scenario, a destination node can not be reached from the source s (i.e. it is isolated because the connections leading to it fail), then we call this situation a *disconnection*. If, in a scenario, there is at least one disconnection, we call that scenario a *flawed scenario*. Based on this, for each scenario, the evaluation of a solution is done according to how much in total disconnections occur on the destination nodes and how much of the scenarios are flawed.

For comparing the three-steps approach and the classical approach, instances were randomly generated. Given that the number of nodes $|V|$ and the number of destinations $|D|$ are passed as arguments to the instance generator, the following procedure is followed:

- a 2-dimensional area with size 100x100 is prepared;
- for each node, a random sized circle with a maximum radius R (R being a parameter), is generated;
- for each looping arc (i, i), $\underline{r}_{ij}$ and $\overline{r}_{ij}$ are set as 0;
- for each arc (i, j) with $i \neq j$, $\underline{r}_{ij}$ becomes B added the minimum distance between the circles of nodes i and j to the power of 4 and $\overline{r}_{ij}$ becomes B added the maximum distance between the circles of nodes i and j to the power of 4, where B is the base transmitting cost, set as 1 in our experiments;
- One of the nodes is declared as the source terminal and other $|D|$ nodes are declared as destination terminals.

To sum up, the procedure above associates circles for each node assumes that the nodes are positioned within the areas covered by those circles. Then, power requirement intervals are calculated according to the minimum and maximum distances of those circles. By generating circles of random radius for each node, the situation of having the same sized intervals for all r_{ij} is avoided.

Let us first analyze the solutions provided by both approaches according to α=0.9 on a single instance. The instance was generated by following the procedure above, with R=1, with 20 nodes and 10 destination nodes. The results are shown in Table 1. Considering that the solutions in Table 1 are same-cost solutions, it can be seen that the number of disconnections is significantly decreased by the three-steps approach. While it is difficult to come up with a single β value which will work best on all instances, from the table it can be concluded that a huge gap between α and β decreases the success of the three-steps approach. For this particular instance, the best working β seem to be around 0.85.

Table 1. Experimental results on a randomly generated instance

Approach	Disconnections	Flawed scenarios
Classical	36273	6475
Three-Steps with $\beta = 0.5$	10983	7701
Three-Steps with $\beta = 0.6$	9106	6743
Three-Steps with $\beta = 0.7$	7192	5657
Three-Steps with $\beta = 0.8$	5400	4499
Three-Steps with $\beta = 0.85$	4390	3796
Three-Steps with $\beta = 0.87$	5044	3921

By following the procedure for generating instances, multiple instances were generated with different number of nodes, different number of destination nodes and different values for *R* which controls the sizes of intervals of the uncertain data. Each instance was solved by both approaches with α=0.8, β=0.75; α=0.9, β=0.87; and α=0.95, β=0.93. β values were chosen according to the averaged success of the three-steps approach over all considered instances. Results of the experiments can be seen in tables 2, 3, 4 and 5, representing experiments with *R*=1, *R*=2, *R*=5, *R*=10, respectively. In these tables, each horizontal area (group of two rows separated by horizontal lines) represents 20 randomly generated instances with the specified number of nodes and destination nodes. Within each horizontal area, each row represents results obtained with different α and β values. The columns "Cost Save (%)" show how much, in percentage, the solution cost (which is the same for both the classical approach and the three-steps approach since the same α value is used for them on each experiment) was decreased in comparison to the cost of the conservative (α=1) solution, calculated as 1 - *Cost* / *ConservativeCost*. The column groups "Classical Approach" and "Three-Steps Approach" show how much in average the results have disconnections (column denoted by "D.") and flawed scenarios (column denoted by "F.S."). The column group "Improvement (%)" show, how much in average are the numbers of disconnections and flawed scenarios are improved by the three-steps approach, in comparison to the classical approach. On an instance, the improvement as percentage is calculated as 1-(*a*/*b*), *a* being the number of disconnections or flawed scenarios of the solution of the three-steps approach and *b* being the number of disconnections or flawed scenarios of the solution of the classical approach.

Table 2. Experimental results on instances generated with *R*=1

Terminals, Destinations	α, β	Cost Save (%)	Classial Approach		Three-Steps Approach		Improvement (%)	
			D.	F.S.	D.	F.S.	D.	F.S.
10, 5	0.8, 0.75	11.21	20290.3	6816.35	12399.1	5968.4	36.50	12.08
	0.9, 0.87	5.60	11380.95	4172.5	6675.5	3502.6	38.36	15.29
	0.95, 0.93	2.80	6037.45	2319	3201.95	1884.45	42.43	17.04
10, 9	0.8, 0.75	11.00	24591.25	6171.4	12093.15	5430.45	39.99	10.86
	0.9, 0.87	5.50	13108.55	3650.9	6044.3	3049.8	41.27	4.41
	0.95, 0.93	2.75	6807.05	1970.2	2780.05	1625	44.80	15.08
20, 10	0.8, 0.75	11.42	51216	8680.7	33525.8	8324.75	35.12	4.43
	0.9, 0.87	5.71	30336.05	6121.65	18353.95	5642.9	41.16	8.38
	0.95, 0.93	2.86	16581.9	3676.75	9615.7	3379.8	44.27	8.62
20, 19	0.8, 0.75	11.36	96753.55	9055.85	56647.35	8659.75	41.56	4.40
	0.9, 0.87	5.68	57460.7	6748.35	33631.15	6328.2	41.28	6.17
	0.95, 0.93	2.84	31454.2	4228.25	18502.05	3927.65	42.34	7.67
30, 10	0.8, 0.75	12.01	55495.4	9372.2	37451.85	9184.1	33.16	2.03
	0.9, 0.87	5.99	33471.15	7292.45	21343.35	6981.1	37.63	4.28
	0.95, 0.93	2.99	18596.75	4698	11506.9	4525.25	40.15	3.54
30, 20	0.8, 0.75	11.79	117065.65	9753.9	67257.2	9572.95	42.03	1.88
	0.9, 0.87	5.89	73187.65	8206.05	35806.75	7731.8	49.79	5.83
	0.95, 0.93	2.95	41285.55	5640.55	17926.95	5077.2	55.00	9.88
30, 29	0.8, 0.75	11.70	177218.85	9782.2	110519.4	9554.75	36.63	2.33
	0.9, 0.87	5.85	111365.1	8278.1	58146.75	7955.15	46.29	3.79
	0.95, 0.93	2.92	62791.6	5697	29180.8	5426.55	52.93	4.45

Table 3. Experimental results on instances generated with *R*=2

Terminals, Destinations	α, β	Cost Save (%)	Classial Approach		Three-Steps Approach		Improvement (%)	
			D.	F.S.	D.	F.S.	D.	F.S.
10, 5	0.8, 0.75	11.85	15779.5	6173.25	10010.5	5630	31.04	9.14
	0.9, 0.87	5.92	8348.3	3568.25	5208.25	3222.3	30.33	9.42
	0.95, 0.93	2.96	4328.55	1917.05	2598.85	1733.65	32.17	8.87
10, 9	0.8, 0.75	12.14	27083.45	6476.75	15035.65	5720.75	36.05	11.23
	0.9, 0.87	6.07	14285.95	3812.8	7185.3	3215.35	40.59	14.80
	0.95, 0.93	3.03	7392.25	2060.15	3429.75	1745.2	43.90	14.47
20, 10	0.8, 0.75	12.89	44900.3	8525.35	27646.6	8102.9	35.11	5.01
	0.9, 0.87	6.45	25678.6	5909.7	13733.95	5365.3	41.60	9.30
	0.95, 0.93	3.22	13791.05	3535.15	6696.45	3132.1	47.94	12.51
20, 19	0.8 0.75	12.66	81451.3	8950.85	39866.9	8217.1	48.55	8.51
	0.9, 0.87	6.33	46087.65	6451.65	20888.7	5726.9	51.03	12.02
	0.95 0.93	3.16	24291.75	3931.05	10206.85	3438.15	53.76	13.23
30, 10	0.8 0.75	13.42	53459.5	9211.25	33258.1	8825.7	37.18	4.36
	0.9, 0.87	6.72	31944.5	6984.7	18092.3	6445.75	41.97	8.21
	0.95 0.93	3.36	17646.3	4430.9	8585.15	3982.2	48.55	10.21
30, 20	0.8, 0.7	13.18	108737.85	9646.55	70539.15	9602.2	35.21	0.45
	0.9, 0.87	6.59	64843.05	7798.35	39072.4	7677.9	40.06	1.44
	0.95 0.93	3.29	36634.2	5199.85	18375.3	4815.5	53.48	7.94
30, 29	0.8 0.75	13.47	151890.15	9695.3	91034.45	9653.5	38.36	0.42
	0.9, 0.87	6.73	89855	7958.95	43270.35	7615.15	50.57	4.24
	0.95 0.93	3.36	49242	5294	18224.1	4817.2	60.72	8.77

Table 4. Experimental results on instances generated with *R*=5

Terminals, Destinations	α, β	Cost Save (%)	Classial Approach		Three-Steps Approach		Improvement (%)	
			D.	F.S.	D.	F.S.	D.	F.S.
10, 5	0.8 0.75	15.46	16103.15	5520.45	9809.15	4781.3	35.54	12.99
	0.9, 0.87	7.72	8471.2	3134.45	4931.15	2618.95	36.87	15.81
	0.95 0.93	3.86	4351.7	1664.95	2300.55	1321.75	40.93	18.52
10, 9	0.8 0.75	14.94	27932.1	6120.05	11318.55	5062.65	54.76	17.28
	0.9, 0.87	7.47	14832.45	3569.05	5510.8	2792.75	57.56	21.76
	0.95 0.93	3.74	7669.7	1939.3	2778.8	1504.75	57.66	22.19
20, 10	0.8 0.75	16.02	40083.05	8382.8	23446.8	7978.5	39.17	4.81
	0.9, 0.87	8.00	21661.15	5631.55	11778.1	5124.55	42.31	8.70
	0.95 0.93	4.00	11288.6	3290.05	5357.8	2746	49.42	15.60
20, 19	0.8 0.75	16.03	72826.6	8865.3	44544.95	8756.35	36.42	1.18
	0.9, 0.87	8.02	39304.2	6249.4	20603.45	5993.2	44.57	3.97
	0.95 0.93	4.01	20640.5	3724.45	9670.4	3571.05	48.76	3.75
30, 10	0.8 0.75	17.00	50081.2	8828.4	36303.25	8702.95	28.65	1.60
	0.9, 0.87	8.50	28845.35	6264.2	18531.05	6061.5	37.48	3.81
	0.95 0.93	4.25	15104.55	3703.55	7371.25	3401.65	50.39	8.60
30, 20	0.8 0.75	16.12	103116.9	9376.9	61560.2	9168.25	39.99	2.29
	0.9, 0.87	8.05	61408.5	7263.85	3485.85	6983.45	43.64	4.19
	0.95 0.93	4.03	33453.8	4645.15	17703	4516.75	46.37	3.29
30, 29	0.8 0.75	16.77	129703.5	9521.8	66174.85	9387.3	47.19	1.38
	0.9, 0.87	8.38	70615.65	7422.15	30569.05	7075.95	53.39	4.44
	0.95 0.93	4.19	36570.05	4720.85	11899.9	4352.3	64.67	7.29

Table 5. Experimental results on instances generated with *R*=10

Terminals, Destinations	α, β	Cost Save (%)	Classial Approach		Three-Steps Approach		Improvement (%)	
			D.	F.S.	D.	F.S.	D.	F.S.
10, 5	0.8 0.75	16.85	13152	5357.4	8297.3	4992.85	34.09	6.65
	0.9, 0.87	8.41	6695.85	2927.15	4025.95	2659.3	36.23	9.03
	0.95 0.93	4.20	3402.35	1521.3	1786.4	1327.9	39.83	10.78
10, 9	0.8 0.75	17.17	23572	5906.2	13549.55	5414.05	33.10	7.77
	0.9, 0.87	8.57	12013.7	3313.95	5995.9	2847.95	40.05	12.68
	0.95 0.93	4.29	6101.75	1740.85	2916.15	1506.05	39.75	10.67
20, 10	0.8 0.75	18.25	34826.55	7501.8	19182.85	6761	45.53	10.40
	0.9, 0.87	9.11	17292	4544.5	8178.05	3892.85	50.06	14.75
	0.95 0.93	4.56	8798.8	2457.35	3605.85	1979.45	56.76	20.23
20, 19	0.8 0.75	17.68	60889.9	7908.5	33507.2	7717.4	43.77	2.46
	0.9, 0.87	8.82	32434.25	5059.2	16336.3	4795.2	48.98	5.21
	0.95 0.93	4.41	16464.5	2809.7	6774.2	2582	55.24	7.76
30, 10	0.8 0.75	16.77	129703.5	9521.8	66174.85	9387.3	47.19	1.38
	0.9, 0.87	9.15	22213.65	5284.55	11910.8	4766.2	46.15	9.77
	0.95 0.93	4.57	11577.6	3005.4	5720.9	2643.4	48.90	11.10
30, 20	0.8 0.75	18.33	40803.95	8045.45	24001.6	7556.55	40.66	5.97
	0.9, 0.87	9.22	47779.2	6219.95	24387.3	5997.3	47.03	3.26
	0.95 0.93	4.61	24803.6	3675.8	10766.95	3459.75	56.01	5.57
30, 29	0.8 0.75	18.45	89886.15	8911	52252.05	8822.55	41.39	0.92
	0.9, 0.87	9.13	53855.95	6388.95	24372.85	6147.4	50.31	4.11
	0.95 0.93	4.56	28081.6	3780	11384.05	3640.4	56.70	5.02

The results in tables 2, 3, 4 and 5 show that, among the considered instances, the three-steps approach can significantly reduce the number of disconnections most of the time. Improvements in the numbers of flawed scenarios can also be observed. Interestingly, there were cases where the numbers of flawed scenarios were higher for the three-steps approach than the classical approach, however, in most of such cases, the numbers of disconnections were still reduced by the three-steps approach. This reflects into the average results as higher improvement percentages for the number of disconnections, in comparison to the number of flawed scenarios. From the instance groups where all nodes except the source are destinations, it can be seen that the three-steps approach can be applied to broadcasting problems (in which the objective is to transmit to all nodes except the source node), in addition to multicasting problems.

In the experimental results, it becomes visible that, with α=0.8, the number of flawed scenarios increases considerably for both the classical approach and the three-steps approach (despite the improvements it introduces over the classical approach). The practical values for α seems to be around or greater than 0.9. Also, it can be seen that the flawed scenarios keep increasing with the size of the instance, when the same value for α is considered.

Calculation of the flawed scenarios

In this section, we present formulations for estimating the number of flawed scenarios, to provide an alternative to finding it by scenario sampling. A solution, generated by the classical approach, or the three-steps approach, is a set of transmission power value for each terminal. According to these transmission power values, for each arc (i, j), there is a probability to reach terminal j. In this section, we provide the formulations to calculate the probability to reach all destination terminals $d \in D$ from the source terminal s.

Let us first define the concept of *important arcs*. An arc is an *important arc* if it exists within a non-looping path which starts from the source terminal s and reaches at least one destination terminal. The set of all important arcs are denoted by A^*.

Since, the arcs are probabilistic, depending on the scenario, different important arcs can be enabled or disabled. Therefore, we define a *configuration* $K \subseteq A^*$ as a set of important arcs, which represents the assumption that each important arc $(i, j) \in K$ can reach terminal j and each important arc $(i', j') \in (A^* \setminus K)$ can not reach j'. Now we define a *connective configuration* as a configuration in which the arcs provide at least one path to each destination terminal $d \in D$, from the source terminal s.

To estimate the number of flawed scenarios, first, all important arcs are identified. This can be done by following all the paths from source terminal s to all destination terminals by using a recursive depth-first search.

After identifying all important arcs, each possible configuration $K \subseteq A^*$ is listed and its probability is calculated. For this purpose, we begin by formulating the function $ProbArc(i, j)$ which calculates the probability for arc (i, j) to reach terminal j:

$$ProbArc(i,j) = \begin{cases} 1 & \text{if } \rho_i \geq \overline{r}_{ij} \\ 0 & \text{if } \rho_i < \underline{r}_{ij} \\ \frac{\rho_i - \underline{r}_{ij}}{\overline{r}_{ij} - \underline{r}_{ij}} & \text{otherwise} \end{cases}$$

By using the function $ProbArc(i,j)$, now we can formulate the function to find the probability of a configuration as:

$$ProbConfiguration(K) = \prod_{(i,j)\in K} ProbArc(i,j) \cdot \prod_{(i,j)\in A^*\setminus K} (1 - ProbArc(i,j))$$

The probability of having all destinations reached, *ProbFlawless*, is then calculated as the sum of the probabilities of all connective configurations:

$$ProbFlawless = \sum_{K\subseteq A^*} ProbConfiguration(K)$$

The probability of having at least one destination terminal disconnected is *ProbFlawed*=1-*ProbFlawless*. Among n trials of establishing connection, the number of flawed scenarios is finally estimated as $n \cdot ProbFlawed$.

Estimating the flawed scenarios via mathematical means, this approach can be used to verify the correctness of the sampling engine available. For this purpose, the numbers of flawed scenarios of the solutions of some randomly generated instances were estimated by both sampling engine and the formulations presented in this section. The results of these comparisons are shown in Table 6. The instances were generated with R=1, and solved by the three-steps approach with parameters α=0.9, β=0.87. Within the table, under the column group "Estimated flawed scenarios", the column "Sampling" represents the estimations of the sampling engine, and the column "Formulations" represents the estimations of the approach presented in this section. The column "Errors" represents the error made by the sampling engine, in comparison to the estimations of the formulations, calculated as (a-b)/10000 where a is the result of the sampling engine, b is the result of the formulations and 10000 is the number of scenarios. In the table, it can be seen that the absolute values of the errors made by the sampling engine are always below 1%. Therefore, the results of the sampling engine can be verified as close to the formulations.

Table 6. Flawed scenario estimations via the sampling engine and via the formulations presented in Section 5.

			Estimated flawed scenarios		
Instance	Nodes	Destinations	Sampling	Formulations	Errors
1	10	5	2655	2671.01	-0.16%
2	10	5	1926	1944.24	-0.18%
3	10	5	2472	2499.67	-0.28%
4	10	5	1405	1430.72	-0.26%
5	10	5	1851	1838.57	0.12%
6	10	9	3432	3372	0.60%
7	10	9	1997	2028.36	-0.31%
8	10	9	5480	5445.85	0.34%
9	10	9	1026	1050.29	-0.24%
10	10	9	2347	2354.51	-0.08%
11	20	10	4407	4450.17	-0.43%
12	20	10	6710	6734.37	-0.24%
13	20	10	3245	3208.72	0.36%
14	20	10	1632	1600.92	0.31%
15	20	10	2836	2834.71	0.01%

Conclusions

A three-steps approach was discussed for multicasting on wireless networks, which takes into account the uncertainty on the power requirement values between the terminals.

The three-steps approach takes a classical mathematical programming approach as a base and adds two more steps which make adjustments on the transmission powers on the nodes to make the network more robust. Therefore, the three-steps approach can also be interpreted as a two-step addition to a classical mathematical programming approach. These additional steps can be implemented as polynomial-time procedures, therefore the complexity class of the approach is the same with the classical mathematical programming.

The experiments show that the additional two steps introduced by the three-steps approach can be useful in improving the robustness of the solutions, compared to the classical mathematical programming approach.

Acknowledgments—N.E. Toklu was supported by the Swiss National Science Foundation through project 200020-140315/1: "Robust Optimization II: applications and theory". The authors would like to thank an anonymous referee for the valuable suggestions.

References

Babonneau F, Vial JP and Apparigliato R (2010). Handbook on "Uncertainty and Environmental Decision Making", chapter Robust optimization for environmental and energy planning, pages 97–126. Springer Verlag.

Ben-Tal A and Nemirovski A (1998). Robust convex optimization. Mathematics of Operations Research, 23(4):769–805.

Ben-Tal A and Nemirovski A (2000). Robust solutions of linear programming problems contaminated with uncertain data. Mathematical Programming, 88:411–424.

Bertsimas D, Gamarnik D and Rikun AA (2010). Performance analysis of queuing networks via robust optimization. Operations Research, 59(2):455–466.

Bertsimas D and Sim M (2003). Robust discrete optimization and network flows. Mathematical Programming Series B, 98:49–71.

Bertsimas D and Sim M (2004). The price of robustness. Operations Research, 52(1):35–53.

Karmarkar N (1984). A new polynomial time algorithm for linear programming. Combinatorica, 4(4):373–395.

Leggieri V, Nobili P, and Triki C (2008). Minimum power multicasting problem in wireless networks. Mathematical Methods of Operations Research, 68:295–311.

Montemanni R, Gambardella LM and Das AK (2005). The minimum power broadcast problem in wireless networks: a simulated annealing approach. In Proceedings of IEEE WCNC 2005 - Wireless Communications and Networking Conference.

Montemanni R and Mahdabi P (2011). A linear programming-based evolutionary algorithm for the minimum power broadcast problem in wireless sensor network. Journal of Mathematical Modelling and Algorithms, 10(2):145–162.

Rappaport T (1996). Wireless Communications: Principles and Practices. Prentice Hall.

Soyster AL (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research, 21(5):1154–1157.

Wieselthier JE, Nguyen GD and Ephremides, A (2002). Energy-efficient broadcast and multicast trees in wireless networks. Mobile networks and applications, 7(6):481–492.