Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 948521, 11 pages
http://dx.doi.org/10.1155/2014/948521

Research Article

Constructing Fair Destination-Oriented Directed Acyclic Graphs for Multipath Routing

Katarzyna Kalinowska-Górska and Fernando Solano Donado

Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Correspondence should be addressed to Katarzyna Kalinowska-Górska; k.kalinowska@stud.elka.pw.edu.pl

Received 7 February 2014; Accepted 24 March 2014; Published 28 April 2014

Academic Editor: Hanan Luss

Extensive research in the field of telecommunications has been done on the techniques of multipath routing, as they offer many advantages over conventional single-path routing methods. Some of these techniques make use of the so-called Destination-Oriented Directed Acyclic Graphs (DODAGs) which are constructed on the networks, usually in a distributed way. However, while defining methods of forming DODAGs, the authors of multipath algorithms tend to overlook a possibly significant issue which could, in a way, define the quality of a given DODAG in the context of multipath routing, namely, providing an equitable distribution of the paths between the nodes in the newly created DODAG. In this paper, a few requirements for constructing a "fair" DODAG are identified in the context of multipath routing. An optimization algorithm that tries to find an equitable solution according to these requirements is also presented. Three DODAG-creation algorithms that appear in the literature are simulated and compared against this equitable solution, and none of them is getting close to it in terms of fairness in the distribution of the paths. Moreover, two interesting properties of equitable solutions are revealed in the simulations.

1. Introduction

A Destination-Oriented Directed Acyclic Graph (DODAG) is a term used in [1] to describe a directed acyclic graph with exactly one root, where a root is a node which has no outgoing edges. Diverse multipath routing algorithms make use of DODAGs, such as [2–8]. While conventional single-path routing techniques form directed spanning trees rooted at the destination, multipath algorithms use DODAGs to provide one or more paths between a node-root pair (A directed tree is also a DODAG, but of a more restricted form.). The networks and DODAGs considered in this paper are simple graphs. As an example, a simple DODAG is shown in Figure 1.

It is worth noting that in a full multipath routing procedure two different tasks have to be performed: (1) determining a set of paths per node-destination pair and (2) distributing the flow on these paths. In some algorithms these tasks are easily separable, like in the case of equal-cost multi-path (ECMP), where the demand from the source to the destination is split uniformly between all equal-cost shortest paths. In other algorithms, such as iterative gradient minimization algorithms [9], these two steps merge and cannot be executed separately.

In scenarios where network topology changes are infrequent and the amount of energy is limited, for example, in wireless sensor networks for environmental monitoring, separating these two tasks could yield better overall performance, considering the energy consumed for calculations and transmission. By neglecting small changes in the topology, CPU cycles required for recalculating the set of paths (first task) can be spared and the number of signaling messages exchanged between the nodes can be reduced. In addition, total node's memory requirements can be reduced, as only one task needs to be tackled at a time. Therefore, smaller chipsets (with less energy consumption per cycle) can be employed.

Still, which of the two strategies, that is, executing both tasks jointly or separately, performs better, depends much on the particular application and is left out of the scope of this paper. Here, we consider such applications where both tasks can be solved separately without a substantial

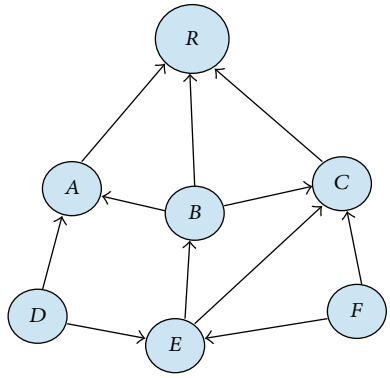

Figure 1: An example DODAG with the root at node R.

decrease in performance. In particular, we assume that a set of paths is fixed for long periods of time and flow reallocation procedures are executed more often, for example, once link qualities, nodes' capabilities or congestion state in the network change. As a considerable amount of literature has already been published on flow allocation algorithms, in this paper, we focus on the first task: how to find a large set of paths that can be used for long periods of time and which provides enough flexibility, taking into account possible flow reallocations. This can be achieved by maintaining a once-constructed DODAG in the network, with occasional local and global repairs when needed, as in [1].

Many different DODAGs might be constructed in the network for multipath routing. Hence, an obvious question arises; that is, what properties should the constructed DODAG have? The objectives of multipath routing include, among other things, providing resilience against node/link failures, minimizing the risk of link overload, or balancing the energy consumption of battery-powered nodes in case of Low-Power and Lossy Networks (LLNs). On the basis of these (and other) objectives, a conclusion can be made that during the lifetime of a single DODAG, a node would probably need a few paths for path switching, flow splitting, or introducing flow redundancy. This means that, when considering two nodes with a similar distance to the destination, it is better when both have several paths to the root in the DODAG than one having tens of paths and the other with only one or two paths available. This observation leads to an objective for the DODAG construction algorithm, which is to provide a *fair* distribution of paths per node-root pair in the DODAG. Obviously, this objective is strongly constrained by the network topology; in particular, it is never possible to obtain equal, other than one, number of paths in every node. This is due to the fact that in a simple DODAG there always exists at least one node which has exactly one path to the root. This is stated as Proposition 1 in the next section and proved in Appendix B.

There are more reasons why the *fairness* objective mentioned above does not simply mean trying to equalize the numbers of paths per node in the resulting DODAG. First of all, leveling down is undesired, which means that apart from equalizing the numbers of paths, their simultaneous maximization is required. Second, in most cases, the average number of paths of a node in the DODAG depends on its distance from the root. Typically, it grows as the distance increases (unless many long roundabout paths exist in the DODAG that lead from nodes closer to the root through nodes farther from the root). Third, the quality of considered paths is important. In particular, it could be of great advantage to reduce the length/cost of the paths used for routing and/or to provide node/link disjointness of the paths in the path sets [10]. Moreover, the number of paths in a graph grows exponentially as a function of the number of links. Merely enumerating all the paths would pose a serious computational problem, even for relatively small networks (e.g., of less than 50 nodes), which means that it is not feasible to consider all of them in reasonable time.

These observations imply that a solution is needed as follows:

(i) it tries to equalize, without leveling down, the numbers of paths per node at the same minimum hop count distance from the root (at the same "level");

(ii) when trying to provide an equitable distribution of the paths, it does not take all of them into consideration but reasonably selects subsets of all possible paths instead based on their properties, for example, their length.

Which properties of the paths selected for equalization are the most important depends on the particular application. Therefore, a solution is proposed in this paper which does not define *a priori* the desired path sets per node but allows potential users to define their own candidate path sets, one set per node, which are appropriate in a particular situation. An equitable distribution of the paths per node would then be provided, only taking into account the given path sets. These path sets might satisfy some predefined requirements; for example, they might represent shortest paths sets or disjoint paths sets or include other paths which are, for some reason, of interest to the user.

It is proved in Appendix A that even a simple problem of maximizing the number of paths from a single node to the DODAG root is NP-complete. Therefore, to achieve an equitable and yet efficient distribution of given paths, an IP optimization problem with a lexicographic maximin objective has been formulated, and an appropriate algorithm from [11] for solving this problem has been chosen. Solving the optimization problem results in a simple DODAG rooted at the destination node, with equitable distribution of input paths among all the nodes.

Of course, a constructed DODAG will typically contain more paths than just the ones belonging to the set provided by the user for optimization. These additional paths might be used for flow allocation as well if needed; for example, they might serve as backup paths when all of the input paths defined for a node are affected by a link/node failure. However, it is assumed that since they were not included by the user in the input path sets, they might be of "worse quality" (e.g., greater cost). In any case, since the number of additional paths of a node is not controlled by

the optimization algorithm, their distribution will probably be random and no *fairness* will be achieved in general. On the other hand, for high enough levels, so many paths will be available per node that trying to obtain a general fair distribution makes little sense, since each node will get more paths than it will need anyway. What is important is providing each node with the best paths, as far as possible, which, again, justifies the approach taken in this paper. As for the nodes at lowest levels, if the input path sets are reasonably chosen, any additional paths in the DODAG will be very roundabout, therefore not very useful.

Authors of some of the DODAG-creation algorithms appearing in the literature, like LMR and TORA [5, 6], ignore the number and properties of paths available in the resulting DODAGs. In the IDAGs approach [8], two node- or link-independent DAGs are constructed, guaranteeing each node to have at least two node- or link-disjoint paths. MARA-MC [7] is proved by the authors to compute a large number of paths for a large fraction of source-destination pairs, when executing a DODAG construction procedure for every destination node. Authors of the MDVA algorithm [2] only check its performance (i.e., message overhead and convergence time) and do not consider properties like the number of paths at all. However, MDVA and other LFI-based algorithms by the same authors [3, 4] are compared to MARA-MC in [7] and proved to yield worse results regarding the average number of paths in the constructed DODAGs.

The remaining part of the paper is organized as follows. Section 2 describes the formulation of the optimization problem and the algorithm chosen for its solution. Section 3 presents the simulations and a comparison of three algorithms from the literature with the solution obtained by solving the lexicographic problem when providing sets of k shortest paths per node. Moreover, additional properties of the most equitable solutions are shown. Section 4 concludes the paper. Proofs for statements included in the paper are presented in Appendices A and B. Appendix C contains parameter values of the algorithm used for the generation of test networks.

2. Optimization Problem with Lexicographic Maximin Objective: Formulation and Algorithm

The problem of equitable distribution of the paths among the nodes in the graph corresponds in fact to a resource allocation problem [11], where different activities compete over limited resources which have to be allocated *fairly* among them. In this case, nodes correspond to activities and a single objective function of a node expresses the number of paths "allocated" to that node. The resources which are limited, or rather constrained by the network topology, are the directions of the links. The way of assigning these directions determines the set of paths available in the DODAG.

A lexicographic maximin objective [11] has been chosen for this problem. It can be viewed as an extension of a simple maximin objective. In this approach, first the number of paths for the node with fewest paths is maximized. Then, the number of paths for the node with second fewest paths is maximized, but without decreasing the smallest value calculated in the first step. Next, the number of paths for the node with third fewest paths is maximized, without decreasing the value of the first two and so forth. Hence, solving a lexicographic maximin problem means finding a solution vector (of numbers of paths of the nodes) which is lexicographically largest of all feasible vectors, when the values in the vectors are sorted in a nondecreasing order. It is not known in advance which node will occupy which lexicographic position in the solution vector. A lexicographic maximin (or minimax) solution is Pareto optimal.

It is worth mentioning here that a simple maximin objective would achieve nothing due to the following fact.

Proposition 1. *In a simple DODAG, there will always exist at least one node with exactly one path to the root.*

It will be one of the root's neighbours. Therefore, the minimum number of paths of all nodes in the DODAG will always be equal to one. If only predefined sets of paths are considered, the minimum number of input paths of all nodes in the DODAG will never be more than one. The proof of Proposition 1 is presented in Appendix B.

Applying lexicographic maximization to the DODAG construction problem with input path sets ensures that at least one candidate path per node survives in the DODAG, as long as the following two conditions are met:

(i) at least one path per node is inserted into the formulation;

(ii) in the provided path sets, there is at least one combination of paths containing at least one path per node where no paths exclude each other due to conflicting edge directions; that is, all these paths can coexist in a DODAG.

At least one path per node will be included in the DODAG due to the fact that lexicographic maximization will not allow a zero value in the first position in the solution vector, if it is possible to obtain a minimum equal to 1. The above conditions are met, for example, when providing nodes' shortest paths to the root (as they form a tree).

2.1. The Formulation. The generic network inserted into the formulation (No specific type of network is of interest, although the simulations were performed on random *ad hoc* networks.) is modeled as a simple, strongly connected directed graph $G = (N, E)$, where N is the set of nodes and E is the set of links. In addition, if there exists an edge from vertex m to vertex n, there also exists an edge in the reverse direction (from vertex n to vertex m); that is, the graph is symmetric. A set of candidate paths $p = 1, 2, \ldots, P$ is provided, each path represented a sequence of C_p directed edges. The identifier of the destination (root) node is denoted by r. Neighbours of node m or r are indexed by n. The following variables are present in the formulation:

(i) $y_{mn} = 1$ if the (m, n) edge is included in the solution (output DODAG), 0 otherwise;

(ii) $Y_p = 1$ if a candidate path p is included in the DODAG, 0 otherwise;

(iii) U_m is an integer variable representing the number of paths from a node m to the root in the DODAG but only counting the paths included in the P set inserted into the formulation. No other paths available in the DODAG are counted by this variable.

The problem is formulated as follows:

$$\text{lexmax } \left\{U^{(m)}(Y) = \left[U_{m1}(Y), U_{m2}(Y), \ldots, U_{m|N|-1}(Y)\right]\right\}, \tag{1}$$

where

$$U_{m1}(Y) \le U_{m2}(Y) \le \cdots \le U_{m|N|-1}(Y). \tag{2}$$

(i) Apart from the root, each node has at least one outgoing edge:

$$\sum_n y_{mn} \ge 1 \quad \text{for each } m \in N,\ m \ne r. \tag{3}$$

(ii) All edges incoming to the root are included in the DODAG:

$$y_{nr} = 1 \quad \text{for each neighbour } n \in N. \tag{4}$$

(iii) At the same time, all edges outgoing from the root are excluded from the DODAG:

$$y_{rn} = 0 \quad \text{for each neighbour } n \in N. \tag{5}$$

Note 1. Instead of constraints (4) and (5) and variables y_{nr} and y_{rn}, binary constants might be inserted.

(iv) Of each edge pair (m, n) and (n, m) not adjacent to the root, not more than one should be selected:

$$y_{mn} + y_{nm} \le 1 \quad \text{for each pair of neighbouring nodes } m, n \in N,\ m \ne r,\ n \ne r. \tag{6}$$

Note 2. In the case of maximizing the number of paths, there will always be one direction selected; that is, the constraint will always be met with equality.

(v) All possible cycles have to be eliminated. Cycles of length 1 do not exist in the network graph, as it is assumed to be simple. Cycles of length 2 are already eliminated by constraints (6). Hence, it is required to eliminate cycles of length $K = 3, 4, \ldots, |N|$. For this purpose, the following constraints can be formulated:

$$y_{k1k2} + y_{k2k3} + \cdots + y_{kKk1} \le K - 1$$
$$\text{for each } K \text{ interconnected nodes, } k_1, k_2, \ldots, k_K \in N. \tag{7}$$

(vi) If any of the edge variables belonging to a path p is equal to 0, then the variable Y_p is also equal to 0, which gives $Cp - 1$ constraints (8) per path (not C_p as, due to constraints (4), edges incoming to the root are always included in the DODAG):

$$Y_p \le y_{mn} \quad \text{for each pair of consecutive nodes } m, n \in N \text{ on the path } p, \text{ except } r\ (n \ne r). \tag{8}$$

On the other hand, if a path p is included in the DODAG due to maximization ($Y_p = 1$), then all of its edges will be included in the DODAG; that is, all y_{mn} variables of this path will be equal to 1.

(vii) The last expression represents the number of paths from node m to the root:

$$U_m = \sum_p Y_p \quad \text{for each } m \ne r \text{ and paths } p \text{ originating at node } m. \tag{9}$$

2.2. Proof of Correctness. Constraints presented in the previous subsection are sufficient to construct a valid DODAG due to the following facts:

(i) all cycles are eliminated due to constraints (6) and (7);

(ii) each node has at least one path to the root. This path does not necessarily belong to the candidate paths set inserted into the formulation. It might be one of the additional paths resulting from the edge orientation of the solution. This requirement basically means that no other root than the one given is created in the resulting graph; therefore, it is a DODAG. This can be proved in the following way: due to the constraints (3), any arbitrary node different than the root has at least one outgoing edge. Following one of the outgoing edges, it is possible to reach either the root or another node, which also has at least one outgoing edge. Due to the fact that no cycles exist, which means that it is impossible to come back to any of the nodes traversed previously, and the fact that by applying constraints (3) and (5) the root is the only node with no outgoing edges, the whole process can finish only at the root.

2.3. The Lexicographic Algorithm. The presented DODAG construction problem belongs to the class of equitable resource allocation problems with integer decisions, where the number of possible distinct outcomes of the performance functions is limited and relatively small. Here, a single performance function, that is, a $U_{mi}(Y)$ function representing the number of paths of a single node, might assume all integer values from 0 to k_m, where k_m is the number of candidate paths inserted into the formulation for this node. Hence, the set of possible distinct outcomes of all performance functions contains all integer values between 0 and $k_{\max}$, where $k_{\max}$ is the largest number of candidate paths inserted for any node.

An algorithm designed for solving this class of problems is presented in [11], Section 7.2.3 *Lexicographic Minimization of Counting Functions*. The algorithm in its original form solves the problem of lexicographic minimization. The number of performance functions' possible distinct outcomes determines the maximum number of lexicographic iterations. The basic idea of this algorithm is the following. The set of all possible distinct outcomes has to be provided. Special counting functions are then constructed, each of them representing the number of times that a single distinct outcome appears in the solution. These functions are then iteratively minimized in the following way. First, the number of occurrences of the largest possible outcome is minimized. Then, the number of occurrences of the second largest outcome is minimized without increasing the first one and so forth. The algorithm terminates after reaching the last (smallest) distinct outcome or in an earlier iteration, if a unique solution is found. Due to the fact that the counting functions are represented as optimization problems, additional constraints and continuous variables have to be inserted in each lexicographic iteration, extending the problem and transforming it from an IP to a MIP problem.

The lexicographic maximization objective can be transformed into lexicographic minimization objective, similar to the maximin-minimax conversion given in Section 1.2 of [11]:

$$\begin{aligned}\operatorname{lexmax}\left\{U^{(m)}(Y)=\left[U_{m1}(Y),U_{m2}(Y),\ldots,U_{m|N|-1}(Y)\right]\right\}&\\ =-\operatorname{lexmin}\left\{-U^{(m)}(Y)\right.&\\ \left.=\left[-U_{m1}(Y),-U_{m2}(Y),\ldots,-U_{m|N|-1}(Y)\right]\right\},&\end{aligned}\tag{10}$$

where

$$-U_{m1}(Y)\geq -U_{m2}(Y)\geq\cdots\geq -U_{m|N|-1}(Y).\tag{11}$$

2.4. The Exact Number of Distinct Outcomes. As mentioned already, the set containing all integers between 0 and $k_{\max}$ can be used as the set of all possible distinct outcomes of the performance functions in this formulation. This method is approximate and results in an overhead, since it is very likely that two or more paths inserted into the formulation for a single node will exclude each other from the resulting DODAG because of using one or more edges of opposite direction between the same pair of nodes. To obtain the minimal upper bound on the possible distinct outcomes set, a maximum independent set count problem would have to be solved for every node, as proved in Appendix A, and the maximum of the obtained values should be used as the upper bound. However, due to the NP-hardness of the maximum independent set problem, it might be of advantage to omit the exact computation of the maximum distinct outcome value, even though a crude approximation increases the maximum number of lexicographic iterations.

2.5. Size of the Formulation: Reduction of the Number of Cycle Elimination Constraints. The cycle elimination constraints (7), due to their numerosity, incur heavy computational load while solving the formulation. Maximally $2\cdot(\binom{N}{3}+\binom{N}{4}+\cdots+\binom{N}{N})$ (in the case of a complete graph) constraints (7) could appear in the problem. Although on average this number will be smaller, the number of cycles in a graph, *ergo* the number of constraints, grows exponentially as a function of the size of the graph (in terms of edges). To reduce the number of inserted cycle elimination constraints, the following method has been employed (based on a suggestion of A. Tomaszewski):

(i) the formulation is solved without constraints (7);

(ii) if cycles exist in the resulting graph, appropriate (7) cycle elimination constraints are inserted. Only a few cycles are searched for (e.g., up to 5);

(iii) the extended formulation is solved again, until the output graph is a DODAG; that is, it does not contain any cycles.

This method trades solving one large IP problem for iterative solving of several smaller IP problems. In the last of the solved problems, all possible cycles will be eliminated, yet avoiding inclusion of unnecessary constraints. When solving the whole lexicographic formulation, all cycle elimination constraints added in the ith iteration are transferred to the $i+1$th lexicographic iteration.

3. Simulations and Results

3.1. Generated Networks. Random *ad hoc* networks were generated for the simulations. WPA *ad hoc* network generation algorithm presented in [12] was used, with a small modification and with values of the parameters chosen so that generated networks were always connected and of reasonable density (limiting the density limits the number of existing paths in the network graphs). All values for parameters used in the WPA algorithm as well as a short remark about the modification are given in Appendix C.

The output of the WPA is an undirected graph, where the length of each edge represents the distance between the two nodes placed on a 2D surface. In the simulations, each undirected edge was replaced by a pair of inverse directed edges. For one of the tested algorithms, a weight was assigned to each directed edge, representing 2.45 GHz signal attenuation in free space (This frequency is used, for example, in Zigbee technology.). The weight was calculated according to (7) in [13] and length of the edge used as the distance, assuming meters as units. A random deviation of ±10% was also added to each link weight.

Networks comprising 5 to 50 nodes were generated, with a step of 5. For each network size, 50 random networks were generated. The first generated node in the network was always selected as the root node.

3.2. Input Paths. Sets of k shortest paths, one set per node other than the root, were inserted into the formulation, where *shortest* means of least hop count. Yen's k shortest path algorithm was used for calculating these sets [14]. BFS algorithm was used as the shortest path subalgorithm. k values were equal for every node in the network. They were

Table 1: Comparison of four tested algorithms: average number of paths per node at different levels 1–6 for 50-node networks. Levels (1–6) were present in most (47–50 out of 50) networks.

Level	Shortest-multipath	TORA	MARA-MC	k-shortest-lex-maximin
1	2.17	2.25	3.22	3.72
2	4.67	4.39	5.50	5.85
3	7.00	6.32	6.97	7.06
4	8.57	7.69	7.66	8.00
5	9.65	8.42	8.04	8.72
6	10.46	9.44	8.55	9.43

Table 2: Comparison of four tested algorithms: average variance in numbers of paths per node at different levels 1–6 for 50-node networks.

Level	Shortest-multipath	TORA	MARA-MC	k-shortest-lex-maximin
1	1.64	2.24	4.12	4.08
2	5.77	5.27	4.28	2.56
3	9.02	8.36	4.65	2.59
4	9.25	8.29	6.12	3.01
5	10.29	11.93	7.72	3.70
6	9.93	10.83	11.39	5.32

dependent on the size of the network and equal to 5 for networks of sizes 5 to 10, 10 for 15 to 30-node networks, and 15 for 35 to 50-node networks.

3.3. Tests. DODAGs formed by three distributed algorithms that appear in the literature were compared against the k-shortest-lexicographic solution: the Shortest-multipath DODAG as in MDVA [2], using link costs as given in Section 3.1, the MARA-MC DODAG [7], and a DODAG approximating the output graph of the TORA *ad hoc* multipath routing algorithm [6]. The approximation mentioned above avoids random irregularities specific to the graphs constructed by the TORA algorithm. Hence, to form a DODAG, the nodes are simply ordered by the hop count metric (edges are directed from the node with the greater hop count to the node with the smaller hop count), which, in fact, makes the resulting DODAG similar to a Shortest-multipath DODAG when link costs are equal to 1. Ties in hop counts are resolved using nodes' unique identifiers. This simplification might incur slightly different results than TORA would achieve in reality, but it is required to provide the possibility of performing the simulations in a simple way (i.e., without employing more complex network simulators).

An especially interesting algorithm is MARA-MC [7], which solves an all-to-one maximum edge connectivity problem with optimality. The authors' simulations show that this algorithm obtains significantly more paths in the resulting DAGs than other compared algorithms and that it calculates a large number of paths for a large fraction of source-destination pairs. It is therefore interesting to check how well it performs in comparison to the lexicographic algorithm. It is worth noting that the all-to-one maximum connectivity objective as defined in Section 4.A of their paper does not have any meaning in the case of nonmultigraphs, as the minimum connectivity between a node and a simple DAG root will always be equal to 1, regardless of the algorithm used. This is an implication of Proposition 1. A possibly significant drawback of the MARA-MC algorithm is that it completely ignores the length of the resulting paths.

3.4. Results. Tables 1 and 2 present the average value and the average variance in the number of candidate (shortest) paths per node at different levels, in the DODAGs constructed by the three algorithms from the literature and by the lexicographic algorithm, when testing 50-node networks. At level 1, one node with the number of paths equal to 1 was excluded from the calculations, since, due to Proposition 1, it is always present, regardless of the algorithm used (as its only path is the shortest one (hop count = 1), it is always included in the input path set).

While for levels 2–6 no substantial differences in the average values can be observed, Table 2 shows that for these levels the equitable solution achieves significantly smaller variance values than the three other algorithms. These two observations mean that, for these levels, the numbers of shortest paths of different nodes in the equitable solution are much more balanced. The closest to it is the MARA-MC algorithm, which has also outperformed the other two algorithms considerably as regards variance values at levels 2–5. At level 1, the relation between the variance values and average values is similar for all the algorithms, although the equitable solution and MARA-MC obtain larger average numbers of paths.

Another interesting observation is that the Shortest-multipath algorithm, with the link costs calculated basing on the 2D distance, obtains greater average numbers of shortest hop count paths than the TORA approximation algorithm, which is based on the hop count metric. This happens probably because, in the simulated Shortest-multipath, a node of a smaller cost would most likely have paths of smaller hop count lengths than a node of a greater cost. In the case of the TORA algorithm, no additional information is available

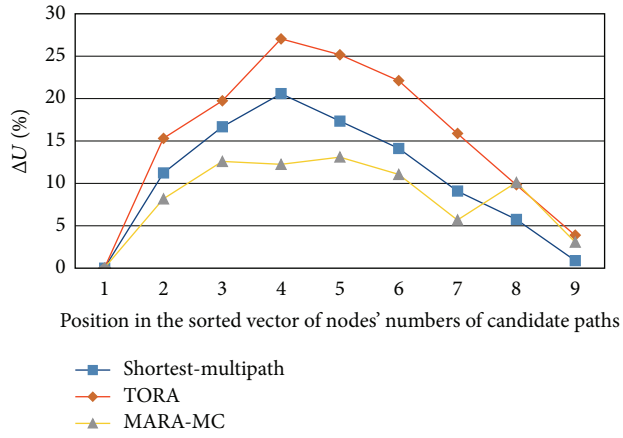

Figure 2: Relative comparison of the three algorithms for 10-node networks using the lexicographic k-shortest solution as a reference. ΔU values are relative to the reference (lexicographic) solution.

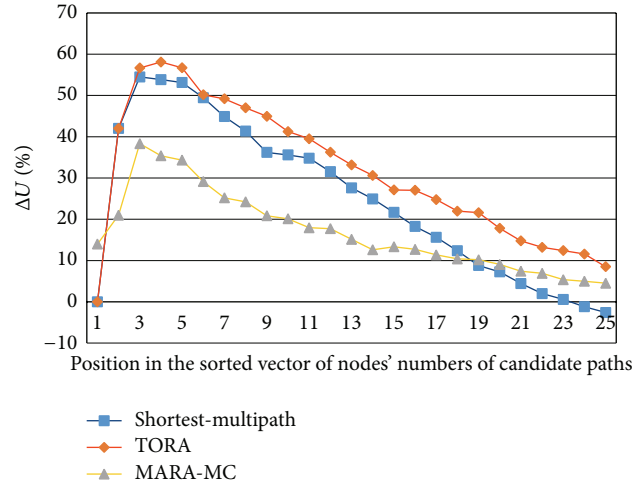

Figure 3: Relative comparison of the three algorithms for 50-node networks using the k-shortest lexicographic solution as a reference: first 25 positions in the sorted solution vectors.

to grade the nodes at the same hop count distance from the root; hence, a significant fraction of edges is assigned at random (using node identifiers).

Figure 2 shows a comparison of the three algorithms on 10-node networks, with the lexicographic k-shortest solution used as a reference. The ΔU value reflects the average gap to the reference solution on individual lexicographic positions and is defined as follows:

(i) let U_{i_ref} = the number of paths (only taking into account the input k shortest paths) in a given position in the equitable solution vector of the ith network, and let U_{i_alg} = the number of paths in the same position in the vector obtained by the tested algorithm for the ith network (previously sorted in a nondecreasing order so that a lexicographic comparison using individual positions is possible), $i = 1, 2, \ldots, 50$;

(ii) let $\Delta U_i = ((U_{i\_\mathrm{ref}} - U_{i\_\mathrm{alg}})/U_{i\_\mathrm{ref}}) \times 100[\%]$;

(iii) then, $\Delta U = (\sum_i \Delta U_i)/50$.

Figure 2 shows, again, that the MARA-MC algorithm is the best of the three algorithms in terms of equitability in the distribution of shortest paths, as it has, on average, vectors whose values in the lowest positions are closest to the equitable solution vectors.

An interesting observation can be made that all algorithms achieve a 0% gap in the first lexicographic position. This is due to the fact mentioned previously, namely, that one of the root's neighbours in every simple DODAG will always have exactly one path to the root, with this path being one of this node's shortest paths (hop count = 1). This node probably always occupies the first lexicographic position in the case of 10-node networks.

However, this is clearly not always the case when dealing with 50-node networks, as shown in Figure 3. It can be observed that the MARA-MC algorithm sometimes happens to have a nonzero gap in the first position, which means that, in some cases, there exists at least one node in the DODAG obtained by MARA-MC which has no paths available from

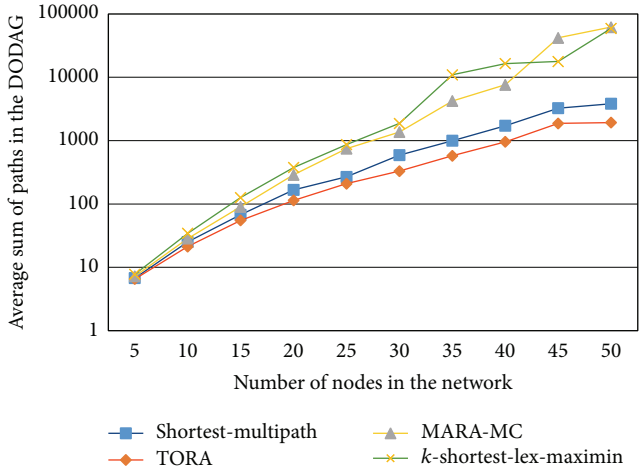

Figure 4: Average sum of all paths in the DODAGs.

the set inserted into the formulation for that node—which means that this node has only longer paths than the $k = 15$ shortest possible paths and occupies the first lexicographic position instead of the root's 1-path neighbour. Moreover, although MARA-MC performs better in the overall comparison to the two other algorithms, it is still far from equitability achieved by the lexicographic algorithm (~20%–38% gaps in positions 2–10).

Figure 4 shows the average sums of all paths in the obtained DODAGs; this time, all possible paths were enumerated (not only the candidate paths from the shortest path sets inserted into the lexicographic formulation but also the additional paths resulting from the obtained edge orientation). It can be concluded from the plot that the most balanced solutions, like the k-shortest lexicographic solution and the MARA-MC solution, also achieve the greatest numbers of paths in general.

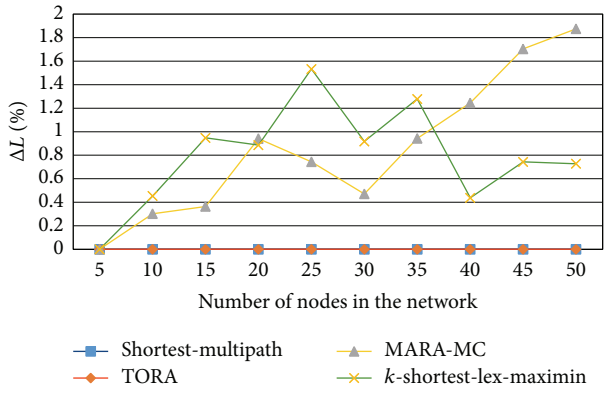

Figure 5: Average DODAG-to-network shortest path length gap in function of the size of the network.

Figure 5 shows the average gap between the hop count length of the shortest path of a single node available in the formed DODAG and the length of this node's actual shortest path, that is, this node's shortest path in the network. Hence, the ΔL value is defined as follows:

(i) let $L_{\text{in\_opt}}$ = the length of the shortest path of a node n in ith network, and let $L_{\text{in\_alg}}$ = the length of the shortest path for this node in the DODAG obtained by the algorithm;

(ii) let $\Delta L_{\text{in}} = ((L_{\text{in\_alg}} - L_{\text{in\_opt}})/L_{\text{in\_opt}}) \times 100[\%]$;

(iii) hence, $\Delta L = (\sum_i \sum_n \Delta L_{\text{in}})/(50(N-1))$, where $N - 1$ = the number of nodes in the network excluding the root.

An interesting observation can be made, namely, both MARA-MC and the lexicographic k-shortest maximin algorithm do not achieve zero ΔL values for networks greater than 5 nodes, which means that the output DODAGs do not always include the whole shortest path trees. This in turn means that to achieve fairness in terms of the numbers of input paths per node in a DODAG, even when considering sets of shortest paths as it has been done in the simulations, a trade-off has to be made between the equitability and availability of the shortest paths of some nodes.

4. Conclusions

A lexicographic optimization formulation has been proposed in this paper, solution of which achieves an equitable distribution of the paths available per node in a constructed DODAG. This property could be important in the context of multipath routing, where network topology changes are infrequent. The formulation allows insertion of the user's own sets of paths per node. This property makes the mechanism adjustable to particular requirements imposed by the application on the structure of paths, although a well-thought-out choice of the input path sets is necessary to obtain reasonable results.

A comparison of three distributed algorithms that appear in the literature: MARA-MC, Shortest-multipath, and TORA, and the lexicographic algorithm has been presented, with k shortest paths per node being inserted into the formulation. The lexicographic solution has been proved to obtain the best results in terms of providing a fair distribution of the shortest paths of the nodes at the same minimum hop count distance from the root (at the same level).

Two interesting properties of the k-shortest equitable solution have been observed. First, it has been shown that providing efficient maximin fairness in the numbers of paths per node increases the overall number of paths in the resulting DODAG. Second, even though sets of shortest paths were included in the optimization problem, the resulting DODAGs did not always contain full shortest path trees.

It has been shown that of the three distributed algorithms, MARA-MC achieves the best results, although it still does not get close to the k-shortest equitable solution in low lexicographic positions. Moreover, it turns out that, in some cases of 50-node networks, a node could be found in the MARA-MC DODAG whose shortest path available in the graph was longer than its first 15 shortest paths in the network. These observations lead to a conclusion that designing a distributed algorithm to construct a *fair* DODAG, that is, with equitable distribution of the paths per node, is still an open question.

Appendices

A. Proof of NP-Completeness of the Simplified Problem

The simplified problem is stated as follows. Given a simple directed graph $G = (N, E)$, two nodes $s, t \in N$ and a list of paths $p_1, p_2, \ldots, p_P \in P$, from node s to node t, find a subset of edges such that no two edges between the same pair of nodes are included (i.e., no edges of conflicting directions are included) and the number of available paths between s and t is maximized. Let the simplified problem be called MAX-PATHS.

It is possible that two or more paths from the P set will exclude each other because of using one or more edges of opposite directions between the same pair of nodes. For example, if, for the network shown in Figure 6, the following three paths belong to the P set: (1) s-a-t, (2) s-a-b-t and (3) s-b-a-t, then the maximum possible number of paths from s to t for the given set will be equal to 2, as paths s-a-b-t and s-b-a-t cannot both be included due to the conflict in assignment of the direction of edge ab.

There might be more than two counter-direction paths. This problem can be modeled as shown in Figure 7. Vertices $p_1 - p_5$ represent an example set of paths between s and t. An edge exists between a pair of vertices if the corresponding paths cannot be picked together due to at least one edge conflict. Therefore, the maximum number of paths from node s to t that can be chosen together is equal to the size of the maximum independent set of this graph. The vertices in Figure 7 that belong to the maximum independent set of the graph have been marked green. Hence, in this case, the objective value is equal to 3, although 5 paths were considered originally.

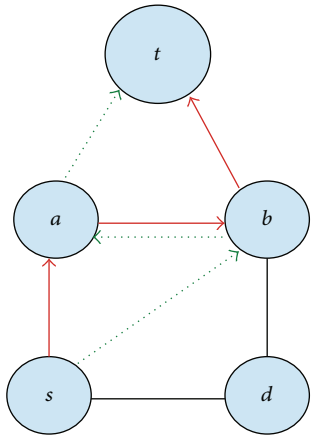

Figure 6: If for node s, paths s-a-b-t (red solid) and s-b-a-t (green dotted) are given, then these paths will not both be included due to the conflict in the used direction of edge ab.

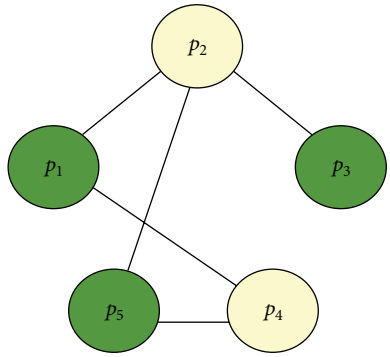

Figure 7: An edge exists between a pair of vertices, if the corresponding paths cannot coexist. For example, path p_3 cannot exist together with path p_2. Paths belonging to the maximum independent set have been marked green. In this case, the maximum set of paths that can be picked together is equal to 3.

Therefore, to show that the MAX-PATHS problem is NP-complete, instances of the maximum independent set (MIS) problem can be mapped to the MAX-PATHS problem. It can be shown that a valid solution to MIS is also a valid solution to MAX-PATHS problem and that a nonvalid solution to MIS is not a valid solution to MAX-PATHS.

Consider an MIS problem instance, modeled with a graph $G'(N', E')$. A corresponding instance of the MAX-PATHS problem, modeled by a network graph $G(N, E)$ and the set P of paths, can be constructed in the following way.

(1) For every edge $e' \in E'$ adjacent to a pair of nodes $n'_1, n'_2 \in N'$ in the MIS problem, create two vertices $n_{e'1}, n_{e'2} \in N$ in the MAX_PATHS problem.

(2) In MAX_PATHS, between each pair $n_{e'1}$, $n_{e'2}$, add two edges $e_1, e_2 \in E$ of opposite directions, hence obtaining a bipartite graph.

(3) In each MAX_PATHS pair $n_{e'1}$, $n_{e'2}$, label one of the e_1, e_2 edges with node n'_1 of the corresponding edge e' in MIS, and the other with the node n'_2.

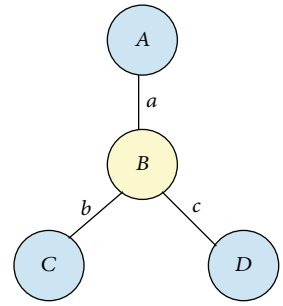

Figure 8: G' graph of the MIS problem. Vertices A, C, and D belong to the maximum independent set.

(4) In MAX_PATHS, add the source and destination nodes $s, t \in N$.

(5) For each node $n' \in N'$ in MIS, build a path $p_{n'} \in P$ from s to t in MAX_PATHS, going through all the edges labeled with n', and adding necessary edges $e \in E$: between s and the $N \setminus \{s, t\}$ set, between the $N \setminus \{s, t\}$ set and t, and between the nodes in the $N \setminus \{s, t\}$ set, belonging to different pairs $n_{e'1}, n_{e'2}$. The order of constructing the paths is arbitrary.

Proposition 2. *An independent set in G' corresponds to a set of nonconflicting paths in G.*

In the MAX_PATHS problem, there is one path $p_{n'} \in P$ from s to t for each node $n' \in N'$ in MIS. Each of these paths goes through exactly $2M_{n'}$ nodes, where $M_{n'}$ is the number of n''s neighbours in G'. More precisely, each path crosses $M_{n'}$ pairs of $n_{e'1}, n_{e'2}$. Every time a path is selected, an edge orientation between $n_{e'1}$ and $n_{e'2}$ is fixed, forbidding the selection of a conflicting path.

(6) Notice that if an independent set on G' corresponds to a set of paths that do not have conflicting edges in G, the maximum independent set in MIS will correspond to the maximum number of nonconflicting paths from s to t in MAX_PATHS.

The mapping process has been shown in Figures 8, 9, 10, and 11.

B. Proof of Proposition 1

Proposition 1 is proved as follows:

(i) consider a network comprising a single node which is the root of the DODAG;

(ii) add a neighbouring node A with a single edge directed to the root. Node A has exactly one path to the root. To increase node A's number of paths to the root without cycles, another node has to be added that has a connection both to the root (but not via node A) and from node A;

(iii) add a node B, with an edge directed to the root and an edge incoming from node A. Node A has now two

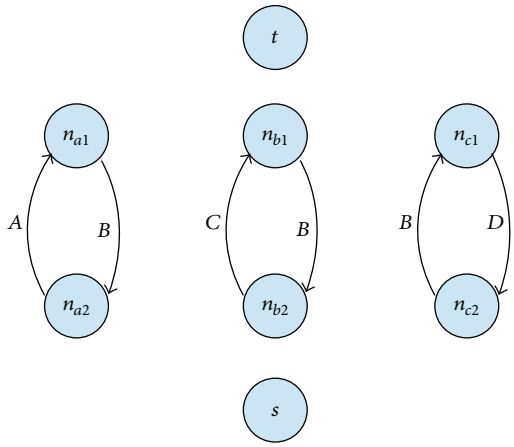

FIGURE 9: G mapping after step 4.

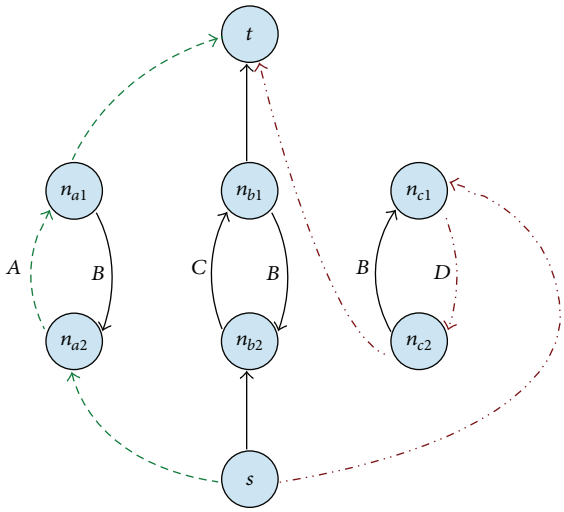

FIGURE 10: A, C, and D paths belonging to the maximum independent set. These paths do not have any conflicting edges.

paths to the root; however, node B is now the one that has only one path to the root;

(iv) adding an edge from node B to node A will result in a cycle; therefore, the only way to provide node B with a second path is to connect it to another node, which has either a direct link to the root (is another neighbour of the root) or has a path to the root which goes through another neighbour of the root. This neighbour will, again, either have only one path to the root or its other paths will eventually lead to another root's neighbour with this property.

The proof is shown in Figures 12, 13, and 14.
The proof of this proposition implicates that

(i) the minimum number of paths of a single node to the root in a simple DODAG is always equal to one;

(ii) the minimum connectivity between a single node and the root in a simple DODAG is always equal to one;

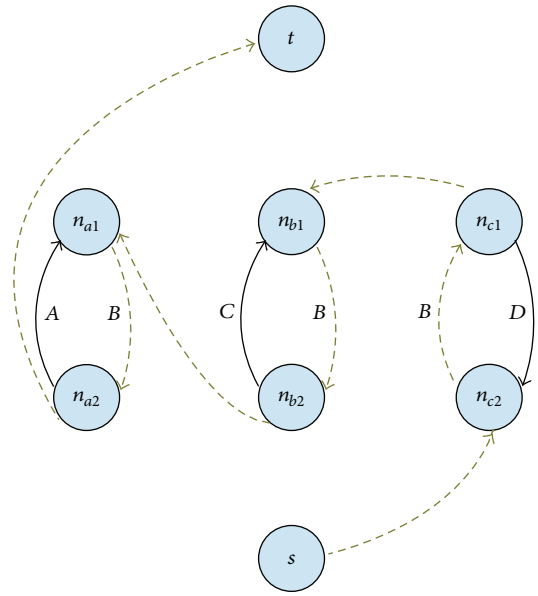

FIGURE 11: B path, conflicting with the paths from the maximum independent set due to different selection of edge directions.

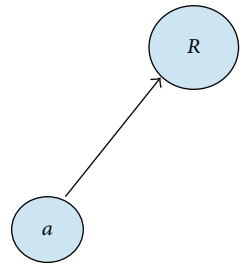

FIGURE 12: Node a has only one path to the root.

TABLE 3: Parameter values for the WPA algorithm.

Parameter	Value
r	12
d_0	6
l	100

(iii) this one path is the shortest path of the node (hop count = 1).

C. Parameter Values for the WPA Algorithm

Table 3 contains parameter values for the WPA algorithm. The same values were used for networks of different sizes.

Change in the Generation Procedure. Instead of calculating the transmission range after generating node positions according to the length of resulting edges, it was assumed to be equal to r to ensure connectivity and prevent repeated generation of node positions.

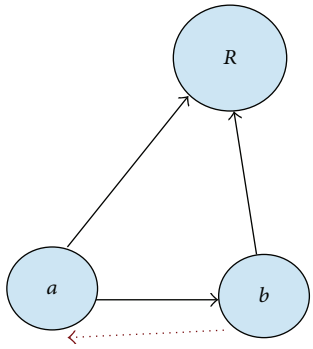

Figure 13: Node a has now two paths to the root, but node b has just one. Creating a second path for node b by adding an edge directed to node a will create a cycle.

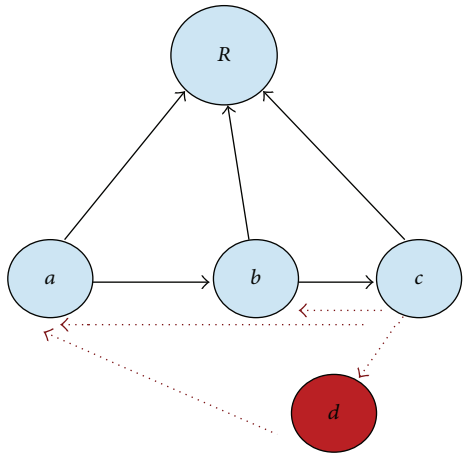

Figure 14: A similar situation happens when a third node is added. And so on.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Dr. Artur Tomaszewski for suggesting the iterative method of simplifying the formulation by reducing the number of cycle elimination constraints, mentioned in Section 2.5 of this paper, and Dr. Hanan Luss for a small but valuable hint and his kind willingness to help. The research presented in this paper was partially supported by the National Science Centre (Poland) under Grant 2011/01/B/ST7/02967 "Integer programming models for joint optimization of link capacity assignment, transmission scheduling, and routing in fair multicommodity flow networks." The research presented in this paper has received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under Grant Agreement no. 269985.

References

[1] T. Winter, P. Thubert, A. Brandt et al., "RPL: IPv6 routing protocol for low-power and lossy networks," Internet Engineering Task Force RFC 6550, 2012.

[2] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: a distance-vector multipath routing protocol," in *Proceedings of the 20th Annual Joint Conference on the IEEE Computer and Communications Societies (IEEE INFOCOM '01)*, vol. 1, pp. 557–564, Anchorage, Alaska, USA, April 2001.

[3] S. Vutukury and J. J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information," in *Proceedings of 8th IEEE International Conference on Computer Communications and Networks*, pp. 534–539, Boston, Mass, USA, 1999.

[4] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '99)*, pp. 227–238, Cambridge, Mass, USA, 1999.

[5] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *Wireless Networks*, vol. 1, no. 1, pp. 61–81, 1995.

[6] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution (IEEE INFOCOM '97)*, p. 1405, Kobe, Japan, April 1997.

[7] Y. Ohara, S. Imahori, and R. van Meter, "MARA: maximum alternative routing algorithm," in *Proceedings of the 28th Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 298–306, Rio de Janeiro, Brazil, April 2009.

[8] S. Cho, T. Elhourani, and S. Ramasubramanian, "Independent directed acyclic graphs for resilient multipath routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 153–162, 2012.

[9] M. Pióro and D. Medhi, "Chapter 5, section 5.5: gradient minimization and other approaches for convex programming problems," in *Routing, Flow and Capacity Design in Communication and Computer Networks*, Morgan Kaufmann, Boston, Mass, USA, 2004.

[10] K. Zhang and H. Gao, "Finding multiple length-bounded disjoint paths in wireless sensor networks," *Wireless Sensor Network*, vol. 3, no. 12, pp. 384–390, 2011.

[11] H. Luss, "Chapter 1: introduction. Chapter 2: nonlinear resource allocation. Chapter 7: equitable resource allocation with integer decisions, section 7.2: problems with a limited number of distinct outcomes," in *Equitable Resource Allocation: Models, Algorithms, and Applications*, John Wiley & Sons, New York, NY, USA.

[12] F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, "Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 597–615, 2008.

[13] S. Wloczysiak, "Extending 2.4 GHz ZigBee short range radio performance with skyworks front-end modules: RF signal propagation attenuation," *Microwave Journal*, 2009.

[14] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.