Electric Power Systems Research 241 (2025) 111278

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Efficient handling of radiality constraints for large-scaled power distribution networks

Ludovico Nati [a,*], Renato Bruni [b], Marco Maccioni [a], Alberto Geri [a]

[a] Department of "Ingegneria Astronautica, Elettrica ed Energetica", University "Sapienza" of Rome, Italy
[b] Department of "Ingegneria Informatica, Automatica e Gestionale", University "Sapienza" of Rome, Italy

ARTICLE INFO

Keywords:
Distribution networks
Graph models
Radiality constraints
Combinatorial optimization
Constraint generation
Lazy constraints

ABSTRACT

Power distribution networks are usually characterized by a radial topology and therefore the related optimization problems require radiality constraints in their formulations. However, practical instances may have very large size, and the number of radiality constraints may grow faster than the size of the instance. Hence, the arising optimization models may become computationally intractable due to their huge dimension. This work proposes a combinatorial optimization approach to overcome this issue. Our approach is based on the combinatorial formulation of the radiality constraints and their delayed and efficient generation in the model, following a separation-optimization scheme. We find an optimal acyclic spanning subgraph subject to both technical side constraints and topological radiality requirements on a graph representing the distribution network. This can be done without considering all the exponentially many radiality constrains from the beginning of the model solution, but introducing only the ones that are needed to make the solution feasible. It turns out that the number of required constraints is much smaller than the number of all possible radiality constraints, also because some of the electric constraints already favor the elimination of infeasible configurations. Computational experiments on reconfiguration benchmarks from the literature show the effectiveness of the proposed approach.

1. Introduction

The vast majority of distribution networks (DNs) have a meshed topology but are radially operated, i.e. each *secondary substation* must be supplied by a single *primary substation*, and the connection must have no loops. Radial operation is preferred for a number of technical reasons, such as simple protection scheme coordination and short circuit current reduction [1,2].

In general, optimization problems involving DNs are formulated by means of graph-based models, normally imposing some *radiality constraints* (RC) within a *mixed-integer program* (MIP) encoding the graph model [1–5]. In more detail, a DN is usually modeled as an *undirected* graph, taking the nodes as *vertices* and the branches or lines as *edges*. The two electrical conditions above correspond in the graph model to the identification of a *subgraph* such that: *(a)* the subgraph *spans* all vertices of the graph; *(b)* the subgraph has no *loops* or *paths* between primary substations; *(c)* each connected component of the subgraph must include exactly one primary substation. In other words, we search for particular *spanning trees* or *forests* [6].

Radiality check of DNs is a task often performed by the Distribution System Operator (DSO): in normal operation the DN is periodically reconfigured in order to minimize power losses; in fault conditions, after the detection and isolation of the faulted branch, the DN is reconfigured to restore service to the largest possible number of customers while optimizing other objectives relevant to the DSO. When dealing with real-world distribution networks, we are particularly interested in studying techniques to solve large sized instances, for several reasons. Firstly, because electric networks constantly grow in size and complexity, and since all technological improvements constantly drive this process ahead, such a trend will likely continue in the future. Moreover, for large instances the choice of the solution approach usually makes the difference between having good quality solutions in acceptable times or not, whereas in the case of small instances the choice of the solution algorithm appears less crucial.

Several approaches to ensure radiality of a DN have been proposed in the literature. Works [7,8] use a Spanning Tree (ST) formulation; however, this ST formulation is a necessary but not sufficient condition to ensure radiality, as shown in [4,9]. Ref. [10] proposes a radial constraint representation based on a loop elimination approach. Models based on single commodity flow formulation are proposed in [11,12],

* Corresponding author.
E-mail addresses: ludovico.nati@uniroma1.it (L. Nati), bruni@diag.uniroma1.it (R. Bruni), marco.maccioni@uniroma1.it (M. Maccioni), alberto.geri@uniroma1.it (A. Geri).

https://doi.org/10.1016/j.epsr.2024.111278
Received 9 August 2024; Received in revised form 8 November 2024; Accepted 17 November 2024

whereas a spanning forest model specifically designed to fully enable the topological flexibility of DNs is detailed in [13]. Work [14] proposes a formulation based on the concept of maximum density in graph theory to enforce radiality. Various formulations are also reviewed and compared in [15].

However, radiality formulations require a number of constraints that grows quite fast with the size of the instance, sometimes even *exponentially*. Therefore, the produced models tend to have very big size. This, together with the fact that solving the larger instances is particularly required in practice, leads to the need for solving larger and larger optimization models. Since in general the computational complexity of solving optimization models grows with the size of those models, the larger instances remained unsolved. Therefore, recent years have seen the proposal of more flexible formulations to address the challenges posed by large DNs and avoid the use of exponentially many constraints.

Aiming at ensuring that the DN has no loops and paths between primary substations, Ref. [16] proposes an efficient algorithm able to find all the simple cycles (i.e. each connected cycle without repeated vertices) in a graph through the construction of a loop basis and adding constraints that ensure the opening of at least one line of each simple cycle. In [5], an iterative approach to identify only the most important loop constraints is proposed to overcome scalability issues of existing RC formulations. However, both [5,16] do not add constraints dynamically. In [2], an algorithm specifically tailored for large DNs and able to add constraints dynamically whenever a continuous solution is obtained within the execution of a branch-and-cut is proposed to boost the branching process. However, the algorithm in [2] only works on arbitrarily selected critical nodes.

On the other hand, a powerful approach to the solution of very large models developed in the field of Combinatorial Optimization is the *constraint generation* strategy, see e.g. [17,18]. This approach is based on the solution of an optimization model not containing (i.e., relaxing) some set of constraints (usually the ones making the model difficult). Such a technique proceeds with the iterative generation (i.e., introduction in the model) of only some of the constraints from the above set: those that are needed to cut away infeasible solutions possibly obtained in the solution process. The constraints added at integer infeasible solutions are also known as "lazy constraints" and are the ones strictly necessary to ensure the correctness and the feasibility of the solution found. Note that generally only a small subset of the set of all relaxed constraints has to be added. Constraint generation requires some effort to be carefully tailored on the specific model, in order to be computationally convenient. In particular, the algorithm to check the relaxed constraints and generate the violated ones, called *separation procedure*, has to be developed for the specific type(s) of relaxed constraints. If not properly designed, this technique could even increase the computational burden. The convergence of the algorithm to an optimal solution (except for numerical imprecisions) is guaranteed, if the separation procedure is correctly designed, see e.g. [18,19].

Introducing the full features of this approach in the solution of a radiality constraints formulation is still an open problem, as it require identifying an appropriate formulation and designing a convenient separation procedure. The present work addresses at closing this research gap, by proposing a new formulation of radiality constraints, called Ω_{RC}, based on two sets of inequalities: a first set enforcing the elimination of cycles and of paths including more than one primary substation; a second set imposing that each secondary substation is connected to at least one primary substation. In case only one primary substation appears in the system, the proposed formulation still works; it will simply reduce to finding a spanning tree. The proposed formulation is tight but requires a number of constraints that grows exponentially with the size of the instance. However, we have been able to develop an innovative constraint generation strategy and implement it using a callback routine in GUROBI Optimizer 10.0.1.

We present experiments on reconfiguration problems using our formulation Ω_{RC}, alone or in combination with some of the constraints taken from two other known formulations: the so called single-commodity flow + number difference equation (SCF0, [1,3]) and a combination of SCF and spanning tree constraints (SCF + ST [20]), obtaining the two variants called here Ω_{RC0} and Ω_{RC} + ST. We also compare the above results to those produced by the original versions of SCF0 and SCF + ST on publicly available standard IEEE DNs.

Hence, the main contributions of this work are the following.

- A novel formulation of radiality constraints which is tight but requires exponentially many constraints is presented. Some variants, obtained by including in our formulation constraints taken from other known formulations for the same problem, are also presented.
- The large models corresponding to this formulation can be practically solved through a constraint generation approach making use of lazy constraints, which is an advanced technique in combinatorial optimization successfully applied in various fields including logistics, manufacturing and transportation [17,18].
- The application of a constraints generation approach requires the design of a separation procedure. As another innovative point, we propose an efficient separation procedure (polynomial in complexity) for DN, which is used to generate in some cases more than one constraint at a time [17,21].
- Comparisons with state-of-the-art formulations on publicly available IEEE DNs show the soundness and effectiveness of the proposed approach. Moreover, the proposed approach exhibits the best advantages on the larger instances.

The rest of this article is organized as follows. Section 2 provides the basic problem description by introducing a graph model of DNs, and by describing the generic DN optimization model (DNO). Section 3 presents some convenient formulations of the problems, which however present the drawback that the number of constraints grows exponentially with the size of the graph. Section 4 describes the combinatorial optimization technique used to generate only the needed radiality constraints and not all the exponentially many. Section 5 reports computational results on real-life IEEE reconfiguration instances going from moderate to very large size. Finally, Section 6 draws conclusions. Furthermore, Appendix contains the technical evaluation of the strength of the proposed formulation.

2. Problem description

A DN usually contains: a set R of primary substations, also known as *root* (or *supply*) nodes, and a set D of secondary substations, also known as *demand* (or *load*) nodes, connected by the branches of the network. The undirected graph model $G(V, E)$ of the DN will therefore have the set of *vertices* $V = V^R \cup V^D$, with V^R the set of *root vertices* and V^D that of *demand vertices*, and the set of *edges* E associated with the above branches and such that the graph is connected. Edges and vertices may have additional attributes (e.g., capacity, cost) according to the specific DN problem.

The *basic* problem to establish a radial configuration for the DN consists in finding a subgraph $H(V_H, E_H)$ of $G(V, E)$ satisfying the following *basic radiality* conditions:
(a) the subgraph *spans* all vertices of G (i.e., $V_H = V$);
(b) the subgraph has no *loops* (i.e., it is acyclic) or *paths* between primary substations
(c) by calling k the number of roots, the subgraph H has exactly k *connected* components (i.e., it is a *spanning forest* composed of k trees) and each of them contains *exactly* one root vertex.

Condition *(c)* enforces the desired subgraph $H(V_H, E_H)$ satisfying the radiality constraints to be a *k-rooted spanning forest* [6,21].

The search for such *k-rooted spanning forest* $H(V, E_H)$ can be encoded into an optimization model, where a set of binary decision

variables x associated with all the edges will provide, upon solving, the *incidence vector* of H. This optimization model will pursue the objective (1) that is relevant for the considered DN application, and will be generally subject to *radiality constraints* RC (2) and to *technical side constraints* SC (3). We denote by Ω_{RC} and Ω_{SC} the sets of points satisfying respectively RC and SC. Technical side constraints (3) often require in their expression a set of additional variables y, which may be real-valued. Usually, such constraints impose the power flow equations, the operational limits of flows through branches and the acceptable bus voltages, whereas variables y represent branch flows and bus voltages. Hence, the conceptual DN optimization (DNO) model is:

$$\min \ F(x,y) \tag{1}$$

$$\text{Subject to :} \quad x \in \Omega_{RC} \tag{2}$$

$$x, y \in \Omega_{SC} \tag{3}$$

$$x \in \{0,1\} \tag{4}$$

$$y \in R \tag{5}$$

In the literature, RC are formulated in different ways [1–5], and some formulations refer to the *spanning forest* integer polytope based on the *loop-elimination constraints*, see (A.2) in the Appendix, hence they contain a number of constraints growing exponentially with the size of the instance.

These formulations allow the solution of moderate-sized DNO problems [16]. However, the presence of exponentially many radiality constraints (2), the often non-linear objective $F(x,y)$ (1), and the side constraints SC (3) make these models very computationally demanding: large-size DN instances rapidly become computationally intractable. In this work, we propose an alternative and stronger reformulation for the radiality constraints (2), for which we will be able to develop an efficient separation procedure.

3. Radiality constraints formulation

We now introduce the basic ideas of our reformulation, starting with the classical spanning tree formulation. Let S be a generic subset of V, and E_S the set of edges of the subgraph induced by S in G. For graph $G(V,E)$, the formulation of the spanning tree problem, i.e. the known *spanning-tree polytope* P_T, is:

$$P_T = \left\{ \begin{array}{l} x \in \{0,1\}^{|E|} \ \text{ such that} \\ \displaystyle\sum_{e \in E} x_e = |V| - 1, \\ \displaystyle\sum_{e \in E_S} x_e \le |S| - 1 \quad \forall S \subset V, \ |S| \ge 2 \end{array} \right\}. \tag{6}$$

The first constraint in the formulation forbids the creation of a *cycle* containing all the vertices, since that would require $|V|$ edges and we only impose $|V|-1$ edges. Similarly, for each subset S, the constraints in the second group implement the *loop elimination* request [6,22] by allowing only $|S|-1$ edges from E_S. Clearly, the number of possible subsets S is exponential in $|V|$, and so it is the number of constraints in the second group. On the positive side, polytope P_T is integer, and its vertices are exactly the incidence vectors of the spanning trees of $G(V,E)$ [17]. Note that this loop elimination condition, taken from the Dantzig, Fulkerson, Johnson formulation [23], is known to be particularly strong and it is stronger than enumerating all possible cycles and imposing that at least one edge of each cycle is open, as shown in the Appendix. See also [24] for further details on formulation strength. However, the obtained tree is not bounded to respect the above condition (c), as shown in the following example. Hence, we need to extend this stronger version of loop elimination constraints to obtain full radiality.

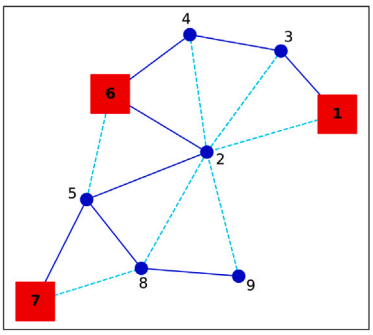

Fig. 1. Spanning subgraph (in solid lines) composed of only one connected component and with 3 roots: nodes 1, 6 and 7.

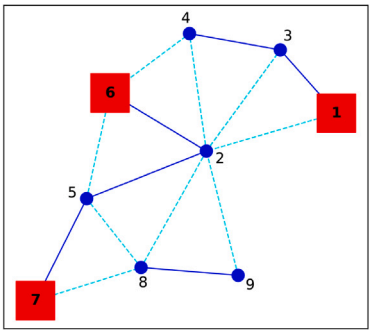

Fig. 2. Spanning forest composed of 3 connected components and with 3 roots (nodes 1, 6 and 7), but not having one root in each connected component.

Example 1. Fig. 1 depicts a case with $n = 9$ vertices, where the roots, represented in red squares, are $V^R = \{1,6,7\}$, and the demand vertices are $V^D = \{2,3,4,5,8,9\}$. They are connected by the 15 edges $E = \{(1,2),(1,3),(2,3),(2,4),(2,5),(2,6),(2,8),(2,9),(3,4),(4,6),(5,6),(5,7),(5,8),(7,8),(8,9)\}$. A solution satisfying constraints (6) is $x = (0,1,0,0,1,1,0,0,1,1,0,1,1,0,1)$, which is reported in the figure by drawing in solid blue the edges at 1, and in dotted cyan the edges at 0. It is easy to see that it has exactly $n-1 = 8$ edges at 1 and that it has no loops. However, it does not respect the above condition (c).

If we replace the constraint on the number of edges $\sum_{e \in E} x_e = |V| - 1$ with $\sum_{e \in E} x_e = |V| - k$, we split the single spanning tree into k different spanning trees, hence a spanning forest. Thus, we obtain the *k-spanning forest polytope* P_{kSF} of $G(V,E)$, i.e. the formulation of a spanning forest having k connected components, each of which is a tree.

$$P_{kSF} = \left\{ \begin{array}{l} x \in \{0,1\}^{|E|} \ \text{ such that} \\ \displaystyle\sum_{e \in E} x_e = |V| - k, \\ \displaystyle\sum_{e \in E_S} x_e \le |S| - 1 \quad \forall S \subset V, \ |S| \ge 2 \end{array} \right\} \tag{7}$$

This polytope preserves integrality [17,21]. If $k = |V^R|$, we are producing a number of trees equal to the number of roots. However, we still may have trees in which there is not exactly one root. Fig. 2 shows such an example.

Example 2. Fig. 2 represents the same case with $n = 9$ vertices and 15 edges. A solution satisfying constraints (7) is for example $x = (0,1,0,0,1,1,0,0,1,0,0,1,0,0,1)$, which is reported in the figure by drawing in solid blue the edges at 1, and in dotted cyan the edges at 0. It is easy to see that it has exactly $n - k = 6$ edges at 1 and that it contains 3 trees. However, it still does not respect condition (c).

Condition (c) further requires $H(V, E_H)$ to be a *k-rooted spanning forest* of $G(V, E)$. Hence, we propose to use the formulation that we call P_{kRSF} polytope:

$$P_{kRSF} = \left\{ \begin{array}{l} x \in \{0,1\}^{|E|} \text{ such that} \\ \sum_{e \in E} x_e = |V| - k, \\ \sum_{e \in E_S} x_e \le \min(|S| - 1,\ |S \cap V^D|) \\ \qquad \forall S \subset V,\ |S| \ge 2 \end{array} \right\} \tag{8}$$

Now the constraints in the second group enforce the selected subgraph to have no *loops* and at most one *root vertex* in each *connected component* (tree). This holds because we are requiring that the number of edges for each subset S is not larger than both $|S|-1$ and the number of demand nodes that happen to be in S. If for example S has 6 nodes, 4 demand nodes and 2 roots, then we can only take up to 4 edges in it. On the other hand, when the subset composed only of those 2 root nodes is considered (all subsets must be considered), no edges in it are allowed, so the 4 edges in the example subset S can only connect 1 root per connected component. Fig. 3 shows such an example.

Example 3. Fig. 3 represents the same case with $n = 9$ vertices and 15 edges. A solution satisfying constraints (8) is for example $x = (0,1,0,0,1,1,0,0,1,0,0,\ 0,0,1,1)$, which is reported in the figure by drawing in solid blue the edges at 1, and in dotted cyan the edges at 0. It is easy to see that it has exactly $n - k = 6$ edges at 1 and that it contains 3 trees, each of which has exactly 1 root, so it finally respects the above conditions (c).

Note that this set of constraints defines an equivalent but stronger formulation than the enumeration formulation proposed in [2,16] where all the cycles or paths between two roots must have at least an open edge. This insight constitutes one of the contributions of this work; a detailed proof of this is in Appendix. This means that even the strongest condition (c) in the radiality constraints is now satisfied. However, it is easy to see that the number of constraints in the second group grows with $O(2^{|V|})$. Therefore, the size of this model increases very rapidly with the size of the DN instance.

For the implementation of our separation procedure, instead of (8), we prefer to use the following equivalent reformulation for Ω_{RC}, where the role of both sets of nodes V^R and V^D is explicit. This because we noticed in our experiments that this reformulation often allows slightly shorter solution times, especially in the case of large instances. We express those constraints using the *cutset* $\delta(S)$ associated with every possible subset S of the set of demand nodes V^D [6,22].

Proposed Ω_{RC} formulation

$$\Omega_{RC} = \left\{ \begin{array}{l} x \in \{0,1\}^{|E|} \text{ such that} \\ \sum_{e \in \delta(S)} x_e \ge 1, \quad \forall S \subseteq V^D,\ S \ne \emptyset \\ \sum_{e \in E_S} x_e \le \min(|S| - 1,\ |S \cap V^D|) \\ \qquad \forall S \subset V,\ |S| \ge 2 \end{array} \right\} \tag{9}$$

Formally, constraints in the first group of (9) prescribe the selection of at least one edge of the cutset $\delta(S)$ associated with any subset S of the set of demand nodes V^D. This means that, for every nonempty subset of demand nodes, there is at least one edge connecting them to something else, hence to a root node (or to another demand node that in turn must be connected to a root node due to another constraint on a larger S), so that islands (i.e., connected components without a root node) are forbidden. Note that transfer nodes, if present, can be treated by simply considering them demand nodes with demand = 0. Note also that even the constraints in the first group of (9) are exponentially many.

We also recap here two state-of-the-art complete radiality constraints formulations: *SCF0*, presented in [1,3], and *SCF+ST* presented in [4], that will be used for comparison.

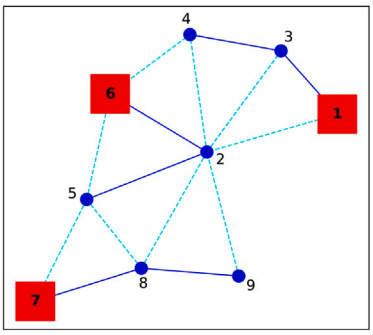

Fig. 3. Spanning forest composed of 3 connected components and with 3 roots (nodes 1, 6 and 7), having exactly one root in each connected component. This is a radial configuration.

Single Commodity Flow (SCF) is a formulation which guarantees connectivity of all demand nodes to at least one root by assuming that each demand node has a fictitious demand of a commodity and each root is a fictitious source of that commodity. A direction is defined for each edge and the fictitious flow is defined positive or negative according to this direction. The symbols $\delta(i)^+, \delta(i)^-$ of (10) refer to the inward and outward star of node i with respect to the above directions. The SCF equations are:

$$SCF = \left\{ \begin{array}{l} x \in \{0,1\}^{|E|},\quad F \in R^{|E|} \text{ such that} \\ \sum_{e \in \delta(i)^+} F_e + D_i = \sum_{e \in \delta(i)^-} F_e \qquad \forall i \in V^D \\ -|V^D| x_e \le F_e \le |V^D| x_e \qquad \forall e \in E. \end{array} \right\} \tag{10}$$

The first constraint of (10) imposes the flow conservation for all the demand nodes, and the second constraint ensures that the flow can be different from 0 only on active branches. Moreover, if we take the first constraint of (8) and all constraints of (10) we obtain the known formulation called SCF0, which imposes, in addition to SCF, that the number of active branches is equal to the number of demand nodes.

The spanning tree (ST) constraints instead are based on parent–child relations of the branches and provide a necessary but not sufficient condition to ensure a radial topology to the network [4,9]. In these case, for each $(i,j) = e \in E, i, j \in V$ we define two sets of binary variables $b1_{i,j}$ and $b2_{i,j}$. $b1_{i,j}$ is equal to 1 if i is the parent of j and vice-versa for $b2_{i,j}$.

$$ST = \left\{ \begin{array}{l} x, b1, b2 \in \{0,1\}^{|E|} \text{ such that} \\ b1_{i,j} + b2_{i,j} = x_e \qquad \forall e = (i,j) \in E \\ \sum_k b1_{k,i} + \sum_k b2_{i,k} = 1 \qquad \forall i \in V^D \\ b1_{k,i}, b2_{i,k} = 0 \qquad \forall i \in V^R \end{array} \right\} \tag{11}$$

The first equation of (11) states that, if line (i,j) is connected, either i is parent of j or vice-versa. The second equation imposes that every demand node has exactly one parent, whereas the last equation states that root nodes have no parent. Now, if we take all the constraints of (10) and (11) we obtain the formulation called SCF+ST.

Finally, by adding to our formulation (9) some of the constraints taken from other formulations, we obtain the following two new formulation variants for the radiality constraints:

- Ω_{RC} formul. (9) and all constraints of (11), called Ω_{RC}+ST formulation;
- Ω_{RC} formul. (9) and the first constraint of (8), called Ω_{RC0} formulation.

4. Separation procedure

As remarked above, the constraints in both the first and the second group of Ω_{RC} (9) are exponentially many. Nevertheless, we avoid the

explicit use of those constraints by introducing a *Separation Procedure* (SP) to either *certify* the feasibility of a given solution (x, y) or *generate* a valid constraint of (9) violated by it. This technique is also known as *constraint generation*, or *separation optimization*, and constitutes a very powerful technique used in combinatorial and integer optimization [17,18]. This approach is especially useful for the considered RC models which, despite being strong formulations, have exponentially many constraints.

In practice, we start the solution of DNO by *relaxing* (i.e., removing) the RC (2) and postponing their generation (i.e., insertion into the formulation) until they are actually needed. Then, for every solution found, we apply the separation procedure to obtain one of the two following answers: (a) that solution violates radiality constraints, and at least one of them is generated and added to the formulation; (b) that solution does respect all radiality constraints, even those not explicitly added in the formulation.

Remarkably, it turns out that only a few of the exponentially many radiality constraints must be added to the formulation to obtain a solution satisfying all of them, as shown by the experiments in the next Section. This approach helps to reduce the solution time, and so it also improves the quality of the solutions achievable within a reasonable time. This can be obtained because the solver only deals with a smaller and more manageable MIP *reduced model*, which is initialized including: the objective (1), all side constraints (3), the integrality constraints (4), and relaxing all the RC (2). The main difficulty in implementing such a solution approach is the design of an efficient separation procedure, which must be developed for the specific constraints that we use.

The proposed SP, described in Algorithm 1, is based on the identification and use of the connected components of the subgraph $H(V, E_H)$ of $G(V, E)$, with $E_H = \{e \in E : x_e = 1\}$, and works as follows. We call $H'(V, E_{H'})$ the graph correspondent to an incumbent solution (x', y'), that is a *best so far* binary solution of the *reduced model* (RM) obtained during the branch-and-cut execution. The SP is divided in two parts: the first searches for violated constraints in the set

$$\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S \subseteq V^D,\ S \neq \emptyset,$$

while the second part searches for them in the set

$$\sum_{e \in E_S} x_e \leq \min(|S| - 1,\ |S \cap V^D|) \quad \forall S \subset V,\ |S| \geq 2.$$

Algorithm 1 Separation Procedure

1: **Initialization:** Extract x', y' from the incumbent solution and define the corresponding graph $H'(V, E_{H'})$
2: **Find connected components:**
$connComponents \leftarrow$ PartitionIntoConnComp(H')
3: **for** each *component* in *connComponents* **do**
4: $S \leftarrow component$ ▷ get the current component
5: **if** $S \subseteq V^D$ **then** ▷ Cond 1
6: Compute *newConstraint*1
7: AddLazyConstraint(*newConstraint*1)
8: **end if**
9: **if** $|S \cap V^R| > 1$ **then** ▷ Cond 2
10: Compute *newConstraint*2
11: AddLazyConstraint(*newConstraint*2)
12: **end if**
13: **if** $|S \cap V^R| \leq 1$ ς $|E_S \cap E_{H'}| > |S| - 1$ **then** ▷ Cond 3
14: Compute *newConstraint*3
15: AddLazyConstraint(*newConstraint*3)
16: **end if**
17: **end for**

The first part accepts (x', y') and builds the corresponding graph model $H'(V, E_{H'})$; then it partitions the set of nodes into connected components by using a *depth-first-search* (DFS) in $H'(V, E_{H'})$. After this, it cycles over all connected components. If the connected component S has only demand nodes (without root nodes) and so $S \subseteq V^D$ (Cond 1), then the corresponding violated constraints $\sum_{e \in \delta(S)} x_e \geq 1$ is generated.

The second part checks S for the following conditions:

- $|S \cap V^R| > 1$. In this case, the connected component has more than one root (Cond 2) and the violated constraint $\sum_{e \in E_S} x_e \leq \min(|S| - 1,\ |S \cap V^D|)$ is generated and added to the model.
- $|S \cap V^R| \leq 1$ but $|E_S \cap E_{H'}| > |S| - 1$, that is the active edges of the incumbent solution having both endpoints in S close a cycle (Cond 3). In this case, the connected component is not a tree and the corresponding violated constraint $\sum_{e \in E_S} x_e \leq \min(|S| - 1,\ |S \cap V^D|)$ is generated and added to the model.

If no constraint is generated by SP, then all requirements on the presence of islands, cycles and multiple roots in the connected components are met. Therefore, the considered solution (x', y') satisfies all radiality constraints, even those that were not explicitly generated. Thus, that solution is feasible for DNO.

To explore the complexity of the proposed SP, let n be $|V|$ and m be $|E_{H'}|$. The SP can be efficiently performed, because the partition in connected components is a standard algorithm based on a DFS requiring $O(n + m)$ steps. The subsequent phase is a cycle over the connected components, which means that the number of iterations is $O(n)$. Each single iteration contains the testing of 3 **if** conditions, followed by the possible computation of the corresponding constraint. Each of the first 2 **if** conditions requires a scan over the nodes, hence $O(n)$, while the computation of the corresponding constraint requires a scan over the edges, hence $O(m)$. The third **if** requires a scan over nodes and over edges, hence $O(n + m)$, while the computation of the corresponding constraint requires a scan over the edges, hence $O(m)$. So, the total complexity of the second part is $O(n(n + m))$, which is therefore the complexity of the whole SP. Thus, the complexity of the proposed SP is low order polynomial.

The overall conceptual scheme of the proposed separation-optimization method is illustrated in the *flow chart* reported in Fig. 4, where the optimization process of RM is assumed to be based on the *branch-and-cut* method. During the solution of RM, when an integer solution is found, it is compared to the *best so far* optimal solution, called *incumbent* solution. In case the new integer solution is better than the incumbent, it represents a *candidate* incumbent solution. For each of such candidate incumbent solutions obtained during the optimization process, the SP is applied either to *certify* its feasibility (so the candidate will become the new incumbent) or to *generate* a valid (lazy) constraint to add to the current RM formulation and continue the optimization process while maintaining all the generated constraints. At the end of the iterations of the RM optimization process, the latest found incumbent solution is optimal for the DNO problem since there are no violated constraints in RC.

The implementation of this approach allows to overcome the burden discussed in the literature [3,4] to explicitly generate all the exponentially many RC. Furthermore, it is facilitated by the possibility of using *lazy constraint callbacks* [18] offered by the major modern solvers available for RM. Note that a separation procedure based on the idea of the *max-flow/min-cut* algorithm can be developed in order to identify a single violated constraint in the first set of constraints in (9) even at fractional solutions in the branch-and-cut process, as the one proposed in [2].

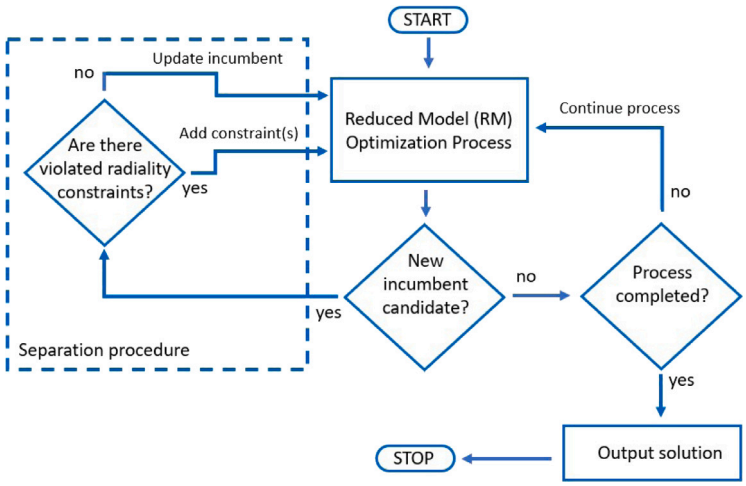

Fig. 4. Conceptual scheme of the separation-optimization.

5. Experimental results

This section compares the performance of the three following proposed formulations, all using SP (1):

- Ω_{RC} formulation (9)
- Ω_{RC} (9) and the first constraint of (8) called Ω_{RC0}
- Ω_{RC} (9) and all constraints (11) called Ω_{RC} + ST

and these two state-of-the art formulations, already described in Section 3:

- SCF0 [1,3], composed of the first constraint of (8) and all constraints of (10)
- SCF + ST [4], composed of all the constraints of (10) and all those of (11).

We use these five strategies for the *reconfiguration* of publicly available IEEE networks [25]. The distribution network is modeled through the Kirchhoff's laws equations presented on page 9 of [26], which employ the linearized network model proposed in [27]. These problems are taken from [28]; the objective is the minimization of a convex quadratic function representing *loss minimization*. Note that, when solving real-world reconfiguration problems, the emphasis is generally not on the *pure* optimality of the solution, but rather in finding in *short times*, and sometimes in real-time (in the sense of compulsory time constraints) a solution that could even be near-optimal. For this reason, in our experiments we chose a maximum time limit of 1 h, with checks at smaller intervals of 15 min and 30 min, and on the other hand we slightly relax the numerical precision by accepting, as optimal solutions, those having an optimality gap $\leq 5\%$ or $\leq 10\%$. Recall that the optimality gap in a minimization problem is defined during the solution process as the percentage distance between the current best integer solution and the highest lower bound on the integer solution, and it decreases as the branching algorithm approaches the optimal solution.

All the models are implemented in Python and solved by means of the Gurobi Optimizer 10.0.1 with the default solver settings, except for parameter MIPGAP used to accept solutions with the described optimality gaps, setting PRESOLVE at level 0 instead of −1 (to prevent the solver from altering the model), and using the lazy constraints callback for formulations Ω_{RC}, Ω_{RC0}, Ω_{RC} + ST. The experiments are conducted on a workstation with Intel Core i7-7820X processor (3.6 GHz clock speed) and 64 GB RAM running MS Windows 7 operating system.

In more detail, our instances have been obtained from the six testcases available in [25] going from small to very large size, namely bus_32_1, bus_83_11, bus_135_8, bus_201_3, bus_873_7 and bus_10476_84. Moreover, we added two new testcases of intermediate size, named here bus_2619_21 and bus_5835_47. The first number in the name of each testcase specifies the number of buses of the DN, i.e. the number of nodes in the graph model, while the second number refers to the number of MV feeders, i.e. the number of root nodes in the graph model; see also [25] for additional details. Testcase bus_2619_21 is obtained by combining three times bus_873_7 testcase, and testcase bus_5835_47 contains the first 47 feeders of bus_10476_84 testcase. Lastly, in each testcase all branches are candidate to be opened/closed to perform network reconfiguration.

In the first two Tables, for each instance/model, we allow a computation time of 3600 s, when exceeded we report (*). For each instance, we report if the optimization model has been solved within that time at the chosen accuracy (gap within 5% or 10%); the value of the best solution obtained in the time limit (for the cases ended in timeout, we add the symbol *); the corresponding optimality gap; the computation time and the number of generated constraints. When there is a *clear winner* on an instance, we highlight the winning results in boldface. We deem that there is no clear winner when the instance is solved with all formulations with very similar results or when the instance remains unsolved within the time limit with all formulations. Hence, a clear winner can be found when: only some of the formulations allow to solve the instance within the time limit and the chosen optimality gap, and moreover the difference in the solutions obtained is numerically appreciable.

The analysis of the results shows that the proposed Ω_{RC} and its variants has a comparable behavior to the two state-of-the-art formulations on small instances, which are optimally solved by all approaches. On the contrary, Ω_{RC} and its variants exhibits a distinctly better behavior on the larger instances. In particular, in Table 1 there is a clear winner (as defined above) in 2 cases, and that is always $\mathbf{\Omega_{RC}}$. In Table 2 there is a clear winner in 4 cases, and that is $\mathbf{\Omega_{RC}}$ in 3 cases and its variant $\mathbf{\Omega_{RC}}$+**ST** in the remaining case (and with $\mathbf{\Omega_{RC}}$ very near to it). Thus, it appears that the proposed formulation and its variants are able to find better quality solutions (i.e. with smaller values of loss) for the larger instances.

The constraints dynamically added by the SP are generally few, despite the fact that their number is much larger, and that the complete formulation requires an exponential number of them. This confirms the adequacy of the proposed method. When the number of added constraints is 0, this means that, during the optimization process, the SP successfully examined all radiality constraints and discovered that they are already satisfied. Hence, even in this case, the obtained optimal solution is guaranteed to respect all radiality conditions. We hypothesize that the technical constraints in SC tend to force the solutions to a certain degree of connectivity and radiality even when radiality constraints are missing.

Table 1
Comparison on network reconfiguration allowing an optimality gap of ≤5%. Symbol (*) means terminated for timeout after 3600 s.

Case study	Ω_{RC}	SCF0	Ω_{RC0}	SCF+ST	Ω_{RC} + ST	Additional data
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	13.5	13.5	13.5	13.5	13.5	Solution value
bus_32_1	0%	0%	0%	0%	0%	Optimality gap
	0.05	0.1	0.1	0.1	0.08	Time elapsed (s)
	3	0	0	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	37.3	38.2	37.9	38	38	Solution value
bus_83_11	2.6%	4.8%	3%	3.6%	3.4%	Optimality gap
	0.01	0.3	0.9	0.4	0.2	Time elapsed (s)
	0	0	6	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	43.5	43.3	43.2	43.4	43.1	Solution value
bus_135_8	2.4%	2.8%	3.6%	2.7%	2.9%	Optimality gap
	2.9	1.6	1.6	1.6	0.4	Time elapsed (s)
	53	0	18	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	68.1	65.9	66.3	66.8	66.7	Solution value
bus_201_3	4.6%	4.9%	4.9%	4.2%	4.1%	Optimality gap
	1328.1	1446.5	1488.4	24.2	25.4	Time elapsed (s)
	48	0	14	0	0	# added constraints
	Y	N	N	Y	Y	Solved in time (Y/N)
	5726.7	6699.5*	6328.9*	5841.6	5852.0	Solution value
bus_873_7	**3.0%**	16.6%*	11.8%*	4.9%	4.5%	Optimality gap
	0.25	(*)	(*)	19.9	99	Time elapsed (s)
	0	0	0	0	0	# added constraints
	Y	N	N	Y	N	Solved in time (Y/N)
	17,180.0	27,351.6*	22,798.2*	17,443.7	17,872.6*	Solution value
bus_2619_21	**3.0%**	39.1%*	26.6%*	4.5%	6.1% *	Optimality gap
	1.49	(*)	(*)	125	(*)	Time elapsed (s)
	0	0	0	0	0	# added constraints
	N	N	N	N	N	Solved in time (Y/N)
	66,991.5*	71,152.4*	70,131.0*	68,624.1*	64,708.9*	Solution value
bus_5835_47	8.9%*	14.2%*	13.0%*	11.1%*	5.5% *	Optimality gap
	(*)	(*)	(*)	(*)	(*)	Time elapsed (s)
	1	0	2	0	0	# added constraints
	N	N	N	N	N	Solved in time (Y/N)
	112,592.1*	143,251.9*	144,237.4*	144,704.6*	111,139.9*	Solution value
bus_10476_84	6.7%*	26.7%*	27.2%*	27.4%*	5.5% *	Optimality gap
	(*)	(*)	(*)	(*)	(*)	Time elapsed (s)
	18	0	0	0	0	# added constraints

Table 2
Comparison on network reconfiguration allowing an optimality gap of ≤ 10%. Symbol (*) means terminated for timeout.

Case study	Ω_{RC}	SCF0	Ω_{RC0}	SCF+ST	Ω_{RC} + ST	Additional Data
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	13.5	13.8	13.9	13.9	13.6	Solution value
bus_32_1	6.1%	8.3%	8.7%	9.1%	7.0%	Optimality gap
	0.007	0.08	0.063	0.079	0.057	Time elapsed (s)
	0	0	0	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	37.3	38.2	40.4	40.0	39.4	Solution value
bus_83_11	2.6%	4.8%	9.2%	8.4%	7.6%	Optimality gap
	0.013	0.306	0.646	0.344	0.163	Time elapsed (s)
	0	0	6	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	43.5	43.3	44.0	45.5	43.4	Solution value
bus_135_8	5.4%	3.9%	5.6%	9.6%	5.2%	Optimality gap
	0.028	2.786	2.174	0.71	0.367	Time elapsed (s)
	4	0	10	0	0	# added constraints
	Y	Y	Y	Y	Y	Solved in time (Y/N)
	67.7	67.6	68.1	69.7	68.4	Solution value
bus_201_3	7.8%	7.8%	8.1%	9.4%	8.8%	Optimality gap
	0.466	1.494	4.951	12.59	0.564	Time elapsed (s)
	14	0	10	0	0	# added constraints
	Y	N	N	Y	Y	Solved in time (Y/N)
	5726.7	6699.5*	6328.9*	6090.8	6151.3	Solution value

(*continued on next page*)

Table 2 (*continued*).

Case study	Ω_{RC}	SCF0	Ω_{RC0}	SCF+ST	Ω_{RC} + ST	Additional Data
bus_873_7	**3.0%**	16.6%*	11.8%*	8.8%	9.7%	Optimality gap
	0.225	(*)	(*)	14.28	3.29	Time elapsed (s)
	0	0	0	0	0	# added constraints
	Y	N	N	Y	Y	Solved in time (Y/N)
	17,180.0	27,351.6*	22,798.2*	18,445.3	18,248.8	Solution value
bus_2619_21	**3.0%**	39.1%*	26.6%*	9.7%	8.7%	Optimality gap
	1.35	(*)	(*)	72.21	18.58	Time elapsed (s)
	0	0	0	0	0	# added constraints
	Y	N	N	N	Y	Solved in time (Y/N)
	66,991.5	71,152.4*	70,131.0*	68,624.1*	**65,936.3**	Solution value
bus_5835_47	8.9%	14.2%*	13.0%*	11.1%*	**7.4%**	Optimality gap
	65.52	(*)	(*)	(*)	108.08	Time elapsed (s)
	1	0	2	0	0	# added constraints
	Y	N	N	N	Y	Solved in time (Y/N)
	112,592.1	143,251.9*	144,237.4*	144,704.6*	114,691.3	Solution value
bus_10476_84	**6.8%**	26.7%*	27.2%*	27.4%*	8.5%	Optimality gap
	20.16	(*)	(*)	(*)	932.87	Time elapsed (s)
	0	0	0	0	0	# added constraints

Table 3
Analysis of the evolution of the solution process on Network Reconfiguration. Symbol — means that the optimal solution was already found at the previous time check.

Case study	Ω_{RC}	SCF0	Ω_{RC0}	SCF+ST	Ω_{RC} + ST	
	0%	0%	0%	0%	0%	Optimality gap at 15’
	–	–	–	–	–	Optimality gap at 30’
bus_32_1	–	–	–	–	–	Optimality gap at 60’
	13.5	13.5	13.5	13.5	13.5	Solution value at 15’
	–	–	–	–	–	Solution value at 30’
	–	–	–	–	–	Solution value at 60’
	0%	0%	0%	0%	0%	Optimality gap at 15’
	–	–	–	–	–	Optimality gap at 30’
bus_83_11	–	–	–	–	–	Optimality gap at 60’
	37.3	37.3	37.3	37.3	37.3	Solution value at 15’
	–	–	–	–	–	Solution value at 30’
	–	–	–	–	–	Solution value at 60’
	0%	0%	0%	0%	0%	Optimality gap at 15’
	–	–	–	–	–	Optimality gap at 30’
bus_135_8	–	–	–	–	–	Optimality gap at 60’
	42.5	42.5	42.5	42.5	42.5	Solution value at 15’
	–	–	–	–	–	Solution value at 30’
	–	–	–	–	–	Solution value at 60’
	0%	5.0%	5.2%	0%	0%	Optimality gap at 15’
	–	4.9%	4.3%	–	–	Optimality gap at 30’
bus_201_3	–	4.8%	4.0%	–	–	Optimality gap at 60’
	65.9	65.9	66.3	65.9	65.9	Solution value at 15’
	65.9	65.9	66.0	65.9	65.9	Solution value at 30’
	65.9	65.9	66.0	65.9	65.9	Solution value at 60’
	1.9%	16.7%	11.8%	3.3%	3.0%	Optimality gap at 15’
	1.9%	16.6%	11.8%	1.9%	2.3%	Optimality gap at 30’
bus_873_7	1.9%	16.6%	11.8%	1.8%	1.6%	Optimality gap at 60’
	5726.7	6699.5	6328.9	5799.2	5783.3	Solution value at 15’
	5726.7	6699.5	6328.9	5728.4	5753.4	Solution value at 30’
	5726.7	6699.5	6328.9	5728.4	5717.4	Solution value at 60’
	3.0%	39.1%	26.6%	3.4%	6.1%	Optimality gap at 15’
	3.0%	39.1%	26.6%	3.4%	6.1%	Optimality gap at 30’
bus_2619_21	3.0%	39.1%	26.6%	3.4%	6.1%	Optimality gap at 60’
	17,180	27,351.6	227,98.2	17,363.1	17,872.6	Solution value at 15’
	17,180	27,351.6	227,98.2	17,363.1	17,872.6	Solution value at 30’
	17,180	27,351.6	227,98.2	17,363.1	17,872.6	Solution value at 60’
	8.9%	14.2%	13%	25%	5.5%	Optimality gap at 15’
	8.9%	14.2%	13%	25%	5.5%	Optimality gap at 30’
bus_5835_47	8.9%	14.2%	13%	11.1%	5.5%	Optimality gap at 60’
	66,991.5	71,152.4	70,131	81,368.1	64,708.9	Solution value at 15’
	66,991.5	71,152.4	70,131.	81,368.1	64,708.9	Solution value at 30’
	66,991.5	71,152.4	70,131	68,624.1	64,708.9	Solution value at 60’
	6.8%	26.7%	27.2%	27.4%	12.9%	Optimality gap at 15’
	6.8%	26.7%	27.2%	27.4%	5.5%	Optimality gap at 30’
bus_10476_84	6.8%	26.7%	27.2%	27.4%	5.5%	Optimality gap at 60’
	112,592.1	143,251.9	144,237.4	144,704.6	120,490.0	Solution value at 15’
	112,592.1	143,251.9	144,237.4	144,704.6	111,139.9	Solution value at 30’
	112,592.1	143,251.9	144,237.4	144,704.6	111,139.9	Solution value at 60’

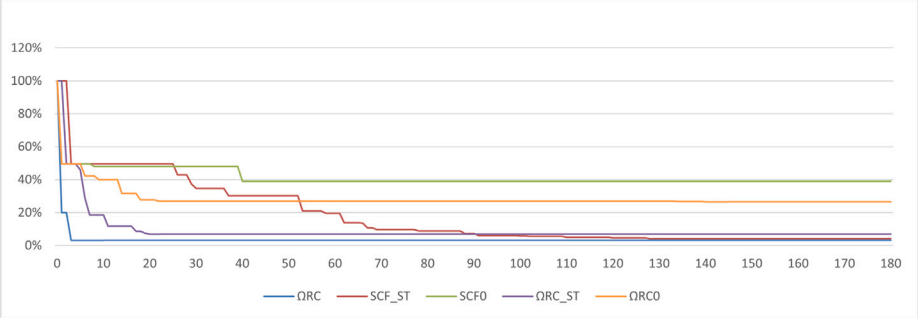

Fig. 5. Reduction of the optimality gap over time for instance bus_2619_21.

We also provide, in Table 3, an analysis of the evolution of the solution process in short times, by considering the best solution obtained within: 15 min; 30 min and 60 min. This because, in real-world cases, it would be particularly useful to be able to reconfigure the network in short times. In this case, MIPGAP was set at 0%, to allow the possibility to reach the optimal solution if that could be done within the time limit. When the optimal solution (with gap 0%) is already obtained in one of the shorter times, for the subsequent time checks there is no need to try to improve it, so we simply report symbol -. Again, we observe that, when results are not all aligned, Ω_{RC} and its variants generally allow to reach good solutions faster than the other state-of-the-art formulations, in particular for the larger instances.

We finally provide an even more detailed analysis of the optimality gap reduction over time for instance bus_2619_21 in Fig. 5. On the vertical axis we report the gap value, with all formulations stating from 100%, and on the horizontal axis computational time in seconds, limited to the first 3 min. For all formulations, the gap sharply decreases in short times, and we can observe that the fastest and most pronounced decrease is obtained with Ω_{RC}, which reaches 3%. After that, the gap tends to remain more constant for longer times, since for all formulations the exploration of all the remaining open problems in the branching tree remains quite time consuming.

6. Conclusions

Solving the larger instances of the DNO problem is particularly needed in practice, however most optimization formulations for this problem suffer from combinatorial explosion, since they need exponentially many constraints to impose the radiality conditions. Therefore, large-scale problems easily become computationally intractable. In this work, we consider the use of a polyhedral formulation for the radiality constraints in power DN reconfiguration, showing the potentiality of its use in a separation-optimization scheme. Our computational results for a set of standard test instances from the literature indicate that the proposed lazy constraints approach based on a strong combinatorial formulation is comparable or superior with the state-of-the-art models in terms of the overall quality of solutions and number of instances solved. Moreover, the proposed approach shows a significantly better behavior for large-size instances. These results suggest that the use of the proposed modeling and algorithmic approach may be key for the solution of larger and larger instances. This can foster further research in this field exploiting combinatorial optimization techniques. Future research might include the extension of the proposed approach to network planning problems, or to the case of microgrid formation, where de-energized nodes and flexible boundaries of microgrids created by distributed generators are allowed.

CRediT authorship contribution statement

Ludovico Nati: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Renato Bruni:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Marco Maccioni:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Data curation, Conceptualization. **Alberto Geri:** Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Sapienza, Italy research grant RM120172B870E2E2 is gratefully acknowledged. The authors want to thank the late Prof. Carlo Meloni for his support to this research.

Appendix. Comparison of formulations

We show here that the set of constraints (A.1), which is the second set of constraints in (9), is stronger (in the sense that it has a smaller LP feasible region) than the enumeration formulation of cycles and paths between two roots (A.2), presented in [2,16], even if it defines exactly the same set of feasible integer points:

$$\sum_{e\in E_S} x_e \le \min(|S|-1,\ |S\cap V^D|) \qquad \forall S\subset V,\ |S|\ge 2 \tag{A.1}$$

$$\sum_{e\in L} x_e \le |L|-1 \qquad \forall L\in\mathcal{L} \tag{A.2}$$

with $\mathcal{L}$ being the family of all loops and all paths between any two roots in the network.

First we show that a set of edges E', defined by an incidence vector x', contains a path between two roots or a loop if and only if it violates (A.1). In other words, the graph configurations forbidden by (A.2) are also forbidden by (A.1).

Necessity: Suppose x' defines a set E' which contains a path between two roots or a loop in the network. Let $P'\subseteq E'$ be the set of edges of this path or loop, and let S' be the set of nodes defined by P'. Define $E_{S'}$ as the set of edges with both endpoints in S', and note that $P'\subseteq E_{S'}$ and that $E_{S'}$ may be different from E'. Then, the constraint of type (A.1) corresponding to S' is violated. Indeed, if P' is a path between two roots, then $\sum_{e\in E_{S'}} x_e \ge |S'|-1$ and $|S'\cap V^D| = |S'|-2$ (or $\le$ if there are more than 2 roots in that path), so the constraint is not satisfied.

If P' is a loop (and we are not in the previous case with more than one root), then $\sum_{e\in E_{S'}} x_e \ge |S'| \not\le |S'|-1$, so again the constraint is not satisfied.

Sufficiency: Suppose that x' defines a set E' which does not contain a path between two roots or a loop in the network but violates (A.1). It follows that there is at least one induced subgraph with a set of nodes S with at most a number of edges greater than $\min(|S|-1,\ |S\cap V^D|)$.

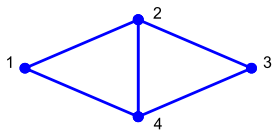

Fig. A.6. Cycle with 4 nodes and a chord.

It means that such subgraph must contain at least a path between two of its roots or a cycle, which is a contradiction. This proves that the two formulations define exactly the same sets of feasible integer points.

Now we show that (A.1) is stronger than (A.2) by showing that, if a vector $0 \leq x \leq 1$ violates (A.2), then it also violates (A.1), but the converse is not true.

If a constraint of (A.2) is violated for a given L', then the constraint of (A.1) corresponding to the set of the nodes S' of L' (i.e., the set of nodes of the path between two roots or of the loop) is violated as well. Indeed both violated inequalities are a sum of positive variables, in the left hand side of (A.2) the sum is over L' while in the corresponding inequality of (A.1) the sum is over $E_{S'} \supseteq L'$. Thus, given the same positive values of the x variables, $\sum_{e \in E'_S} x_e \geq \sum_{e \in L'} x_e$. On the other hand, for the right hand sides we have:

$$\min(|S'| - 1,\ |S' \cap V^D|) \leq |L'| - 1.$$

Therefore, any $0 \leq x \leq 1$ that is outside the formulation (A.2) is also outside the formulation (A.1).

However, if a vector $0 \leq x \leq 1$ violates (A.1), then it does not necessarily violates (A.2). As an example, take a cycle over 4 demand nodes with a chord in it (see Fig. A.6) and consider a solution $\bar{x}$ assigning each edge a value of 0.65. This graph contains 3 cycles: one over the 4 edges $C_1 = \{(1,2)(2,3)(3,4)(4,1)\}$ and two over 3 edges $C_2 = \{(1,2)(2,4)(4,1)\}$ and $C_3\{(2,3)(3,4)(4,2)\}$. The corresponding constraints (A.2) become

$$\begin{cases} C_1 : \ 0.65 \times 4 \leq 4 - 1 \rightarrow 2.60 \leq 3 \\ C_2 : \ 0.65 \times 3 \leq 3 - 1 \rightarrow 1.95 \leq 2 \\ C_3 : \ 0.65 \times 3 \leq 3 - 1 \rightarrow 1.95 \leq 2 \end{cases}$$

and they are all satisfied. Hence, solution $\bar{x}$ is within the polyhedron defined by the (A.2) constraints. Instead, at least one of the constraints of (A.1) is not satisfied: the one corresponding to the set of all nodes $\{1,2,3,4\}$ summing over all the 5 edges.

$$0.65 \times 5 \leq \min(4 - 1, 4) \rightarrow 3.25 \leq 3$$

Hence, solution $\bar{x}$ is outside the polyhedron defined by the (A.1) constraints, and so the formulation given by the (A.1) is stronger and we proved the claim.

Data availability

No data was used for the research described in the article.

References

[1] M. Lavorato, J.F. Franco, M.J. Rider, R. Romero, Imposing radiality constraints in distribution system optimization problems, IEEE Trans. Power Syst. 27 (1) (2012) 172–180.

[2] K. Pang, C. Wang, N.D. Hatziargyriou, F. Wen, Y. Xue, Formulation of radiality constraints for optimal microgrid formation, IEEE Trans. Power Syst. 38 (6) (2023) 5341–5355.

[3] R.A. Jabr, Polyhedral formulations and loop elimination constraints for distribution network expansion planning, IEEE Trans. Power Syst. 28 (2) (2013) 1888–1897.

[4] Y. Wang, Y. Xu, J. Li, J. He, X. Wang, On the radiality constraints for distribution system restoration and reconfiguration problems, IEEE Trans. Power Syst. 35 (4) (2020) 3294–3296.

[5] J. Gorka, L. Roald, Efficient representations of radiality constraints in optimization of islanding and de-energization in distribution grids, Electr. Power Syst. Res. 213 (2022) 108578.

[6] T.L. Magnanti, L.A. Wolsey, Optimal trees, Handbooks Oper. Res. Management Sci. 7 (1995) 503–615.

[7] J.A. Taylor, F.S. Hover, Convex models of distribution system reconfiguration, IEEE Trans. Power Syst. 27 (3) (2012) 1407–1413.

[8] Y. Li, J. Xiao, C. Chen, Y. Tan, Y. Cao, Service restoration model with mixed-integer second-order cone programming for distribution network with distributed generations, IEEE Trans. Smart Grid 10 (4) (2019) 4138–4150.

[9] H. Ahmadi, J.R. Marti, Mathematical representation of radiality constraint in distribution system reconfiguration problem, Int. J. Electr. Power Energy Syst. 64 (2015) 293–299.

[10] C. Xu, S. Dong, Y. Liu, S. Zeng, S. Zhang, Y. Xu, Analysis of radial constraint representation methods for distribution networks, in: 2020 IEEE Power & Energy Society General Meeting, PESGM, Montreal, QC, Canada, 2020, pp. 1–5.

[11] T. Ding, Y. Lin, G. Li, Z. Bie, A new model for resilient distribution systems by microgrids formation, IEEE Trans. Power Syst. 32 (5) (2017) 4145–4147.

[12] Y. Lin, B. Chen, J. Wang, Z. Bie, A combined repair crew dispatch problem for resilient electric and natural gas system considering reconfiguration and DG islanding, IEEE Trans. Power Syst. 34 (4) (2019) 2755–2767.

[13] S. Lei, C. Chen, Y. Song, Y. Hou, Radiality constraints for resilient reconfiguration of distribution systems: Formulation and application to microgrid formation, IEEE Trans. Smart Grid 11 (5) (2020) 3944–3956.

[14] S. Sun, G. Li, C. Chen, Y. Bian, Z. Bie, A novel formulation of radiality constraints for resilient reconfiguration of distribution systems, IEEE Trans. Smart Grid 14 (2) (2023) 1337–1340.

[15] M. Mahdavi, H.H. Alhelou, P. Gopi, N. Hosseinzadeh, Importance of radiality constraints formulation in reconfiguration problems, IEEE Syst. J. 17 (4) (2023) 6710–6723.

[16] A. Borghetti, A mixed-integer linear programming approach for the computation of the minimum-losses radial configuration of electrical distribution networks, IEEE Trans. Power Syst. 27 (3) (2012) 1264–1273.

[17] B. Korte, J. Vygen, Combinatorial Optimization, Sixth ed., in: Theory and Algorithms, Springer-Verlag, Berlin, Germany, 2018.

[18] L.A. Wolsey, Integer Programming, second ed., John Wiley & Sons, Hoboken, NJ, USA, 2020.

[19] G. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, Hoboken, NJ, USA, 1999.

[20] N.G. Paterakis, A. Mazza, S.F. Santos, O. Erdinç, G. Chicco, A.G. Bakirtzis, J.P.S. Catalão, Multi-objective reconfiguration of radial distribution systems using reliability indices, IEEE Trans. Power Syst. 31 (2) (2016) 1048–1062, (2016).

[21] R. Cordone, F. Maffioli, On the complexity of graph tree partition problems, Discrete Appl. Math. 134 (2004) 51–65.

[22] G. Laporte, Generalized subtour elimination constraints and connectivity constraints, J. Oper. Res. Soc. 37 (5) (1986) 509–514.

[23] G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale Traveling-Salesman Problem, J. Oper. Res. Soc. Am. 2 (4) (1954) 393–410.

[24] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency, Springer-Verlag, Berlin, Germany, 2003.

[25] R. Kavasseri, C. Ababei, Reds: Repository of distribution systems, 2024, [Online]. Available: http://dejazzer.com/reds.html. (Accessed November 2024).

[26] G. Muñoz-Delgado, J. Contreras, J.M. Arroyo, Distribution system expansion planning, in: F. Shahnia, A. Arefi, G. Ledwich (Eds.), Electric Distribution Network Planning, first ed., Springer, Singapore, 2018, pp. 1–40.

[27] S. Haffner, L.F.A. Pereira, L.A. Pereira, L.S. Barreto, Multistage model for distribution expansion planning with distributed generation—Part I: Problem formulation, IEEE Trans. Power Deliv. 23 (2) (2008) 915–923.

[28] A. Geri, M. Maccioni, C. Meloni, L. Nati, A. Palazzoli, Power distribution network configuration applying the corridor method, Comput. Ind. Eng. 186 (2023) 109709.