An MILP-based approach to tree partitioning with minimal power flow disruption and generator coherency constraints

Leon Lan, Alessandro Zocca
Department of Mathematics, Vrije Universiteit Amsterdam, NL
{l.lan, a.zocca}@vu.nl

***Abstract*—Tree partitioning has recently been proposed in the power systems literature as a less severe alternative to controlled islanding. In this paper, we formulate an optimization problem to tree partition a network with minimal power flow disruption and generator coherency constraints. We propose a single-stage MILP formulation to compute optimal solutions. Numerical experiments show that our MILP-based approach drastically decreases the power flow disruption when compared to an earlier proposed two-stage approach based on spectral clustering. Moreover, using a search space reduction procedure based on the Steiner Tree problem, the MILP-based approach computes near-optimal tree partitions in sub-second time for instances up to 500 buses.**

***Index Terms*—cascading failures, controlled islanding, failure localization, MILP, tree partitioning**

I. Introduction

When a cascading failure becomes uncontrollable in transmission power systems, a last-resort emergency measure is to split the network into separated islands in a procedure called *controlled islanding* [1]. As the resulting islands are no longer interconnected, the impact of any line failure is confined within the island where it occurred. Furthermore, islanding helps to stabilize or restore the network in a more controllable manner. Despite the widespread attention that this topic has received in the academic literature [2], [3], [4], [5], [6], controlled islanding remains to be hardly used in practice due to its severity.

A recent paper [7] proposed to replace controlled islanding with a less drastic measure named *tree partitioning*. The high-level idea of tree partitioning is, instead of creating islands (i.e., disconnected components), to temporarily switch off lines to obtain "quasi-island" clusters that are interconnected in a tree-like manner (see Figure 1). As shown in [7] and [8], any line failure in a tree-partitioned network causes a power flow redistribution (and hence possible subsequent failures) only within the cluster where it occurred. This means that a tree-partitioned network has the same line failure localization properties as the corresponding islanding strategy,

thus offering similar potential in mitigating cascading failures. However, tree partitioning has several advantages with respect to controlled islanding, including (i) the avoidance of load shedding, (ii) a smaller impact on the surviving network (needing fewer switching actions and less power flow redistribution), (iii) no need for generator adjustments and (iv) no need for re-synchronization when returning to normal operations.

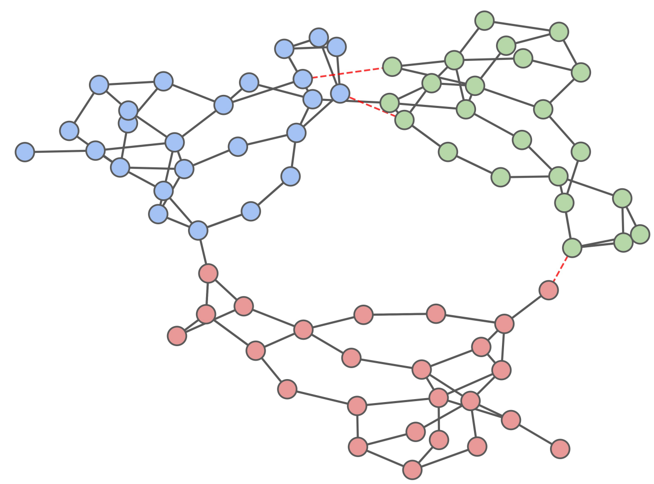


Fig. 1. A tree-partitioned IEEE-73 network. The removal of three transmission lines (in red) splits the network into three different clusters that are connected in a tree-like manner.

Given a specific network with a specific power injection configuration, it is not obvious how to determine the best tree partition: a large number of "quasi-island" clusters would offer stronger failure localization guarantees, but achieving this may require an excessive number of switching actions that could make the network unstable or unreliable. To identify the best achievable tree partition, two different optimality criteria have been considered in [7].

The first one is to minimize the network congestion, ensuring that tree partitioning does not lead to exceptionally high congestion levels on the surviving network. This is also the objective function at the core of the recursive approach for tree partitioning proposed by [8].

In this paper, we focus instead on the second criteria proposed in [7], namely minimizing the impact of the switching actions on the transient stability of the tree-partitioned network. This criterion has often been considered also in the islanding literature [3], [6], [9]. The corresponding optimization problem tries to minimize the sum of absolute flows on the switched off lines while ensuring specific generator coherency constraints. So far, the tree partitioning problem been solved with a heuristic two-stage approach in [7], [8]. The first stage identifies the best partition of the network into clusters using spectral clustering methods, while the second stage solves one of the two aforementioned optimization problems.

In this paper we propose a single-stage MILP-based approach to solve the tree partitioning problem with minimal impact on the transient stability. As opposed to the heuristic two-stage method, our approach is exact and computes optimal solutions for all the considered network instances. Moreover, we demonstrate for a large number of test networks that the MILP-based approach drastically outperforms the two-stage heuristics while having acceptable running times. To improve these running times even further, we use an existing search space reduction procedure by [9] that enhances our MILP-based approach to obtain near-optimal and even sub-second solutions for medium-sized network instances up to 500 buses.

The outline of this paper is as follows. In Section II, we formalize the concept of tree partitions. Section III introduces the tree partitioning problem formulation. In Section IV, we briefly review the two-stage approach of [7] and in Section V we introduce our MILP-based approach. Section VI presents the numerical results and we conclude the paper in Section VII.

II. Tree Partitions in Power Networks

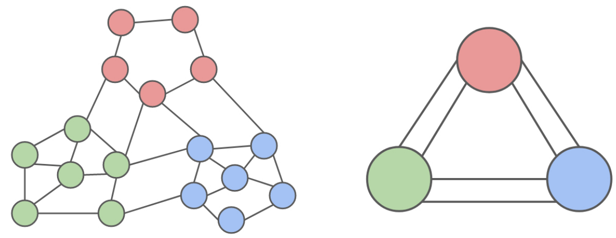
A. Graph theory preliminaries

We model the transmission network as a connected, directed graph $G = (V, E)$, where V is the set of vertices (buses) and E the set of edges (transmission lines). We denote by $n = |V|$ the number of buses and by $m = |E|$ the number of lines. We denote by f_{ij} the active power flow from bus i to bus j. Furthermore, we assume that G is a simple graph, where if needed we merge parallel lines into a single one by summing their power flows.

A k-*partition* of G is the collection $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_k\}$ of k non-empty, disjoint vertex sets $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_k$ called *clusters* such that $\bigcup_{i=1}^{k} \mathcal{V}_k = V$. We denote by $[k] = \{1, 2, \dots, k\}$ the set of integers from 1 to k. Given a partition $\mathcal{P}$, a line $(i, j) \in E$ is called an *internal edge* if both i and j belong to the same cluster and *cross edge* otherwise. The *reduced graph* $G_{\mathcal{P}}$ is the graph whose vertices are the clusters in $\mathcal{P}$ and where an edge is drawn for each cross edge connecting two different clusters. See Figure 2 for an example. Note that the reduced graph is allowed to have multiple edges between two vertices, hence it could be a multigraph. We say that $\mathcal{P}$ is a *tree partition* of G if the reduced graph $G_{\mathcal{P}}$ is a tree. In this case, each of the cross edges in the reduced graph $G_{\mathcal{P}}$ corresponds to a *bridge* or *cut-edge* of network G.

B. Failure localization properties

The failure or the disconnection of a transmission line causes the power to be *globally* redistributed on the remaining lines, some of which can overload and also get disconnected – a phenomenon to which we refer to as *failure propagation*. It was shown in [8], [10], [11], [7] that line failures do not propagate across bridges, meaning that line failures in a tree-partitioned network only impact the flows on lines in the same cluster. Most power networks, however, have a very meshed structure and only have trivial bridges, making them very prone to non-local line failure propagation [8].



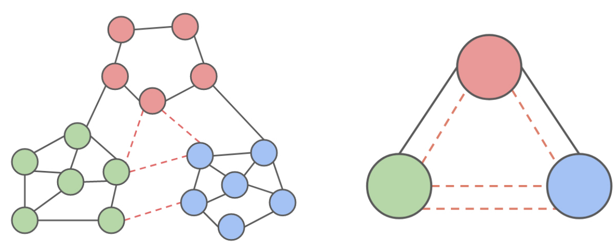
(a) (b)

Fig. 2. (a) A graph G with colors representing the partition $\mathcal{P}$. (b) The reduced graph $G_{\mathcal{P}}$ corresponding to the partition in (a).

III. Tree partitioning with minimal impact on transient stability

A. Problem statement

The *tree partitioning* procedure identifies a partition and selects a subset of its cross edges to switch off such that the clusters of the considered partition are connected in a tree-like manner (see Figure 3). More formally, given a power network $G = (V, E)$, we want to identify a partition $\mathcal{P}$ and a subset of lines $\mathcal{E}$ to be switched off such that $\mathcal{P}$ is a tree partition of the post-switching network $G^{\mathcal{E}} = (V, E \setminus \mathcal{E})$. The switching actions should be carefully chosen since, being line disconnections themselves, they impact the flows on the remaining lines and the stability of the network. In the rest of this section we explore how to optimally and safely tree partition a given network by means of line switching actions.



(a) (b)

Fig. 3. (a) A tree-partitioned network $G^{\mathcal{E}}$ with colors representing the partition $\mathcal{P}$ and the switched off lines $\mathcal{E}$ depicted in red. (b) The corresponding reduced graph $(G^{\mathcal{E}})_{\mathcal{P}}$, with the two newly created bridges in black.

B. Tree partitioning design principles: transient stability

Inspired by the design principles behind islanding schemes [3], [6], [9], we want to identify a set of switching actions that do not compromise *transient stability*, which refers to the ability of the power network to maintain frequency synchronization when subjected to a severe disturbance [12].

To ensure transient stability of a tree-partitioned network, we focus on two main aspects. First, the tree-partitioned network must ensure *generator coherency*, i.e., synchronous generators

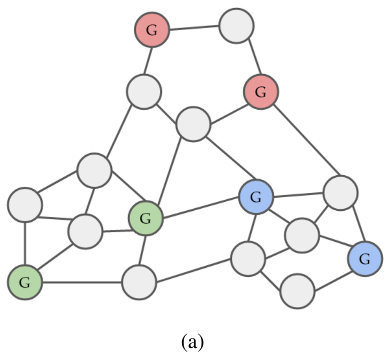


Fig. 4. The two-stage approach performed on a small network instance. (a) The original network with three coherent generator groups. (b) The first stage identifies a partition $\mathcal{P}$ with three clusters, each containing a coherent generator group. (c) The second stage removes a subset of lines $\mathcal{E}$ such that $\mathcal{P}$ is a tree partition of the new network.

that exhibit a similar response after a large disturbance must be grouped together in the same cluster in the post-switching network. Indeed, non-coherent generator groups might lead to generator tripping and eventually the collapse of the network [3]. Secondly, we want to minimize the impact of the switching actions on the remaining lines. In controlled islanding, this is usually quantified using the *power flow disruption*, defined as the total sum of absolute flows on the switched off lines. When tree partitioning a network, it always remains connected and there is no actual power disruption – this is one of its key advantages over islanding. Nonetheless, it is reasonable to use the same quantity to assess the impact on the network due to power flow redistribution.

C. Optimization problem

We now formulate an optimization problem to tree partition a power network with minimal power flow disruption and generator coherency constraints. Consider a power network $G = (V, E)$ with the absolute line power flows $|\boldsymbol{f}|$ as edge weights and let $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k \subset V$ denote the k coherent generator groups. The goal is to identify a partition $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k\}$ and to remove a set of lines $\mathcal{E} \subset E$ with minimal power flow disruption $\sum_{(i,j)\in\mathcal{E}} |f_{ij}|$ such that the following are satisfied:

1) the post-switching network $G^{\mathcal{E}} = (V, E\backslash\mathcal{E})$ is connected,
2) $\mathcal{P}$ is a tree partition of $G^{\mathcal{E}}$, and
3) each cluster in $\mathcal{P}$ contains a set of coherent generators, i.e., $\mathcal{G}_r \subset \mathcal{V}_r, \forall r \in [k]$.

The identification of coherent generator groups is not the key challenge addressed in the present paper. In fact, we follow the common approach used in the islanding literature and consider the slow coherency criterion as described in, e.g., [3], [13], [14].

IV. Two-stage approach

In this section, we revisit an existing two-stage approach introduced by [7]. These two stages, illustrated in Figure 4,

arise naturally from the tree partitioning procedure as presented in Section III-A. In the first stage, a partition $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k\}$ is identified so that each cluster (i) groups a set of coherent generators and (ii) is such that the corresponding induced subgraph is connected. In the second stage, one seeks the subset of cross edges $\mathcal{E} \subset E_C(\mathcal{P})$ with minimal power flow disruption whose removal makes the reduced graph a tree.

In [7], the tree partitioning problem was addressed using the aforementioned two-stage approach. The authors propose to solve the first stage using a constrained version of spectral clustering, which is a fast approximation algorithm to minimize the so-called *expansion constant* of a network [15]. Using the best partition returned by spectral clustering, they then proposes to solve the second stage using a maximum weight spanning tree algorithm on the reduced graph. This algorithm identifies the set of cross edges with the largest total joint absolute power flow and selects them as bridges for the post-switching network. The remaining cross edges, whose removal causes the minimal power flow disruption by construction, correspond to the set $\mathcal{E}$ of lines to be switched off.

Despite the two-stage approach being a natural way to tree partition a power network, there are two reasons why it may lead to sub-optimal results. The first is that the two-stage approach is heuristic in nature, as the power flow disruption highly depends on the quality of the identified partition $\mathcal{P}$. Even though spectral clustering methods generally produce good partitioning solutions in terms of power flow disruption, there is no guarantee that the identified partition is the best with respect to the objective in the considered tree partitioning problem. Second, constrained spectral clustering is known to be NP-hard [16]. Heuristic methods [3], [6], [17] have been introduced to address this issue, but there is no guarantee that the generator coherency constraints are satisfied.

V. MILP formulation

We present an MILP formulation for the tree partitioning problem as introduced in Section III-C. MILP-based approaches have been successfully considered in islanding studies [4], [9], [18] and allow for more flexibility in adding additional constraints or in considering different objective functions when compared to spectral clustering. Besides this, the main advantage of using an MILP-based approach in this context is that we can overcome the heuristic nature of the two-stage approach, and instead compute *truly optimal* tree partitions. The high-level idea of the MILP approach is identical to the two-stage approach, yet its formulation, which we will now outline, allows us to solve both stages jointly.

We say that a line is *inactive* if it is switched off and *active* otherwise. In other words, the inactive lines are those in the subset $\mathcal{E}$, whereas the active lines are those remaining in the tree-partitioned network $G^{\mathcal{E}} = (V, E \setminus \mathcal{E})$.

Let the decision variables $z_{ij} \in \{0,1\}, (i,j) \in E$ indicate whether or not line (i,j) is active. Note that we assume that only cross edges may become inactive, which we will address in more detail later. We introduce the following objective function:

$$\min \sum_{(i,j)\in E} |f_{ij}|(1 - z_{ij}), \tag{1a}$$

where we minimize the total power flow disruption due to the inactive lines. Let the decision variables $x_{ir} \in \{0,1\}, i \in V, r \in [k]$ indicate whether or not bus i belongs to cluster r. Adding the following constraint ensures that coherent generators are grouped together in the same cluster:

$$x_{ir} = 1, \qquad \forall r \in [k], \forall i \in \mathcal{G}_r. \tag{1b}$$

Next, let the decision variables $y_{ijr} \in \{0,1\}, (i,j) \in E, r \in [k]$ indicate whether or not line (i,j) is an internal edge of cluster r, i.e., both endpoints i and j belong to cluster r. We then introduce the partitioning constraints:

$$\sum_{r=1}^{k} x_{ir} = 1, \qquad \forall i \in V, \tag{1c}$$

$$y_{ijr} \leq x_{ir}, \qquad \forall (i,j) \in E, \forall r \in [k], \tag{1d}$$

$$y_{ijr} \leq x_{jr}, \qquad \forall (i,j) \in E, \forall r \in [k], \tag{1e}$$

$$y_{ijr} \geq x_{ir} + x_{jr} - 1, \qquad \forall (i,j) \in E, \forall r \in [k]. \tag{1f}$$

Equation (1c) ensures that each bus belongs to exactly one cluster and (1d) to (1f) represent the linearized version of the expression $y_{ijr} = x_{ir}x_{jr}$, which states that a line (i,j) is an internal edge of cluster r if and only if i and j belong to the same cluster r. Moreover, if a line (i,j) is not an internal edge, i.e., $y_{ijr} = 0$ for all $r \in [k]$, then this implies that (i,j) is a cross edge.

Let the decision variables $w_{ij} \in \{0,1\}, (i,j) \in E$ denote whether or not line (i,j) is an active cross edge. We add the following constraint:

$$\sum_{r=1}^{k} y_{ijr} + w_{ij} = z_{ij}, \qquad \forall (i,j) \in E. \tag{1g}$$

To describe the implications of (1g), we distinguish between whether a line (i,j) is an internal edge or a cross edge. If (i,j) is an internal edge, then the constraint ensures that the line must be active, since the summation in the left-hand side is equal to 1 and thus $z_{ij} = 1$. If instead (i,j) is a cross edge, then it can be either active (i.e., $w_{ij} = 1$) or inactive (i.e., $w_{ij} = 0$), and its value will be correspondingly reflected in the value of z_{ij}. In words, (1g) ensures that internal edges are always active, whereas cross edges may be either active or inactive.

Having defined the partitioning constraints and the switching actions, we now need to ensure that the active cross edges connect the clusters in a tree-like manner. We first restrict the number of active cross edges in the post-switching network:

$$\sum_{(i,j)\in E} w_{ij} = k - 1, \qquad \forall (i,j) \in E. \tag{1h}$$

Next, we add the *single commodity flow* constraints to ensure connectivity between all buses [19]. The idea is to assign a *source* bus that sends a fictitious unit of commodity flow to the remaining $n-1$ *demand* buses, which is possible if and only if the graph is connected. We define the decision variables $q_{ij} \in \mathbb{R}, (i,j) \in E$ as the commodity flow and we assign bus 1 as the source bus. Then, the single commodity flow constraints are given as follows:

$$\sum_{(1,j)\in E} q_{1j} - \sum_{(j,1)\in E} q_{j1} = n - 1, \tag{1i}$$

$$\sum_{(i,j)\in E} q_{ij} - \sum_{(j,i)\in E} q_{ji} = -1, \qquad \forall i \in V \setminus \{1\}, \tag{1j}$$

$$-(n-1)z_{ij} \leq q_{ij} \leq (n-1)z_{ij}, \qquad \forall (i,j) \in E. \tag{1k}$$

Equation (1i) ensures that the net commodity flow, defined as the outgoing minus the incoming commodity flow, of the source bus is exactly $n-1$, i.e., it produces $n-1$ units of commodity flow. Similarly, (1i) ensures that the net commodity flow of the demand buses is -1, meaning that they consume one unit of commodity flow. Finally, (1k) ensures that the inactive lines carry no commodity flow.

To summarize, (1h) to (1k) together ensure that we have (i) $k-1$ active cross edges and (ii) a post-switching network that is connected. Hence, the clusters must be connected in a tree-like manner, where the active cross edges correspond to newly-formed bridges in the post-switching network. This completes our MILP description for the tree partitioning problem.

A. Search space reduction

A potential drawback of using an MILP-based approach is that the running times may grow exponentially large as a function of the instance size. To tackle this issue, we now introduce an ad-hoc *search space reduction* procedure to restrict the solution space, hence decreasing the running time of the proposed MILP method when considering large networks.

The search space reduction procedure was originally introduced by [9] to restrict the solution space in their MILP formulation for controlled islanding. Since we want coherent generators to belong to the same connected cluster, for each pre-identified coherent group we first pre-compute a sub-tree connecting all its generators with the least number of lines. We then pre-assign all the buses of this sub-tree to the same cluster in the MILP. To this end, we solve several instances of the classical *Steiner tree problem*, which can be stated as follows:

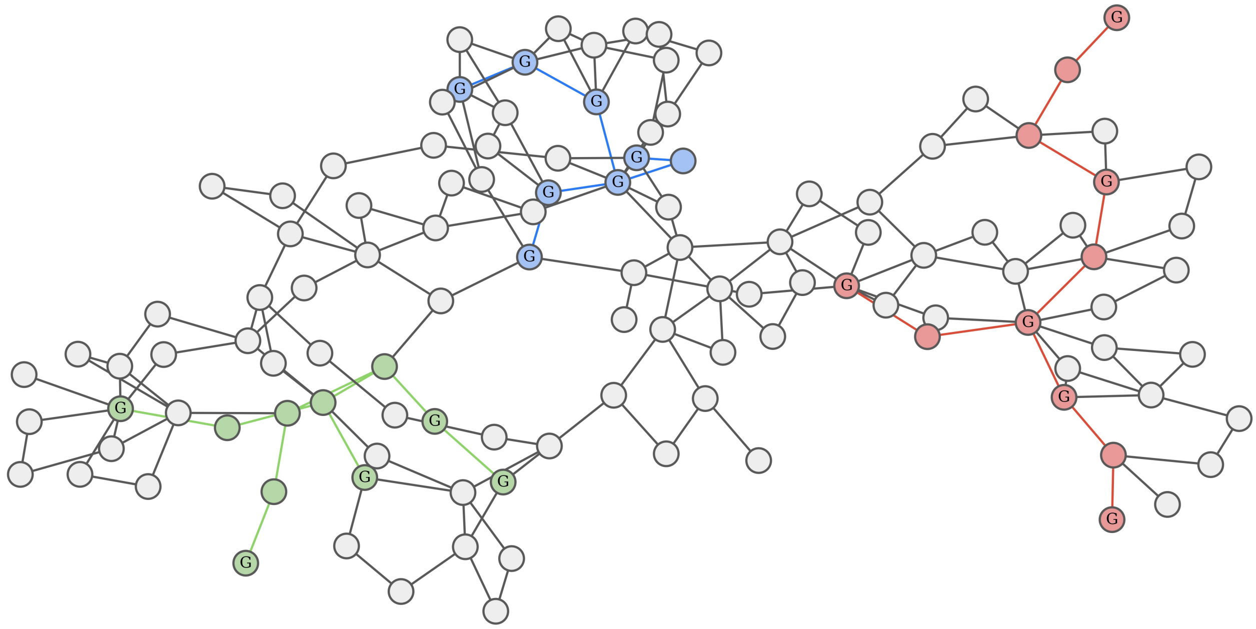


Fig. 5. IEEE-118 network with pre-computed Steiner trees connecting $k = 3$ coherent generator groups of size 5 (green), 7 (blue) and 6 (red). The respective Steiner trees consist of 10, 8 and 11 buses together with 9, 7 and 10 lines.

given an unweighted graph $G = (V, E)$ and a set of terminal vertices R, the goal is to compute a tree $S = (V_S, E_S)$, referred to as *Steiner tree*, with minimum number of edges such that the terminal vertices are spanned by the tree, i.e., $R \subseteq V_S$.

The search space reduction works as follows. We solve the Steiner tree problem for each coherent generator group $\mathcal{G}_r, r \in [k]$, where we let $\mathcal{G}_r$ be the terminal vertex set. This then yields k Steiner trees $S_r, r \in [k]$ that each connect the coherent generators and we extend the earlier defined MILP (1) by adding the following constraints:

$$x_{ir} = 1, \qquad \forall r \in [k], \forall i \in V(S_r), \tag{2a}$$

$$y_{ijr} = 1, \qquad \forall r \in [k], \forall (i,j) \in E(S_r). \tag{2b}$$

In words, (2) fixes the computed Steiner trees as partial solutions to the MILP, which reduces the number of feasible solutions of the MILP. In general, this leads to faster running times.

Figure 5 shows an example of the search space reduction procedure on the IEEE-118 network with $k = 3$ coherent generator groups. The search space reduction procedure identifies an additional 11 buses along with 26 lines, which are then added to the MILP using (2). The solution of the corresponding MILP is shown in Figure 6.

We address several potential issues that may arise from the search space reduction procedure. First, the Steiner tree problem is known to be NP-hard and thus the authors in [9] resort to a heuristic algorithm. However, dedicated Steiner tree algorithms are able to compute optimal solutions within less than one second on instances with over thousand buses [20]. Hence, we use an exact algorithm. Second, solving the Steiner tree problem separately for each generator group may sometimes result in Steiner trees that overlap, i.e., some buses and lines may belong to more than one Steiner tree. This issue is not specifically addressed in [9], but it will lead to infeasibility when solving the corresponding MILP. We correct this by removing the overlapping buses and lines, as well as lines adjacent to the overlapping buses. This correction removes all potential infeasibility issues. Third, it is not guaranteed that the optimal solution of the original MILP formulation (1) can still be obtained after introducing the search space reduction. In the next section, we compare the solutions between the original MILP and the search space reduced MILP.

VI. Results

We now evaluate the performance of the approaches from the previous sections to solve the tree partitioning problem. To this end, we implemented the two-stage approach (from now on denoted as 2-ST) proposed in [7], using constrained spectral clustering [6] and a modified Prim's algorithm to solve the maximum weight spanning tree problem. Moreover, we implemented our MILP-based approach, both with and without search space reduction. We henceforth refer to the first as SSR and to the other one simply as MILP.

In our numerical experiments we use test cases from the `PGLib-OPF` library [21]. From each test case we extract a power network $G = (V, E)$, where the power injections $\boldsymbol{p}$ and flows $\boldsymbol{f}$ are computed by solving a DC-OPF problem. We remark that our results can be easily extended to AC-based power flows. The coherent generator groups are computed based on the slow coherency methodology outlined in [3], where the generator inertia constants H are assumed to be equal to 1 for

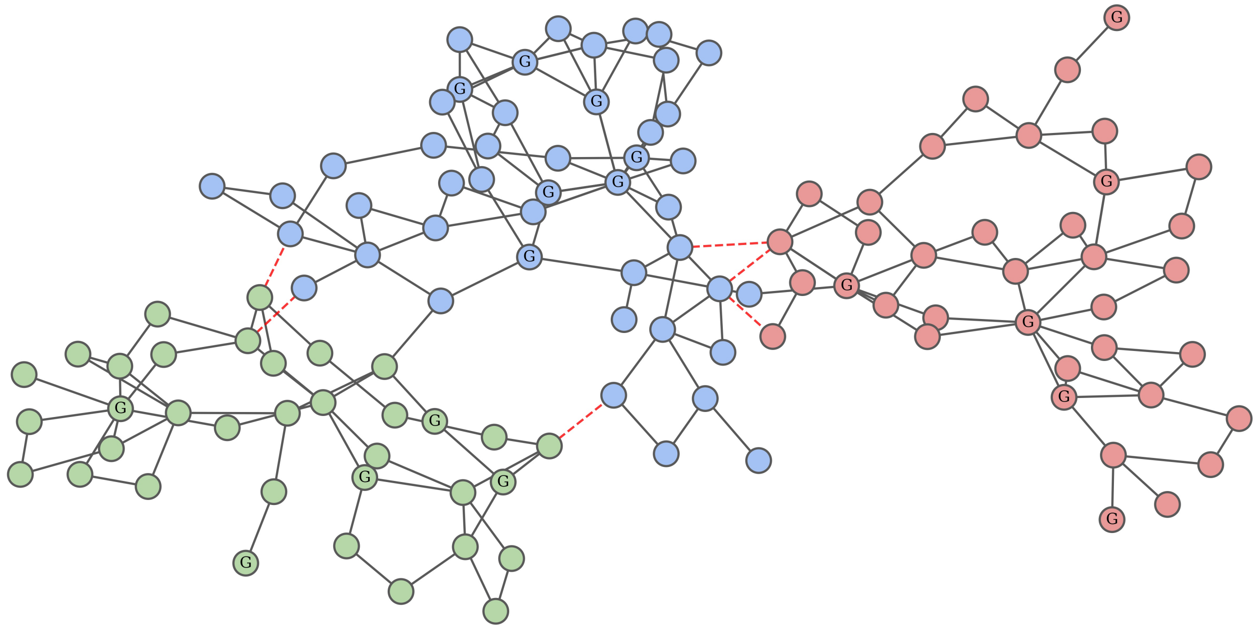


Fig. 6. Tree-partitioned IEEE-118 network obtained after the removal of 6 lines (red dashed), resulting in three clusters of size 35 (green), 46 (blue) and 37 (red). The solution is computed using the MILP algorithm with search space reduction and has a total power flow disruption of 200.27 MW.

all generators. The experiments are performed using an Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 and 16GB RAM. All our algorithms have been implemented in Python and the MILPs are solved using the commercial software Gurobi. More details on the code implementation can be found in [22].

A. 2-ST vs. MILP

In this subsection, we compare the 2-ST algorithm against our MILP-based approach without search space reduction. We remark that the number of test cases is limited due to the challenge of computing feasible clustering solutions in 2-ST, since it is an NP-hard problem as noted earlier in Section IV.

Table I presents the computed objective value and running time of both methods for various test networks and values of k. We also report the percentage decrease in objective value that MILP achieves with respect to 2-ST. In all shown instances, MILP computed the optimal value of the problem within several seconds. Compared to 2-ST, MILP drastically decreased the power flow disruption by at least 40% in more than half of the instances. For small instances and low values of k, the running times of both methods were comparable. However, the running time of MILP was an order of magnitude larger than 2-ST for some of the larger instances (e.g., IEEE-300, $k = 4$). Nevertheless, the results show that the tree partitioning problem can be solved effectively using the MILP-based approach.

TABLE I
PERFORMANCE COMPARISON BETWEEN 2-ST AND MILP.

		Objective value (MW)			Running time (s)	
Case	k	2-ST	MILP	decrease	2-ST	MILP
IEEE-73	2	87.43	87.43	-0.00%	0.02	0.06
	3	415.31	162.37	-60.90%	0.01	0.01
	4	1119.70	559.28	-50.05%	0.01	0.03
	5	1247.50	754.07	-39.55%	0.01	0.19
IEEE-118	2	283.07	283.07	-0.00%	0.04	0.01
	3	200.54	200.27	-0.13%	0.04	0.01
	4	655.01	417.79	-36.22%	0.03	0.01
	5	1026.52	723.82	-29.49%	0.03	0.02
GOC-179	2	2430.13	1850.06	-23.87%	0.07	0.06
	3	1596.20	181.47	-88.63%	0.07	0.02
	4	1887.20	472.47	-74.96%	0.06	0.02
	5	1914.90	500.17	-73.88%	0.06	0.09
PSERC-240	2	5927.07	2426.04	-59.07%	0.17	0.02
	3	3575.03	2451.76	-31.42%	0.13	0.03
	4	5322.38	2714.01	-49.01%	0.14	0.50
	5	5133.82	3787.17	-26.23%	0.13	3.83
IEEE-300	2	2445.36	240.77	-90.15%	0.10	0.02
	3	2179.62	1029.13	-52.78%	0.22	0.07
	4	1857.64	1325.02	-28.67%	0.15	2.59
	5	1357.50	1024.59	-24.52%	0.27	0.67
GOC-500	2	2031.52	516.97	-74.55%	0.57	0.03
	3	1836.22	981.33	-46.56%	0.72	0.49
	4	1077.98	601.56	-44.20%	0.48	0.48
	5	1611.07	1092.64	-32.18%	0.52	2.05

B. MILP vs. SSR

The previous experiment shows that MILP may take up to several seconds to optimally tree partition a network. Due to the fast nature of cascading failures, it is important that emergency measures, and hence tree partitioning procedures, can compute solutions in sub-second time [6]. To make MILP run faster, we use the search space reduction procedure as described in Section V-A and compare SSR against MILP to analyze the trade-off between objective value and speed. We present the results here for $k = 5$ generator groups to demonstrate the scalability of both approaches. Table II compares the MILP and SSR algorithms in terms of objective value and running times. The running time of SSR includes the time from solving the Steiner tree problems, although we note that this takes at most 0.2 seconds for each network. We also report the percentage increase in objective value of SSR compared to MILP, which is always nonnegative since MILP always computes the optimal solutions.

TABLE II
PERFORMANCE COMPARISON BETWEEN MILP AND SSR WITH $k = 5$ COHERENT GENERATOR GROUPS.

Case	Objective value (MW)			Running time (s)	
	MILP	SSR	increase	MILP	SSR
IEEE-30	15.88	15.88	0.00%	0.10	0.01
IEEE-57	324.65	324.65	0.00%	0.10	0.04
IEEE-73	754.07	761.56	0.99%	0.19	0.02
PEGASE-89	1769.46	1814.47	2.54%	29.18	2.07
IEEE-118	723.82	723.82	0.00%	0.02	0.02
GOC-179	500.17	500.17	0.00%	0.09	0.06
ACTIV-200	57.08	57.08	0.00%	0.34	0.09
PSERC-240	3787.17	3787.17	0.00%	3.83	0.30
IEEE-300	1024.59	1200.09	17.13%	0.67	0.07
GOC-500	1092.64	1092.64	0.00%	2.05	0.72
GOC-793	1211.56	1373.82	13.39%	238.30	4.40
GOC-2000	1174.51	1174.51	0.00%	8.24	6.43
SOP-2737	1117.52	1219.37	9.11%	151.75	17.29

We see that the solutions from SSR achieved the same objective value as MILP in 8 out of 13 cases. In the other cases, we observe an increase in the objective value of at most 17% in IEEE-300. Most importantly, SSR decreased the running time by one order of magnitude in most cases and attained sub-second running times for most instances below 500 buses. Thus, SSR greatly decreases the running time at the mere expense of a small increase in the objective value.

VII. CONCLUSION

In this paper, we formulated an optimization problem to compute tree partitions with minimal power flow disruption and generator coherency constraints. We revisited a heuristic two-stage approach based on spectral clustering and introduced a novel single-stage MILP-based approach to improve it. Our numerical results demonstrated that the MILP-based approach computes optimal solutions with acceptable running times, drastically outperforming the two-stage heuristic in terms of objective value. After introducing an ad-hoc search space reduction procedure to the MILP-based approach, we also showed that its running times can be significantly improved. In particular, we showed that near-optimal solutions can be computed in sub-second time for instances below 500 buses, demonstrating its potential to be used in emergency situations. In future work we envision further enriching the tree partition problem by including a wider range of stability conditions, e.g., voltage stability and frequency stability. On the numerical side, the usage of decomposition methods could further improve the computational efficiency of the MILP-based approach for larger network instances.

REFERENCES

[1] N. Senroy and G. Heydt, "A Conceptual Framework for the Controlled Islanding of Interconnected Power Systems," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 1005–1006, May 2006.

[2] Q. Zhao, K. Sun, D.-Z. Zheng, J. Ma, and Q. Lu, "A study of system splitting strategies for island operation of power system: A two-phase method based on OBDDs," *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1556–1565, Nov. 2003.

[3] L. Ding, F. M. Gonzalez-Longatt, P. Wall, and V. Terzija, "Two-Step Spectral Clustering Controlled Islanding Algorithm," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 75–84, Feb. 2013.

[4] P. Trodden, W. Bukhsh, A. Grothey, and K. McKinnon, "MILP formulation for controlled islanding of power networks," *International Journal of Electrical Power & Energy Systems*, vol. 45, no. 1, pp. 501–508, 2013.

[5] R. J. Sanchez-Garcia, M. Fennelly, S. Norris, N. Wright, G. Niblo, J. Brodzki, and J. W. Bialek, "Hierarchical Spectral Clustering of Power Grids," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2229–2237, Sep. 2014.

[6] J. Quirós-Tortós, R. Sánchez-García, J. Brodzki, J. Bialek, and V. Terzija, "Constrained spectral clustering-based methodology for intentional controlled islanding of large-scale power systems," *IET Generation, Transmission & Distribution*, vol. 9, no. 1, pp. 31–42, Jan. 2015.

[7] J. W. Bialek and V. Vahidinasab, "Tree-Partitioning as an Emergency Measure to Contain Cascading Line Failures," *IEEE Transactions on Power Systems*, 2021.

[8] A. Zocca, C. Liang, L. Guo, S. H. Low, and A. Wierman, "A Spectral Representation of Power Systems with Applications to Adaptive Grid Partitioning and Cascading Failure Localization," *arXiv:2105.05234*, 2021.

[9] P. Demetriou, A. Kyriacou, E. Kyriakides, and C. Panayiotou, "Applying exact MILP formulation for controlled islanding of power systems," in *2016 51st International Universities Power Engineering Conference (UPEC)*, Sep. 2016, pp. 1–6.

[10] L. Guo, C. Liang, A. Zocca, S. H. Low, and A. Wierman, "Failure localization in power systems via tree partitions," in *2018 IEEE Conference on Decision and Control*, 2018, pp. 6832–6839.

[11] ——, "Line failure localization of power networks part i: Non-cut outages," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 4140–4151, 2021.

[12] J. Machowski, Z. Lubosny, J. W. Bialek, and J. R. Bumby, *Power System Dynamics: Stability and Control*. John Wiley & Sons, Jun. 2020.

[13] H. You, V. Vittal, and X. Wang, "Slow Coherency-Based Islanding," *IEEE Transactions on Power Systems*, vol. 19, no. 1, pp. 483–491, 2004.

[14] I. Tyuryukanov, A. C. K. Kaltsikis, M. Popov, M. A. M. M. van der Meijden, and V. Terzija, "Generator grouping cutset determination based on tree construction and constrained spectral clustering," *The Journal of Engineering*, vol. 2018, no. 15, pp. 1309–1314, 2018.

[15] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[16] X. Wang, B. Qian, and I. Davidson, "On Constrained Spectral Clustering and Its Applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, Jan. 2014.

[17] A. Esmaeilian and M. Kezunovic, "Prevention of Power Grid Blackouts Using Intentional Islanding Scheme," *IEEE Transactions on Industry Applications*, vol. 53, no. 1, pp. 622–629, Jan. 2017.

[18] A. Kyriacou, P. Demetriou, C. Panayiotou, and E. Kyriakides, "Controlled Islanding Solution for Large-Scale Power Systems," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1591–1602, Mar. 2018.

[19] B. Gavish and S. C. Graves, "The Travelling Salesman Problem and Related Problems," Massachusetts Institute of Technology, Operations Research Center, Working Paper, Jul. 1978.

[20] M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl, "A Dual Ascent-Based Branch-and-Bound Framework for the Prize-Collecting Steiner Tree and Related Problems," *INFORMS Journal on Computing*, vol. 30, no. 2, pp. 402–420, May 2018.

[21] S. Babaeinejadsarookolaee and et al., "The Power Grid Library for Benchmarking AC Optimal Power Flow Algorithms," *arXiv:1908.02788*, 2021.

[22] L. Lan, "Github repository: bridge-blocks." [Online]. Available: https://github.com/leonlan/bridge-blocks