Refining bridge-block decompositions through two-stage and recursive tree partitioning

Leon Lan, Alessandro Zocca

Department of Mathematics, Vrije Universiteit Amsterdam, NL

{l.lan, a.zocca}@vu.nl

Abstract—In transmission networks power flows and network topology are deeply intertwined due to power flow physics. Recent literature shows that specific network substructures named *bridge-blocks* prevent line failures from propagating globally. A two-stage and recursive *tree partitioning* approach have been proposed to create more bridge-blocks in transmission networks, improving their robustness against cascading line failures. In this paper we consider the problem of refining the bridge-block decomposition of a given power network with minimal impact on the maximum congestion. We propose two new solution methods, depending on the preferred power flow model. More specifically, (i) we introduce a novel MILP-based approach that uses the DC approximation to solve more efficiently the second-stage optimization problem of the two-stage approach and (ii) we show how the existing recursive approach can be extended to work with AC power flows, drastically improving the running times when compared to the pre-existing AC-based two-stage method.

Index Terms—power system robustness, bridge-block decomposition, line congestion, failure localization, MILP, tree partitioning

I. INTRODUCTION

Historical blackouts have shown that transmission line failures play an important role in the initiation of cascading failures [1]. The complex network structure of power grids in combination with the underlying power flow physics gives rise to complicated cascading failure patterns, which often exhibit non-local propagation of line failures. A recent paper [2] shows that specific graph structures, called *bridge-blocks*, ensure that line failures propagate only locally. In particular, it is shown that line failures cannot propagate across lines that act as bridges in the network. These failure localization results have been obtained through studying analytical properties of the line outage distribution factor using the DC approximation, while [3], [4] report similar results when using AC power flows.

Despite the potential that bridges may offer in preventing non-local failure propagation, most power grid networks have not been designed with this principle in mind. It is illustrated in [2] that most test power networks have a *trivial bridge-block decomposition*, i.e., they have one large bridge-block comprising a very large fraction of the network, thus potentially allowing line failures to propagate through the entire network (see Figure 1).

Aiming to improve the bridge-block decomposition of a power grid and, hence, its robustness against line failures,
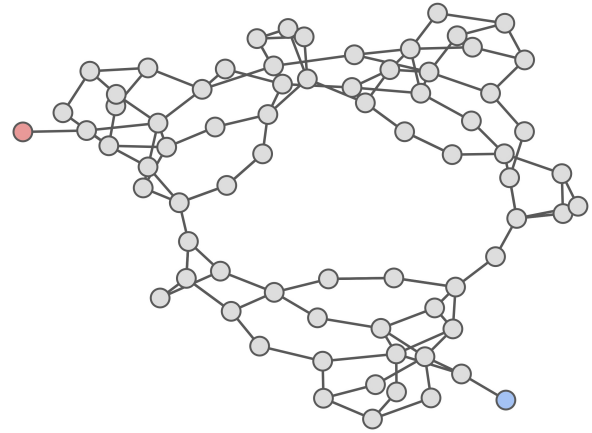
Fig. 1. The bridge-block decomposition of the IEEE-73 network, which has one large bridge-block (in gray) of size 71 and two trivial bridge-blocks of size 1 (in other colors). A line failure in the large gray bridge-block could potentially affect any other line that belongs to it.

[2] proposes to temporarily switch off a set of carefully selected lines, in an adaptive fashion with respect to the current generation and demand patterns. Line switching is a consolidated electricity grid management paradigm that received a lot of attention in the literature. The most well-known applications are *Optimal Transmission Switching* [5], [6], where switching actions aim at maximizing economic efficiency of generation dispatch, and *Controlled Islanding* [7], [8], where lines are switched off to split the network into disconnected islands as a last-resort emergency measure to stop cascading failures.

A heuristic two-stage procedure called *tree partitioning* has been proposed by [2] and [3] (using DC and AC power flow models, respectively) to judiciously switch off lines to improve the bridge-block decomposition of a given power network, while minimizing the impact on line congestion. This two-stage heuristic first identifies suitable clusters and then, by means of switching actions, makes sure that they are connected in a tree-like manner, hence becoming bridge-blocks. The computational bottleneck of this approach resides in the second stage, since it requires evaluating an exponential number of spanning trees to find the best way to connect the identified clusters. Since the brute-force method does not scale well with large networks, [2] also proposes an alternative recursive approach for tree partitioning, which refines the network bridge-block decomposition at each iteration.

There are two major contributions in this paper. First, we introduce an MILP-based algorithm that solves the second-stage problem *exactly* under the DC power flow model. Our algorithm overcomes the computational bottleneck formed by the existing brute-force algorithms, while producing better results than the recursive algorithm in the considered test cases. Second, aiming to extend tree partitioning to the more realistic AC power flow setting, we modify the recursive approach proposed in [2] to work with AC power flows. For several test cases, our AC variant of the recursive approach yields solutions qualitatively comparable to those returned by the two-stage brute-force approach under AC power flows, but runs drastically faster and has the potential to scale well for large networks.

The paper is organized as follows. After some preliminaries in Section II, Section III formally introduces the optimization problem for bridge block decomposition refinement and revisits two existing approaches to tree partitioning. In Section IV, we outline our MILP-based algorithm for the second stage problem using DC power flows and in Section V we propose an AC variant of the recursive algorithm. Section VI presents the numerical results and we conclude the paper with Section VII.

II. PRELIMINARIES

A. Power network model

We model a 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. Each bus $i \in V$ has a *net power injection* p_i, where $p_i > 0$ is interpreted as *injected* power and $p_i \leq 0$ as *consumed* power at bus i. Each line $\ell = (i, j) \in E$ has a capacity $c_\ell = c_{ij} > 0$, denoting its rating, i.e., the maximum power that the line can carry.

Throughout this paper, we will use both DC and AC power flow models. For a full description of AC power flow models, we refer the reader to [1]. We describe here a lossless DC power flow model in which generation always matches demand, i.e., $\sum_{i=1}^{n} p_i = 0$. We refer to any such vector $\boldsymbol{p}$ of power injections as *balanced*. Let $\theta_i \in \mathbb{R}$ denote the *phase angle* of bus i. For each line $\ell = (i, j)$, let $f_\ell = f_{ij} \in \mathbb{R}$ denote the active power flow and let $b_\ell = b_{ij} > 0$ denote the *line susceptance*. Given a vector of power injections $\boldsymbol{p} \in \mathbb{R}^n$, the corresponding line flows $\boldsymbol{f} \in \mathbb{R}^m$ and phase angles $\boldsymbol{\theta} \in \mathbb{R}^n$ are obtained by solving the *DC power flow equations*:

$$p_i = \sum_{j:(i,j)\in E} f_{ij} - \sum_{j:(j,i)\in E} f_{ji}, \qquad \forall i \in V, \quad (1a)$$

$$f_{ij} = b_{ij}(\theta_i - \theta_j), \qquad \forall (i,j) \in E. \quad (1b)$$

Equation (1a) guarantees flow conservation and (1b) captures the flow dependency on susceptance and angle differences. The DC power flow equations (1) admit a unique power flow solution $\boldsymbol{f}$ for each balanced injection vector $\boldsymbol{p}$. The solution for the phase angles $\boldsymbol{\theta}$ is unique up to an arbitrary reference angle: without loss of generality, we select bus n as the *reference bus* with phase angle $\theta_n = 0$.

B. Tree partitions and bridge-block decomposition

We now briefly review some graph-theoretical terminology. A *k-partition* of the network G is a collection $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k\}$ of non-empty, disjoint vertex sets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ called *clusters* such that $\bigcup_{i=1}^{k} \mathcal{V}_k = V$. We denote by $[k] = \{1, 2, \ldots, k\}$ the set of integers from 1 to k, which will be used to denote the clusters. Given a partition $\mathcal{P}$, a line (i, j) is called an *internal edge* if both i and j belong to the same cluster and a *cross edge* otherwise. We denote by $E_C(\mathcal{P})$ the set of cross edges determined by partition $\mathcal{P}$. 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. Note that it is possible for the reduced graph to have multiple edges between two vertices, and thus to be a multigraph.

We say that a partition $\mathcal{P}$ is a *tree partition* of G if the reduced graph $G_\mathcal{P}$ is a tree (see Figure 2a). A *bridge* is a cut-edge for the graph i.e., an edge whose removal would disconnect it. The *bridge-block decomposition* of a graph is its partition in disconnected subgraphs that is obtained after removing all bridges in the graph (see Figure 2b). Each cluster in the bridge-block decomposition is called a *bridge-block*. It is easy to show that the bridge-block decomposition is a tree partition, and, in particular, is the *irreducible* one, i.e., the maximal by inclusion [9]. Consequently, there could exist tree partitions that are not the bridge-block decomposition, as shown by comparing Figure 2a and 2b. Given any tree partition $\mathcal{P}$, we henceforth say that the bridge-block decomposition is always *as fine* as $\mathcal{P}$, meaning that every bridge-block is contained in some cluster of the tree partition.
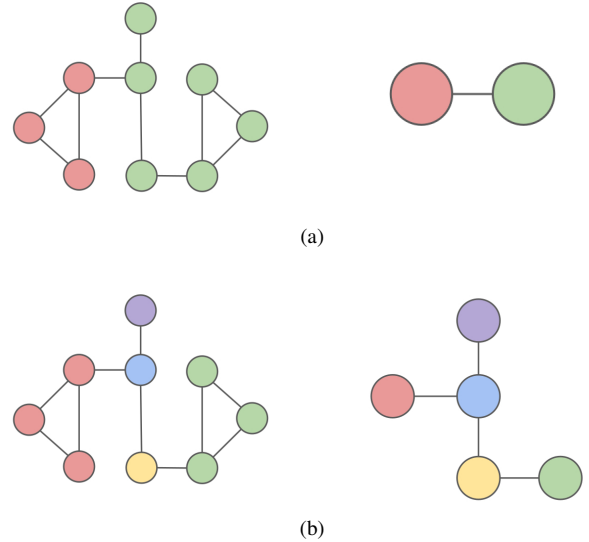

(a)

(b)

Fig. 2. (a) A tree partition of a graph G and the corresponding reduced graph. (b) The bridge-block decomposition of G and the corresponding reduced graph. Note that, being the bridge-block decomposition, it is at least as fine as any tree partition of the same network and in fact is finer than (a).
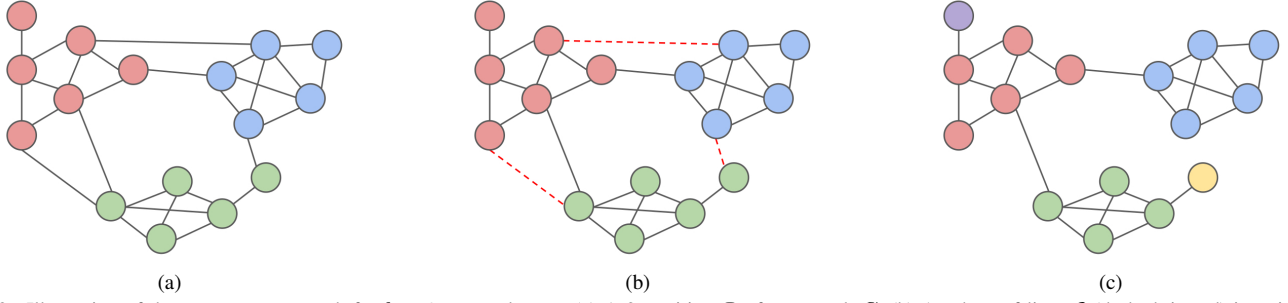
Fig. 3. Illustration of the two-stage approach for $k = 3$ target clusters. (a) A 3-partition $\mathcal{P}$ of a network G. (b) A subset of lines $\mathcal{E}$ (dashed, in red) is switched off, turning $\mathcal{P}$ into a tree partition of $G^{\mathcal{E}}$. (c) The resulting bridge-block decomposition of $G^{\mathcal{E}}$, which is slightly finer than the identified partition $\mathcal{P}$.

III. BRIDGE-BLOCK DECOMPOSITION REFINEMENT

A. Motivation

After the failure or disconnection of a transmission line, its original power flow gets *globally* redistributed as prescribed by power flow physics on the remaining lines, some of which can overload and also get disconnected. We refer to this phenomenon as *failure propagation*. It was shown in [2], [3], [9], [10] that a line failure does not propagate across bridges, but instead only impacts the flows on lines that belong to the same bridge-block. Most power networks, however, have a very meshed structure and trivial bridge-block decompositions [2], making them very prone to non-local line failure propagation, see, e.g., Figure 1.

The number of (non-trivial) bridge-blocks can be increased by switching off lines in a procedure named *bridge-block decomposition refinement* [2]. Nevertheless, as there are an exponential number of lines to consider, there is no obvious way to select which lines to remove in general. [2] proposes a bottom-up approach, named *tree partitioning*: a target partition is first identified using clustering methods, and then these clusters are ensured to be connected in a tree-like manner, transforming the identified partition into a tree partition. More formally, given a power network $G = (V, E)$, the goal of tree partitioning is 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 $G^{\mathcal{E}} = (V, E \setminus \mathcal{E})$.

This tree partitioning procedure guarantees that the post-switching network has a bridge-block decomposition that is at least as fine as the tree partition $\mathcal{P}$.

B. Line congestion

Let g_ℓ denote the *congestion level* on line ℓ, whose specific formula depends on the used power flow model. Under the DC approximation we define the congestion level as the non-negative ratio $g_\ell := |f_\ell| / c_\ell$. Under the AC power flow model the calculation of the congestion is more involved and the apparent power should be considered [3]. In either case, we say that a line ℓ is congested if $g_\ell > 1$.

It is desirable that the switching actions do not cause any of the remaining lines to become congested as a result of flow redistribution, but this is hard to ensure up front due to the complexity of the power network and the underlying power

flow physics. We thus select the set $\mathcal{E}$ of lines to be switched off that minimizes the *maximum congestion*, defined as

$$\gamma(\mathcal{E}) := \max_{\ell \in E \setminus \mathcal{E}} \widetilde{g}_\ell, \qquad (2)$$

where $\widetilde{g}_\ell$ is the post-switching congestion level on line ℓ, calculated using the power flow equations on $G^{\mathcal{E}}$ assuming that the power injections $\boldsymbol{p}$ are unchanged. Note that keeping the maximum congestion below 1 directly implies that the post-switching network has no congested lines. Furthermore, line flows may slightly exceed the capacities as long as subsequent remedial actions are undertaken to alleviate the congestion [3].

C. Problem statement

We now formulate an optimization problem to tree partition a network with minimal impact on the maximum congestion based on [2] and [3]. Given a power network $G = (V, E)$ with balanced power injections $\boldsymbol{p}$ and a positive integer $k \geq 2$, the goal is to identify a k-partition $\mathcal{P}$ and a set of lines $\mathcal{E} \subset E$ to be switched off such that we minimize the maximum congestion $\gamma(\mathcal{E})$ and satisfy the following properties:

- the post-switching network $G^{\mathcal{E}} = (V, E \setminus \mathcal{E})$ is connected;
- $\mathcal{P}$ is a tree partition of $G^{\mathcal{E}}$.

Determining the optimal number k of clusters is outside the scope of this paper, and we henceforth assume that k is a given input parameter. Note further that the problem formulation does not specify any minimum size for the identified clusters in $\mathcal{P}$. However, in view of the considered spectral methods that intrinsically use normalization, obtaining a partition with trivial clusters is a rather rare occurrence.

D. Two-stage approach

We describe the two-stage heuristic approach proposed by [2] to tree partition a network with minimal impact on the congestion. These two stages arise naturally because identifying $\mathcal{P}$ and $\mathcal{E}$ simultaneously is computationally intractable for the optimization problem at hand. The two-stage approach decouples them: the first stage computes the partition $\mathcal{P}$ and the second stage returns the corresponding best subset $\mathcal{E}$.

1) First stage – Optimal Bridge-blocks Identification: For a given k, the first stage aims to find a sensible partition of the power network into k clusters so that the identified partition can then be transformed into a tree partition in the second stage. A "good" partition must have few cross edges between clusters
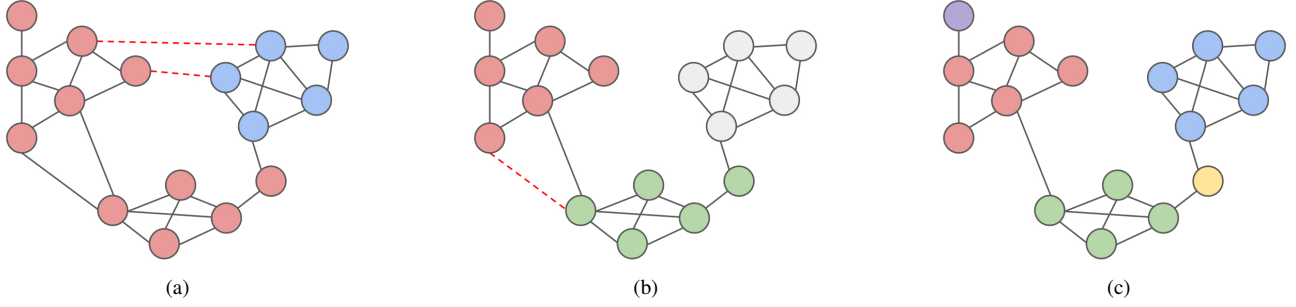
Fig. 4. Illustration of the recursive approach for $k = 3$ target clusters. (a) The first iteration refines the network into a large red bridge-block and a smaller blue bridge-block. (b) The second iteration refines the largest bridge-block from the previous iteration, i.e., the red one in (a). (c) The resulting post-switching network after $k − 1 = 2$ iterations and its bridge-block decomposition. Note that the latter is identical to the post-switching bridge-block decomposition obtained using the two-stage approach (cf. Figure 3), yet the post-switching network and hence the congestion levels are slightly different.

and with very modest power flows. Indeed, switching off too many lines or lines with large power flows often leads to large power flow redistribution and results in severe congestion in the resulting network. We thus seek a partition that has (i) very few (and/or low weight) cross edges between the clusters and (ii) clusters that are similar in size and balanced in terms of total net power. In [2], the authors formulate this as an optimization problem, named the *Optimal Bridge-blocks Identification (OBI)* problem, using a power-flow-weighted version of the *network modularity* problem [11] restricted to k-partitions.

The modularity maximization problem is NP-hard [12] and thus so is the OBI problem, hence two strategies are considered in [2] to obtain good approximate solutions. The first one is spectral clustering, using either the normalized Laplacian matrix L_N or the normalized modularity matrix B_N of the network with absolute power flows as edge weights. Both these variants approximate the solution of the same problem: [13] shows that the optimal solution to the normalized cut problem for a fixed number of target clusters k (i.e., spectral clustering using L_N) is identical to that of the normalized modularity problem (i.e., spectral clustering using B_N). However, due to differences in their eigensystems, both algorithms often yield slightly different results and thus we consider them both separately as *Spectral L_N* and *Spectral B_N* for our numerical results. The second strategy employs the *Fastgreedy* algorithm [14], which is a fast heuristic algorithm that creates clusters in a hierarchical agglomerate way.

2) Second stage – Optimal Bridge Selection: In [2], the second stage is formulated as an optimization problem, named *Optimal Bridge Selection (OBS)*, where the identified partition from the first stage is turned into a tree partition on the post-switching network. Consider the power network $G = (V, E)$ and the k-partition $\mathcal{P}$ obtained by solving the OBI problem. The goal of the OBS problem is to remove a subset of cross edges $\mathcal{E} \subset E_C(\mathcal{P})$ such that we minimize the maximum congestion $\gamma(\mathcal{E})$ on the post-switching network $G^{\mathcal{E}} = (V, E \setminus \mathcal{E})$, where we assume that the power injections $\boldsymbol{p}$ are unchanged, and such that $\mathcal{P}$ is a tree partition of $G^{\mathcal{E}}$.

An alternative formulation of the OBS problem, which will be leveraged for the MILP-based algorithm, can be given using an explicit definition of the reduced graph. We define the reduced graph of G corresponding to $\mathcal{P}$ as $G_{\mathcal{P}} = ([k], E_C(\mathcal{P}))$, i.e., as the graph whose vertices are the clusters indexed from 1 to k and whose edges are the cross edges between them. Solving the OBS problem then corresponds to finding a spanning tree T on the reduced graph $G_{\mathcal{P}}$, such that the removal of lines $\mathcal{E} = E_C(\mathcal{P}) \setminus T$ minimizes the maximum congestion $\gamma(\mathcal{E})$ on the post-switching network $G^{\mathcal{E}}$.

The OBS problem is particularly difficult since, for any given subset $\mathcal{E}$, we need to recalculate the power flows on the post-switching graph $G^{\mathcal{E}}$ in order to obtain the the maximum congestion $\gamma(\mathcal{E})$. In previous studies [2], [3], the OBS problem has been solved using brute-force algorithms that enumerate all spanning trees. However, since their number is exponential in the number of clusters k, any brute force approach is intractable for large instances.

E. Recursive approach

The hardness of the OBS problem limits the applicability of the two-stage approach to larger network instances. However, if the selected partition $\mathcal{P}$ consists of only $k = 2$ clusters, the number of spanning trees in the reduced graph $G_{\mathcal{P}}$ is exactly equal to the number of cross edges. Therefore, the OBS problem can be solved much faster in this case than when considering $k > 2$ clusters at once. This key idea is at the core of the *recursive approach* to tree partitioning, introduced by [2] precisely to overcome the hardness of the OBS problem.

In the recursive approach, the largest bridge-block in the network is iteratively refined into two smaller bridge-blocks until the desired number of clusters is obtained. In other words, at every iteration of the recursive algorithm, one first solves the OBI problem restricted to 2-partitions and then solves the OBS problem given the resulting bipartition. Hence, given the number k, the recursive algorithm solves the OBI and OBS problem for a total of $k − 1$ times. Figure 4 demonstrates how the recursive approach works for a small network.

The main advantage of the recursive approach is that is much faster than the two-stage approach, having time complexity linear in k. The OBS problem can now be solved using a brute-force algorithm that considers a linear number of spanning trees on the reduced graph and computes the corresponding power flows and congestion levels on the post-switching network. Consequently, the running time of the recursive approach

mainly depends on the speed of the clustering algorithm and the number of clusters k to be obtained.

Even when starting with the same network, the solutions from the two-stage and recursive approaches could differ from each other as shown by comparing Figure 3 and 4, and hence lead to different congestion levels. Later in Section VI we will compare the performance between these two approaches (both using DC and AC power flows).

IV. MILP FORMULATION FOR THE OBS PROBLEM USING DC POWER FLOWS

In this section, we present our first main contribution where we show how to solve the OBS problem using an exact MILP-based algorithm that uses the DC power flow model. The MILP-based algorithm overcomes the bottleneck imposed by the earlier proposed brute-force algorithms, consequently allowing the two-stage approach to be solved more efficiently.

Assuming the partition $\mathcal{P}$ is given, we start by describing in detail how the OBS problem can be cast into an MILP. We say that a line is *inactive* if it is switched off and *active* otherwise. Let the decision variable $\gamma \in \mathbb{R}$ denote the maximum congestion and let the decision variables $f_{ij} \in \mathbb{R}, (i,j) \in E$ denote the active power flow on the lines. The objective is to minimize the maximum congestion in the network:

$$\min \quad \gamma, \tag{3a}$$

which is subject to

$$\gamma \geq |f_{ij}|/c_{ij}, \qquad \forall(i,j) \in E. \tag{3b}$$

First, we introduce constraints to obtain a spanning tree on the reduced graph. With slight abuse of notation, we henceforth identify a cross edge $\ell \in E_C(\mathcal{P})$ with indices (i,j) when we refer to the line in the original network G and (u,v) when we refer to the reduced graph $G_{\mathcal{P}}$, where i, j are bus indices and u, v are cluster indices. Recall that k is the number of clusters in $\mathcal{P}$ and thus also the number of vertices in the reduced graph. Let the binary decision variables $y_{uv} \in \{0, 1\}, (u,v) \in E_C(\mathcal{P})$ indicate whether or not a cross edge is active. We introduce a cardinality constraint on the number of cross edges:

$$\sum_{(u,v) \in E_C(\mathcal{P})} y_{uv} = k - 1. \tag{3c}$$

Moreover, to ensure that the post-switching reduced graph is connected, we add the *single commodity flow* constraints [15]. The main idea is to assign a source vertex that sends a fictitious unit flow to all other vertices, which is possible if and only if the graph is connected. Let the decision variables $q_{uv} \in \mathbb{R}, (u,v) \in E_C(\mathcal{P})$ denote the commodity flow on the cross edges and we assign vertex 1 as the source vertex. Then, the single commodity flow constraints are expressed as follows:

$$\sum_{(1,v) \in E_C(\mathcal{P})} q_{1v} - \sum_{(v,1) \in E_C(\mathcal{P})} q_{v1} = k - 1, \tag{3d}$$

$$\sum_{(u,v) \in E_C(\mathcal{P})} q_{uv} - \sum_{(v,u) \in E_C(\mathcal{P})} q_{vu} = -1, \quad \forall u \in [k] \setminus \{1\}, \tag{3e}$$

$$-(k-1)y_{uv} \leq q_{uv} \leq (k-1)y_{uv}, \quad \forall (u,v) \in E_C(\mathcal{P}). \tag{3f}$$

Equation (3d) ensures that the net commodity flow, defined as the outgoing minus the incoming commodity flow, of the source vertex is exactly $k - 1$, i.e., it produces $k - 1$ units of commodity flow. Similarly, (3e) ensures that the net commodity flow of the demand vertices is -1, meaning that they consume one unit of commodity flow. Equation (3f) ensures that the inactive lines carry no commodity flow. Hence, since (3c) to (3f) ensure that there are $k - 1$ cross edges and the reduced graph is connected, this implies that we obtain a spanning tree on the reduced graph.

Next, we model the impact of the switching actions in terms of congestion on the post-switching network. Let the decision variables $\theta_i \in \mathbb{R}, i \in V$ denote the phase angle of each bus. We first consider the cross edges:

$$f_{ij} \leq b_{ij}(\theta_i - \theta_j) + M_{uv}(1 - y_{uv}), \quad \forall \ell \in E_C(\mathcal{P}), \tag{3g}$$

$$f_{ij} \geq b_{ij}(\theta_i - \theta_j) - M_{uv}(1 - y_{uv}), \quad \forall \ell \in E_C(\mathcal{P}), \tag{3h}$$

$$f_{ij} \leq M_{uv}y_{uv}, \quad \forall \ell \in E_C(\mathcal{P}), \tag{3i}$$

$$f_{ij} \geq -M_{uv}y_{uv}, \quad \forall \ell \in E_C(\mathcal{P}). \tag{3j}$$

Equations (3g) and (3h) ensure that the DC power flow equations hold on the active cross edges, whereas (3i) and (3j) switch off the DC power flow equations for all inactive cross edges. We set the big-M value M_{uv} at four times the capacity of the corresponding line. Moreover, we add the following constraints:

$$f_{ij} = b_{ij}(\theta_i - \theta_j), \quad \forall (i,j) \in E \setminus E_C(\mathcal{P}), \tag{3k}$$

$$\sum_{(i,j) \in E} f_{ij} - \sum_{(j,i) \in E} f_{ji} = p_i, \quad \forall i \in V, \tag{3l}$$

$$\theta_n = 0. \tag{3m}$$

Equation (3k) models the DC power flows equations on the internal edges, which are all active by assumption, and (3l) and (3m) ensure flow conservation and assign n as the reference bus.

In summary, the MILP formulation (3) contains (i) a spanning tree formulation on the reduced graph and (ii) DC power flow equations to compute the congestion on the post-switching network. Assuming the partition is obtained solving the OBI problem (cf. Section III-D), the number of cross edges is relatively small, meaning that modeling the spanning tree formulation requires a relatively small number of constraints and decision variables. Hence, the size of the MILP depends mostly on the network instance size $|V|$ and $|E|$. In Section VI-B, we compare the performance between the newly improved two-stage approach and recursive approach assuming DC power flows.

V. AC MODIFICATION OF THE RECURSIVE APPROACH

The objective function of the OBS problem requires to calculate the congestion $\gamma(\mathcal{E})$ in the post-switching network $G^{\mathcal{E}}$ for all possible sets $\mathcal{E}$ of switching actions. Ideally one would calculate line flows and thus the maximum congestion using an AC power flow model, but this is not feasible when solving the OBS problem with brute force [3], [2] or with the

MILP approach proposed in the previous section. Both these methods strongly rely on the linear DC power flow model. The recursive approach, which was originally introduced to solve the tree partitioning problem using the DC power flow model, does not suffer from the same scalability issue and it is thus possible to incorporate AC power flow calculations as subroutine. This simple yet crucial modification enables the recursive approach to solve the tree partitioning problem with AC power flows, being the first algorithm to do so with time complexity that is linear in k. In Section VI-C, we compare the performance between the recursive and two-stage approach using AC power flows, where we use a brute-force algorithm to solve the OBS problem in the two-stage approach.

VI. NUMERICAL RESULTS

In this section, we evaluate the performance of the discussed methods to solve the optimization problem introduced in Section III. In our numerical experiments we use test cases from the `PGLib-OPF` library [16]. The experiments are performed using an Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 and 16 GB RAM. For more implementation details see [17].

A. Post-switching bridge-block decompositions

In this section we quickly look at the quality of the bridge-block decompositions obtained by tree partitioning various test networks. We use the two-stage approach on several test networks with $k = 5$ target clusters, where the OBI problem is solved using Spectral L_N and the OBS problem is solved by the MILP-based algorithm. Table I shows the bridge-block decomposition characteristics of the pre-switching and post-switching networks. In particular, observe that the bridge-block decomposition of the pre-switching networks all consist of a single bridge-block encompassing a large fraction of the buses. The tree partitioning procedure creates post-switching networks that have *at least* 5 non-trivial bridge blocks, but often slightly more due to unintended formation of new bridges (e.g., see Figure 4). The new non-trivial bridge-blocks are smaller in size, and, consequently, the new networks are more robust against failure propagation. Similar results with respect to the post-switching bridge-block decomposition have been obtained

TABLE I
BRIDGE-BLOCK DECOMPOSITION DUE TO TREE PARTITIONING WITH $k = 5$ TARGET BRIDGE-BLOCKS.

	pre-switching non-trivial bridge-blocks		post-switching non-trivial bridge-blocks	
Case	#	sizes	#	sizes largest 5
IEEE-30	1	$\{27\}$	5	$\{7, 3, 3, 3, 3\}$
IEEE-118	1	$\{109\}$	5	$\{39, 19, 19, 9, 8\}$
GOC-179	1	$\{150\}$	8	$\{40, 30, 30, 15, 13\}$
ACTIV-200	3	$\{125, 4, 3\}$	6	$\{37, 20, 15, 12, 4\}$
IEEE-300	3	$\{206, 3, 3\}$	8	$\{58, 46, 36, 19, 16\}$
GOC-500	3	$\{354\}$	5	$\{92, 84, 59, 53, 28\}$
GOC-793	1	$\{500\}$	6	$\{96, 61, 53, 47, 41\}$
RTE-1888	2	$\{881, 5\}$	8	$\{228, 165, 158, 146, 131\}$

when using other clustering algorithms, i.e., Fastgreedy and Spectral B_N, as well as when using the recursive method.

TABLE II
COMPARISON BETWEEN THE MILP-BASED TWO-STAGE AND RECURSIVE APPROACH WITH $k = 5$ AND USING DC POWER FLOWS.

Case	$ALG(\mathcal{P})$	$\gamma(\mathcal{E})$		Running time (s)	
		MILP	R-DC	MILP	R-DC
IEEE-118	Fastgreedy	1.57	**1.14**	0.23	0.49
	Spectral B_N	1.78	**1.00**	0.31	1.13
	Spectral L_N	**1.21**	**1.21**	0.26	0.69
GOC-179	Fastgreedy	**1.38**	**1.38**	0.51	0.19
	Spectral B_N	**1.38**	**1.38**	0.93	0.26
	Spectral L_N	**1.24**	1.51	0.70	0.35
IEEE-300	Fastgreedy	**1.16**	1.20	0.50	0.76
	Spectral B_N	**1.09**	1.68	0.59	1.12
	Spectral L_N	**1.09**	1.22	0.42	0.81
GOC-500	Fastgreedy	**1.28**	2.38	2.93	1.26
	Spectral B_N	**1.01**	2.36	1.16	1.83
	Spectral L_N	**1.01**	2.39	1.27	1.07
GOC-793	Fastgreedy	**1.50**	1.54	5.88	1.67
	Spectral B_N	**1.44**	2.64	3.09	1.59
	Spectral L_N	1.79	**1.34**	3.82	1.55
RTE-1888	Fastgreedy	**1.00**	1.88	4.56	5.20
	Spectral B_N	**1.00**	1.06	5.51	8.56
	Spectral L_N	1.10	**0.86**	12.81	4.36

B. Two-stage vs. recursive approach: DC power flows

Having demonstrated how tree partitioning can improve the quality of a network's bridge-block decomposition, we now investigate its impact on the maximum congestion under the assumption of DC power flows. Specifically, we compare the performance of the two-stage and recursive approach, where the former is solved by using the MILP-based algorithm for the OBS problem. We denote the two approaches by MILP and R-DC, respectively. We select $k = 5$ to demonstrate the scalability of our MILP-based algorithm and use the three different clustering algorithms as presented in Section III-D1, which we denote by $ALG(\mathcal{P})$.

The algorithms were run on a large collection of test networks. For each of them, the initial power injections p and power flows f were obtained by solving a DC-OPF problem. All presented test cases are selected such that the pre-switching maximum congestion is exactly 1, i.e., no line carries more power flow than its capacity.

Table II reports the objective value and the running time of both MILP and R-DC. For each test case and partitioning method, we highlighted the best solution in terms of maximum congestion in bold. MILP computed the best solutions in most of the presented test cases, often with much lower congestion levels than R-DC. The running times of both methods were comparable for smaller instances, but for large instances MILP took up to 2-4 times longer than R-DC to compute.

The results show that MILP performs better than 2-RC in terms of maximum congestion. Most importantly, MILP has running times comparable to R-DC, meaning that MILP drastically improves the running times of earlier proposed brute-force algorithms for the OBS problem and allows the two-stage

TABLE III
COMPARISON BETWEEN THE BRUTE-FORCE-BASED TWO-STAGE AND
RECURSIVE APPROACH WITH $k = 5$ AND USING AC POWER FLOWS.

Case	$ALG(\mathcal{P})$	$\gamma(\emptyset)$	$\gamma(\mathcal{E})$		Running time (s)	
			BF	R-AC	BF	R-AC
IEEE-30	Fastgreedy	1.07	**1.02**	2.13	1.53	0.05
	Spectral L_N	1.07	**1.02**	2.13	1.65	0.06
	Spectral B_N	1.07	**1.02**	1.06	1.84	0.08
EPRI-39	Fastgreedy	0.89	**1.11**	**1.11**	0.86	0.05
	Spectral L_N	0.89	**0.82**	1.11	0.50	0.07
	Spectral B_N	0.89	1.11	**1.09**	1.34	0.07
IEEE-73	Fastgreedy	0.95	0.96	**0.95**	1.01	0.12
	Spectral L_N	0.95	**1.21**	1.45	1.72	0.14
	Spectral B_N	0.95	**1.21**	1.21	2.06	0.15
IEEE-118	Fastgreedy	1.11	**1.11**	1.11	5.77	0.10
	Spectral L_N	1.11	**1.14**	1.15	3.44	0.18
	Spectral B_N	1.11	1.16	**1.11**	2.91	0.17
ACTIV-200	Fastgreedy	0.63	0.72	**0.69**	7.97	0.09
	Spectral L_N	0.63	0.72	**0.63**	9.08	0.16
	Spectral B_N	0.63	0.71	**0.63**	10.48	0.19

approach to be used for large instances. Lastly, the experiments show there is no single best clustering algorithm to be used for tree partitioning. This is partly due to the fact that the OBI problem does not take into account other important network characteristics, such as pre-switching congestion levels and line susceptances. Future work should look into possibly new formulations that take these factors into account.

C. Two-stage vs. recursive approach: AC power flows

We now compare the performance of the two-stage and recursive approach when using AC power flows. Since the MILP-based formulation of the OBS problem cannot account for the nonlinear AC power flows, we consider the brute-force (BF) variant of the two-stage approach here instead. We henceforth denote the recursive approach by R-AC.

Table III reports the performance of BF and R-AC in terms of maximum congestion and running time on five test cases, where we again highlighted the best solution in terms of maximum congestion in bold. For each test case, the initial power injections and power flows were computed using AC-OPF, but note that the pre-switching congestion $\gamma(\emptyset)$ was not strictly 1. The two-stage and recursive approach show similar performance in terms of maximum congestion: roughly one half of the best solutions were produced by BF, whereas the other half were computed by R-AC. However, most importantly, the recursive approach runs extremely fast with sub-second running times for all considered cases. Lastly, no single clustering algorithm worked best in minimizing the maximum congestion.

VII. CONCLUSION

In this paper we considered an optimization problem to refine the bridge-block decomposition of power networks by means of line switching actions while having minimal impact on the maximum congestion. We revisited a heuristic two-stage tree partitioning procedure and proposed an MILP-based algorithm to solve its second stage using DC power flows. Numerical experiments on several test networks show that the improved two-stage approach performs better than the recursive approach, while overcoming the computational bottleneck of earlier brute-force algorithms. Furthermore, we proposed a modification of the recursive approach of [2] to account for AC power flows. We show that, at least on five small instances, the recursive approach obtains objective values similar to the pre-existing brute-force two-stage approach, while drastically improving the running times.

Our numerical experiments suggest that the quality of the partitions plays an important role in achieving low post-switching congestion. As future work, we envision finding new optimization problem formulations to determine better partitions. Moreover, we plan to extend our current MILP for the OBS problem to account for AC power flows based on sophisticated linearization techniques in the literature [18].

REFERENCES

[1] D. Bienstock, *Electrical Transmission System Cascades and Vulnerability: An Operations Research Viewpoint*. SIAM, Dec. 2015.

[2] 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.

[3] J. W. Bialek and V. Vahidinasab, "Tree-Partitioning as an Emergency Measure to Contain Cascading Line Failures," *IEEE Trans. Power Syst.*, 2021.

[4] L. Guo, C. Liang, A. Zocca, S. H. Low, and A. Wierman, "Line Failure Localization of Power Networks Part II: Cut Set Outages," *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4152–4160, 2021.

[5] S. R. Salkuti, "Congestion Management Using Optimal Transmission Switching," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3555–3564, 2018.

[6] K. W. Hedman, S. S. Oren, and R. P. O'Neill, "Optimal transmission switching: Economic efficiency and market implications," *Journal of Regulatory Economics*, vol. 40, no. 2, pp. 111–140, Oct. 2011.

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

[8] 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.

[9] 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.

[10] ——, "Line Failure Localization of Power Networks Part I: Non-Cut Outages," *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4140–4151, 2021.

[11] M. E. J. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006.

[12] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner, "Maximizing Modularity is hard," *arXiv:physics/0608255*, Aug. 2006.

[13] L. Yu and C. Ding, "Network community discovery: Solving modularity clustering via normalized cut," in *Proceedings of the 8th Workshop on Mining and Learning with Graphs*, ser. MLG '10, Jul. 2010, pp. 34–36.

[14] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, Dec. 2004.

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

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

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

[18] C. Coffrin and P. Van Hentenryck, "A linear-programming approximation of AC power flows," *INFORMS Journal on Computing*, vol. 26, no. 4, pp. 718–734, 2014.