A Submodular Approach for
Electricity Distribution Network Reconfiguration

Ali Khodabakhsh*, Ger Yang*, Soumya Basu*, Evdokia Nikolova*, Michael C. Caramanis†,
Thanasis Lianeas*, Emmanouil Pountourakis*

* Department of Electrical and Computer Engineering, University of Texas at Austin
{ali.kh, geryang, basusoumya, thanasis, manolis}@utexas.edu, nikolova@austin.utexas.edu
† Department of Mechanical Engineering, Boston University
mcaraman@bu.edu

Abstract

Distribution network reconfiguration (DNR) is a tool used by operators to balance line load flows and mitigate losses. As distributed generation and flexible load adoption increases, the impact of DNR on the security, efficiency, and reliability of the grid will increase as well. Today, heuristic-based actions like branch exchange are routinely taken, with no theoretical guarantee of their optimality. This paper considers loss minimization via DNR, which changes the on/off status of switches in the network. The goal is to ensure a radial final configuration (called a spanning tree in the algorithms literature) that spans all network buses and connects them to the substation (called the root of the tree) through a single path. We prove that the associated combinatorial optimization problem is strongly NP-hard and thus likely cannot be solved efficiently. We formulate the loss minimization problem as a supermodular function minimization under a single matroid basis constraint, and use existing algorithms to propose a polynomial time local search algorithm for the DNR problem at hand and derive performance bounds. We show that our algorithm is equivalent to the extensively used branch exchange algorithm, for which, to the best of our knowledge, we pioneer in proposing a theoretical performance bound. Finally, we use a 33-bus network to compare our algorithm's performance to several algorithms published in the literature.

1. Introduction

Distribution networks are usually built as interconnected mesh networks, but are normally configured (via switches) and operated as radial networks (i.e. trees, in graph theoretic terms), to simplify overload protection [1]. The entire network can be thought of as a forest consisting of rooted trees. Each tree consists of a substation (root) and a number of customers (users) that are serviced via so-called distribution feeders (distribution lines starting at the substation). Switches located throughout the network allow dynamic reconfiguration of the distribution network through switching operations; the opening or closing of a switch corresponds to the removal or addition of an edge, respectively.

The goal of distribution networks is to deliver the power from substations to users, but notably, substantial losses of up to 13% occur as electric power flows over distribution lines [2]. As a result, *Distribution Network Reconfiguration* (DNR) is a major tool focusing on the dynamic identification of a spanning tree that optimizes a performance measure such as load flow balancing or total line loss minimization. We select the latter, namely the minimization of losses for a given hourly load flow, as the objective of the reconfiguration problem. Similar issues in meshed transmission networks have been addressed in the literature recently (see [3] and references therein).

Our results: In this paper, we analyze the DNR problem via a submodular optimization approach. In particular, we give the following results:

1. We prove that the DNR problem is strongly NP-hard. We do this through a polynomial reduction from 3-PARTITION problem, which is defined in Section 4 (see [4] for more details). To the best of our knowledge, the computational hardness of this problem has not been studied so far.

2. We formulate the DNR problem as a supermodular minimization problem subject to a single matroid basis constraint (we define supermodularity and matroid later in Section 5.1). Supermodularity is motivated by the fact that losses are quadratic in the current flowing over each branch of the distribution network. Furthermore, the matroid basis constraint ensures the radial structure and guarantees that all the buses are connected to the substation.

3. We observe that the local search algorithm for solving the supermodular minimization problem is equiv-

HICSS

alent to the well-known *branch exchange* algorithm. Hence, we obtain the first theoretical result on why the branch exchange algorithm performs well in practice.

The proposed submodular framework sheds some light on the algorithmic structure of the optimization problems in distribution networks. Although for the DNR problem we are mostly providing a theoretical justification for an existing heuristic, as it is evident in other lines of work in energy systems (see [5, 6, 7] for example), the theoretical study of such problems can help to either find new algorithms or improve the efficiency of existing ones.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 gives a concise formulation of the problem; and its computational complexity is studied in Section 4. The submodular framework is proposed in Section 5. Section 6 describes the algorithm and its performance guarantee. Section 7, provides numerical results and comparison with different algorithms. Finally, Section 8 concludes the work.

2. Related Work

DNR has been studied extensively in the literature. One of the most common heuristic algorithms is the *branch exchange* suggested by Civanlar *et al.* [8] and implemented by Baran and Wu [9], who considered loss minimization and load balancing objectives. Starting from a feasible tree configuration, the branch exchange algorithm transfers some loads in each iteration by (i) closing an open switch to create a loop in the network, followed by (ii) opening one of the closed switches in that loop to arrive at another feasible solution with a lower cost. The algorithm terminates when no further improvements are possible. This algorithm has been used as a benchmark against different DNR algorithms with the 12.6kV network of Fig. 1 employed for numerical comparisons.

An improved branch exchange algorithm was proposed by Miguez *et al.* [10] who tried to expand the space of available changes in the local search, hence eliminating some local minima of the standard algorithm. The idea of improved branch exchange is to investigate improvement from a pair of exchanges, once there is no improvement by a single branch exchange. Peng and Low [11] proposed an algorithm to do each step of branch exchange efficiently by solving only 3 optimal power flow equations (OPF), regardless of the size of the network. Their algorithm helps to find the best switch to open in order to minimize any convex increasing cost function, assuming that an open switch has already been closed. These improvements still provide
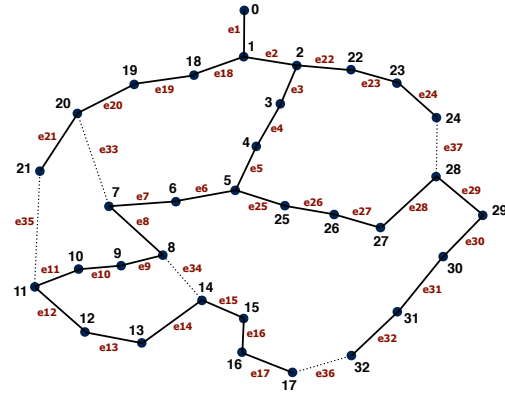

Figure 1: 33-bus network [17].

no theoretical guarantee on the output of the branch exchange algorithm.

Unlike the branch exchange algorithm that maintains a tree structure during its execution, there are other heuristic algorithms that start with the meshed network (obtained by closing all the tie switches) or the disconnected network (obtained by opening all the switches) and proceed to open/close switches one by one until a radial configuration is achieved [12, 13, 14]. Shirmohammadi and Hong [13] proposed one such algorithm that starts with the meshed network and proceeds with iterations that open the switch with the smallest current. No theoretical performance guarantees have been obtained for this algorithm.

For small networks such as the 33-bus example of Fig. 1, the global optimal configuration can be discovered by brute-force enumeration. An efficient enumeration approach proposed in [15], lists all the spanning trees in a clever way that generates each tree exactly once, and calculates losses by adjusting the losses of the previous spanning tree. The drawback of this method is that it is not practical for larger networks, since a network has exponentially many spanning trees [16].

The joint DNR and OPF problem was considered in [17] using Benders decomposition to decompose the global problem to master and slave subproblems. The master level determines the binary variables by solving a mixed-integer non-linear program using CPLEX. The slave level solves the OPF non-linear program using the CONOPT solver. Again, solving integer programs is computationally intractable for large networks.

Many other approaches like genetic algorithms [18, 19], particle swarm optimization [1], ant colony algorithms [20], artificial neural networks [21, 22], etc. have been utilized to solve this problem. A survey of different algorithms for the DNR problem can be found in [2]. What is conspicuously missing in all these previous works is a rigorous theoretical performance guarantee.
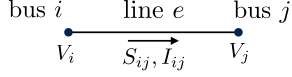
Figure 2: Power flow variables.

To close this gap, we consider a submodular approach to the DNR problem. Since switching binary decisions render DNR a non-linear combinatorial optimization problem, additional structure like submodularity or supermodularity enables finding an approximate solution efficiently.

3. Problem Formulation

In this section we present the power flow equations and employ some simplifying assumptions to model the problem in graph theoretic terms. We model the distribution network as a graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of buses (nodes) and $\mathcal{E}$ is the set of lines (undirected edges). We assume that a single substation is located at node 0, and the other nodes are load buses with given active and reactive power demands (p_i, q_i), for all $i \in \mathcal{N} \setminus \{0\}$. We are looking for a spanning tree rooted at bus 0 (i.e., a tree that connects all the loads to the root through a single path) which minimizes the total resistive loss.

Letting $V_i = |V_i| e^{\mathbf{i}\theta_i}$ represent the complex voltage at bus i, we adopt the relaxed branch model of [11, 23] that allows us to ignore the phase angles of voltages and currents in radial networks. Let $Z_e = R_e + \mathbf{i}X_e$ be the impedance of line $e \in \mathcal{E}$. We also use $S_{ij} = P_{ij} + \mathbf{i}Q_{ij}$ to express the branch power flow from bus i to bus j, and I_{ij} to express the current from bus i to j. A summary of our notation is depicted in Fig. 2.

Assumption 1 ([11, A2]). *Voltage variation across the distribution network can be neglected. Using per unit (p.u.) representation, we assume that $|V_i| = 1$ p.u. for all nodes $i \in \mathcal{N}$.*

This assumption is realistic since in practice voltage at every bus is kept within an allowable range such as $(0.95, 1.05)$ p.u., and impacts losses (the objective function of DNR) at a smaller order of magnitude than different spanning trees. Moreover, this assumption does not change significantly the ordering of spanning trees based on the associated line losses.

Assumption 2. *The impact of line losses on line flows is negligible relative to the power demands at the buses of the network.*

This assumption implies that the power flow on each line $e \in \mathcal{E}$ is almost equal to the total demand of the buses that are receiving power through that line. Specif-

ically, for a given spanning tree, if we denote the set of successors of an edge $e \in \mathcal{E}$ by c_e, then we have:

$$P_e = \sum_{i \in c_e} p_i \quad \text{and} \quad Q_e = \sum_{i \in c_e} q_i, \tag{1}$$

where p_i and q_i are the active and reactive power demands at bus i. Note that by P_e we mean the power flowing on line e in the direction from the root of the tree to the leaves (parent to child). In Section 7, we verify the validity of these assumptions in detail.

If we denote the loss of line $e = \{i, j\} \in \mathcal{E}$ by L_e, then we have:

$$L_e = R_e \times |I_{ij}|^2. \tag{2}$$

In addition, in the relaxed model we have:

$$|V_i|^2 |I_{ij}|^2 = P_{ij}^2 + Q_{ij}^2. \tag{3}$$

Combining (1), (2), and (3) with Assumption 1 implies that:

$$L_e = R_e \left[\left(\sum_{i \in c_e} p_i \right)^2 + \left(\sum_{i \in c_e} q_i \right)^2 \right].$$

Given a spanning tree (ST) we can sum up the line losses L_e over all the edges of the tree to find the total loss. Thus, the optimal reconfiguration problem with the goal of loss minimization can be written as the following optimization problem:

$$\min_{ST} \sum_{e \in ST} R_e \left[\left(\sum_{i \in c_e} p_i \right)^2 + \left(\sum_{i \in c_e} q_i \right)^2 \right], \tag{P1}$$

where the minimization is over all the spanning trees of $\mathcal{G}(\mathcal{N}, \mathcal{E})$.

4. Hardness Result

In this section we prove that the DNR problem is strongly NP-hard in general by a reduction from the 3-PARTITION problem [4]. A computational hardness result is more powerful when it is derived for a more restricted setting—since the hardness implication then holds for any generalization of the setting. Here we derive a hardness result for the special case of unit demands, where the objective function of the optimization problem (P1) reduces to a simpler function that is just counting the number of successors. In particular we make the following assumptions:

$$\begin{aligned} R_e &= 1 \qquad \forall e \in \mathcal{E}, \\ p_i &= 1 \qquad \forall i \in \mathcal{N} \setminus \{0\}, \\ q_i &= 0 \qquad \forall i \in \mathcal{N} \setminus \{0\}. \end{aligned}$$
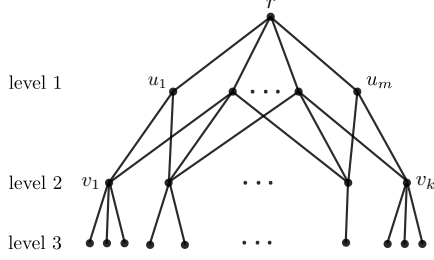
Figure 3: Polynomial reduction.

Under these assumptions, the optimization problem (P1) reduces to the following problem:

$$\min_{ST} \quad \sum_{e \in ST} (\text{number of successors of } e \text{ in } ST)^2. \quad \text{(P2)}$$

Although these assumptions may not be realistic, they transform the problem into an explicit combinatorial problem (without any power flow variable or parameter) and help us to analyze the computational complexity of the reconfiguration problem. The resulting complexity applies then to more general and realistic settings, as mentioned above.

We show that even the unit-demand case is strongly NP-hard. We prove this by a reduction from the 3-PARTITION problem defined as follows.

Definition 1. (3-PARTITION) *In the* 3-PARTITION *problem we have a multiset of $k = 3m$ integers summing to mB with each integer strictly between $B/4$ and $B/2$. The task is to partition these numbers into m triplets each with a sum of B.*

It is well known [4] that in the 3-PARTITION problem, deciding whether a given multiset can be partitioned into balanced triplets or not, is strongly NP-complete, i.e., it is NP-complete even if the numbers are bounded by a polynomial in the length of the input.

Theorem 1 (Hardness result). *Distribution network reconfiguration problem (P1) is strongly NP-hard.*

Proof. We propose a polynomial reduction from the 3-PARTITION problem to the unit-demand case of reconfiguration problem (P2). Given an instance of the 3-PARTITION problem we build an instance of the reconfiguration problem such that the optimal spanning tree reveals the answer to the 3-PARTITION problem (if it exists). Given $k = 3m$ integers $\{a_1, a_2, ..., a_k\}$, we construct a network as shown in Fig. 3. There is a root r, m nodes $u_1, ..., u_m$ connected to the root, $k = 3m$ nodes $v_1, ..., v_k$ each connected to all of u_i's (thus v_i's and u_j's form a complete bipartite graph) and for each v_i we have $a_i - 1$ nodes connected to it.

Lemma 2 below proves that all the lines between the root r and the u_j's are part of the optimal tree. Moreover, all the lines between levels two and three appear in every spanning tree, so the only choices are on the lines between levels one and two. In particular, we have to connect each v_i to exactly one u_j, i.e., make one of m choices.

When we connect node v_i to node u_j, the corresponding edge gets a cost of a_i^2, since there are $a_i - 1$ nodes in level three and the edge has a_i successors including v_i. This cost is independent of the choice of u_j, so the total cost for the edges between levels one and two is the same for all the spanning trees. The cost to be minimized is thus the total cost of the edges between root r and the u_j's.

Let S_j be the set of indices of the children of u_j, i.e.,

$$S_j = \{i \mid (u_j, v_i) \in \text{Tree}\},$$

then the cost related to edge (r, u_j) is:

$$C_{(r,u_j)} = \left(1 + \sum_{i \in S_j} a_i\right)^2,$$

where 1 counts for the node u_j itself. Now the total cost of the spanning tree is:

$$C = \sum_{j=1}^{m} C_{(r,u_j)} + \sum_{i=1}^{3m} a_i^2 + \sum_{i=1}^{3m} (a_i - 1)$$

$$= \sum_{j=1}^{m} \left(1 + \sum_{i \in S_j} a_i\right)^2 + \sum_{i=1}^{3m} a_i^2 + (mB - 3m). \quad (4)$$

As mentioned earlier, the second and third terms are constants since they are independent of the choice of the spanning tree. Using the fact that the S_j's are disjoint and $\cup_{j=1}^{m} S_j = \{1, ..., k\}$, we also have:

$$\sum_{j=1}^{m} \left(1 + \sum_{i \in S_j} a_i\right) = m + \sum_{j=1}^{m} \sum_{i \in S_j} a_i$$

$$= m + \sum_{i=1}^{3m} a_i = m + mB = m(B+1). \quad (5)$$

Lemma 1. *The minimum of $\sum_{i=1}^{n} x_i^2$ given that $\sum_{i=1}^{n} x_i = C$ for a constant $C \in \mathbb{R}$ is achieved when $x_i = C/n$ for all $1 \leq i \leq n$.*

By Lemma 1 and (5), the minimum possible cost of (4) is obtained when:

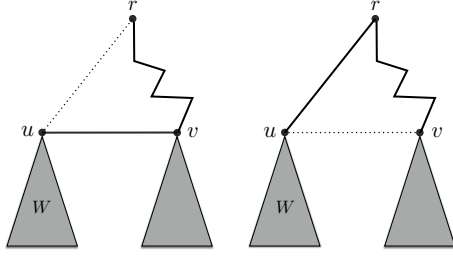$$1 + \sum_{i \in S_j} a_i = B + 1 \qquad \forall j \in \{1, ..., m\},$$

Figure 4: Proof of Lemma 2.

and the optimal value is:

$$C_{min} = m(B+1)^2 + \sum_{i=1}^{3m} a_i^2 + (mB - 3m).$$

Note that this optimal cost is achieved when a_i's are partitioned into m subsets with sum B, but there is no restriction on the size of S_j's. This means that node u_j can have any number of v_i's connected to it, while in the 3-PARTITION problem we want to partition the a_i's into m triplets. The property $B/4 < a_i < B/2$ ensures that this minimum can only be achieved when $|S_j| = 3$ for all j. If for any j' we have $|S_{j'}| > 3$, then we get:

$$\sum_{i \in S_{j'}} a_i > 4 \times \frac{B}{4} = B,$$

and the partition cannot be balanced. Similarly if $|S_{j'}| < 3$, then we get:

$$\sum_{i \in S_{j'}} a_i < 2 \times \frac{B}{2} = B.$$

In conclusion, the algorithm for the unit-demand case finds the tree corresponding to the 3-PARTITION answer (if it exists), and if it outputs some unbalanced tree, this means that the 3-PARTITION does not exist. If each a_i is bounded by a polynomial in k, the constructed network has polynomial number of nodes, hence any polynomial time algorithm for the unit-demand case provides a pseudo-polynomial time algorithm for the 3-PARTITION problem which is not possible unless $P = NP$. $\square$

Lemma 2 proves the only remaining part of the hardness proof.

Lemma 2. *With uniform line resistances ($R_e = R, \forall e \in \mathcal{E}$), the optimal tree includes all the edges adjacent to the root.*

Proof. We prove this by contradiction. Assume that we have an optimal tree which does not choose edge (r, u) as shown in Fig. 4 on the left. Let W be the total load weight of subtree connected to u (including u). Since we have a tree, this subtree is connected to the root through

another node v. Node v may have other children and also may be connected to the root via one or more edges. Now we claim that this tree cannot be optimal since we can exchange edge (u, v) with edge (r, u) and improve the objective value as shown in the right tree. To see this, note that both edges (u, v) in the left tree and (r, u) in the right tree have costs RW^2, but the exchange of (u, v) with (r, u) decreases the load on all the edges of the path from r to v by W, hence decreasing the total cost. This contradicts the optimality of the first tree. $\square$

Note that the unit-demand case is a special case of Lemma 2.

5. Supermodular Structure

In the previous section we showed that DNR is strongly NP-hard, but if we find some additional structure such as submodularity or supermodularity in the problem, we may be able to provide approximation algorithms, which provides a rigorous worst-case performance guarantee. Here we first define this structure in addition to some required background about matroid constraints, and then we show that the DNR problem has this structure.

5.1. Submodularity and Supermodularity

Let V be a finite set, called the ground set. We use 2^V to denote the set of all subsets of V, called the power set. A set function $f : 2^V \mapsto \mathbb{R}$ is submodular if it has the *diminishing returns* property, namely adding an element to a bigger set is less valuable than adding it to a smaller set.

Definition 2 (Submodularity). *A set function $f : 2^V \mapsto \mathbb{R}$ with a ground set V is submodular if:*

$$f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y),$$

for every $X \subseteq Y \subseteq V$, $u \in V \backslash Y$.

Function f is said to be supermodular if $-f$ is submodular (or the above inequality holds in the other direction). Supermodularity captures an increasing returns property. A function is said to be modular if it is both submodular and supermodular.

Definition 3 (Monotonicity). *A set function $f : 2^V \mapsto \mathbb{R}$ is said to be monotone increasing if $f(X) \leq f(Y)$ for any $X \subseteq Y \subseteq V$.*

Definition 4 (Matroid [24]). *Let V be a finite set, and let $\mathcal{I}$ be a collection of subsets of V. The pair $\mathcal{M} = (V, \mathcal{I})$ is a matroid if the following conditions hold: (1) If $B \in \mathcal{I}$, then $A \in \mathcal{I}$ for all $A \subseteq B$, (2) If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists $v \in B \backslash A$ such that $A \cup \{v\} \in \mathcal{I}$.*

A set $A \in \mathcal{I}$ is called an independent set. The collection $\mathcal{I}$ is called the set of independent sets of the matroid $\mathcal{M}$. A maximal independent set (an independent set that has maximum size) is a base of the matroid. It is easy to show that all the bases of a matroid have the same number of elements.

5.2. Set Function Formulation of DNR

Considering the formulation of DNR (P1), the optimization problem is over all the spanning trees of the original graph. We would like to encode the two properties of "being a tree" and "touching all the vertices of the graph" into a set of constraints. In order to do this, we need to define a set of variables as follows. These variables also help to determine the successors of an edge in any arbitrary tree.

- For any edge $e \in \mathcal{E}$ we define a variable x_e that indicates if that edge is included in the tree or not (the number of variables is equal to the number of lines in the distribution network).
- Corresponding to any variable x_e, where $e = \{i, j\}$, we also define y_{ij}^k and y_{ji}^k for all $k \in \mathcal{N}$, which indicate the position of node k compared to edge $e = \{i, j\}$. If there is a simple path from i to k including $\{i, j\}$, then $y_{ij}^k = 1$ and if there is a simple path from j to k including $\{i, j\}$, then $y_{ji}^k = 1$. In other words, $y_{ij}^k = 1$ means that edge $\{i, j\}$ is chosen and j is on the path from i to k. If $x_e = 0$, then both y_{ij}^k and y_{ji}^k are zero.

The following theorem, inspired by the integer programming formulation for the minimum spanning tree problem [25, 26], explains how we use these variables to characterize the spanning trees explicitly.

Theorem 2 (**Feasible set characterization**). *There is a one-to-one correspondence between the spanning trees of $\mathcal{G}(\mathcal{N}, \mathcal{E})$ and the feasible set specified by the following set of constraints:*

$$\sum_{e \in \mathcal{E}} x_e = n - 1 \tag{6}$$

$$y_{ij}^k + y_{ji}^k = x_e \qquad \forall e = \{i, j\} \in \mathcal{E}, \forall k \in \mathcal{N} \tag{7}$$

$$x_e + \sum_{k \neq i, j} y_{ik}^j = 1 \qquad \forall i, j \in \mathcal{N} : e = \{i, j\} \in \mathcal{E} \tag{8}$$

$$x_e, y_{ij}^k, y_{ji}^k \in \{0, 1\} \qquad \forall e = \{i, j\} \in \mathcal{E}, \forall k \in \mathcal{N} \tag{9}$$

If we write the total loss as a function of the binary variables above, we end up with an integer program formulation of (P1). For a given spanning tree T (equivalently, a feasible set of values for the binary variables), and an edge $e = \{i, j\} \in T$, the variables y_{ij}^k and y_{ji}^k induce a partition of the vertices $\mathcal{N}$ into two sets which are exactly the two connected components of the tree

obtained by removing $\{i, j\}$. The set that does not include the root (assuming vertex 0 is the root), is the set of successors of e in T. In other words, if $y_{ji}^0 = 1$, and c_e is the set of its successors, then we have:

$$c_e = \{k \in \mathcal{N} : y_{ij}^k = 1\}.$$

Note that $y_{ji}^0 = 1$ is not an additional assumption, since the edges are not directed, and hence for the edges in the tree, one of the pairs (i, j) or (j, i) satisfies this condition.

Using this new description of successors, we can rewrite the objective function as:

$$\sum_{e \in ST} R_e \left[\left(\sum_{i \in c_e} p_i \right)^2 + \left(\sum_{i \in c_e} q_i \right)^2 \right] =$$

$$\sum_{i, j : \{i, j\} \in \mathcal{E}} R_{ij} y_{ji}^0 \left[\left(\sum_{k \in \mathcal{N}} y_{ij}^k p_k \right)^2 + \left(\sum_{k \in \mathcal{N}} y_{ij}^k q_k \right)^2 \right], \tag{10}$$

where the inner summations are over all nodes, but the y_{ij}^k's guarantee that we only count the successors, and the term y_{ji}^0 outside guarantees that we calculate each edge of the tree exactly once and in the correct direction with respect to the root.

So, the following optimization problem is equivalent to (P1):[1]

$$\min \quad \sum_{\{i, j\} \in \mathcal{E}} R_{ij} y_{ji}^0 \left[\left(\sum_{k \in \mathcal{N}} y_{ij}^k p_k \right)^2 + \left(\sum_{k \in \mathcal{N}} y_{ij}^k q_k \right)^2 \right]$$

$$s.t. \quad (6), (7), (8), (9). \tag{P3}$$

Now we show that (P3) is equivalent to a supermodular minimization problem with a single matroid basis constraint. The objective function (10) is not supermodular over $\mathcal{E}$, but we create a similar set function that is supermodular and is equal to (10) when constraints (6–9) hold (i.e., for spanning trees). A corollary to the following theorem shows that the feasible set in (P3) is indeed a matroid basis constraint.

Theorem 3 (**Cycle Matroid** [24]). *Let $\mathcal{G}(\mathcal{N}, \mathcal{E})$ be an undirected graph. Define the set T to be the collection of all subsets of $\mathcal{E}$ that form a forest (i.e., the subset is acyclic). In other words, $A \in T$ iff $A \subseteq \mathcal{E}$ and edges in A do not form a cycle. Then $\mathcal{M} = (\mathcal{E}, T)$ is a matroid called the cycle matroid of graph $\mathcal{G}$ (also known as graphic matroid).*

---

[1] we use $\sum_{\{i, j\} \in \mathcal{E}}$ instead of $\sum_{i, j \in \mathcal{N} : \{i, j\} \in \mathcal{E}}$ for simplicity.

Corollary 1. *Assuming that graph $\mathcal{G}$ is connected, the bases of the cycle matroid $\mathcal{M}$ are the spanning trees of $\mathcal{G}$, which all have cardinality $|\mathcal{N}| - 1$. Therefore, constraints (6–9) are equivalent to a single matroid basis constraint on $\mathcal{E}$.*

Now we introduce the supermodular set function over $\mathcal{E}$. For any $A \subseteq \mathcal{E}$, we define:

$$f(A) = \sum_{\{i,j\} \in \mathcal{E}} R_{ij} z_{ji}^0 \left[\left(\sum_{k \in \mathcal{N}} z_{ij}^k p_k \right)^2 + \left(\sum_{k \in \mathcal{N}} z_{ij}^k q_k \right)^2 \right] \tag{11}$$

The only difference between (10) and (11) is that we replaced the y_{ij}^k's with z_{ij}^k's, and z_{ij}^k is defined similar to y_{ij}^k except that it can be any non-negative integer (compared to $0, 1$) and it counts the number of paths in A starting with $\{i,j\}$ and going to k. Clearly, for spanning trees there cannot be more than one path between any arbitrary pair of vertices, therefore $z_{ij}^k = y_{ij}^k$ and this implies the equality of (10) and (11) when constraints (6–9) hold.

Theorem 4 (Supermodularity). *Objective function (11) is a supermodular set function over $\mathcal{E}$, provided that the p_i's and q_i's are non-negative.*

Proof. The sum of supermodular set functions is supermodular, so we only need to prove the supermodularity for a fixed edge $\{i,j\} \in \mathcal{E}$. We can also drop positive constants like R_{ij}. Define $f_{ij}(A)$ and $f'_{ij}(A)$ as follows:

$$f_{ij}(A) = z_{ji}^0 \left(\sum_{k \in \mathcal{N}} z_{ij}^k p_k \right)^2, f'_{ij}(A) = z_{ji}^0 \left(\sum_{k \in \mathcal{N}} z_{ij}^k q_k \right)^2$$

We now prove that $f_{ij}(A)$ is supermodular. A similar proof works for $f'_{ij}(A)$. We want to show that:

$$f_{ij}(A \cup \{e\}) - f_{ij}(A) \le f_{ij}(B \cup \{e\}) - f_{ij}(B), \tag{12}$$

for every $A \subseteq B \subseteq \mathcal{E}$, $e \in \mathcal{E}$, and $e \notin B$. For any k, let a_{ij}^k be the change in z_{ij}^k when we add e to A, i.e.:

$$a_{ij}^k = z_{ij}^k(A \cup \{e\}) - z_{ij}^k(A),$$

where $z_{ij}^k(A)$ is just z_{ij}^k, calculated based on the edges in A. Similarly, let b_{ij}^k be defined for B and $B \cup \{e\}$. We have $a_{ij}^k \le b_{ij}^k$, because any new path in A created by adding e is also a new path in B. Another fact is that $z_{ij}^k(A) \le z_{ij}^k(B)$, because adding more edges cannot decrease the number of paths between any pair of vertices (i.e., $f(A)$ is a monotone increasing function). Now we prove (12):

$$f_{ij}(B \cup \{e\}) - f_{ij}(B) \tag{13}$$

$$= z_{ji}^0(B \cup \{e\}) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(B \cup \{e\}) p_k \right)^2 \tag{14}$$

$$- z_{ji}^0(B) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(B) p_k \right)^2$$

$$= (z_{ji}^0(B) + b_{ji}^0) \left(\sum_{k \in \mathcal{N}} b_{ij}^k p_k + \sum_{k \in \mathcal{N}} z_{ij}^k(B) p_k \right)^2 \tag{15}$$

$$- z_{ji}^0(B) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(B) p_k \right)^2$$

$$\ge (z_{ji}^0(A) + b_{ji}^0) \left(\sum_{k \in \mathcal{N}} b_{ij}^k p_k + \sum_{k \in \mathcal{N}} z_{ij}^k(B) p_k \right)^2 \tag{16}$$

$$- z_{ji}^0(A) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(B) p_k \right)^2$$

$$\ge (z_{ji}^0(A) + b_{ji}^0) \left(\sum_{k \in \mathcal{N}} b_{ij}^k p_k + \sum_{k \in \mathcal{N}} z_{ij}^k(A) p_k \right)^2 \tag{17}$$

$$- z_{ji}^0(A) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(A) p_k \right)^2$$

$$\ge (z_{ji}^0(A) + a_{ji}^0) \left(\sum_{k \in \mathcal{N}} a_{ij}^k p_k + \sum_{k \in \mathcal{N}} z_{ij}^k(A) p_k \right)^2 \tag{18}$$

$$- z_{ji}^0(A) \left(\sum_{k \in \mathcal{N}} z_{ij}^k(A) p_k \right)^2$$

$$= f_{ij}(A \cup \{e\}) - f_{ij}(A). \tag{19}$$

In (14), (15) we just applied the definitions of f_{ij} and b_{ij}^k, respectively. In (15), the aggregate coefficient of $z_{ji}^0(B)$ is positive, so using the fact that $z_{ji}^0(B) \ge z_{ji}^0(A)$, we get (16). To get (17), note that quadratic function $(x + \alpha)^2 - x^2 < (y + \alpha)^2 - y^2$ for $x < y$ and fixed $\alpha > 0$. Setting $\alpha = \sum_{k \in \mathcal{N}} b_{ij}^k p_k$, and the fact that $z_{ij}^k(A) \le z_{ij}^k(B)$ implies (17). Finally, (18) is implied by $a_{ij}^k \le b_{ij}^k$, which proves the supermodularity of $f_{ij}(A)$. We have

$$f(A) = \sum_{\{i,j\} \in \mathcal{E}} R_{ij} \left(f_{ij}(A) + f'_{ij}(A) \right),$$

therefore $f(A)$ is supermodular. $\qquad\square$

Table 1: Comparison of different DNR algorithms on the 33-bus network of Fig. 1

Algorithm	Method	Open Lines	Loss (kW)
Proposed	Submodular Local Search	$7, 9, 14, 32, 37$	139.552
Morton and Mareels [15]	Brute-Force	$7, 9, 14, 32, 37$	139.552
Gomes *et al.* [12]	Greedy on Mesh Network	$7, 9, 14, 32, 37$	139.552
Khodr and Martinez-Crespo [17]	Benders Decomposition	$7, 9, 14, 32, 37$	139.552
Wu *et al.* [20]	Ant Colony	$7, 9, 14, 28, 32$	139.976
Shirmohammadi and Hong [13]	Optimal Current Pattern	$7, 10, 14, 32, 37$	140.279
Baran and Wu [9][3]	Branch Exchange	$11, 28, 31, 33, 34$	146.832
Initial Configuration		$33, 34, 35, 36, 37$	202.670

6. Algorithm and Performance Guarantee

In the previous section we showed that the DNR problem (P3) is equivalent to a supermodular minimization problem subject to a single matroid basis constraint. Unless $P = NP$, it is not possible to approximate the minimum of a supermodular function within any factor [27], in contrast with the related problem of maximizing a submodular function which admits a constant factor approximation algorithm [28]. We adapt the approximation algorithm for the submodular maximization problem under matroid constraints, proposed by Lee *et al.* [28], to solve the DNR problem, but we have to convert the supermodular function to a non-negative submodular function (by negating and shifting). This conversion affects the multiplicative approximation guarantee, as shown in Theorem 5. The algorithm, which is based on local search, is described in Algorithm 1.

Algorithm 1 Distribution Network Reconfiguration for Loss Minimization

---

1: **Input:** Configuration $\mathcal{G}(\mathcal{N}, \mathcal{E})$, bus demands (p_k, q_k), line resistances (R_{ij}), ϵ.
2: **Output:** Spanning tree for minimizing the total loss.
3: **Initialize** T with an arbitrary spanning tree.
4: **while** 1 **do**
5:    **if** there exist $e \in \mathcal{E} \backslash T$ and $e' \in T$ such that $(T \backslash \{e'\}) \cup \{e\}$ is a spanning tree and $f((T \backslash \{e'\}) \cup \{e\}) < (1 - \epsilon) f(T)$ **then**
6:       $T \leftarrow (T \backslash \{e'\}) \cup \{e\}$
7:    **else**
8:       break
9:    **end if**
10: **end while**
11: **return** T

The algorithm starts with an arbitrary spanning tree T. Then at each iteration, it looks for two edges $e \in \mathcal{E} \backslash T$ and $e' \in T$ such that swapping those two edges makes

another spanning tree with loss at most $(1 - \epsilon) f(T)$. If such a pair exists, it updates T and repeats the exchange process, otherwise the algorithm terminates and outputs the locally optimal spanning tree.

Theorem 5 (**Performance guarantee**). *Let T_{alg} be the output of Algorithm 1, and T^* be the optimal spanning tree, i.e., $T^* = argmin\{f(T) : T \subseteq \mathcal{E}, T$ is a spanning tree$\}$. Let $M = f(\mathcal{E})$, which is an upper bound on $f(A)$ for all $A \subseteq \mathcal{E}$, then:*

$$M - f(T_{alg}) \geq \left(\frac{1}{6} - \epsilon\right)\left(M - f(T^*)\right). \quad (20)$$

Proof. This is a corollary of [28, Theorem 22], which provides a $(\frac{1}{6} - \epsilon)$-approximation algorithm for maximizing any non-negative submodular function over bases of a matroid $\mathcal{M}$.[2] Here we use $M - f(A)$ as the non-negative submodular function, and the spanning trees are the bases of the cycle matroid discussed in Theorem 3. $\square$

Even though Algorithm 1 is based on the local search approximation algorithm for maximizing non-monotone submodular functions [28], it is equivalent to the branch exchange heuristic algorithm which has been used since the late 1980s [9]. This establishes that Theorem 5 provides the first proof of a performance bound, and hence a performance guarantee for the branch exchange algorithm.

7. Experiments

Table 1 shows the results of our experiments on the 33-bus network of Fig. 1. The parameters of the network can be found in [1]. All the active and reactive power demands are positive for this network as assumed

---

[2]That theorem requires $\mathcal{M}$ to have at least two disjoint bases. We can solve this (if necessary) by adding dummy edges with very high resistances (to make sure that the algorithm never selects them). Moreover, their algorithm performs another local search which allows deletion of elements, but that run yields the empty set in our case (due to the monotonicity), hence does not apply to the DNR problem.
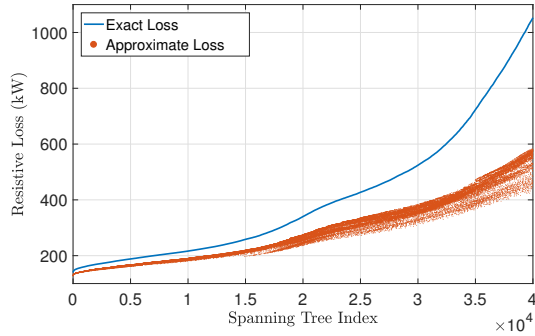
Figure 5: Comparing losses from the simplified model with the exact values.

Figure 6: Rankings based on exact and approximate losses for the best 5000 spanning trees.

in Theorem 4. The simulations have been done by using the MATPOWER package in MATLAB [29]. The results show that in this case, our submodular approach finds the globally optimal configuration, which was found in [15] (by enumerating all 50751 spanning trees). In [9], 2 different approximate power flow methods with different accuracies have been used and we also believe that there are inconsistencies regarding the parameters of the network in the literature[3]; that is why results reported in [9] differ from what we obtained by Algorithm 1. Clearly, the output of the local search algorithms depends on the initialization. We used the initial configuration (Fig. 1) as the initial spanning tree in our simulation. Further, to check the robustness with respect to the initial tree, we repeated the simulations with 1000 random initial trees, all of which ended with the same optimal solution.

In order to check the validity of our assumptions (see the problem formulation in Section 3), we compare the losses of spanning trees as measured in (P1) with the exact losses obtained from MATPOWER. The result is shown in Fig. 5. The blue line is the exact loss curve where the spanning trees are sorted in the order of increasing total loss. The red dots also show the loss for each tree obtained from the simplified model. We observe that the approximate loss is generally increasing, which means that it can be used in the local search algorithm. In fact performing the local search with either exact loss or approximate loss results in the same globally optimal tree, reported in Table 1. As expected, approximate loss estimates are less accurate for trees with higher losses, since the resistive losses approach the order of magnitude of load demands in such networks (hence contradicting Assumption 2). On the other
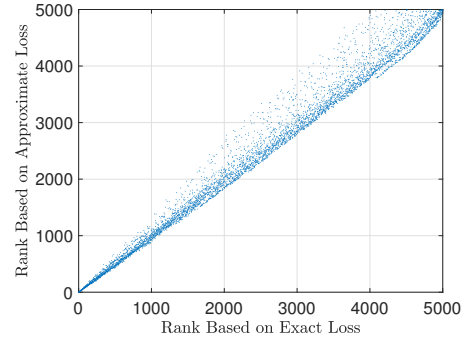
hand, for trees with smaller losses (which are indeed the target of our optimization problem) the simplified loss approximates the exact loss very well.

Fig. 6 also compares the rank of the top 5000 spanning trees based on the exact and approximate losses. Ideally, we would like the simplified losses to preserve the rankings (which would result in a $y = x$ line in this plot). We observe that no single spanning tree faces a significant change in its ranking.

8. Conclusion

In this paper, we studied the distribution network reconfiguration problem (DNR) for loss minimization through a submodular optimization approach. We proved that this problem is NP-hard even if the demands and the line resistances are all equal to one. We formulated this problem as a supermodular minimization problem subject to a matroid basis constraint. We then used the algorithm for maximizing non-monotone submodular functions under matroid constraints, to give a polynomial time algorithm for the DNR problem with a performance guarantee. The algorithm was equivalent to the branch exchange algorithm that was known previously, but for which no theoretical guarantees were available. By discovering a submodular structure in the problem, we pioneered the derivation of a performance bound on the branch exchange algorithm.

Although supermodular minimization cannot be approximated in general, there are approximation algorithms for the case when the supermodular function has bounded curvature (see [30, 31] for the definition of curvature and the approximation algorithms). The formulation studied in this paper does not have bounded curvature. One interesting question that arises is whether the DNR problem can be formulated as minimizing a supermodular function with bounded curvature. A posi-

---

[3]The resistance of the branch between bus 6 and bus 7 is 0.7114Ω in [9], but 1.7114Ω in [1]. We used the latter value in all our simulations.

tive determination would imply a multiplicative constant factor approximation (compared to Theorem 5 which includes the upper bound M) and would provide a significant improvement.

9. References

[1] K. Sathish Kumar and S. Naveen, "Power system reconfiguration and loss minimization for a distribution system using catfish PSO algorithm," *Frontiers in Energy*, vol. 8, no. 4, pp. 434–442, 2013.

[2] R. J. Sarfi, M. Salama, and A. Chikhani, "A survey of the state of the art in distribution system reconfiguration for system loss reduction," *Electric Power Systems Research*, vol. 31, no. 1, pp. 61–70, 1994.

[3] E. A. Goldis, X. Li, M. C. Caramanis, A. M. Rudkevich, and P. A. Ruiz, "AC-Based topology control algorithms (TCA)–A PJM historical data case study," in *IEEE 48th Hawaii International Conference on System Sciences (HICSS)*, pp. 2516–2519, 2015.

[4] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York, 1979.

[5] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran, "Towards scalable voltage control in smart grid: a submodular optimization approach," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, p. 20, 2016.

[6] M. G. Damavandi, V. Krishnamurthy, and J. R. Martí, "Robust meter placement for state estimation in active distribution systems," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1972–1982, 2015.

[7] N. Gensollen, V. Gauthier, M. Marot, and M. Becker, "Submodular optimization for control of prosumer networks," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 180–185, 2016.

[8] S. Civanlar, J. Grainger, H. Yin, and S. Lee, "Distribution feeder reconfiguration for loss reduction," *IEEE Transactions on Power Delivery*, vol. 3, no. 3, pp. 1217–1223, 1988.

[9] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.

[10] E. Míguez, J. Cidrás, E. Díaz-Dorado, and J. L. García-Dornelas, "An improved branch-exchange algorithm for large-scale distribution network planning," *IEEE Transactions on Power Systems*, vol. 17, no. 4, pp. 931–936, 2002.

[11] Q. Peng and S. H. Low, "Optimal branch exchange for feeder reconfiguration in distribution networks," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, pp. 2960–2965, 2013.

[12] F. V. Gomes, S. Carneiro, J. L. R. Pereira, M. P. Vinagre, P. A. N. Garcia, and L. R. Araujo, "A new heuristic reconfiguration algorithm for large distribution systems," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1373–1378, 2005.

[13] D. Shirmohammadi and H. W. Hong, "Reconfiguration of electric distribution networks for resistive line losses reduction," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1492–1498, 1989.

[14] T. E. McDermott, I. Drezga, and R. P. Broadwater, "A heuristic nonlinear constructive method for distribution system reconfiguration," *IEEE Transactions on Power Systems*, vol. 14, no. 2, pp. 478–483, 1999.

[15] A. B. Morton and I. M. Mareels, "An efficient brute-force solution to the network reconfiguration problem," *IEEE Transactions on Power Delivery*, vol. 15, no. 3, pp. 996–1000, 2000.

[16] W. Kocay and D. L. Kreher, *Graphs, algorithms, and optimization*. CRC Press, 2016.

[17] H. Khodr and J. Martinez-Crespo, "Integral methodology for distribution systems reconfiguration based on optimal power flow using Benders decomposition technique," *IET generation, transmission & distribution*, vol. 3, no. 6, pp. 521–534, 2009.

[18] W. Lin, F. Cheng, and M. Tsay, "Distribution feeder reconfiguration with refined genetic algorithm," *IEEE Proceedings-Generation, Transmission and Distribution*, vol. 147, no. 6, pp. 349–354, 2000.

[19] B. Enacheanu, B. Raison, R. Caire, O. Devaux, W. Bienia, and N. Hadjsaid, "Radial network reconfiguration using genetic algorithm based on the matroid theory," *IEEE Transactions on Power Systems*, vol. 23, no. 1, pp. 186–195, 2008.

[20] Y. K. Wu, C. Y. Lee, L. C. Liu, and S. H. Tsai, "Study of reconfiguration for the distribution system with distributed generators," *IEEE transactions on Power Delivery*, vol. 25, no. 3, pp. 1678–1685, 2010.

[21] H. Kim, Y. Ko, and K. Jung, "Artificial neural-network based feeder reconfiguration for loss reduction in distribution systems," *IEEE Transactions on Power Delivery*, vol. 8, no. 3, pp. 1356–1366, 1993.

[22] M. Kashem, G. Jasmon, A. Mohamed, and M. Moghavvemi, "Artificial neural network approach to network reconfiguration for loss minimization in distribution networks," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 4, pp. 247–258, 1998.

[23] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on power Delivery*, vol. 4, no. 1, pp. 725–734, 1989.

[24] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, vol. 24. Springer Science & Business Media, 2002.

[25] R. K. Martin, "A sharp polynomial size linear programming formulation of the minimum spanning tree problem," *Graduate School of Business, University of Chicago, Chicago, IL*, 1986.

[26] S. Raghavan, *Formulations and algorithms for network design problems with connectivity requirements*. PhD thesis, Massachusetts Institute of Technology, 1994.

[27] S. Mittal and A. S. Schulz, "An FPTAS for optimizing a class of low-rank functions over a polytope," *Mathematical Programming*, pp. 1–18, 2013.

[28] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the 41st Annual ACM symposium on Theory of Computing*, pp. 323–332, ACM, 2009.

[29] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2011.

[30] V. P. Il'ev, "An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function," *Discrete Applied Mathematics*, vol. 114, no. 1, pp. 131–146, 2001.

[31] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," in *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1134–1148, Society for Industrial and Applied Mathematics, 2015.