Theory of Distribution Network Evaluation Tool

Takeru Inoue

Aim of this paper. This paper describes the *theory* of DNET (Distribution Network Evaluation Tool) in detail. We aim to provide several analytical methods in DNET but the current version just support power loss minimization, and so this paper also describes the theory of loss minimization only. Since this paper describes assumptions, algorithms, and implementation issues in detail, it is a bit long as you see; the readers will find the shorter paper that summarizes our basic idea [12]. This paper does not address the usage of DNET at all; the readers will find the usage in the README file of the DNET package.

DNET was implemented by Takeru Inoue, but its theory was developed with many people, as seen in [12]. I would like to list their names to appreciate the contributions: Keiji Takano, Takayuki Watanabe, Jun Kawahara, Ryo Yoshinaka, Akihiro Kishimoto, Koji Tsuda, Shin-ichi Minato, and Yasuhiro Hayashi.

Abstract

Network operation requires a fast and reliable reconfiguration method to reduce resistive line losses in coordination with smart metering systems. The loss minimization is, unfortunately, quite a complex optimization problem, and so the research community has failed to develop any reconfiguration method that works in large-scale networks with guaranteed optimality. This paper proposes a novel method that yields tight bounds on the minimum loss without losing the scalability. We first derive relaxed problems to bound the minimum loss with a feasible configuration, and then devise several algorithms that find the bounds efficiently. These algorithms rely on zero-suppressed binary decision diagrams (ZDDs), which can handle a huge number of feasible configurations in a compressed manner. We finally conduct thorough tests with promising results. A brief discussion on distributed generators is also provided.

1 Introduction

Distribution networks, which consist of several feeders with a number of switches, are generally operated in a radial structure under the operational constraints such as line capacity and voltage profiles. The network structure can be *reconfigured* by changing the open/closed status of the switches to reduce resistive line losses. In a more constrained energy environment coming in the near future, operators would be more interested in the loss minimization and consequently in the network reconfiguration process enhancement.

Table 1: Comparison between existing methods and our method

Optimization method	Quality of solutions	Scalability (# of switches)
Heuristics and metaheuristics	"Good" solution without bound	Thousands
Our method	Relaxed solution with tight bounds	Hundreds
Brute-force method	Optimal solution	Tens

The "smart grid" is now gathering much attention to overcome the energy problem; distribution automation and advanced metering infrastructure are being installed in distribution management systems [29], and more switches will be introduced for the fine grain operation. However, efficient loss minimization methods, which are required to reconfigure the grid with many switches in coordination with the smart metering systems, are still an imperative piece.

The loss minimization is a highly complex combinatorial, non-differentiable, and non-linear optimization problem, due to the large number of variables (switches in a network), and due to the non-linear characteristics of the constraints used to model the electrical behavior of the system [28, 6]. Moreover, the astronomical number of solutions (feasible configurations) involved in the problem makes it very hard to solve. In recent years, several optimization methods have been studied for this problem. Most of them rely on approximate techniques like heuristics [19, 8, 4, 17, 28, 16] or metaheuristics [7, 23, 13, 11, 10, 6], in order to reduce the computational complexity. These methods can scale well with a large distribution network, but do not guarantee the optimality nor give any bound on the optimal value. The brute force method proposed in [22] guarantees the optimality, while it does not scale in a network of practical size, which includes several hundred switches [28, 18, 13]. Optimization methods should guarantee the optimality or provide tight bounds without losing the scalability.

This paper proposes a novel optimization method to establish tight bounds on the minimum loss value at given load demands under the constraints of radiality, line capacity, and voltage profiles. Our contributions are summarized as follows.

- We relax the loss minimization problem under appropriate modeling assumptions. The relaxed problems give a lower bound as well as an upper bound with a feasible configuration. (Section 3)
- We also propose a solution method for the relaxed problems. The method scales well by introducing zero-suppressed binary decision diagrams (ZDDs) [20, 15], which are used to efficiently handle the huge number of configurations. (Section 5)
- Our method is tested with practical distribution networks, and we show that it provides tight bounds with acceptable scalability. (Section 6)

Table 2: Symbols for distribution networks

$D = (C, S)$	A distribution network
$M_i = (C_i, S_i)$	An independent component of the network
$\mathcal{M} = \{M_1, \ldots\}$	A set of independent components
$C = \{c_1, \ldots\}$	A set of sections
C_0	A set of root sections
C_i	A set of sections in component M_i ($i \geq 1$)
C^{ex}	A set of sections including extended sections
C_i^{up}, C_i^{down}	Sets of up/down-stream sections for section c_i
$S = \{s_1, \ldots\}$	A set of switches
S_i	A set of switches in component M_i
$X \subset S$	A configuration (a set of closed switches)

Table 3: Symbols for power flows

I_i	Load current of section c_i
$Z_i = R_i + jX_i$	Impedance (resistance and reactance) of section c_i
J_i	Line current of section c_i
$J_i^{\max}$	Line capacity of section c_i
V_i	Voltage magnitude at the sending end of section c_i
V_0	Sending voltage at the transformer
$V^{\min}$	Lower limit of voltage magnitude
P_i	Power loss on section c_i
P	Total power loss in the network

The features of the proposed method are compared with those of the existing methods in Table 1.

The rest of this paper is organized as follows. Section 2 describes the loss minimization problem, and Section 3 introduces our relaxed problems. After briefly reviewing ZDDs in Section 4, Section 5 proposes the solution method for the relaxed problems. Section 6 tests our method and shows the results. Section 7 summarizes related work, and finally Section 8 concludes this paper with discussion about distributed generators in our method.

2 Loss Minimization Problem

This section formulates the loss minimization problem. Symbols used for distribution networks and those for power flows are presented in Tables 2 and 3, respectively.

We first describe our network model. We consider three-phase alternating current systems, but distribution networks are represented on a per-phase basis

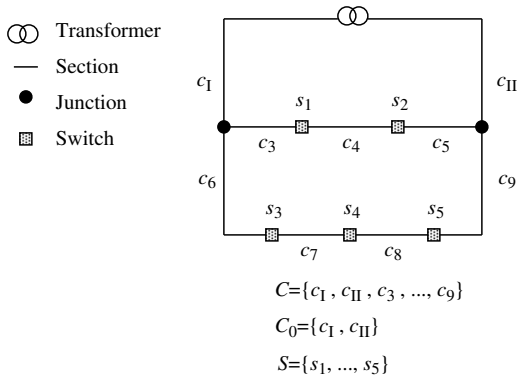

Figure 1: An example of a distribution network.

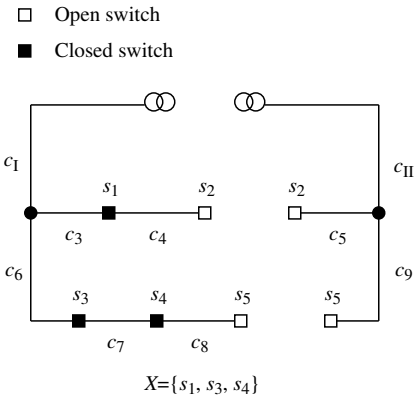

Figure 2: An example of a configuration in the network of Fig. 1.

in this paper for simplicity. Sectionalizing and tie switches are not distinguished, and they are just called switches. Each part of a distribution feeder with load separated by switches or junctions is called a section. Sections that are directly connected to a substation are called root sections; root sections are energized in any configuration, because no switch intervenes between a root section and a transformer in a substation. Given a status of all switches, a network configuration is uniquely determined. A configuration following a radial structure consists of *tree*-like feeders, and each feeder is rooted at a root section. For a section on a feeder, sections on the same feeder is separated into upstream sections and downstream sections.

Figure 1 shows an example of a distribution network, $D = (C, S)$, which has nine sections, $C = \{c_{\mathrm{I}}, c_{\mathrm{II}}, c_3, \ldots, c_9\}$, and five switches, $S = \{s_1, \ldots, s_5\}$. The set of root sections, C_0, includes sections c_{I} and c_{II} (we identify root sections by Roman numerals). Figure 2 shows a configuration of this network. In this paper,

we represent a configuration by a set of closed switches; the configuration in Fig. 2 is given by $X = \{s_1, s_3, s_4\}$. This configuration has two radial feeders rooted at the root sections; one feeder consists of sections c_{I}, c_3, c_4, c_6, c_7, and c_8, and the other includes sections c_{II}, c_5, and c_9. Sections on a same feeder are grouped into upstream and downstream; e.g., for section c_{I} on the feeder rooted at c_{I}, sections are grouped into upstream $c_{\mathrm{I}}^{\mathrm{up}} = \{\}$ and downstream $c_{\mathrm{I}}^{\mathrm{down}} = \{c_3, c_4, c_6, c_7, c_8\}$ (c_{I} is included in neither upstream nor downstream), and for section c_7 on the same feeder, sections are grouped into upstream $C_7^{\mathrm{up}} = \{c_{\mathrm{I}}, c_6\}$ and downstream $C_7^{\mathrm{down}} = \{c_8\}$ (sections on another branch, c_3 and c_4, are included in neither upstream nor downstream sets).

We next discuss power flow. We assume that section load is represented as constant current [11] and it is uniformly distributed on the section. If the load is given in power, not current, load current is estimated by dividing the load power by the sending line voltage. Line current of section c_i is given by sweeping backward to sum up downstream section loads [23], as follows,

$$J_i = \sum_{c_j \in C_i^{\mathrm{down}}} I_j + I_i. \tag{1}$$

Voltage magnitude at the sending end of c_i is given by,

$$V_i = V_0 - \sum_{c_j \in C_i^{\mathrm{up}} \cup c_i} Z_j \left[\sum_{c_k \in C_j^{\mathrm{down}}} I_k + \frac{I_j}{2} \right]. \tag{2}$$

Active power loss on c_i is given by,

$$P_i = R_i |J_i|^2. \tag{3}$$

Finally, the loss minimization problem is formulated as follows.

$$\begin{aligned} \text{minimize} \quad & P = \sum_{c_i \in C} P_i, && (4) \\ \text{subject to} \quad & X \text{ is a spanning rooted forest,} && (5) \\ & J_i \le J_i^{\max} \text{ and } V_i \ge V^{\min}, \forall c_i \in C. && (6) \end{aligned}$$

We explain the variable, objective function, and constraints below.

The variable in this problem is the open/closed status of switches. We denote the status by a set of closed switches, X, as noted above. Given a status of all switches, the network configuration is uniquely determined, and then we are allowed to evaluate the objective function (4) and the constraints (5) and (6).

The objective of this problem is to minimize the total loss in the network, which is given by (4).

The first constraint given by (5) is *topological constraints*, which include the network radiality and the load connectivity. The radiality is satisfied if the network includes no loop; this implies the network forms a forest, that is a set of disjoint trees. The load connectivity is satisfied if the forest is spanning all

Table 4: Accents

$x^\star$	Optimal value or optimal solution in the original problem
$\overline{x}$	Upper bound, or relaxed solution for upper bound
$\underline{x}$	Lower bound, or relaxed solution for lower bound

sections and every tree is rooted at a root section (i.e., all loads are energized from a substation). We call such a forest a *spanning rooted forest* in this paper. The network configuration must be a spanning rooted forest to satisfy the topological constraints. The configuration in Fig. 2 is an example of the spanning rooted forest. Since every load is energized through a root section for the load connectivity, we have the following identity,

$$\sum_{c_i \in C_0} J_i = \sum_{c_j \in C} I_j. \tag{7}$$

The second constraints given by (6) are *electrical constraints*, which include line capacity and voltage profiles. For all sections in the network, the line current must not exceed the capacity and the voltage magnitude must be equal to or greater than the lower limit.

We solve this problem, given a distribution network $D = (C, S)$, and given load profile I_i and impedance Z_i for all sections $c_i \in C$.

3 Relaxed Problems

This section introduces the relaxed problems to derive upper and lower bounds on the minimum loss. Section 3.1 discusses our assumptions under which the relaxed problems will be discussed. Sections 3.2 and 3.3 describe the relaxed problems for the upper bound and the lower bound, respectively. Accents used in this paper are presented in Table 4.

3.1 Assumptions

Section load as constant current. As shown in (1), we assume section loads are given as constant current. This assumption plays a key role in our optimization method; we will divide objective function (4) into several term groups called *independent components*, as discussed in Section 3.2. We also assume that load current must not be negative (we will briefly discuss negative loads to represent distributed generators).

No root switch. We assume that every switch above a *root junction* is closed in the minimum loss configuration; a root junction is a junction that comes earlier than any other junction from a transformer. In other words, such switches are ignored in the loss minimization process. We then redefine a root section as a section between a root junction and a transformer, since such a section is

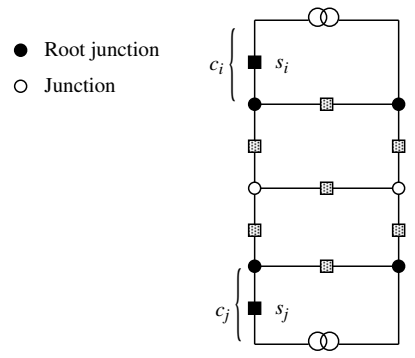

Figure 3: An example of root junctions.

energized in any configuration under this assumption, same as the original definition (it is worth noting that the neighbors of root sections are also energized and so we will extend root sections in Section 5.1).

In Fig. 3, switches s_i and s_j, which are above the root junctions, are assumed to be closed. Sections c_i and c_j, which include these closed switches, are root sections in the new definition. In Fig. 1, the two junctions are all root junctions.

This assumption is reasonable in the loss minimization problem as follows. First, a section in the upstream has larger line current than another section in the downstream, as shown in (1). In addition, since power loss is proportional to the square of line current as in (3), the total loss in the network strongly depends on the line current of upstream sections. Not in order to increase the line current of any upstream sections, the loads have to be assigned to each feeder fairly. For such fair load balancing, upstream switches are unlikely to be open; if an upstream switch is open, neighbor feeders have to energize larger areas, which increases the total loss.

Independent component. Following this assumption, we partition a distribution network into small *independent components*; given a configuration of an independent component, the power loss in the component can be determined independently with other components. These components are obtained by dividing a network at root junctions. Since all components are disjoint, we have $C_i \cap C_j = \emptyset$ and $S_i \cap S_j = \emptyset$, for $M_i, M_j \in \mathcal{M}$ and $i \neq j$. Every section other than root sections is included in a component, that is $\bigcap_{M_i \in \mathcal{M}} C_i = C \setminus C_0$. Every switch is also included in a component, and we have $\bigcap_{M_i \in \mathcal{M}} S_i = S$ (we ignore switches on root sections as noted above).

Figure 4 shows an example of independent components. The network has two independent components; one includes sections c_3, c_4, and c_5, and switches s_1 and s_2; the other includes sections c_6, c_7, c_8, and c_9, and switches s_3, s_4, and s_5. Root sections, c_{I} and c_{II}, are not included in any component.

We are now allowed to determine the power loss in a component independently. Since a component is connected to its outside just at root junctions, the power comes into the component just from the root sections. The power

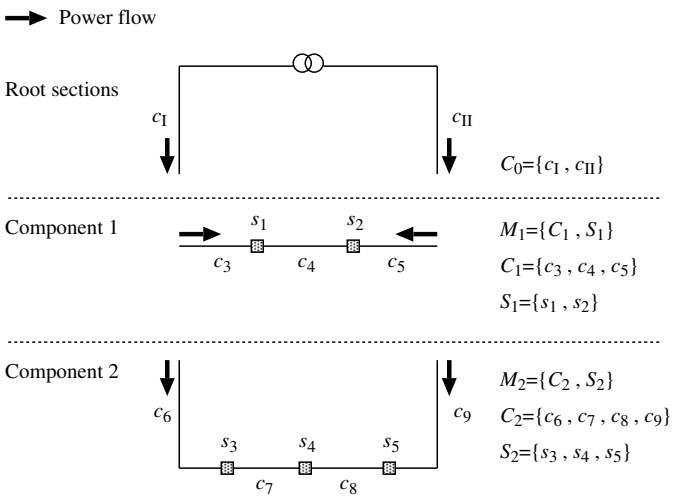

Figure 4: An example of independent components in the network of Fig. 1.

never goes out to other components so as not to create a loop (we will briefly discuss big distributed generators that provide the power beyond the border of components in Section 8). Therefore, given a configuration in a component, the line current in the component is calculated by (1), and the power loss is also determined by (3). Since a component has much fewer configurations than the whole network does (at least we assume so), it is easier to examine all of them.

3.2 Upper Bound

The objective function given by (4) can be divided into losses in root sections and those in components,

$$P = \sum_{c_i \in C_0} P_i + \sum_{M_j \in \mathcal{M}} \sum_{c_k \in C_j} P_k.$$

We ignore the terms of root sections, and relax the original problem as follows,

$$\begin{aligned} \text{minimize} \quad & \sum_{M_j \in \mathcal{M}} \sum_{c_k \in C_j} P_k, \\ \text{subject to} \quad & \text{all constraints, (5) and (6).} \end{aligned} \tag{8}$$

We will solve this relaxed problem by utilizing the component independency in Section 5 (i.e., losses in component M_j, that is $\sum_{c_k \in C_j} P_k$, can be determined independently).

We call the optimal solution of this relaxed problem the *relaxed solution*, and denote it by $\overline{X}$. The relaxed solution is feasible in the original problem as well, because the both problems have the same constraints. The relaxed solution,

therefore, gives the upper bound on the minimum loss of the whole network, as follows,

$$\overline{P} = \sum_{c_i \in C} P_i(\overline{X}) \geq P^\star. \tag{9}$$

It is worth noting that operators can implement this feasible configuration, $\overline{X}$, in their networks.

This relaxation, which ignores losses on root sections, may seem too radical, because root sections yield large losses as discussed in Section 3.1. We, however, believe that the relaxed problem gives a *tight* upper bound, because the line current of a root section is determined by the neighbor sections whose losses are minimized in the relaxed problem. This issue will be discussed with the tests in Section 6.3.

3.3 Lower Bound

We modify the original objective function (4) just to include the root terms, and replace the constraints with the identity (7). The relaxed problem is then given by,

$$\begin{aligned} &\text{minimize} && \sum_{c_i \in C_0} P_i, \\ &\text{subject to} && \text{identity (7)}. \end{aligned} \tag{10}$$

This relaxed problem gives the lower bound of root sections. We analytically solve this problem for J_i of $c_i \in C_0$, and find the relaxed solution,

$$\underline{J_i} = \frac{\sum_{c_j \in C} I_j}{R_i \sum_{c_j \in c_0} 1/R_j}.$$

This relaxed solution is just used to calculate the lower bound, and cannot realized by any actual configuration in general.

The relaxed solution of this subsection gives the lower bound for root sections, while the relaxed solution of the previous problem (8) minimizes the losses of all components. Therefore, the sum of objective values in the both relaxed problems gives the lower bound of the whole network,

$$\underline{P} = \sum_{c_i \in C_0} P_i(\underline{J_i}) + \sum_{M_j \in \mathcal{M}} \sum_{c_k \in C_j} P_k(\overline{X}) \leq P^\star. \tag{11}$$

4 Zero-suppressed Binary Decision Diagrams

This subsection reviews zero-suppressed binary decision diagrams, or ZDDs in short. Symbols used for them are presented in Table 5.

A ZDD is a data structure that efficiently represents a *family* of sets; a family means a *set* of sets in this paper, and it is denoted by a calligraphic

Table 5: Symbols for ZDDs

$\mathcal{Z} = \{X_1, \ldots\}$	A ZDD representing a family of configurations
$N = \{n_0, \ldots\}$	A set of ZDD nodes, which is equivalent to a ZDD
n_0	The root node in a ZDD

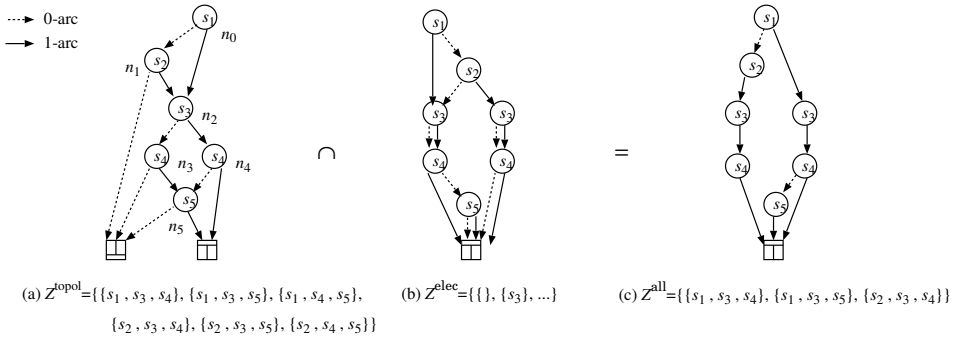

Figure 5: An example of ZDDs and the intersection-operation over (a) and (b).

symbol like $\mathcal{Z}$. A ZDD represents a family of closed-switch sets in this paper, since a configuration is denoted by a set of corresponding closed switches as noted in Section 2.

A ZDD is defined as a labeled directed acyclic graph that satisfies the following properties. There is only one node with indegree 0, which is called the root node of ZDD. Each node has just 2 outgoing arcs, which are labeled by 0 and 1; a node pointed by the i-arc of node n is called i-child of n. Each node is labeled by an item of a universal set, $S = \{s_1, \ldots\}$, which is totally ordered as $s_i \leq s_j$ if $i \leq j$. Nodes are arranged in the ascending order of the labeled items from the root node. There are just two terminals $\bot$ and $\top$, and a path from the root node to $\top$-terminal represents a feasible set of items; each set includes items at which the 1-arc is taken.

Figure 5 (a) shows a ZDD, $\mathcal{Z}^{\mathrm{topol}}$, that represents a family of topologically feasible configurations in the network of Fig. 1. The ZDD requires six nodes, $N = \{n_0, \ldots, n_5\}$, to represent the family, and these nodes are labeled by the switches. The ZDD represents six feasible configurations, as shown in the figure; e.g., a path of n_0, n_2, n_4, n_5, and $\top$, represents a configuration of $\{s_1, s_3, s_5\}$. We see some parent-child relations in the figure; e.g., n_1 is 0-child of n_0. In the rest of this paper, we omit $\bot$-terminal and the corresponding arcs.

A ZDD is said to be *reduced* if the following two properties are satisfied; there is no distinct nodes that have the same label, 0-child, and 1-child; there is no node whose 1-child is $\bot$-terminal. A reduced ZDD can represent a family of sets in a compressed manner, since a part of ZDD is shared among some sets in the family. The size of reduced ZDD, $|N|$, can be logarithmically smaller

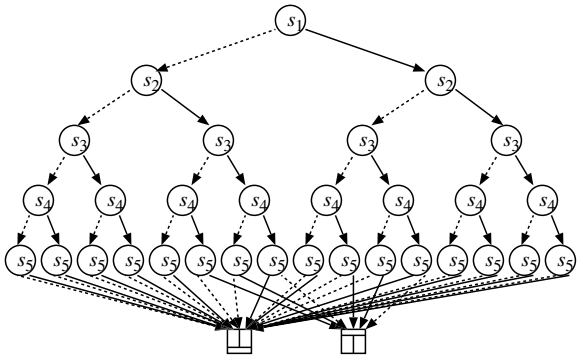

Figure 6: An example of an unreduced ZDD of Fig. 5 (a).

than the number of sets in it. Figure 5 (a) shows a reduced ZDD whose size is $|N| = 6$, while Figure 6 is an unreduced ZDD that represents the same family with Fig. 5 (a).

Several algorithms are defined on ZDDs mainly for performing set operations. Since the complexities depend on the size of the ZDDs other than the number of sets in them, the operations can run very efficiently. We will use the following algorithms in this paper.

- Intersection. Given two ZDDs that represent families $\mathcal{Z}_0$ and $\mathcal{Z}_1$, the intersection is defined by $\mathcal{Z}_0 \cap \mathcal{Z}_1 = \{X | X \in \mathcal{Z}_0 \text{ and } X \in \mathcal{Z}_1\}$ [15, p.141, exercise 203].
- Union. Given two ZDDs that represent families $\mathcal{Z}_0$ and $\mathcal{Z}_1$, the union is defined by $\mathcal{Z}_0 \cup \mathcal{Z}_1 = \{X | X \in \mathcal{Z}_0 \text{ or } X \in \mathcal{Z}_1\}$ [15, p.141, exercise 203].
- Random sampling. Given a ZDD that represents a family of sets, $\mathcal{Z}$, we can uniformly randomly choose a set from the family [15, p.76].
- Count. Given a ZDD that represents a family of sets, $\mathcal{Z}$, we can count the number of sets in the family, [15, p.75, algorithm C].
- Reduce. Given a ZDD, it is reduced [15, pp.84–85, algorithm R].

Binary decision diagrams including ZDDs have been used to find an optimal solution in a linear combinatorial problem [5]. A binary decision diagram is built for each constraint separately, and then Intersection is applied to the diagrams to get solutions satisfying all the constraints. In this paper, we also build a ZDD separately and then use Intersection in the same manner. Figure 5 shows an example; two ZDDs, $\mathcal{Z}^{\text{topol}}$ and $\mathcal{Z}^{\text{elec}}$, are built for the topological and electrical constraints, and the intersection over them yields a ZDD of all the constraints, that is denoted by $\mathcal{Z}^{\text{all}}$. However, the loss minimization is *non*-linear combinatorial problem, and so we need to develop another solution method that can be applied to this problem.

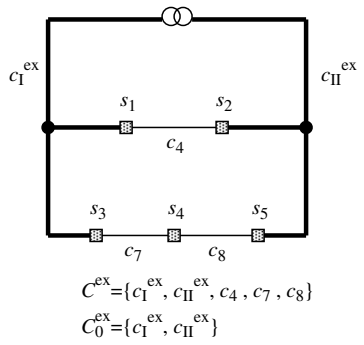

Figure 7: An example of extended sections (thick lines) in the network of Fig. 1.

5 Solution Method for Relaxed Problems

This section describes a solution method that finds the relaxed solution for the upper bound, $\overline{X}$, which is also required to determine the lower bound. Section 5.1 offers preliminaries. Section 5.2 builds a ZDD representing topologically feasible configurations, and Section 5.3 builds another ZDD for the electric constraints. Section 5.4 calculates the intersection of these ZDDs, and Section 5.5 reconstructs it to search for the relaxed solution.

5.1 Preliminaries

We define a total order between switches, which is required to build ZDDs. Since we will calculate power loss for each component in Section 5.5, switches in the same component must be ordered in a sequence. In addition, switches should be ordered based on the proximity in the network, since it is well known that the ZDD size is likely to be small if correlated items (switches) are put closely on the ZDD. To meet these requirements, components are numbered in the breadth-first order, and then switches in each component are also numbered in the breadth-first order. Figure 4 gives an example of this ordering.

We next define a set of sections connected through no switch as an *extended section*. Figure 7 shows an example of extended sections, $c_{\mathrm{I}}^{\mathrm{ex}}$ and $c_{\mathrm{II}}^{\mathrm{ex}}$; e.g., extended section $c_{\mathrm{I}}^{\mathrm{ex}}$ consists of sections c_{I}, c_3, and c_6. The set of sections including extended ones, C^{ex}, is also shown in the figure. Extended sections will be used for the topological constraints, because we do not need to distinguish sections in a same extended section when considering network topologies. We assume that Roman and Arabic numerals are comparable, and also assume that root extended sections have smaller numbers (subscripts) than others; i.e., $i < j$ for $c_i \in C_0^{\mathrm{ex}}$ and $c_j \in C^{\mathrm{ex}} \setminus C_0^{\mathrm{ex}}$.

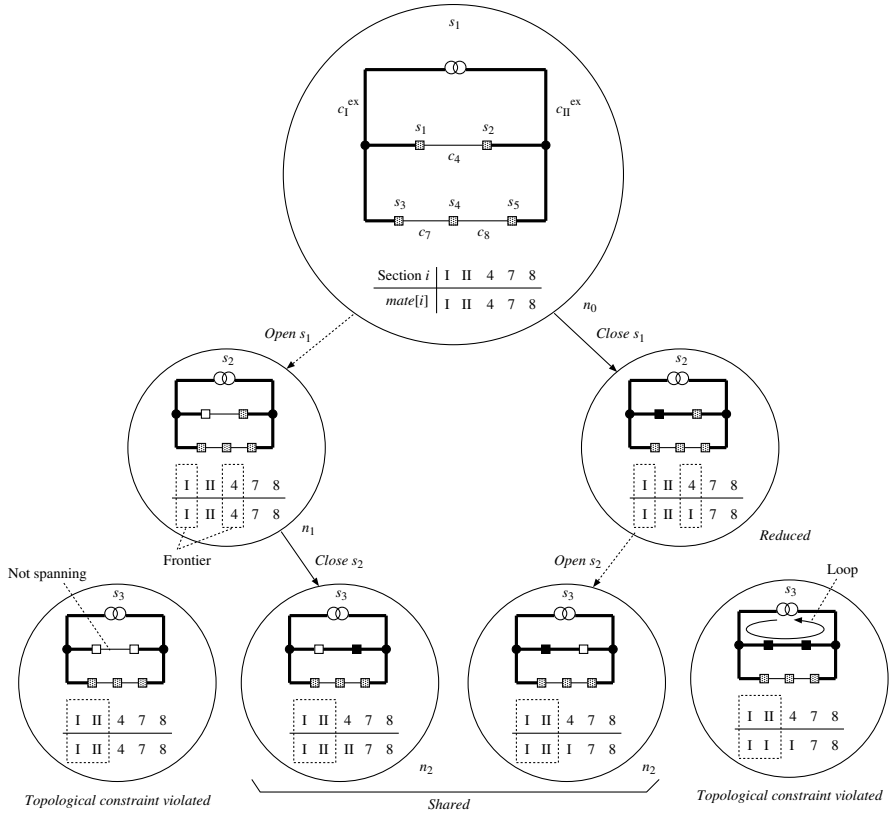

Figure 8: An example of algorithm BUILDT that builds ZDD $\mathcal{Z}^{\mathrm{topol}}$ for the network in Fig. 1; sections represented by thick lines indicate they are energized.

5.2 ZDD for Topological Constraints

This subsection builds a ZDD, $\mathcal{Z}^{\mathrm{topol}}$, that represents a family of configurations satisfying the topological constraints. Each of these configurations must be a spanning rooted forest, as discussed in Section 2. In order to find all such forests efficiently, we extend the algorithm proposed in [15, p.143, exercise 225]; the algorithm finds all paths between a given pair of vertices on a graph, and builds a ZDD to represent the paths.

We describe an outline of the extended algorithm named BUILDT in Fig. 8. This figure shows a process of building the ZDD of Fig. 5 (a). In this algorithm, a ZDD is built in a top-down manner. First, only the root node, n_0, is given; no switch has fixed its status here. Given a status of the first switch, s_1, open or closed, two nodes are created at the level of s_2. Similarly, s_2 chooses its status, which yields four nodes at the level of s_3. The leftmost node and the rightmost node in this level are removed, because they violate the topological constraints. The two center nodes in the same level are shared, since their state

are *equivalent* in the following building process. In this way, BuildT will build $\mathcal{Z}^{\text{topol}}$, as shown in Fig. 5 (a).

In order to efficiently validate the constraints and to quickly check the node equivalency, we introduce an array named *mate*[1] for each node, and we define a set of sections called *frontier* for each level.

- **Mate** of node n, $n.mate[i]$, indicates the status of a section, $c_i \in C^{\text{ex}}$, in the configuration represented by node n. The section status is defined as follows.
  - If $n.mate[i]$ is an Arabic numeral, section c_i is not included in any rooted tree (c_i is not energized).
  - If $n.mate[i]$ is a Roman numeral, section c_i is included in the tree rooted at section of $n.mate[i]$ (c_i is energized).
  - If $n.mate[i] = n.mate[j]$, sections c_i and c_j are included in a same tree (are electrically connected), whether the tree is rooted or not (they are energized or not).
- **Frontier**, F, is a set of sections on the border between fixed switches and not-yet-fixed ones. We are allowed to focus on sections just in the frontier when evaluating the constraints and the equivalency. This is because sections that had left the frontier have already determined their status, and because sections that have not come to the frontier cannot determine their status yet. The equivalency between nodes n_i and n_j is denoted by $n_i.mate \equiv n_j.mate$, and it is actually checked like this; $n_i.mate[k] = n_j.mate[k], \forall c_k \in F'$, where F' is frontier sections in the next level. Similarly, the constraint validation is also done based on the frontier, as will be discussed later.

Figure 8 offers the mates and the frontiers; e.g., for the root node, n_0, it has the mate of $mate[i] = i$ for $i \in \{\text{I}, \text{II}, 4, 7, 8\}$, and its next level frontier is $F' = \{c_I^{\text{ex}}, c_4\}$ as presented at node n_1.

We present the complete algorithm of BuildT in Algorithm 1. BuildT first initializes ZDD nodes in Lines 1–3. The algorithm then creates nodes for each level in Lines 4–17 [2]. Finally, Algorithm Reduce is called to reduce the ZDD at the end. We describe the node creation process in detail with the associated algorithms.

Algorithm CloseS, which is called at Line 8 in BuildT, is given in Algorithm 2. When closing switch s, the algorithm validates the topological constraints for given node n, and updates the mate. The validation is executed as follows; if s's both ends are connected electrically (the mates are equal), or they are energized (the mates are Roman numerals), a loop is created by closing s.

[1] The array name, *mate*, comes from that the array is originally used to maintain path ends in [15]. We did not change the name in this paper, though we focus on forests, not paths.

[2] We assume that a child is $\perp$-terminal until any node will be assigned at Line 10. We also assume that in the statement, if A or B, at Line 8, B is never evaluated when A is true, like C language.

Algorithm 1 BUILDT

```
1: n_0.mate[i] ← i for each c_i ∈ C^ex
2: Z^topol ← {n_0}
3: N ← {n_0}, N' ← ∅ // ZDD nodes at current/next levels
4: for each s_l ∈ S do // level of s_l
5:    for each n_i ∈ N do // n_i is node at current level
6:       for each x ∈ {0,1} do // n_i's 0/1-child
7:          // n_j is new node at next level
            n_j.label ← s_{l+1}, n_j.mate ← n_i.mate
8:          // If valid topology
            if (x = 0 or CLOSES(n_j, s_l)) and LEAVEF(n_j) then
9:             // Find equiv node from N' to share, or use new n_j
               n_k ← (∃n ∈ N' s.t. n.mate ≡ n_j.mate) ? n : n_j
10:            n_i.x_child ← n_k // Set n's x-child
11:            N' ← N' ∪ {n_k}
12:         end if
13:      end for
14:   end for
15:   Z^topol ← Z^topol ∪ {N'}
16:   N ← N', N' ← ∅
17: end for
18: REDUCE(Z^topol)
```

For example, at the right node in the second level of Fig. 8, if s_2 were closed, a loop would be created since the both ends had been energized. If these tests have been passed, we update the mates of frontier sections by the smaller value.

Algorithm LEAVEF is called with node n to check the constraints for sections leaving the frontier. If the section is not energized (the section's mate is Arabic) and the section is not connected to the frontier (no section has the same mate value), then the constraints are violated; e.g., at the left bottom node in Fig. 8, section c_4 is no longer energized.

At Line 9 in Algorithm 1, BUILDT searches for an existing node equivalent to new node n_j from the node set of next level, N'. If the equivalent node is found, it is set to n's child to be shared, same as the center nodes in Fig. 8. Otherwise, the new node, n_j, is set.

Algorithm BUILDT and the associated algorithms are fast and space-efficient, because they process each switch just once and they rely on the small mate array in the constraint validation.

5.3 ZDD for Electrical Constraints

This subsection describes several algorithms that build a ZDD, $\mathcal{Z}^{\text{elec}}$; the ZDD represents a family of configurations satisfying the electrical constraints. Figure 9 shows an outline of the algorithm called BUILDE. This algorithm first builds a ZDD of *feasible feeders* for each root section by using another algorithm named ENUMF; a feasible feeder is a partial network configuration that forms a single

Algorithm 2 CLOSES(n, s)

```
1: c_i, c_j ← s.both_ends // s's both end sections
2: if n.mate[i] = n.mate[j] or
     n.mate[i] and n.mate[j] are Romans then
3:   return false // Constraints violated
4: end if
5: for each c_k ∈ F' do // Update mate just on frontier
6:   n.mate[k] ← Min(n.mate[i], n.mate[j])
       if n.mate[k] = Max(n.mate[i], n.mate[j])
7: end for
8: return true // Constraints not violated
```

Algorithm 3 LEAVEF(n)

```
1: for each c_i ∈ F \ F' do
2:   if n.mate[i] is Arabic and
       ∀c_j ∈ F' s.t. n.mate[i] ≠ n.mate[j] then
3:     return false // Constraints violated
4:   end if
5: end for
6: return true // Constraints not violated
```

tree satisfying the electrical constraints. BUILDE, then, gives $\mathcal{Z}^{\text{elec}}$ by calculating the intersection of these feeder sets for all root sections. We present details of this algorithm in Algorithm 4.

We next describe an outline of algorithm ENUMF to build a ZDD of feasible feeders for a given root section. Figure 10 shows the algorithm running with root section c_2^{ex}. This algorithm enumerates feasible feeders by determining switch status one by one. It begins with the smallest feeder, as shown at Fig. 10 (a). A new feeder is created by closing a neighbor switch of the feeder (Fig. 10 (c)), while the same feeder remains by opening the switch (Fig. 10 (b)). If the new feeder is feasible, a ZDD is built for it and another neighbor switch is processed recursively. If not feasible, the enumeration process stops. Finally, the union of these ZDDs are calculated, as seen at the second term of Fig. 9, because $\mathcal{Z}^{\text{elec}}$ must include one of the enumerated feeders.

We discuss details of ENUMF. Since ENUMF is called with algorithm BORDER as shown in Algorithm 4, we begin with this algorithm.

$$\text{BORDER}(c_0) = \{s \in S : \text{IsNeighbor}(s, c_i), c_i \in C_0^{\text{ex}} \setminus c_0\}.$$

The argument, c_0, is a root section at which feeders to be enumerated are rooted. This algorithm finds neighbor switches of root sections other than c_0. These switches comprise the *border* of feasible feeders rooted at c_0; the feeders cannot beyond the border so as not to create a loop. In Fig. 10, since BORDER is called with c_2^{ex}, the border consists of s_1 and s_3, which are neighbor switches of c_1^{ex}.

We present the complete algorithm of ENUMF in Algorithm 5. ENUMF

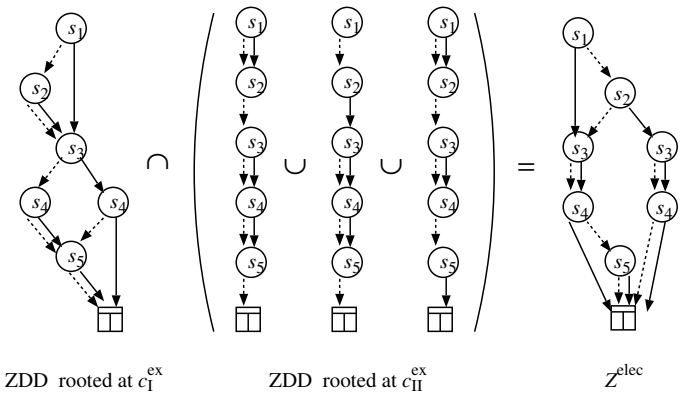

Figure 9: An example of algorithm BUILDE to build ZDD $\mathcal{Z}^{\mathrm{elec}}$ for the network of Fig. 1.

Algorithm 4 BUILDE

1: $\mathcal{Z}^{\mathrm{elec}} \leftarrow U$ // U is the universal set
2: **for each** $c \in C_0^{\mathrm{ex}}$ **do**
3: $\quad \mathcal{Z}^{\mathrm{elec}} \leftarrow \mathcal{Z}^{\mathrm{elec}} \cap$ ENUMF($\emptyset$, BORDER(c))
4: **end for**

maintains a feeder configuration, X, which is represented by a set of closed switches. It also maintains a set of switches, S', that will not to be processed. If a neighbor switch of the current feeder, s, is found at Line 2, we go on the following process. First, s is added to switch set S'. ENUMF is then recursively called by opening s (called with the same feeder configuration, X). We also obtain the new feeder configuration, that is $X \cup s$, by closing s. If the feeder has a tree structure and it satisfies the electrical constraints, we build a ZDD for the new feeder and call ENUMF with the feeder recursively. We present the feeder configuration, X, and the switch set, S', in Fig. 10.

Algorithm SATISFIESE ensures whether all sections in a given feeder, X, satisfy the electrical constraints of (6).

$$\textsc{SatisfiesE}(X) = \neg\exists c_i \in X.\mathit{sections} \text{ s.t. } J_i(X) > J_i^{\max}, V_i(X) < V^{\min}$$

As shown in (1) and (2), line current and voltage magnitude are *monotonic* with respect to section adding; i.e., the electrical constraints cannot be re-satisfied by adding another section to the feeder, once the constraints have been unsatisfied. This is the reason why the enumeration process can stop when the constraints are violated.

BUILDF, which is given in Algorithm 6, builds a ZDD representing a single feeder, X. The ZDD is built in a top-down manner. If a switch, s_k, is not found in feeder configuration X, the switch can be open and 0-child is set. If the switch is not a neighbor of feeder X, the switch can be closed and 1-child is

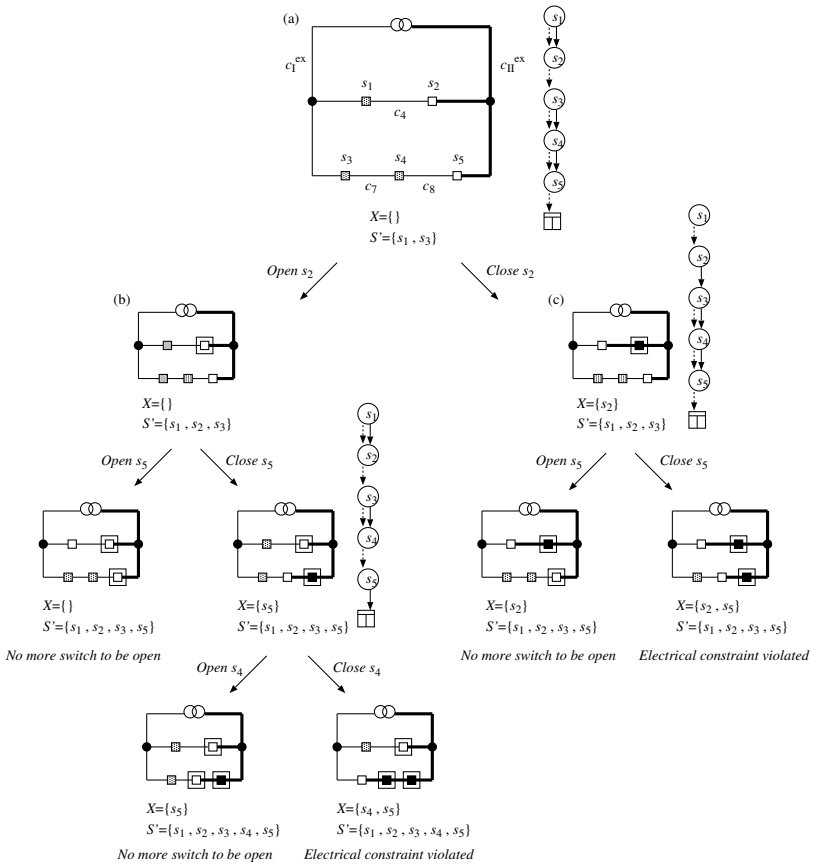

Figure 10: An example of algorithm EnumF for feeders rooted at c_2^{ex} in the network of Fig. 1; sections represented by thick lines are energized and constitute a feeder, and switches of double square indicate their statuses have been determined.

set. We illustrate this algorithm using Fig. 10 (c). Since s_1 is not found in X, 0-child is set. Since s_2 is found in X but not a neighbor, 1-child is set. Since s_3 is not found in X and not a neighbor, both 0-child and 1-child are set.

Configurations included in $\mathcal{Z}^{\mathrm{elec}}$ are radial (a set of trees), but do not care the load connectivity. This omission is rectified by calculating the intersection with $\mathcal{Z}^{\mathrm{topol}}$ in Section 5.4.

BuildE and its associated algorithms are quite efficient, since EnumF enumerates feasible configurations without duplication. Moreover, BuildE can be easily parallelized for each root section, because it runs independently with other root sections.

Algorithm 5 EnumF(X, S')

```
1: 𝒵 ← ∅ // ZDD for feeders
2: if s ∈ X.neighbor_switches \ S' then
3:    S' ← S' ∪ {s} // s is not needed to be processed more
4:    𝒵 ← 𝒵∪ EnumF(X, S') // Open s and call recursively
5:    X ← X ∪ {s} // Close s
6:    if IsTree(X) and SatisfiesE(X) then
7:       𝒵 ← 𝒵∪ BuildF(X)∪ EnumF(X, S')
8:    end if
9: end if
10: return 𝒵
```

Algorithm 6 BuildF(X)

```
1: N ← ∅ // Set of nodes, namely, ZDD for feeder X
2: n_i ← n_0
3: for each s_k ∈ S do
4:    n_j.label ← s_{k+1} // n_j is new node
5:    n_i.0_child ← n_j if s_k ∉ X
6:    n_i.1_child ← n_j if s_k ∉ X.neighbor_switches
7:    N ← N ∪ {n_i}, n_i ← n_j
8: end for
9: return N
```

5.4 Intersection over ZDDs

This subsection builds $\mathcal{Z}^{\text{all}}$, which represents configurations satisfying all the constraints. Since this ZDD is given as the intersection over ZDDs for each constraint, as shown in Fig. 5, we have,

$$\mathcal{Z}^{\text{all}} = \mathcal{Z}^{\text{topol}} \cap \mathcal{Z}^{\text{elec}}.$$

5.5 Search for Relaxed Solution on ZDD

This subsection searches for the relaxed solution, $\overline{X}$, in $\mathcal{Z}^{\text{all}}$. Figure 11 describes an algorithm named Rebuild, which finds the relaxed solution. We introduce *weighted arcs*, and add them to the ZDD as shown in Fig. 11 (b). A weighted arc traverses a component over the ZDD, and the arc's weight represents power loss in the component (power loss can be determined without considering other components, as discussed in Section 3.1). For example, the arc between nodes n_0 and n_2 indicates a single configuration of component 1, that is $\{s_2\}$, and so the weight is power loss of the component with that configuration, namely, $P_1(\{s_2\})$. Similarly, the arc between node n_3 and $\top$-terminal is given by two configurations $\{s_3, s_4\}$ and $\{s_3, s_5\}$, and the weight is the minimum of the corresponding power losses, $\text{Min}(P_2(\{s_3, s_4\}), P_2(\{s_3, s_5\})$. Each path from the root node to $\top$-terminal represents total power loss along all the components, which is equivalent to the objective function (8). We, finally, find the relaxed solution just by searching for the shortest path of the weighted arcs.

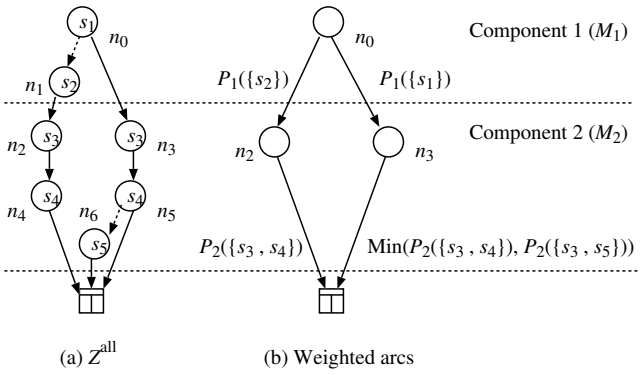

Figure 11: An example of algorithm REBUILD for the network of Fig. 1.

Algorithm 7 REBUILD

```
1: N ← {n_0} // Entry nodes of each component M
2: for each M ∈ 𝓜 do
3:     N ← REBUILDM(N, M) // Update N for next component
4: end for
```

We present the complete algorithm of REBUILD in Algorithm 7. This algorithm just calls REBUILDM for each component with entry nodes of the component. Entry nodes are nodes that have incoming arcs from the previous component; e.g., nodes n_2 and n_3 are entry nodes of component 2 (exceptionally, the root node, n_0, is the entry in component 1, though it has no incoming arc).

REBUILDM in Algorithm 8 sets the weighted arcs in a given component, M_l. First, algorithm FINDX finds all configurations in the component, and also returns associated entry nodes of the next component; each configuration X is found between an entry node of the current component, n_i, and that of the next component, n_j. For each configuration, the power loss (weight) is calculated, and an arc is set between the nodes with the weight if the weight is smaller than that of the existing arc. Finally, the entry nodes of the next component are returned. The power loss in a component, P_l in Algorithm 8, can be cached for future use, because $\mathcal{Z}^{\text{all}}$ may include several common partial configurations. Assuming that REBUILDM is called with $N = \{n_0\}$ and M_1 in Fig. 11, the algorithm sets weighted arcs n_0–n_2 with $P_1(\{s_2\})$ and n_0–n_3 with $P_1(\{s_1\})$, and returns the entry nodes of the next component, n_2 and n_3.

FINDX in Algorithm 9 returns all configurations that begin from a given ZDD node, n, in component M_i. This algorithm follows n's 0- and 1-children recursively, and gets a configuration when getting to the next component. For example, when this algorithm is called with n_0 and M_1 in Fig. 11, it returns $(\{s_2\}, n_2)$ and $(\{s_1\}, n_3)$.

Since the number of configurations in a component is logarithmically smaller

Algorithm 8 REBUILDM(N, M_l)

$N' \leftarrow \emptyset$ // Entry nodes of next component
for each $n_i \in N$ **do**
  $\mathcal{X} \leftarrow$ FINDX$(n_i, M_l, \emptyset)$ // Config's beginning at n_i in M_l
  for each $(X, n_j) \in \mathcal{X}$ **do** // Config and next entry node
    $N' \leftarrow N' \cup \{n_j\}$
    $P_l \leftarrow \sum_{c_k \in C_l} P_k(X)$ // Power loss in M_l by X
    // Find arc between n_i and n_j from $\mathcal{Z}^{\text{all}}$, or create new arc
    $a =$ FindArc(n_i, n_j)
    // Set a with weight P_l in $\mathcal{Z}^{\text{all}}$
    $a.weight \leftarrow P_l$, SetArc(a) **if** $P_l < a.weight$
  end for
end for
return N'

Algorithm 9 FINDX(n, M_i, X)

$\mathcal{X} \leftarrow \emptyset$ // Configurations in M_i
if $n.label \notin M_i$ // If getting to next component
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{(X, n.n)\}$
else
  $\mathcal{X} \leftarrow \mathcal{X} \cup$ FINDX$(n.0_child, M_i, X)$ **if** $n.0\_child \neq \perp$
  $\mathcal{X} \leftarrow \mathcal{X} \cup$ FINDX$(n.1_child, M_i, X \cup \{n.label\})$
end if
return $\mathcal{X}$

than that of the whole network, all the configurations can be investigated in each component. Moreover, we set just a single arc between a node pair, though there can be many configurations; this greatly reduces the search space and makes the shortest path search faster. Power loss calculation in a component can be cached to avoid duplicated calculation.

6 Tests and Results

In this section, the proposed method is tested with hypothetical distribution networks. Section 6.1 describes the hypothetical networks and Section 6.2 shows our implementation. Section 6.3 discusses the quality of solutions to evaluate the relaxed problems introduced in Section 3. Section 6.4 examines the scalability of our solution method proposed in Section 5.

The test criteria are the quality and scalability, as presented in Table 1. For the quality, the proposed method is compared with a brute force method, which guarantees the optimality, while it is not compared with heuristics and metaheuristics because they have no guarantee on the quality. In terms of the scalability, computation time and memory usage are examined with networks of various sizes. We discuss the scalability of the proposed method as well as the brute force method, but do not evaluate that of heuristics and metaheuristics

Table 6: Specification of test network used

# of distribution substations	4
# of feeders	72
# of switches	468
# of sections	648
Total load, $\sum_C I$	287 MW at 2 p.m., and 113 MW at 4 a.m.
Line capacity, $J^{\max}$	300 A
Sending line voltage, V_0	6.6 kV
Minimum voltage, $V^{\min}$	6.3 kV

since they are confirmed scalable enough.

6.1 Test Networks

The hypothetical network used in the tests has been developed based on actual distribution systems and load profiles provided by a large utility in Japan[3]; we call this data the Fukui-TEPCO network and it is publicly available online [3]. The network has four distribution substations, 72 feeders, and 468 switches. The network is divided into 63 independent components. The number of switches in a component is 7.43 on average, while the minimum and maximum are 3 and 20, respectively. We have hourly load profiles on a summer day, and the peak load (2 p.m.) and the bottom load (4 a.m.) were used in the tests. The specification of the network is summarized in Table 6.

To evaluate the scalability, we scaled down the test network while honoring its structure by selecting successive feeders; networks with 20, 39, 59, 78, 99, 118, 235, 352, and 468 switches were used in the tests.

6.2 Implementation

We developed software that implements our method described in Section 5. The topological constraints described in Section 5.2 and the intersection in Section 5.4 are written in C++, and others are written in Python; the Python code is the DNET, while the C++ code is *fukashigi* combinatorial problem solver (fukashigi will be available publicly online, but currently it is shipped with DNET).

We also implemented a brute force method for the comparison, but the implementation does not follow [22] since it supports direct current only.

The tests were conducted on a single computer with Intel Xeon CPU E31290 (3.60 GHz $\times 4$).

[3]There are some benchmark networks provided online, but they could not used for our tests. IEEE power and energy society [1] and North Dakota State University [2] offer many benchmark networks, but their networks have too few switches to evaluate the scalability. Other benchmark networks used in [31] are unavailable due to linkrot.

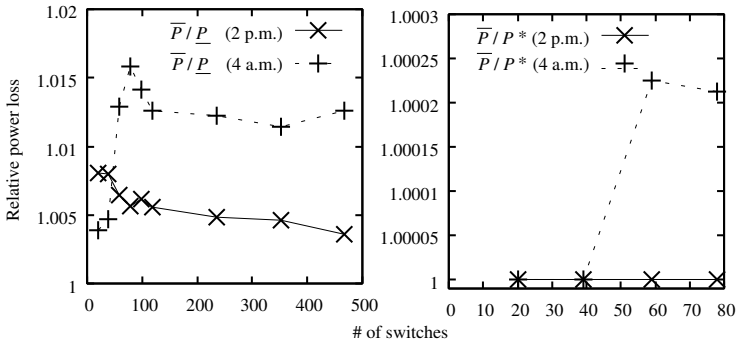

Figure 12: Relative power loss.

6.3 Quality of Solutions

This subsection evaluates the quality of solutions. The proposed method yields the upper bound, $\overline{P}$, and the lower bound, $\underline{P}$, on the minimum loss, while the brute force method found truly minimum loss, $P^\star$.

We evaluate these bounds with the true minimum in Fig. 12. The figure plots ratios between upper and lower bounds, $\overline{P}/\underline{P}$, in the left, as well as ratios between an upper bound and a true minimum, $\overline{P}/P^\star$, in the right. The gap between the lower and upper bounds is less than 1.56 %, and so we can say that the minimum loss is tightly bounded from above and below by the proposed method. Moreover, the upper bounds are consistent with the true minima in six of eight plots, and the gap is only 0.0225 % at most. The true minima could be found just in networks with up to 78 switches, due to the lack of scalability of the brute force method as will be discussed in Section 6.4. It is worth noting that the gap between the lower and upper bounds is often larger at 4 a.m. than at 2 p.m.. This is because the load is less uniformly distributed among feeders at 4 a.m., and the lower bounds are derived ignoring this un-uniformity while the upper bounds are affected by it.

Figure 13 shows the loss in root sections over the total loss, namely, $\sum_{c_i \in C_0} \overline{P}_i/\overline{P}$. We see that the root loss accounts for a large part of the total, that is nearly 30 %. We removed the root loss from the objective function in the relaxed problem for the upper bound, but the upper bounds show good consistency with the true minima in Fig. 12. This result implies that the loss in root sections is strongly correlated with that in component sections as we discussed in Section 3.2. The root loss at 2 p.m. covers a larger part compared to 4 a.m., because the load is mainly placed near substations at 2 p.m. in our load profiles.

We estimate cumulative distribution functions of power loss in the network with 468 switches, and present it in Fig. 14. These distributions are drawn by randomly choosing 1,000 configurations from $\mathcal{Z}^{\text{all}}$ using algorithm RANDOM SAMPLING. The figure also shows the upper and lower bounds, $\overline{P}$ and $\underline{P}$, provided by the proposed method. We see that both bounds are much better than

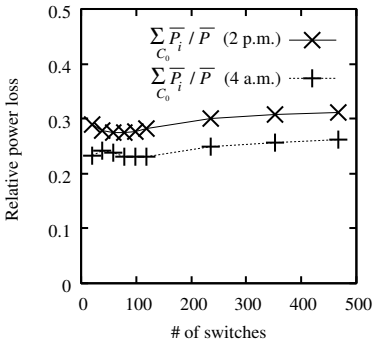

Figure 13: Relative power loss on root sections.

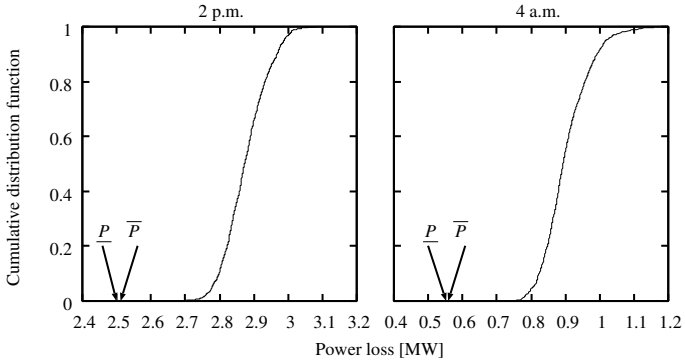

Figure 14: Cumulative distribution function of power loss in the network with 468 switches.

other configurations and the bounds are very close to each other.

We conclude that the relaxed problems introduced in Sections 3.2 and 3.3 yield bounds of good quality. Operators are allowed to reduce the power loss by implementing the relaxed solution, $\overline{X}$, in their network, since the upper bound is expected to be very close to the true minimum.

6.4 Scalability

This subsection evaluates the scalability. We first examine the computation time, and then discuss the memory usage[4].

Figure 15 shows the computation time of the proposed method and that of the brute force method. The proposed method finishes the optimization less than an hour even in the network with 468 switches, while the brute force method takes several hours in much smaller network (it takes 10^{13} years for 468

[4]Since this experiments were conducted with the former version of DNET written in Perl, the results are slightly different with the Python DNET.

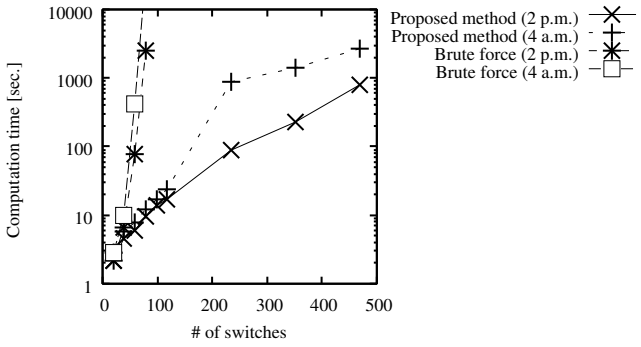

Figure 15: Computation time of the proposed method and the brute force method.

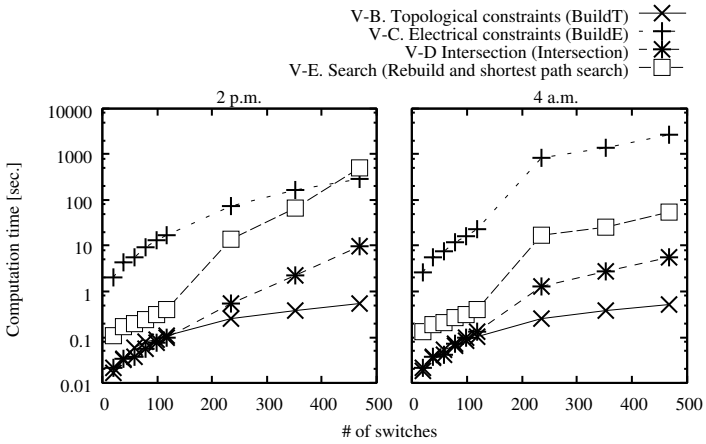

Figure 16: Computation time of each process in the proposed method.

switches estimated by simple extrapolation). The result shows that our method scales well in the network of practical size.

Figure 16 shows the computation time required by each process described in Sections 5.2–5.5. Since the network at 4 a.m. has more electrically feasible configurations because of the smaller load, the total computation time is governed by the enumeration of these configurations. In the network at 2 p.m., since the ZDD structure is more complicated as discussed later, the final search process becomes significant. We then examine these processes in more detail. The computation time for the electrical constraints is mainly covered by algorithm EnumF in the both load profiles. The search process is split roughly in half between algorithm Rebuild and shortest path search. The cache hit ratio in RebuildM is given in Fig. 17; e.g., 99.9 % (5,704,593 in 5,715,906) of loss calculation is skipped in the network with 468 switches at 2 p.m. thanks to the

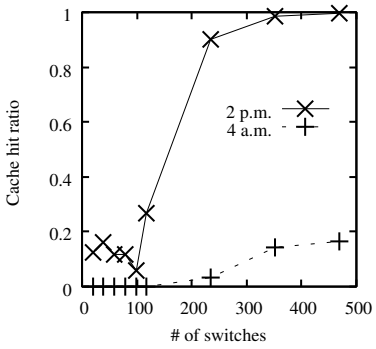

Figure 17: Cache hit ratio in algorithm REBUILDM.

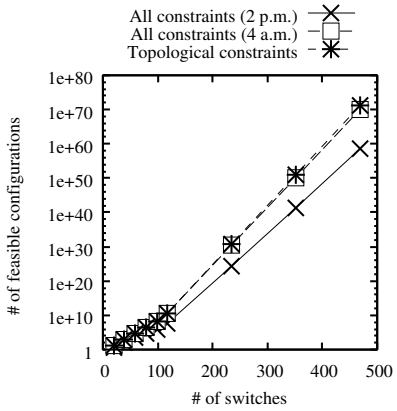

Figure 18: The number of feasible configurations.

cache.

The poor scalability of the brute force method, shown in Fig. 15, is caused by the exponential explosion of configurations. Figure 18 shows the number of configurations that satisfy all the constraints or the topological constraints; these numbers are determined by algorithm COUNT. Since there are an astronomical number of feasible configurations, it is impossible to find the optimum value among them in a brute force manner. The network at 4 a.m. has more feasible configurations than that at 2 p.m., because it is less constrained electrically as mentioned above.

We next discuss the memory usage of proposed method. Figure 19 shows the size of $\mathcal{Z}^{\mathrm{all}}$, which is defined as the number of nodes in it as noted in Section 4. The size is significantly smaller than the number of configurations due to the reduction operation described in Section 4. As mentioned above, configurations are less constrained electrically at 4 a.m., and so the ZDD structure is quite similar to $\mathcal{Z}^{\mathrm{topol}}$. On the contrary, the network has strong electrical constraints

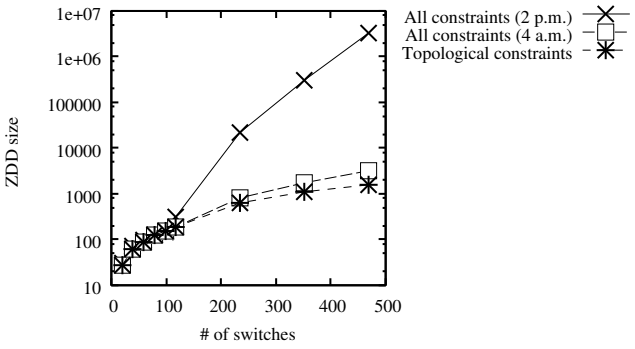

Figure 19: ZDD size.

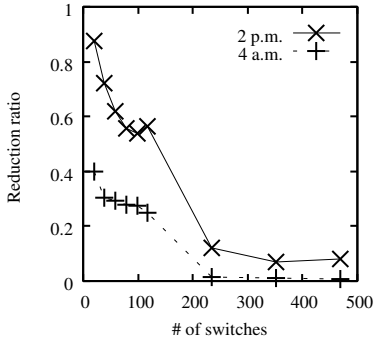

Figure 20: Reduction ratio of weighted arcs.

at 2 p.m., and so the ZDD requires more complicated structure. Nonetheless, the memory usage is quite moderate even at 2 p.m. Since a single ZDD node requires about 32 bytes in the C++ implementation, the ZDD for the network of 468 switches can be stored in memory of about 100 MB and 100 KB for 2 p.m. and 4 a.m., respectively. The Python implementation requires more bytes due to the weighted arcs as well as Python interpreter, but it is still acceptable. Since we set just a single weighted arc between a pair of nodes as mentioned in Section 5.5, the search space is reduced by 91.9 % (5,252,892 of 5,715,906 arcs are eliminated) in the network with 468 switches at 2 p.m. The reduction ratio is shown in Fig. 20.

7 Related Work

7.1 Heuristics

Distribution network reconfiguration for the loss reduction was first studied in [19]. The method used is called the *sequential switch opening*, and it was

followed by [28, 16]. In this method, all switches are closed in the initial state, and then they are opened sequentially to restore to the radial configuration. The *branch exchange* is another heuristic method [8, 4, 17]; the opening of any switch is required to correspond to the closure of another switch, ensuring that the radial structure would be preserved. The branch exchange heuristic was also used in conjunction with another method to improve the efficiency [26]. These heuristics have great scalability, but they do not guarantee the optimality of solutions nor give any bound.

7.2 Metaheuristics

Several metaheuristics are applied to the loss minimization problem. Simulated annealing has been applied in [7, 13]. Genetic algorithms and tabu search were used in [23, 10, 6] and in [11], respectively. As is the case in the heuristic methods, they scale well without proving the optimality.

7.3 Methods with Guaranteed Optimality

A brute force method [22] evaluates all feasible configurations and finds the minimum loss configuration. This method guarantees the optimal solution, but it compromises the scalability; the method cannot be used in a network of practical size, due to the exponential growth of the number of feasible configurations as shown in Section 6.4.

There are some papers that address the optimality or bounds. In [27], the loss minimization problem is relaxed to the linear programming, which guarantees the optimality, but the solution obtained is compromised without bound by the relaxation. Reference [18], which introduced a greedy search with backtracking, mentions that it provides the lower bound, but we cannot evaluate the bound since it shows no derivation and no test result.

7.4 Others

In addition to the reconfiguration, capacitor control is also used for the loss minimization [14, 32]. It is, however, not economically justified often [25].

The loss minimization problem can be extended by introducing other objectives. Multiple objectives including loss minimization are evaluated at a time by using the fuzzy set theory [9]. Reference [30] minimizes *energy* loss for a given period, instead of minimizing power loss at a given moment. The impact of distributed generators on the reconfiguration was studied in [11, 24, 21, 29]. We believe that our proposal gives a solid basis to these extensions.

8 Conclusions

This paper proposed a scalable network reconfiguration method that yields tight bounds on the minimum loss. We first derived the relaxed problems for the

bounds under the appropriate modeling assumptions. We then devised several algorithms that search for the bounds without losing the scalability. These algorithms take advantage of the ZDD's great efficiency to handle a huge number of configurations. The method was tested with the practical distribution networks. The minimum loss was tightly bounded by a gap of about 1 % with a computation time of a few tens of minutes for the network with 468 switches.

We briefly discuss the impact of distributed generators on our method. Distributed generators are often regarded as negative loads in distribution systems [21]. In our method, small distributed generators can be treated in a same manner, but big generators, which provide large power enough to feed neighbor components, cannot be handled in the same way as discussed in Section 3.1. Instead of introducing negative loads, such a big generator should be treated as a substation transformer, and the network would be divided into smaller components according to it. The refinement of this issue is our future work.

Future work also includes tests with actual distribution systems operated by a utility, as well as comparison with heuristics and metaheuristics.

Acknowledgement

We would like to thank Takashi Horiyama and Takeaki Uno for their valuable comments on algorithm EnumF. We wish to acknowledge the helpful advice in problem formulation by Hirotaka Takano.

References

[1] Distribution test feeders - distribution test feeder working group - IEEE PES distribution system analysis subcommittee. http://www.ewh.ieee.org/soc/pes/dsacom/testfeeders/.

[2] REDS: Repository of distribution systems. http://venus.ece.ndsu.nodak.edu/∼kavasseri/reds.html.

[3] Research institute of advanced network technology. http://www.hayashilab.sci.waseda.ac.jp/RIANT/riant_test_feeder.html.

[4] M. Baran and F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *Power Delivery, IEEE Transactions on*, 4(2):1401 –1407, 1989.

[5] M. Behle. *Binary decision diagrams and integer programming*. PhD thesis, Universität des Saarlandes, 2007.

[6] E. Carreno, R. Romero, and A. Padilha-Feltrin. An efficient codification to solve distribution network reconfiguration for loss reduction problem. *Power Systems, IEEE Transactions on*, 23(4):1542 –1551, 2008.

[7] H.-D. Chiang and R. Jean-Jumeau. Optimal network reconfigurations in distribution systems. i. a new formulation and a solution methodology. *Power Delivery, IEEE Transactions on*, 5(4):1902 –1909, 1990.

[8] S. Civanlar, J. Grainger, H. Yin, and S. Lee. Distribution feeder reconfiguration for loss reduction. *Power Delivery, IEEE Transactions on*, 3(3):1217 –1223, 1988.

[9] D. Das. A fuzzy multiobjective approach for network reconfiguration of distribution systems. *Power Delivery, IEEE Transactions on*, 21(1):202 – 209, 2006.

[10] B. Enacheanu, B. Raison, R. Caire, O. Devaux, W. Bienia, and N. Hadj-Said. Radial network reconfiguration using genetic algorithm based on the matroid theory. *IEEE Transactions on Power Systems*, 23(1):186 –195, 2008.

[11] Y. Hayashi and J. Matsuki. Loss minimum configuration of distribution system considering n-1 security of dispersed generators. *Power Systems, IEEE Transactions on*, 19(1):636 – 642, 2004.

[12] T. Inoue, K. Takano, T. Watanabe, J. Kawahara, R. Yoshinaka, A. Kishimoto, K. Tsuda, S. Minato, and Y. Hayashi. Distribution network reconfiguration for tightly bounded minimum loss by ZDDs. Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-12-58, 2012. http://www-alg.ist.hokudai.ac.jp/tra.html.

[13] Y.-J. Jeon, J.-C. Kim, J.-O. Kim, J.-R. Shin, and K. Lee. An efficient simulated annealing algorithm for network reconfiguration in large-scale distribution systems. *IEEE Transactions on Power Delivery*, 17(4):1070 – 1078, 2002.

[14] D. Jiang and R. Baldick. Optimal electric distribution system switch reconfiguration and capacitor control. *IEEE Transactions on Power Systems*, 11(2):890 –897, may 1996.

[15] D. Knuth. *The Art of Computer Programming, volume 4, Fascicle 1: Bitwise Tricks and Techniques; Binary Decision Diagrams.* Addison-Wesley, USA, 2009.

[16] W.-M. Lin and H.-C. Chin. A new approach for distribution feeder reconfiguration for loss reduction and service restoration. *Power Delivery, IEEE Transactions on*, 13(3):870 –875, 1998.

[17] C. Liu, S. Lee, and K. Vu. Loss minimization of distribution feeders: optimality and algorithms. *Power Delivery, IEEE Transactions on*, 4(2):1281 –1289, 1989.

[18] T. McDermott, I. Drezga, and R. Broadwater. A heuristic nonlinear constructive method for distribution system reconfiguration. *Power Systems, IEEE Transactions on*, 14(2):478 –483, 1999.

[19] A. Merlin and H. Back. Search for a minimal-loss operating spanning tree configuration in an urban power distribution system. In *Proc. 5th Power System Computation Conference (PSCC), Cambridge, UK*, pages 1–18, 1975.

[20] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Design Automation, 1993. 30th Conference on*, pages 272 – 277, 1993.

[21] Y. Miyamoto and Y. Hayashi. Evaluation of improved generation efficiency through residential PV voltage control of a clustered residential grid-interconnected PV. In *IEEE PES ISGT Europe*, pages 1 –8, 2010.

[22] A. Morton and I. Mareels. An efficient brute-force solution to the network reconfiguration problem. *IEEE Transactions on Power Delivery*, 15(3):996 –1000, 2000.

[23] K. Nara, A. Shiose, M. Kitagawa, and T. Ishihara. Implementation of genetic algorithm for distribution systems loss minimum re-configuration. *Power Systems, IEEE Transactions on*, 7(3):1044 –1051, 1992.

[24] J. Olamaei, T. Niknam, and G. Gharehpetian. Impact of distributed generators on distribution feeder reconfiguration. In *IEEE Lausanne Power Tech*, pages 1747 –1751, 2007.

[25] G. Peponis, M. Papadopoulos, and N. Hatziargyriou. Distribution network reconfiguration to minimize resistive line losses. In *Athens Power Tech, 1993. APT 93. Proceedings. Joint International Power Conference*, volume 2, pages 601 –605, 1993.

[26] G. Raju and P. Bijwe. An efficient algorithm for minimum loss reconfiguration of distribution system based on sensitivity and heuristics. *Power Systems, IEEE Transactions on*, 23(3):1280 –1287, 2008.

[27] E. Ramos, A. Exposito, J. Santos, and F. Iborra. Path-based distribution network modeling: application to reconfiguration for loss reduction. *Power Systems, IEEE Transactions on*, 20(2):556 – 564, may 2005.

[28] D. Shirmohammadi and H. Hong. Reconfiguration of electric distribution networks for resistive line losses reduction. *IEEE Transactions on Power Delivery*, 4(2):1492 –1498, 1989.

[29] S.-Y. Su, C.-N. Lu, R.-F. Chang, and G. Gutiérrez-Alcaraz. Distributed generation interconnection planning: A wind power case study. *Smart Grid, IEEE Transactions on*, 2(1):181 –189, 2011.

[30] R. Taleski and D. Rajicic. Distribution network reconfiguration for energy loss reduction. *Power Systems, IEEE Transactions on*, 12(1):398 –406, 1997.

[31] H. Zeineldin, E. El-Saadany, M. Salama, A. Alaboudy, and W. Woon. Optimal sizing of thyristor-controlled impedance for smart grids with multiple configurations. *Smart Grid, IEEE Transactions on*, 2(3):528 –537, 2011.

[32] D. Zhang, Z. Fu, and L. Zhang. Joint optimization for power loss reduction in distribution systems. *Power Systems, IEEE Transactions on*, 23(1):161 –169, 2008.