TCS-TR-A-14-70

TCS Technical Report

Verifying Distribution Networks for Secure Restoration by Enumerating All Critical Failures

by

Takeru Inoue, Norihito Yasuda, Shunsuke Kawano,
Yuji Takenobu, Shin-ichi Minato, and
Yasuhiro Hayashi

Division of Computer Science

Report Series A

January 28, 2014

Hokkaido University
Graduate School of
Information Science and Technology

Email: minato@ist.hokudai.ac.jp

Phone: +81-011-706-7682
Fax: +81-011-706-7682

Verifying Distribution Networks for Secure Restoration by Enumerating All Critical Failures

Takeru Inoue*† Norihito Yasuda†‡ Shunsuke Kawano§
Yuji Takenobu§ Shin-ichi Minato‡† Yasuhiro Hayashi§

January 28, 2014

Abstract

If several feeders are cut in a severe accident, distribution networks should be restored by reconfiguring switches automatically with smart grid technologies. Although there have been several restoration algorithms developed to find the new network configuration, they might fail to restore the whole network if the cut critically damaged the network. The networks design has to guarantee that it is restorable under any possible cut for secure power delivery, but it is computationally hard task to examine all possible cuts in a large-scale network with complex electrical constraints. This paper presents a novel method to find all the critical (unrestorable) cuts with great efficiency to verify the network design. Our method first runs a fast screening algorithm based on hitting set enumeration; the algorithm selects suspicious cuts without naively examining all possible cuts. Next, unrestorable cuts are identified from the suspicious ones with another algorithm, which strictly tests the restorability of the network under each suspicious cut without redundantly repeating heavy power flow calculations. Thorough experiments on two distribution networks reveal that our method can find thousands of unrestorable cuts from the trillions of possible cuts in a large 432-bus network with no significant false negatives. This is the first work to reveal the vulnerability of a large-scale distribution network extensively.

1 Introduction

Distribution networks consist of several feeders with many switches. The feeder lines are often cut accidentally by heavy equipment, natural disasters, or intentional attacks, which causes blackouts along the affected feeders. In response to

*takeru.inoue@ieee.org
†ERATO MINATO Discrete Structure Manipulation System Project, Japan Science and Technology Agency, Sapporo, Japan.
‡Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan.
§Faculty of Science and Engineering, Waseda University, Tokyo, Japan.

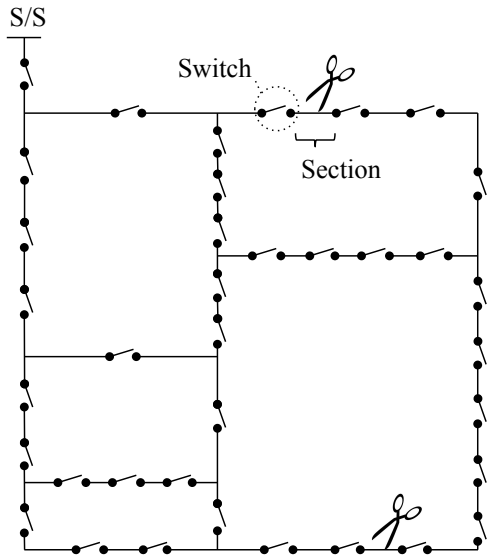

Figure 1: Unrestorable cutset found in a distribution network [1]. Given that line sections indicated by the scissors are cut, the whole network would be still physically connected but no feasible configuration could be found due to the voltage drop constraint. We cannot deliver power to some intact sections, though they have paths to the substation (S/S).

the blackouts, operators attempt to restore the network, providing power to downstream of the affected feeders, by network reconfiguration, that is, by changing the open/close status of switches. In the restoration process, sections containing the cut are located, and are isolated by opening neighboring switches. Then, the downstream sections, which have not been cut but are disconnected from the network by the isolation process, are connected to a neighbor feeder through a tie switch. The new restored configuration must satisfy operational constraints such as the radiality of feeders, line capacity, and voltage drop. Secure and automated restoration is a key motivation to introduce the smart grid technologies.

The restoration process has been well studied, and several restoration algorithms have been proposed [11, 16, 10, 9, 15, 14]; given a set of line sections that have been cut, the algorithms efficiently find a series of switching operations leading to a new feasible (constraint-conformant) configuration. These algorithms, however, can fail to restore all the intact sections. This is because no feasible configuration might remain in the affected network whose topology has been changed by the cuts. Here, we focus on an *unrestorable cutset*, which is defined as a set of cuts such that some intact sections cannot be energized after the cuts have been made due to the absence of feasible configurations. Figure 1 gives an example of an unrestorable cutset found in the well-known distribution network introduced by Baran and Wu [1]. An unrestorable cutset may cause a long-term blackout on several intact sections. Unfortunately, restoration algorithms cannot resolve this

issue, because no feasible configuration exists in the network given the presence of an unrestorable cutset. Since the root cause of this issue is the absence of feasible configurations, this is an issue of network design at deployment time, not an issue for the restoration algorithms. If the network were verified not contain an unrestorable cutset, restoration algorithms could identify a new feasible configuration in an emergency. We need to find all unrestorable cutsets and to eliminate them for secure power delivery before deploying the new distribution network.

Only a few papers have investigated efficient techniques related to finding unrestorable cutsets. The reliability-network-equivalent method [2] calculates the availability of each load point in a distribution network, but it does not find unrestorable cutsets explicitly since the calculation method depends on probability propagation. Reference [9] examined feeder lines to be cut for availability calculation, but it checks them one by one naively and so it does not scale to support large networks. Reference [4] focused on $N-1$ security to ensure whether a network would be feasible under a cutset of any single line, but did not deal with a cutset of multiple lines.

Finding all unrestorable cutsets is a computationally tough problem, because the number of possible cutsets increases quite rapidly with network size and cutset size; i.e., assume there are n line sections in a network and k of the n sections can be cut at a time, we have to examine $\binom{n}{k} = O(n^k)$ combinations of sections to find all unrestorable cutsets. The naive approach, which examines all possible combinations, clearly does not scale well. As will be presented in this paper, lines separated by 18 switches can form an unrestorable cutset, and so we have to deal with a network of large n without dividing it into small pieces. In addition, k can be larger than one, because several lines can be cut in a short time by a large-scale disaster or a series of terror attacks.

We face another issue; we have to conduct a feasibility test for every cutset to check whether the network can deliver power to all intact sections under the cutset. Since the search space is non-convex due to the complex constraints of the distribution network [8], the test is also computationally intractable.

This paper proposes a novel method that efficiently finds (nearly) all unrestorable cutsets. Our method avoids the computation issues by employing a compressed data structure named the zero-suppressed binary decision diagram, or ZDD [12], which allows us to represent a huge non-convex space in a compressed manner, and also to execute efficient algebra on the compressed space. ZDDs have been successfully applied to the loss minimization problem of distribution networks [6]; a non-convex space of 10^{70} configurations were efficiently handled to find an optimal configuration.

Our method is two-fold.

- First, our method efficiently selects unrestorable cutest candidates (suspicious cutsets), which greatly reduces the number of cutsets to be tested rigorously. We employ the *hitting set* enumeration with ZDDs [17] for this

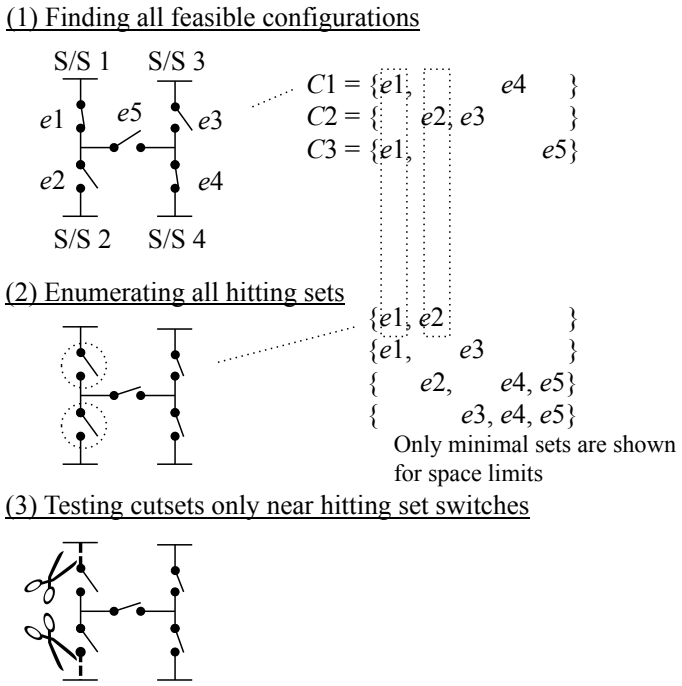

Figure 2: Basic idea to identifying potential unrestorable cutsets. (1) We represent a feasible configuration by a set of closed switches; e.g, two switches e_1 and e_4 are closed in feasible configuration C_1. (2) All hitting sets of the feasible configurations are enumerated; e.g. switches $\{e_1, e_2\}$ form a hitting set, since at least one of them is closed in all feasible configurations C_1–C_3. Opening all switches in the hitting set makes the network infeasible. (3) Cutting all lines near hitting set switches is also likely to make the network infeasible. We focus only on such cutsets and ignore others.

reduction; this approach finds sets of switches that are closed in common in all feasible configurations (Figure 2). In other words, if all switches in the set were to be opened, the network would lose all feasible configurations. This observation leads to our basic idea; cutting lines near the switches of a hitting set is likely to trigger the same consequence, and so we will test only such suspicious cutsets and can ignore the others. (Section 3.3)

- Second, our method tests the suspicious cutsets rigorously and identifies truly unrestorable cutsets. In advance, we built two ZDDs, which represent the topological constraints (radiality) and the electrical constraints (line capacity and voltage drop). After updating just the topological constraints for each suspicious cutset, we conduct the satisfiability test without re-executing complex power flow calculations required for the electrical constraints. (Section 3.2)

We conduct comprehensive computer experiments to evaluate our method with

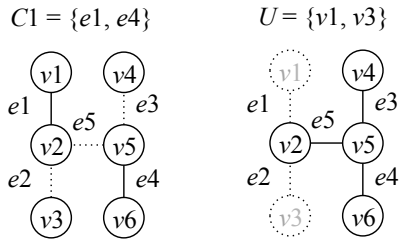

Figure 3: Graph representation of distribution network in Figure 2. In graph $G = (V, E)$, vertex $v \in V$ and edge $e \in E$ represent a line section and switch, respectively, while subgraph $C \subseteq E$ can be a feasible configuration. Our problem is to find every unrestorable cutset $U \subseteq V$.

the well-known Baran and Wu network [1] as well as a large-scale network developed by Fukui University and Tokyo Electric Power Company [3]; the large-scale network closely models a typical Japanese distribution network and includes 432 buses. The results show a remarkable reduction in the number of tests, five orders of magnitude lower. Moreover, each test is completed in less than five seconds on average even in the large-scale network. Finally, thousands of unrestorable cutsets are found in the large-scale network.

The rest of this paper is organized as follows. Section 2 defines our problem, and Section 3 describes the algorithms used to find unrestorable cutsets. Section 4 reports our experiments and the results, and Section 5 concludes this paper.

2 Problem statement

We first describe our network model. A distribution network consists of feeders with switches. Each part of a feeder separated by switches is called a line section, and it has load and impedance. Given the status of all switches, the network configuration is uniquely determined. A configuration that complies with the *topological* constraints consists of radial feeders (loop-free), each of which is connected exclusively to a feeding point at a substation. All line sections must be energized unless they have not been cut. The *electrical* constraints including line capacity and voltage drop are also set.

Following our past work [6], we represent the distribution network as a graph, $G = (V, E)$, as shown in Figure 3. A switch is represented by an edge, $e \in E$, while a line section is a vertex, $v \in V$. If a switch is open, the corresponding edge is removed. In this graph representation, a configuration is represented by a subset of edges that have not been removed; a subset of edges is simply called a subgraph in this paper, unless otherwise confusing. A subgraph representing a feasible configuration is noted by $C \subseteq E$. The topological constraints require feasible subgraph C to be a spanning forest rooted to substations (a forest means a set of disjoint trees), because each feeder has to be a tree and be rooted on a

feeding point, and every vertex has to be covered by a tree. Feasible subgraph C also satisfies the electrical constraints. A set of all feasible subgraphs in graph G is noted by $\mathcal{C}(G)$ and we have,

$$\mathcal{C}(G) = \{C \subseteq E : \text{is\_feasible}(C)\},$$

where argument G will be omitted when it is obvious from the context.

Our definition of cutset is different from that of the graph theory, though it shares the flavor of a vertex cutset. A *cutset* in this paper refers to a set of vertices (line sections) that are being cut at the same time. We give a formal definition of an unrestorable cutset as follows; given an unrestorable cutset, $U \subseteq V$, and a set of such cutsets, $\mathcal{U} \ni U$, it is defined as,

$$\mathcal{U} = \{U \subseteq V : G' = (V \setminus U, E \setminus \text{edges}(U)), |\mathcal{C}(G')| = 0\},$$

where G' is a subgraph without vertices U and their edges (i.e., we cut sections of U and isolate them by opening neighboring switches), and the last term means that G' has no feasible configuration and cannot deliver power to all intact sections. An unrestorable cutset might not make the rest of graph physically disconnected, since it is possible for a configuration to fail to satisfy the electrical constraints even if the graph can be connected.

Our goal is to find $\mathcal{U}$ efficiently. We only consider line failures, since they are much more frequent than other failures like capacitor banks and switches.

3 Finding unrestorable cutsets

This section describes our method that finds unrestorable cutsets. Section 3.1 reviews our past work that enumerates all feasible configurations [6]. Section 3.2 describes a test algorithm to identify unrestorable cutsets from suspicious cutsets, which are selected by the algorithm described in Section 3.3.

3.1 Enumerating all feasible configurations

We briefly describe our past work [6]. We first find a set of all topologically-feasible configurations $\mathcal{C}_t(G)$, and also find another set of all electrically-feasible configurations $\mathcal{C}_e(G)$,

$$\begin{aligned} \mathcal{C}_t(G) &= \{C \subseteq E : \text{is\_topol\_feasible}(C)\}, \\ \mathcal{C}_e(G) &= \{C \subseteq E : \text{is\_elec\_feasible}(C)\}. \end{aligned} \tag{1}$$

A set of configurations satisfying all the constraints, $\mathcal{C}(G)$, is given as the intersection of both sets,

$$\mathcal{C}(G) = \mathcal{C}_t(G) \cap \mathcal{C}_e(G). \tag{2}$$

These three sets are represented as ZDDs, which are built by algorithms presented in [6, III.A], [6, III.B], and [12]. The set size, $|\mathcal{C}(G)|$, can be calculated by another algorithm[7]. Finding $\mathcal{C}_e(G)$ is the most time-consuming process and it takes more than 100 times longer than the other calculation processes, since it involves complex power flow calculations. We assume that $\mathcal{C}_e(G)$ and $\mathcal{C}(G)$ have been obtained before the following algorithms.

3.2 Testing network feasibility for suspicious cutsets

Algorithm 1 is_unrestorable

Input: $G, \mathcal{C}_e(G), S$
Output: r // indicates S is unrestorable or not
  $G' = (V \setminus S, E \setminus \text{edges}(U))$
  $\mathcal{C}_t(G') = \{C \subseteq E \setminus \text{edges}(S) : \text{is\_topol\_feasible}(C)\}$
  $\mathcal{C}(G') = \mathcal{C}_t(G') \cap \mathcal{C}_e(G)$
  if $|\mathcal{C}(G')| = 0$ **then** // if G' has no feasible configuration
    $r =$ True
  else
    $r =$ False
  end if

We describe an algorithm to determine whether a given suspicious cutset is unrestorable in Algorithm 1. Given suspicious cutset S by the algorithm of Section 3.3, we isolate vertices of S by removing their neighboring edges (imitating the opening of the corresponding switches) which yields another graph, G'. We then run the algorithm of (1) that finds topologically-feasible configurations $\mathcal{C}_t(G')$ with G'. We also run the intersection algorithm of (2) over $\mathcal{C}_t(G')$ and $\mathcal{C}_e(G)$. Finally, we count the number of feasible configurations $|\mathcal{C}(G')|$ to know whether the given cutset is unrestorable.

This algorithm is very efficient, because it does not involve the time-consuming power flow calculation required for $\mathcal{C}_e(G)$; it is calculated before this algorithm.

$\mathcal{C}_e(G)$ includes electrically feasible configurations with and without S, but configurations with S cannot be realized in the modified graph G', from which S's vertices are removed. These impossible configurations are, however, eliminated by intersection with $\mathcal{C}_t(G')$, which does not include configurations that use S.

3.3 Selecting suspicious cutsets

We describe an algorithm to select suspicious cutsets, which will be tested by Algorithm 1. We first consider hypothetical switch failures such that the distribution network would be infeasible if the switches could not be closed. Figure 2 gives an example. This network has feasible configurations, but none of them can be realized if switches e_1 and e_2 are kept open, since at least one of them must be

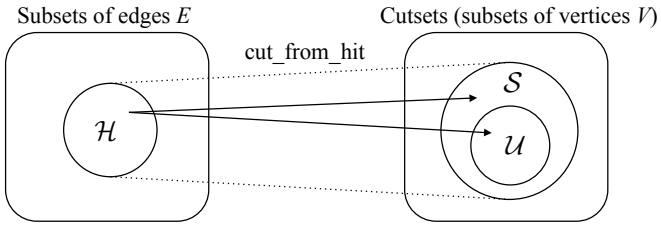

Figure 4: A set of unrestorable cutsets, $\mathcal{U}$, is a subset of suspicious sets, $\mathcal{S}$, which is projected from a set of hitting sets, $\mathcal{H}$, by function cut_from_hit. All unrestorable cutsets are, therefore, mapped just from hitting sets.

closed in any feasible configuration. A set of these switches is called a *hitting set* for the feasible configurations. We give a rigorous definition here. A hitting set for $\mathcal{C}$ is set $H \subseteq E$ where H has non-empty intersection with every configuration $C \in \mathcal{C}$. A set of hitting sets, $\mathcal{H}$, is given by,

$$\mathcal{H} = \{H \subseteq E : H \cap C \neq \emptyset \ \forall C \in \mathcal{C}\}.$$

The hitting set gives us a powerful clue for finding unrestorable cutsets. Our basic idea works as follows; a network that opens a switch has a similar topology with the network that cuts a line connected to the switch, as shown in Figure 2 (3). Based on this idea, we give a theorem to find all unrestorable cutsets from hitting sets. As preparation for the theorem, we define a suspicious cutset, $S \in \mathcal{S}$, which is mapped from a hitting set by a function named cut_from_hit,

$$\mathcal{S} = \{S \in \text{cut\_from\_hit}(H) : \forall H \in \mathcal{H}\},$$

as shown in Figure 4. The function cut_from_hit is defined by,

$$\text{cut\_from\_hit}(H) = \prod_{e \in H} \text{vertices}(e)$$

where $\prod A_i$ represents a Cartesian product of sets A_i. In the example of Figs. 2 and 3, we have,

$$\begin{aligned} \text{cut\_from\_hit}(\{e1, e2\}) &= \{v1, v2\} \times \{v2, v3\} \\ &= \{\{v1, v2\}, \{v1, v3\}, \{v2\}, \{v2, v3\}\}. \end{aligned}$$

Suspicious cutset S can also be called a hitting *vertex* set, since it consists of vertices connected to edges of a hitting set.

Theorem. A set of unrestorable cutsets, $\mathcal{U}$, is a subset of suspicious cutsets, $\mathcal{S}$, as shown in Figure 4 (a sketch of proof is given in the Appendix).

This theorem means that all unrestorable cutsets are mapped from hitting sets,

$$\mathcal{U} \subseteq \{U \in \text{cut\_from\_hit}(H) : \forall H \in \mathcal{H}\},$$

and it allows us to ignore unsuspicious cutsets without the risk of overlooking unrestorable ones. We test only the suspicious cutsets by Algorithm 1, and determine whether they are unrestorable. We cannot skip the tests, since the opposite of this theorem is not true (a hitting set can be mapped to restorable cutsets as well). This is because loads are reduced given a suspicious cutset due to the blackout of isolated sections, while it is not reduced for a hitting set that isolates no section.

Algorithm 2 find_unrestorable

Input: $\mathcal{C}(G)$
Output: $\mathcal{U}$
  $\mathcal{U} = \emptyset$
  $\mathcal{H} = \text{hitting\_sets}(\mathcal{C}(G))$
  for $H \in \mathcal{H}$ **do** // for each hitting set
    for $S \in \text{cut\_from\_hit}(H)$ **do** // for each suspicious cutset
      if is_unrestorable(S) **then**
        $\mathcal{U} = \mathcal{U} \cup \{S\}$
      end if
    end for
  end for

We present Algorithm 2 to find all unrestorable cutsets $\mathcal{U}$. This algorithm first obtains a set of hitting sets $\mathcal{H}$ for feasible configurations $\mathcal{C}$ by [17]. We test suspicious cutsets, S's, mapped from the hitting sets. Our method reduces the number of feasibility tests compared to the naive brute-force approach, which tests each cutset.

3.4 Scalability improvement techniques

We show two techniques that make our method more scalable.

3.4.1 Cutset size limit

We put an upper limit on the cutset size, which is the number of vertices (line sections) in a cutset, based on the maximum number of lines cut at a time. Since the number of hitting sets rapidly increases with their size, this limitation greatly reduces the number of hitting sets to be examined. We can efficiently restrict the hitting set size [7]. In order to identify unrestorable cutset U, we only need to examine hitting sets of $|H| \leq |U|(d-1)$, where d is the maximum vertex degree in the graph, thanks to the following Lemma (details are discussed in the Appendix).

Lemma. Unrestorable cutset U is mapped from hitting sets of $|H| \leq |U|(d-1)$.

3.4.2 Minimal cutsets only

We only focus on *minimal* unrestorable cutsets; minimal unrestorable cutset U is a cutset that does not include any other unrestorable cutset, that is, $U \not\supset U' \, \forall U' \in \mathcal{U}$. Since non-minimal hitting sets can yield non-minimal unrestorable cutsets, we test suspicious cutsets mapped only from minimal hitting sets; hitting sets which are supersets of another hitting set are ignored. Reference [17] also proposed an algorithm that efficiently selects minimal hitting sets.

Minimalization makes our method more efficient, but it can introduce the risk of overlooking some unrestorable cutsets. Suppose no unrestorable cutset were found from a hitting set, and some unrestorable cutsets were found from a superset of the hitting set. This super hitting set would not be minimal and thus would not be examined, and so this unrestorable cutset could be never found. This is a tradeoff between efficiency and accuracy, and is evaluated in the experiments.

4 Experiments

Our method is applied to two datasets, a traditional small distribution network in Section 4.1 and a large-scale network closely modeling a typical Japanese distribution network in Section 4.2. We evaluate our two methods of Algorithm 2 in the experiments; one examines minimal hitting sets only (minimalization), while the other considers supersets of the minimal hitting sets (no minimalization). These methods are compared against the naive brute-force approach, in which feasibility tests are performed on every cutset. The tests generated by our methods and the naive method are executed with Algorithm 1. Our methods were implemented with the efficient graph library named Graphillion [5]. We performed power flow calculation based on [13], but our method is not limited to any specific power flow model. The experiments were conducted using a single core of an Intel Xeon CPU E7-8837 (2.67GHz).

4.1 32-bus network

The first dataset represents the 32-bus network introduced by Baran and Wu [1]. It has 37 switches and only a single substation. This network is a single-phase alternating current system. The sending line voltage is 12.66 kV, and the maximum voltage drop must be within 10 % of the sending voltage. This dataset has no line current constraint. The maximum degree, d, is 3 in the graph representation. The results include unrestorable cutsets of size four or less, $|U| \leq 4$, since we found no minimal unrestorable cutset greater than four.

Figure 5 shows the number of tests performed by each method. Our two methods grew much more slowly than the naive method. Our method with minimalization performed only 88 % fewer tests than the naive method. Many innocent

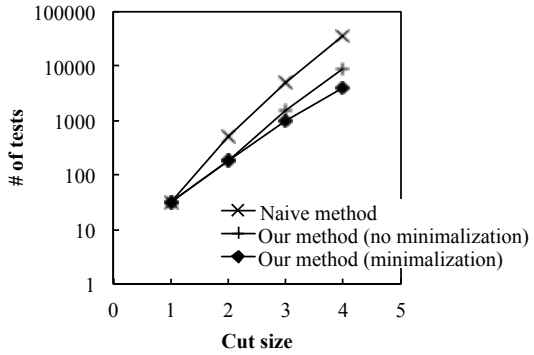

Figure 5: The number of tests versus the cutset size in the 32-bus network. The number of tests is defined as the number of suspicious cutsets $|S|$ in our method or $\binom{n}{k}$ (k is the cutset size) in the naive method. Our methods conducted much fewer tests than the naive method.

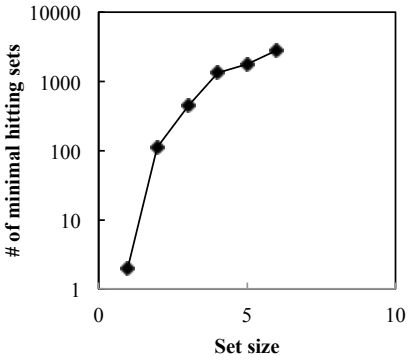

Figure 6: The number of minimal hitting sets $|\mathcal{H}|$ versus their size, $|H|$, in the 32-bus network. It is much smaller than the number of tests performed by the naive method (Figure 5). This is the reason why many unsuspicious cutsets were filtered out in our method.

cutsets were filtered out without performing feasibility tests by our methods[1]. In order to investigate this great reduction in depth, we plot the number of minimal hitting sets in Figure 6. Only a couple of thousand sets were found, a lot fewer than the possible edge combinations. This result indicates that our method successfully selected a small number of suspicious cutsets by utilizing the minimal hitting sets. The number of tests performed by our minimalization method can be greater or smaller than the number of minimal hitting sets. This is because a single hitting set can be mapped to multiple suspicious cutsets by function cut_from_hit, and also

[1]Our methods involve nearly the same number of tests as the naive method when the cutset size is one. This is because every pair of neighboring switches in straight line forms a hitting set, which generates a suspicious cutset from only the intermediate section. Our focus is, however, on the scaling of test numbers against the cutset size, and our methods are much more efficient than the naive method for larger cutsets.

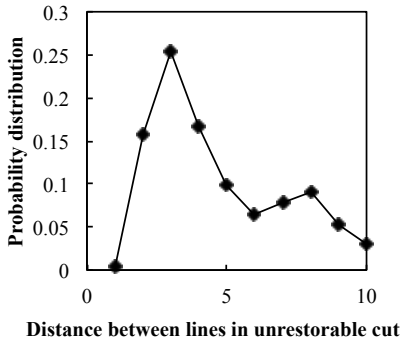

Figure 7: Probability distribution of distance between line sections in a minimal unrestorable cutset, in the 32-bus network. Some minimal unrestorable cutsets include line sections separated by ten switches, and so we have to address cutsets across a wide area.

Table 1: The number of minimal unrestorable cutsets found in 32-bus network

	Naive method and our method (no minimalization)	Our method (minimalization)
$\|U\| = 1$	3	3
2	76	76
3	69	69
4	55	47

some hitting sets share common suspicious cutsets. We found no minimal hitting set of greater than six, as shown in the figure. This is because most big hitting sets include smaller ones, so of course they are not minimal.

The great reduction in the number of tests cannot be achieved just by ignoring line sections far apart. Figure 7 shows distance distribution between line sections in a minimal unrestorable cutset, where the distance is measured by the number of switches between line sections. Line sections in a minimal unrestorable cutset are separated by up to ten switches; actually, the diameter (the maximum distance) of the network is equal to ten. This result implies that we are not allowed to use the distance for filtering out innocent cutsets. Sophisticated methods like ours are required to efficiently find unrestorable cutsets.

Table 1 shows the number of minimal unrestorable cutsets found by each method; an example of unrestorable cutset found is shown in Figure 1. Our method with minimalization overlooked eight unrestorable cutsets ($55 - 47 = 8$), but it found 96 % of them with 88% fewer tests. The probability of finding a minimal unrestorable cutset per test is 3.8 % in our minimalization method, while it is 0.49 % in the naive method. Our method found most of the unrestorable cutsets with high efficiency. This is a good tradeoff between the performance and accuracy.

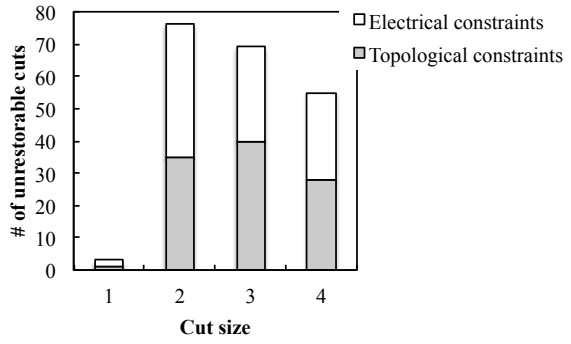

Figure 8: Causes of the unrestorable cutsets in the 32-bus network. Unrestorable cutsets can make the network disconnected topologically, or can make it electrically infeasible without disconnection.

Table 2: Computation time for 32-bus network

Algorithm 1 (average per test)	10.7 msec
Algorithm 2	5.55 sec

Figure 8 clarifies the causes of the unrestorable cutsets. The label "topological constraints" means that the network would be disconnected by the cutsets, while the label "electrical constraints" indicates that the network cannot satisfy the electrical constraints even though it can be connected. Our method found both types of unrestorable cutsets. The topologically unrestorable cutsets make the network physically disconnected. We found many small unrestorable cutsets caused by the electrical constraints, unlike the Fukui-TEPCO network as will be shown; this implies that the network might be so poorly designed that it is likely cause power supply shortages, which makes the constraints too strict.

We also evaluate the performance of our method (Table 2). Testing a suspicious cutset by Algorithm 1 requires only 10.7 msec on average, due to our efficient feasibility test algorithm that involves no power flow calculation. Total CPU time needed for Algorithm 2 is 5.55 seconds. The maximum amount of memory used by our method is 66.7 MB. Calculating feasible configurations, $\mathcal{C}_e$ and $\mathcal{C}$, requires a few hours, as shown in [6].

4.2 Fukui-TEPCO network

We consider the 432-bus network with 468 switches, which was developed in 2006 by Fukui University and Tokyo Electric Power Company (TEPCO) [3]. The network is modeled as a three-phase alternating current system, and consists of residential, industrial, and commercial areas. Among the hourly load profiles, we used the peak load, which imposes the severest electrical constraints on the network.

Table 3: Specification of Fukui-TEPCO network

Number of buses	432
Number of switches	468
Total load	287 MW
Line capacity	390 A
Sending line voltage	6.6 kV
Maximum voltage drop	0.3 kV
Max degree, d	3

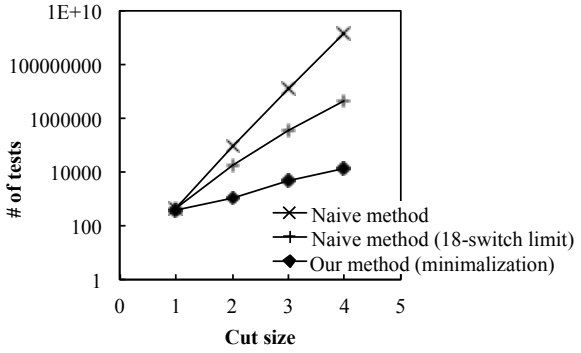

Figure 9: The number of tests versus cutset size for the Fukui-TEPCO network. Our method shows a remarkable reduction in test number, several orders of magnitude, even though the naive method took advantage of the distance limit of line sections.

The specification is given in Table 3[2]. To the best of our knowledge, there is no benchmark network that matches this size and specificity. We conducted the experiments with just our method of minimalization, since other methods did not scale to handle this large-scale network. We examined unrestorable cutsets of size four or less, $|U| \leq 4$, due to the scalability limit.

Figure 9 shows the number of tests. The numbers for the naive methods were just counted without testing. Our method reduced the number of tests by five orders of magnitude compared to the naive method, since the number of minimal hitting sets smaller than nine was 10,539, many fewer than the possible edge combinations. Figure 10 shows the distance distribution of line sections in a minimal unrestorable cutset. Our method found unrestorable cutsets consisting of line sections separated by up to 18 switches, which implies the naive method can work more efficiently with this distance limit. However, it still has to examine 226 times more cutsets than our method, as shown in Figure 9, and so our method outperformed the naive method even with the distance limit. It is worth noting that the

[2]The dataset of Fukui-TEPCO network is available from `http://www.hayashilab.sci.waseda.ac.jp/RIANT/riant_test_feeder.html`

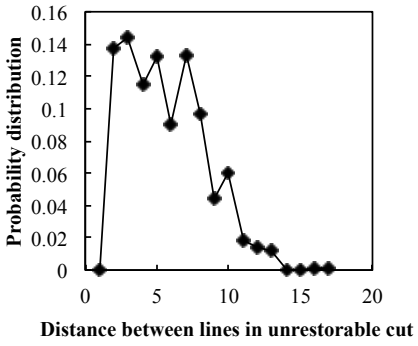

Figure 10: Probability distribution of distance between lines in a minimal unrestorable cutset, in the Fukui-TEPCO network. There are unrestorable cutsets with lines separated by 18 switches.

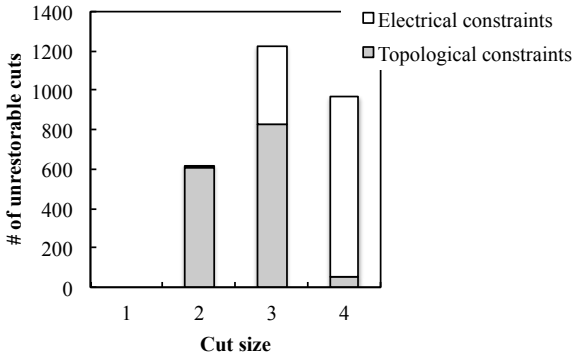

Figure 11: The number of minimal unrestorable cutsets $|\mathcal{U}|$ found by our method, versus their size $|U|$, in the Fukui-TEPCO network. Their causes are also shown.

distance limit has been revealed by our method, and so we could not apply the limit to the naive method without our method.

The number of minimal unrestorable cutsets found by our method is shown in Figure 11 together with their causes. In contrast to Fig. 8, we found that the electrical constraints caused only a few small unrestorable cutsets whose sizes are less than three. However, the network has relatively many topologically unrestorable cutsets that are small, which means the network is topologically not robust. This is because the network is not tightly connected compared to the 32-bus network; their average degrees are 2.17 and 2.25, respectively. We evaluated the coverage of our method using random sampling as follows. We selected 4,771,000 cutsets uniformly and randomly, and found 1,450 minimal unrestorable ones from those selected. Our method did not overlook any of them, which proves its high accuracy.

Table 4 shows the calculation time. Testing a suspicious cutset by Algorithm 1 required only 4.81 seconds on average. Total CPU time for Algorithm 2 was 98,733 seconds (about a day). The maximum amount of memory used through our method

Table 4: Computation time for Fukui-TEPCO network

Algorithm 1 (average per test)	4.81 sec
Algorithm 2	98,733 sec

is 334 GB. This calculation time is long but acceptable, since it would be performed only once at the design stage (the naive method is, of course, unacceptable, because it takes a year even with the distance limit). Calculating a set of feasible configurations $\mathcal{C}$ requires a few hours as shown in [6].

5 Conclusions

In this paper, we have introduced an efficient method to find unrestorable cutsets exhaustively. Although unrestorable cutsets have not gathered much attention in the research community, it is a crucial tool with which to guarantee the restorability of distribution networks. Finding all unrestorable cutsets is a computationally tough problem, but we tackled it with efficient algorithms based on hitting sets and ZDDs; we first select a small number of suspicious cutsets from the many ones possible by using hitting sets, and performed feasibility tests against the selected cutsets with ZDD algorithms without complex power flow calculation. Experiments showed our method reduced the number of tests by five orders of magnitude (relative to the naive approach) with no significant false negatives. Each test was executed in just a few seconds in our method. Our work uncovered the vulnerability of a large-scale distribution network thoroughly for the first time ever.

In future work, we will consider the significance of each unrestorable cutset; e.g., the number of intact sections that cannot be restored under the cutset. We also will develop an optimal strategy to redesign the distribution network based on the significance of the failure and fixing cost.

We believe that our hitting set approach to vulnerability analysis offers generality to some extent and it can be successfully applied to other complex systems.

Acknowledgement

We would like to thank Takahisa Toda for his energetic support in implementing his hitting set enumeration algorithm.

References

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

[2] R. Billinton and P. Wang. Reliability-network-equivalent approach to distribution-system-reliability evaluation. *IEE Proceedings of Generation, Transmission and Distribution*, 145(2):149–153, 1998.

[3] Y. Hayashi, S. Kawasaki, J. Matsuki, H. Matsuda, S. Sakai, T. Miyazaki, and N. Kobayashi. Establishment of a standard analytical model of distribution network with distributed generators and development of multi evaluation method for network configuration candidates. *IEEJ Transactions on Power and Energy*, 126(10):1013–1022, 2006. in Japanese.

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

[5] T. Inoue, H. Iwashita, J. Kawahara, and S. Minato. Graphillion: Software library designed for very large sets of graphs in Python. Technical report, Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-13-65, 2013.

[6] T. Inoue, K. Takano, T. Watanabe, J. Kawahara, R. Yoshinaka, A. Kishimoto, K. Tsuda, S. Minato, and Y. Hayashi. Distribution loss minimization with guaranteed error bound. *IEEE Transactions on Smart Grid*, 5(1):102–111, 2014.

[7] D. E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms Part 1*, volume 4A. Addison-Wesley, USA, 2011.

[8] J. Lavaei, A. Rantzer, and S. Low. Power flow optimization using positive quadratic programming. In *Proceedings of the 18th IFAC World Congress*, 2011.

[9] W. Li, P. Wang, Z. Li, and Y. Liu. Reliability evaluation of complex radial distribution systems considering restoration sequence and network constraints. *IEEE Transactions on Power Delivery*, 19(2):753 – 758, 2004.

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

[11] C.-C. Liu, S. Lee, and S. Venkata. An expert system operational aid for restoration and loss reduction of distribution systems. *IEEE Transactions on Power Systems*, 3(2):619 –626, 1988.

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

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

[14] C. Nguyen and A. Flueck. Agent based restoration with distributed energy storage support in smart grids. *IEEE Transactions on Smart Grid*, 3(2):1029–1038, 2012.

[15] R. Perez-Guerrero, G. Heydt, N. Jack, B. Keel, and A. Castelhano. Optimal restoration of distribution systems using dynamic programming. *IEEE Transactions on Power Delivery*, 23(3):1589–1596, 2008.

[16] D. Shirmohammadi. Service restoration in distribution networks via network reconfiguration. *IEEE Transactions on Power Delivery*, 7(2):952 –958, 1992.

[17] T. Toda. Hypergraph transversal computation with binary decision diagrams. In *Experimental Algorithms*, volume 7933 of *Lecture Notes in Computer Science*, pages 91–102. Springer, 2013.

We give a sketch of proof for Theorem in Section 3.3.

Sketch of proof. We prove it by contradiction. Suppose the Theorem is false; there exists unrestorable cutset U that cannot be any suspicious cutset, which is mapped from a hitting set. This means that any combination of U's neighboring edges is not a hitting set, and so the network must be feasible upon removal of these edges. We analyze the following two cases; when removing vertices U and their edges, the network is (a) physically disconnected or (b) connected but does not satisfy the electrical constraint.

- (a) Removing some of U's edges, the network can be disconnected, and so it is infeasible, as shown in Figure 12 (a). This is a contradiction. (Strictly, let d be the maximum degree of a vertex in U, it is required to remove $d-1$ edges from each vertex of U at most to make the network disconnected.)
- (b) Removing some of U's edges, the network does not satisfy the electrical constraint and is infeasible, because it has more load than the network without U does, as shown in Figure 12 (b). This is a contradiction. (In this case, removing just a single edge from each vertex of U is enough to make the network electrically infeasible.)

∎

We have the Lemma shown in Section 3.4 from the proof, since unrestorable cutset U is mapped from hitting sets of $|U|(d-1)$ or less edges.

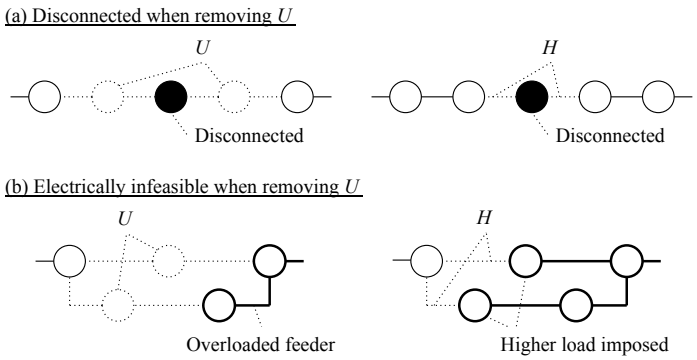

Figure 12: (left) Networks without vertices of U and their edges, and (right) those without some of U's edges (H). If the left networks are infeasible, the right network (a) is disconnected or (b) suffers from higher load, and so they must be also infeasible and a set of these edges is a hitting set.