Optimal Reconfiguration for Supply Restoration with Informed A* Search

Adi Botea, Jussi Rintanen, and Debdeep Banerjee

***Abstract*—Reconfiguration of radial distribution networks is the basis of supply restoration after faults and of load balancing and loss minimization. The ability to automatically reconfigure the network quickly and efficiently is a key feature of autonomous and self-healing networks, an important part of the future vision of Smart Grids.**

We address the reconfiguration problem for outage recovery, where the cost of the switching actions dominates the overall cost: when the network reverts to its normal configuration relatively quickly, the electricity loss and the load imbalance in a temporary suboptimal configuration are of minor importance.

Finding optimal feeder configurations under most optimality criteria is a difficult optimization problem. All known complete optimal algorithms require an exponential time in the network size in the worst case, and cannot be guaranteed to scale up to arbitrarily large networks. Hence most works on reconfiguration use heuristic approaches that can deliver solutions but cannot guarantee optimality. These approaches include local search, such as tabu search, and evolutionary algorithms.

We propose using optimal informed search algorithms in the A* family, introduce admissible heuristics for reconfiguration, and demonstrate empirically the efficiency of our approach. Combining A* with admissible cost lower bounds guarantees that reconfiguration plans are optimal in terms of switching action costs.

***Index Terms*—Smart Grid, power supply restoration, reconfiguration, search methods.**

I. Introduction

Reconfiguration of distribution networks is necessary for maintaining power supply for example during network maintenance and for restoring power after an outage and before the fault leading to it has been repaired [1]. It is also the basis of avoiding overloads and minimizing resistive losses [2].

Fully automatic and efficient reconfiguration is increasingly important for the Smart Grid, the future electricity networks which are envisioned to be autonomous and self-healing. For example, Brown [3] describes the Smart Grid as an "enabling system that can quickly and flexibly be reconfigured". An increasing number of distributed generation and storage devices will make distribution networks more complex and more dynamic, making manual or semi-automatic reconfiguration, due to its slowness, less of an option. Ipakchi and Albuyeh [4] argue that automated switching will be required to address overload, voltage and phase imbalance issues caused by the presence of a substantial number of plug-in electrical vehicles in future distribution networks. Fully automatic reconfiguration should be quick, efficient (in terms of its immediate and long term costs to network operation), reliable, and it should not compromise the safety by violating the operating constraints of the network.

Several approaches to the reconfiguration problem have been proposed. Zhou et al. [5] and Ciric and Popovic [6] use mixed integer linear programming, and Toune et al. [7] use tabu search. All of these works mention supply restoration as their main motivation. A number of other works primarily address loss reduction and load balancing, including the early work by Merlin and Back [8], which proposes a branch and bound algorithm, as well as more recent works, for example by Morton and Mareels [9]. In the latter, the focus is on loss minimization in the long term, and for this reason the cost of the switching operations needed to achieve the desired configuration is not important and it is ignored. Loss minimization is also the focus of recent work by Rao and Narasimham [10]. They introduce a greedy heuristic based on the idea that closing an open switch with a larger voltage difference between its two adjacent lines results in a larger loss reduction. Every time a switch is closed, another switch is opened to maintain the radial structure of the network. In the presence of distributed generation, when small scale power generators are present inside the distribution network, the radiality requirement must be relaxed [11]. We will discuss the issues arising from distributed generation later.

Most of the above works cannot give guarantees for the quality of the reconfiguration plan, in terms of the cost of executing the plan. Those that can, do it so by exhaustive enumeration of all possible plans, which is infeasible for distribution networks of realistic sizes.

As suggested earlier, several criteria can be used to measure the quality of a reconfiguration plan. The most common ones include minimizing the area left without power, supplying most important customers such as hospitals with a higher priority, minimizing the cost of the switching actions, and minimizing the power loss in the system. Some techniques aim at finding plans that are good according to several criteria, treating the problem as a multi-objective optimization problem [12], [13]. Ahuja *et al.* [12] use an algorithm that combines ant colony optimization with artificial immune systems. Kumar *et al.* [13] describe a technique based on genetic algorithms. In practice, such techniques can find good solutions. However, being based on non-systematic search, such as ant colony optimization or genetic algorithms, they cannot guarantee finding optimal solutions and cannot determine whether a solution they have found is optimal. In contrast, systematic search

Adi Botea is with IBM Research, Dublin, Ireland.

Jussi Rintanen is with the Australian National University and the Griffith University, Australia.

Debdeep Banerjee is with the Australian National University.

Part of this research was conducted while all authors were affiliated with NICTA, Australia.

algorithms, such as A* [14], do provide solution optimality guarantees and are guaranteed to find an optimal solution given enough time and memory resources.

In this work we propose an A*-based algorithm that is guaranteed to find a least-cost reconfiguration plan. The cost of a plan is the sum of the costs of the switching actions in the plan. We restrict our attention to *level 1* plans, which never move loads from a healthy feeder to another feeder. The resulting configuration satisfies capacity and other constraints, and will supply power to all non-faulty parts of the network, unless the capacity constraints or the network topology makes this impossible.

In our algorithm, we combine two quality criteria, switching costs and supplied area, and additionally require that all capacity constraints are respected. We ignore other quality criteria, such as minimizing resistive losses. In the case of reconfiguration for maintenance or outage recovery, the original configuration of the network is restored after the necessary repairs have been made, and therefore the temporary configuration will be used only for a relatively short time. Hence, maximizing the supplied area and minimizing the cost of performing the switching actions are more important than the minimization of resistive losses.

The A* search algorithm [14] and related algorithms such as IDA* [15] have two important properties. First, they are guaranteed to find a best possible solution whenever solutions exist. Second, among all such search algorithms that traverse the search space one state at a time, they are guaranteed to make the smallest possible number of search steps [14].

Applying A* to the reconfiguration problem requires defining the search space, the cost function to optimize, and an admissible heuristic (cost lower-bound function) that justifies pruning the search space without losing optimal solutions. The admissible heuristic is a lower bound on the cost-to-go from the current state to a goal state. Admissibility guarantees that an optimal solution is found [16]. The informativeness of a heuristic (that is, its accuracy in terms of the difference between the real and the estimated cost) can have a strong impact on the overall performance of a search algorithm.

The main scientific contribution of this work is the definition of an informative and admissible heuristic for reconfiguration in outage recovery.

Our search space is a collection of network configurations, characterized by the open/closed status of all switches in the network. This low-level state encoding is converted into a more structured representation, which identifies feeders and isolated regions that need to be resupplied with power.

As said earlier, the cost of a switching plan is defined as the sum of the costs of the individual switching actions. Different actions can have different costs. For example, one can consider monetary costs reflecting the wear and tear of the switching devices, and the cost incurred by delays when recovering from outages. Variable action costs allow to encode a problem more realistically but, on the other hand, they often tend to increase the difficulty of a cost-optimal search. As a particular case, if all switching actions in a given network happen to have the same cost, then minimizing the cost of a plan is equivalent to minimizing the number of actions in the plan.

The target (goal) configuration is defined in terms of the conditions it has to satisfy. These include the requirements that capacity constraints of the network are not violated, and that all customers will be supplied (except for customers at the faulty lines).

We evaluate our ideas with networks of a realistic size, containing up to over 200 lines and 200 switches. The results show that our algorithm can compute cost-optimal switching plans quickly in both single-fault and multiple-fault scenarios.

The rest of this paper is structured as follows. The next section reviews further related work. It is followed by a section that defines in detail the problem that we address. In Section IV we provide background information on search algorithms. Section V provides algorithmic details of our approach, including a description and analysis of our new admissible heuristic. An experimental evaluation is presented in Section VI, followed by concluding remarks.

II. Related Work

Heuristic search has been recognized as a viable approach to computing switching plans for distribution networks. However, many of the earlier works do not understand "heuristics" in the same sense as we do, as an *estimate* of the solution cost. Nevertheless, some of the earlier works do use systematic search algorithms for finding reconfiguration plans.

Morelato and Monticelli [17] generate a search tree by defining one binary decision variable for each switch in a network. A node in the search tree is a partial assignment of the binary variables. At each node in the search tree, one variable (switch) is selected for branching, and each of the two possible values (open and closed) generates one branch in the tree. The authors mention breadth-first, depth-first and best-first search as possible search strategies and test their ideas with a depth-first search implementation without using a formalized heuristic, and without considering algorithms that can in practice guarantee the optimality of solutions.

Wu *et al.* [18] use a representation of a search space that is different from that of Morelato and Monticelli [17]. A node encodes a network configuration. A transition from a parent to a successor node is a two-step action (Close Open), chosen in such a way that the radial structure of the network is preserved.

Devi *et al.* [19] address the supply restoration problem by using breadth-first and best-first search strategies. As in [18], states in the search space are complete network configurations, not just partial assignments of binary switch variables. The quality of solutions is defined as the number of switching actions. Best-first search is combined with a heuristic that selects for expansion a node with the smallest overloading. Similarly to the previous algorithms, this work work cannot guarantee the optimality of solutions.

Non-systematic search methods such as genetic algorithms, tabu search and simulated annealing have often been applied to power supply restoration [7], [20], [21], [22]. Since these methods do not keep track of the already visited parts of the search space, they are inherently incomplete and unable to guarantee the optimality of solutions they find.

III. Problem statement

The literature shows a range of variations of the power supply restoration problem, for example with differences in the definition of the state space, the type of solutions sought, and the optimality criteria. Hence it is necessary to formally define the features of the problem that we address in this work.

A *distribution network model* is a graph where nodes model elements of the physical network and edges model connections between elements. We focus mainly on lines, buses and switches. As discussed later in this section, other network elements, such as substations and transformers, can be modelled if desired. Our definition of switches covers all forms of switchgear, including circuit-breakers and reclosers, that can be opened and closed to respectively disconnect and connect two parts of the network. Each element has one or more connections to adjacent elements in the network. A switch has exactly two connections. A line connects with other elements of the network only at its two end points.

A switch can be either open or closed. In the open state a switch physically disconnects the two elements adjacent to it. Switching actions change the state of a switch. The cost of a switching action for a switch s is $\text{cost}(s) > 0$. The cost of a switching plan is the sum of the costs of its switching actions. A plan is cost-optimal if no plan with a smaller cost exists.

Distribution networks have a *meshed topology*, which means that there can be many possible ways (paths from a source) to feed a given load. A meshed topology is useful because it provides several options for reconfiguration. Despite the possibility to feed a load via multiple alternative pathways, distribution networks are usually kept in a *radial configuration*, where a load is fed from only one source, via only one path. This is achieved by setting the open/closed status of disconnectors (switches) in such a way that, if we ignore paths that contain open disconnectors, the network is a collection of tree structures, called *feeders*. Let us emphasize that feeders are disjoint and contain no cycles.

For simplicity, upstream elements such as a zone substation and a transformer that provide power to a given feeder are not represented explicitly as nodes in the graph model. They are abstracted into the root node of the feeder at hand. We assume that the root node of a feeder is a circuit breaker (e.g., a circuit breaker between the transformer and the feeder). Our model, however, is not limited to such an assumption. For example, when several feeders share a substation of a given capacity, but each feeder has a transformer with its own capacity, the substation could become a root node, and each transformer would become an interior node in the graph model. Each of these nodes would be assigned the capacities of the network elements that they model.

We call the feeder root nodes *substation circuit breakers*, to differentiate them from circuit breakers inside the distribution network. As shown later, the root node is assigned a capacity that reflects the capacity of upstream elements such as a transformer or a substation.

Figure 1 shows a sample network used as a running example in the paper. Substation circuit-breakers are drawn as triangles. Other switches are boxes, which can be open (gray contour

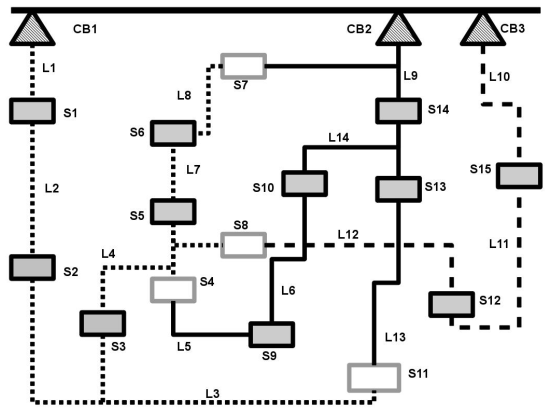


Fig. 1. Toy network used as a running example.

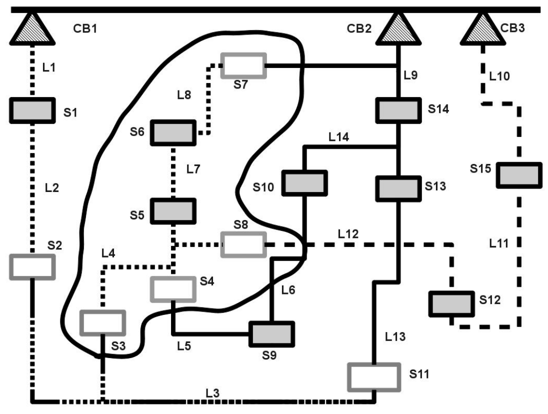


Fig. 2. Example of an isolated region.

and white background) or closed (black contour and gray background). The lines that belong to the same feeder are drawn using the same line pattern (e.g., solid, dashed or dotted). In the picture, loads are skipped for simplicity.

An *isolated region* is a maximal contiguous sub-network that contains no *faulty* lines and that is not supplied with power. If every path from an isolated region r to any substation circuit-breaker contains at least one faulty line, then r cannot possibly be supplied with power. Hence such isolated regions are ignored in the rest of the paper.

Figure 2 shows an example of an isolated region. It contains lines L4, L7, L8 and the two interior switches S5 and S6. The switches on the frontier are S3, S4, S7 and S8. The isolated region is a result of a permanent fault on line L3. To isolate the faulty line, all adjacent switches (S2, S3, S11) are opened and, as a side effect, the isolated region is left without power.

Given an isolated region or feeder r, let $\mathcal{L}(r)$ be the set of all modelled elements (i.e., graph nodes) in r, except for switches. The set $\mathcal{S}(r)$ contains all switches, both open and closed, linked to at least one element that belongs to r. The frontier $\mathcal{F}(r)$ is the collection of all open switches that connect an element in r with an element outside r:

$$\mathcal{F}(r) = \{s \in \mathcal{S}(r) | s \text{ is open} \land |\mathcal{L}(r) \cap \mathcal{E}(s)| = 1\}.$$

For a switch s, $\mathcal{E}(s)$ contains the two elements adjacent to s.

The set $\mathcal{C}(r)$ contains all closed switches inside r:

$$\mathcal{C}(r) = \{s \in \mathcal{S}(r) | s \text{ is closed}\}.$$

A substation circuit-breaker b is assigned a measure $\kappa(b)$ representing the maximum power flow that can be supplied by the upstream network through it. In practice, this is determined by considering the capacity of upstream network elements such as the adjacent transformer or substation.

Let us introduce the capacity and load constraints that involve elements such as buses, lines, and breakers. An element in the graph model e has a capacity $\kappa(e)$ and a local load $\lambda(e)$. The local load can be zero. We assign local loads to load buses, which thus will have a positive λ value. However, the graph model is flexible enough to allow assigning arbitrary capacities and local loads to any elements.

We assume that the κ and the λ values are known and fixed. The load $\lambda(r)$ of a sub-network r, such as a feeder or an isolated region, is the sum of the local loads of the contained elements:

$$\lambda(r) = \sum_{e \in \mathcal{L}(r)} \lambda(e).$$

The total load of a node e in the graph model is computed recursively by summing up the loads of the elements downstream, as well as the local load of e, if any. For example, for a line l, the total load $\gamma(l)$ is the sum of the loads of all elements in the feeder's sub-tree t_l rooted at l and going downstream from l:

$$\gamma(l) = \sum_{e \in \mathcal{L}(t_l)} \lambda(e).$$

For a substation circuit-breaker b, $\gamma(b)$ is defined as the load of the feeder f_b rooted at b, which is the sum of all local loads on all elements of f_b:

$$\gamma(b) = \lambda(f_b) = \sum_{e \in \mathcal{L}(f_b)} \lambda(e).$$

The constraints that we consider state that the total load downstream a given network element e (e.g., line, transformer, substation circuit breaker) should not exceed the capacity of that element:

$$\gamma(e) \leq \kappa(e).$$

The reconfiguration problem we address in this work covers outage recovery after the emergence of permanent faults. We assume that after a fault appears, the circuit-breakers and reclosers go through their pre-programmed sequence of operations to try to recover from the fault, and after this program has completed, the permanent fault has left part of the feeder without power.

More specifically, after a permanent fault, one of the circuit-breakers on a path from the fault to the substation of the corresponding feeder opens automatically. Possible reclosing actions may try to resupply power, but the reclosers will eventually give up and remain open, leaving a part of the feeder without power. The power supply restoration problem involves finding a sequence of switching actions that accomplish the following objectives. First, isolate the fault by opening the nearest switches downstream and upstream. As soon as the fault has been isolated, the circuit-breakers upstream can be closed to resupply part of the feeder. Second, the outage area downstream from the fault is resupplied by further switching actions. The first part is computationally easy (assuming that there is no uncertainty about the fault location). The second part is more challenging. This is the part that we attack with a cost-optimal search algorithm.

As some existing types of switches can't operate under load, some switching actions in a reconfiguration plan might require further attention to make sure they operate safely. We ensure this by temporarily opening the upstream circuit breaker that is the *closest* to the switch at hand s. This minimizes the network areas that will be temporarily disconnected. The additional costs brought by such extra circuit breaker operations need to be taken into account at search time. This can be achieved by adding the extra costs to the cost of operating the switch s. When more than one switch require (temporarily) opening the same circuit breaker, then the extra costs are added only once.

In this work we seek only *level 1* plans, which do not allow to transfer loads from one healthy feeder to another for example to decrease the load of the former. Level 1 plans are often, but not always, sufficient for resupplying power to all healthy lines. Level n plans, which allow load transfers within chains of up to $n > 1$ feeders, are sometimes needed to avoid violating capacity constraints.

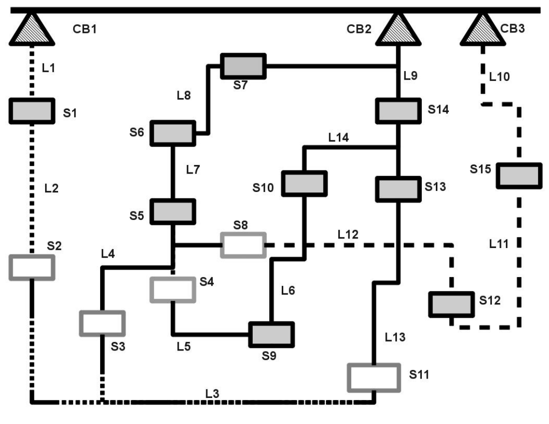
Depending on the quality criteria at hand, goal states (network configurations) can be defined in two alternative ways. A *weak* definition considers as goal states all configurations that respect the load and capacity constraints. Having regions without power is acceptable. Each state has a quality score that measures how much of the network is still without power. According to the weak definition, states where no region is left without power (except, of course, for the faulty lines) are considered optimal.

A *stronger* definition of goal states require that *all* parts of the network (again, except for the permanently faulty lines) are supplied power. An equivalent way to distinguish between weak and strong solutions is that the former allow isolated regions without power, whereas the latter do not.

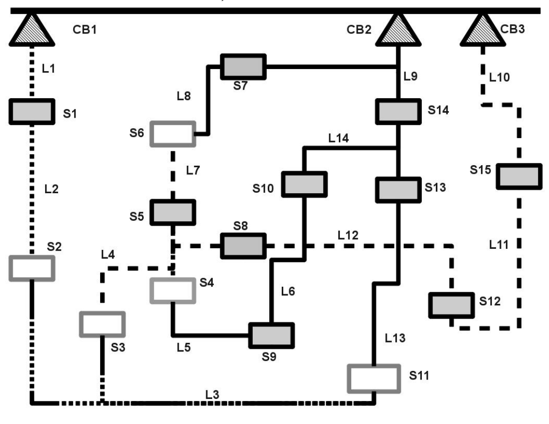
In this work we consider the strong definition. It allows optimizing two distinct quality criteria at a time. First, we require re-supplying all non-faulty regions. Secondly, we seek switching plans of optimal cost (as indicated earlier). Optimizing the cost is more general than minimizing the number of switching actions. The two options are identical when all actions have the same cost. However, in general, different switching actions have different costs.

Figure 3 illustrates two restoration plans for the isolated region shown in Figure 2. For simplicity, the actions for isolating the faulty line L3 and reclosing CB1 are not shown here. We also skip in this example extra circuit breaker operations that might be required to avoid operating switches under load. The first plan resupplies the entire isolated region by connecting it to the feeder of CB2 with one close action. The second plan feeds different parts of the isolated region from different feeders.

Notice that the cost of the first plan is lower than the cost of the second one, since the actions in the latter are a proper



a) Close S7



b) Open S6 Close S7 Close S8

Fig. 3. Sample plans.

superset of the former. However, depending on the actual loads and capacities, it might be the case that the first plan is not valid. For example, assume there is a load bus with a local load of 1 unit attached to that each line in the network. If the capacity of the CB2 circuit-breaker is $\kappa = 7$, then the first plan is invalid because it would require CB2 to feed 8 load buses, which exceeds its capacity.

Assuming that it is possible to resupply all disconnected parts, the algorithm will find a strong solution of minimal cost. In addition, the best *partial* restoration plan (i.e., weak plan) encountered so far in the search, before finding a complete solution, can be cached and provided on demand. This results in an *any-time* functionality, allowing it to provide a weak solution fast and to gradually improve the quality of weak solutions until a strong solution of optimal cost is found or the algorithm proves that no strong solution exists.

IV. Background on Search

Systematic search strategies, such as breadth-first, depth-first and best-first search, explore a space by expanding one *node* at a time. A node encodes a state (i.e., a network configuration in this work) and other information, such as a pointer to the parent node and the cost of the path from the root node to the current node. Expanding a node means generating its successors and adding them to a list of nodes that have been generated but not expanded yet (the Open list). The Open list is initialized to the root node, which represents the initial state of the problem. The exploration continues until a goal node is about to be expanded (in which case a solution has been found) or until the Open list becomes empty (the problem has no solution). Each expanded node is moved from the Open list to the Closed list.

Keeping track of all nodes visited so far (in the Open and Closed lists) is useful for both detecting duplicate states during search and for reconstructing the action sequences that lead to a goal node.

Completeness is an important property of search algorithms. An algorithm is *complete* if it is guaranteed to find a solution whenever one exists.

A* [14] is a complete search algorithm that maintains its Open list ordered according to the f-values of the contained nodes. The f-value of a node n is computed as $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the path from the root node to n, and $h(n)$ is an estimate (called a heuristic) of the cost from n to a goal node. Nodes with a smaller f-value are considered more promising.

A heuristic h is *admissible* if it never overestimates the true smallest cost h^* to a goal node: $\forall n, h(n) \leq h^*(n)$. Using admissible heuristics guarantees that the solutions returned by A* are cost-optimal. The cost of the solution is the sum of the costs of all actions on the path from the root node to the goal node. The larger the value of a heuristic (i.e., closer to the true optimal cost), the faster A* tends to be. Admissible heuristics with larger values are said to be more informative.

Consistency is a property stronger than admissibility (i.e., every consistent heuristic is also admissible). A heuristic h is consistent if, for every parent–successor pair (n, n'), we have $c(n, n') + h(n') \geq h(n)$, where $c(n, n')$ is the cost of the transition from n to n'. The advantage of consistent heuristics is that A* returns cost-optimal solutions without having to expand a state more than once. For more details on A* and heuristic search, see [16].

If no heuristic is used (i.e., nodes are sorted in the Open list according to their g values), then A* is equivalent to uniform-cost search, a blind search strategy that returns cost-optimal solutions [23]. Furthermore, when all actions have the same cost, these two become equivalent to breadth-first search.

When optimality is not required or it is too difficult to achieve, the WA* (Weighted A*) algorithm [24] can be used instead. This is a variant of A* with a parameter W which controls the quality of solutions, trading solution quality to computation time. As opposed to other sub-optimal search algorithms, WA* provides an upper bound on the solution sub-optimality. For example with $W = 1.1$ the algorithm will return a solution with a cost at most 1.1 times the optimum. A higher value of W may lead to finding solutions much faster.

V. Our Reconfiguration Algorithm

The search algorithm we use as the basis of our reconfiguration algorithm is A* [23]. We enhance the standard algorithm with a new heuristic h_A specific to the reconfiguration problem, and with a specialized technique for generating successors. We call h_A the *additive heuristic*.

As shown in Section IV, the heuristic determines how nodes are ordered in the Open list. Besides guiding the search, the additive heuristic has the ability to detect dead-ends in the search space. A dead-end is a state from which no goal state can be reached. Identifying dead-end states eliminates the need to search their subtrees and thus it can lead to significant speed-ups.

The successor generation implements a *partial order reduction* strategy [25], which reduces the number of action sequences that need to be considered. The basic idea is to avoid considering two unrelated actions a_1 and a_2 in both orders, a_1 followed by a_2, and a_2 followed by a_1. This reduces the amount of search needed, without compromising the completeness or the optimality of the algorithm.

Recall that a state (network configuration) σ in our search space is characterized at a low level of abstraction by the open/closed status of all switches and circuit-breakers. Starting from this representation, larger parts of the network, such as feeders and isolated regions, are identified and used in the computation of the heuristic and in the successor generation procedure. The rest of this section provides details on both the heuristic and the successor generation.

A. Additive heuristic

Consider a network state σ. The heuristic $h_A(\sigma)$ estimates (as a lower bound) the cost of a switching plan that reaches a goal state starting from σ. It is defined as

$$h_A(\sigma) = \sum_{r \in \mathcal{I}(\sigma)} h_L(r),$$

where $\mathcal{I}(\sigma)$ is the set of all isolated regions in the state (network configuration) σ. For each isolated region r in the current state, $h_L(r)$ is a lower bound on the cost of resupplying r (i.e., having all loads connected to a substation without violating capacity constraints).

Now we focus on the computation of $h_L(r)$ for a given isolated region r. If k is a lower bound on the number of close actions on the frontier of r required to resupply r, then we need to open at least $k-1$ internal switches in order to maintain the radial structure of the network. Hence a lower bound on the cost to resupply the isolated region r is

$$h_L(r) = k \times \min_{s \in \mathcal{F}(r)} \text{cost}(s) + (k-1) \times \min_{s \in \mathcal{C}(r)} \text{cost}(s).$$

As a simple example, if all switching actions have a constant cost of 1, then $h_L(r) = 2k-1$. We remind the reader that $\mathcal{F}(r)$ contains the frontier switches and that $\mathcal{C}(r)$ is the set of all closed switches in r.

It remains to show how to compute k for a given isolated region r. Intuitively, we compute the maximum amount of power that every switch on the frontier could possibly provide, and then count how many switches would be necessary to provide all the power needed inside the isolated region. Formally, for every open switch s between r and a feeder f, we define the maximum input $\mu(s)$ of s as being the largest amount of power that can be provided by f through s without violating the capacity and load constraints for f. It is computed as

$$\mu(s) = \min_{e \in \pi} (\kappa(e) - \gamma(e)) \tag{1}$$

where π contains all elements e, along the path from f's substation breaker to s, for which capacity constraints have been defined (e.g., elements e can correspond to lines and the substation breaker). The factor $\kappa(e)-\gamma(e)$ is sometimes called the residual capacity of that element [18].

For an open switch s between r and another isolated region r', an upper bound on the maximum input of s is

$$\mu(s) = \max_{e \in \mathcal{E}(s)} (\kappa(e) - \lambda'(e))$$

where $\mathcal{E}(s)$ contains the two elements connected to s, and $\lambda'(e)$ is the load (if any) connected to e (e.g., via a bus) without any switch between e and the load. For all other switches on the frontier of r (e.g., a switch to a faulty line), the maximum input is set to 0.

The switches on the frontier $\mathcal{F}(r)$ are ordered decreasingly according to their maximum input μ. Then, as part of the so-called *k-iteration process*, we keep adding up their maximum inputs until we reach (or exceed) the load $\lambda(r)$ or until all switches have been considered. In the first case, the number of switches considered gives the value k. In the second case, where $\lambda(r)$ cannot be reached even after considering all frontier switches, a dead-end has been discovered and therefore $h_L(r)$ is set to ∞.

The additive heuristic is computed for every state that is encountered in a search. There is a low upper bound on the effort of computing the additive heuristic. Assume that R is an upper bound on the size of an isolated region, F is an upper bound on the size of a feeder, N is an upper bound on the number of the isolated regions, and K is an upper bound on the frontier of an isolated region. Frontier switches between two isolated regions bring only a constant overhead, so they can safely be ignored in this complexity discussion.

When the additive heuristic is computed from scratch, an upper bound on the computation time is

$$O(N \times [(K \times F) + R]).$$

Notice that the values N, K, F and R for an arbitrary state depend on N_0, K_0, F_0 and R_0, the corresponding values in the initial state. For example, N can never increase along an exploration path (i.e., $N \leq N_0$), and F cannot exceed $F_0 + N_0 \times R_0$. In the latter case, equality could be reached when all isolated regions are connected to the same feeder. R cannot exceed $N_0 \times R_0$. The equality could be reached when all isolated regions are first connected to each other in a tree structure. Likewise, $K \leq N_0 \times K_0$.

If desired, the computation of the additive heuristic can be performed incrementally, starting from the value of the parent state. In such a case, the computational requirements are even lower. Only the values that could possibly change have to be re-computed. For example, if a feeder f is not affected by the most recent transition, then all values $\mu(s), s \in \mathcal{F}(f)$ can be re-used from the values computed for the parent state. If the topology of an isolated region r and all the μ values of the switches on its frontier are the same as for the parent state, then the entire value $h_L(r)$ can be re-used from the value computed for the parent state.

B. Other heuristics

In the empirical evaluation reported later we have experimented with two additional heuristics, h_B and h_S, as alternatives to h_A. The first is the trivial blind (null) heuristic $h_B(\sigma) = 0$. With h_B the A* algorithm reduces to blind search. The heuristic h_S is defined as the number of remaining isolated regions in the corresponding state: $h_S(\sigma) = |\mathcal{I}(\sigma)|$. That is, h_S optimistically assumes that each of the remaining isolated regions can be resupplied with only one close action. The h_S heuristic is faster to compute (constant time) than h_A. On the other hand, it can easily be shown that, for any state σ, $h_B(\sigma) \leq h_S(\sigma) \leq h_A(\sigma) \leq h^*(\sigma)$, which suggests that h_A is more informative than h_S. In particular, h_S lacks the ability to detect dead-ends in the search space. See Section VI for an empirical comparison of these heuristics.

C. Successor generation

Search algorithms require a procedure to expand a state (i.e., to generate its successors). A successor state is obtained by performing a transition in the parent state. For each state that is being expanded, we consider two types of transitions. The first type is CLOSE s, where s is a switch on the frontier of an isolated region. Such transitions are useful to connect an entire isolated region with only one switching action. Transitions of the second type are two-action sequences (OPEN s_1 CLOSE s_2), where s_1 is a closed switch inside an isolated region r and s_2 is an open switch on the frontier of r. The cost of a two-action transition is $\text{cost}(s_1) + \text{cost}(s_2)$. Such transitions are useful to connect only a part of an isolated region, which is needed when the load and capacity constraints do not allow resupplying an isolated region with only one switching action.

Grouping switching actions into OPEN-CLOSE pairs is a common strategy reported in the literature (e.g., [18]). We emphasize that this grouping is relevant only when searching for a switching plan, not necessarily when executing the plan. The grouping has no impact on the order in which the switching actions will be executed. The execution order takes into account conditions such as the fact that certain types of switches cannot operate under load.

D. Partial order reduction

When modeled as a search problem, the supply restoration problem features an additional challenge caused by the fact that many orderings of the switching actions are equivalent. This is a well known feature of the problem and work-around solutions specific to various problem formulations have been proposed [18], [17].

As a simple example, consider a network with three isolated regions r_1, r_2 and r_3. For each isolated region r_i there is an action a_i that resupplies it. All sequences $a_1a_2a_3, a_1a_3a_2, \ldots a_3a_2a_1$ are equivalent. Therefore, exploring only one of them will do. In this example there are $3! = 6$ such combinations. The number blows up as the number of isolated regions or the number of actions that are relevant to a given isolated region increase.

To limit the impact of the high number of action orderings, our search strategy focuses on resupplying one isolated region at a time. The search never generates transitions that are relevant to a different isolated region. As soon as the current isolated region is resupplied, a new isolated region among the remaining ones is chosen to be next. This idea preserves both the completeness and the (global) optimality of solutions. Its only effect is that, among several equivalent paths to the same state, some of the paths are pruned away.

In the example, assuming that the three regions are resupplied in the order r_1, r_2, r_3, the algorithm would explore the sequence $a_1a_2a_3$ but would prune all other combinations.

More generally, two or more actions that are relevant to two or more different isolated regions are explored in a fixed order. Irrelevance of the ordering of actions for the same isolated region is still not observed properly, but the number of different but equivalent orderings of actions for one region is much smaller than the number of orderings when actions for different regions are considered.

E. Adaptation to Distributed Generation

Distributed generation is an important part of Smart Grids, leading to more efficient and reliable distribution networks [26], [27], [28]. Part of the increased reliability is due to the possibility of supplying parts of the distribution network from the local generation capacity even when parts of the distribution network have no power. A main change caused by distributed generation is the loss of radiality of the network: a load may be simultaneously supplied both from the substation and a small-scale local generator.

In fault situations it may be useful to operate part of a distribution network in an islanding mode, in which the distributed generators are the sole source of power. In this setting the generator can simply be viewed as a distribution substation, except that the loads connected to the generator have to closely match the power being generated.

Our algorithms can be adapted to electricity networks with distributed generation. The exact way of adapting the algorithms depends on the assumptions made about how distributed generation works in the model at hand.

For example, let us make the following assumptions. Each feeder maintains a tree structure, but a feeder can contain zero or more distributed generators, besides its connection to a substation (circuit-breaker). Hence one load could be supplied at the same time from several sources (at most one substation and zero or more local generators). The output (which we call capacity κ to be consistent with our terminology) of each local generator is fixed and known.

We perform the following two modifications to the additive heuristic h_A, obtaining a new heuristic h_D that works in the distributed generation framework outlined earlier.

First, we need to take into account that the isolated region at hand r might have local generators available. Therefore, in the k-iteration process, we replace $\lambda(r)$ with $\lambda(r) - \beta(r)$. The term $\beta(r)$ is the sum of outputs (capacities) of all local generators contained in r.

Second, in Equation 1, the presence of local generators in the feeder at hand f might increase the maximal input μ of a frontier switch s. In Equation 1, the residual capacity of each element e could increase because local generators in e's subtree could supply part of the combined load $\gamma(e)$. Thus, in the modified computation of $\mu(s)$, $\gamma(e)$ is replaced with $\gamma(e) - \beta(e)$, where $\beta(e)$ is the combined capacity of all local generators in e's subtree.

Assume, for simplicity, that local generators do not export power outside a feeder f through any frontier switch s. (The alternative case is equally easy but we skip it to keep the discussion short.) It can be shown that $\mu(s)$ computed as described earlier remains an upper bound on the maximal power that a feeder f could provide through a frontier switch s. In addition, the values k computed with the modified k-iteration process remain a lower bound on the number of the close actions needed to supply an isolated region r with all the needed power that is not generated locally inside r. Therefore, h_D is an admissible heuristic.

Properties of the reconfiguration algorithm stated earlier are preserved. For example, the algorithm will output a plan that will supply all customers, unless no such plans exist. Obviously, the availability of local generation increases the likelihood that all customers could be supplied after reconfiguration.

VI. Experiments

We have run experiments on two networks of different sizes. The *small* network has 54 circuit-breakers, 139 switches and 138 lines. The *large* one contains 81 circuit-breakers, 210 switches and 207 lines. For illustration purposes, we show the topology of about half of the small network in Figure 4.

The network topologies resemble distribution networks in suburban areas in Australia. Starting from these topologies, we have added the following parameters. The κ value of a given circuit-breaker is randomly set to either 20 or 100. Each switch has a randomly assigned but fixed cost from 1 to 5. Each line has a capacity of 10. A busbar with a local load of 1 is connected to each line.

We consider multiple-fault scenarios, with a maximum of 20 faults per network. Multiple simultaneous faults in one distribution feeder are unlikely, but possible for example during storms. The experiments with up to 20 simultaneous faults are just to demonstrate scalability to very complex fault scenarios.

For each number of faults between 1 and 20, we generate 50 problem scenarios with the faulty lines selected randomly in each case. As a result, we obtain 2,000 scenarios, 1,000 for each network. Some of the generated problem instances turned out to be too difficult for blind search (very quickly violating the memory bound), and we don't include statistics on those scenarios in the result diagrams.

Our program is implemented in Java 1.6. All experiments are run on a Linux machine with a 2.4 GHz Intel Dual Core processor. Each process is allocated 1.5 GB of memory.

We evaluate four versions of the algorithm described in Section V. Three versions are based on A* and differ only in the heuristic function that they employ. Each version uses h_B, h_S and h_A respectively. All three algorithm versions produce cost-optimal solutions. The main difference is in their speed and memory consumption. The fourth program version that we evaluated uses WA* algorithm with the h_A heuristic and $W = 2$.

Figures 5–7 summarize the results: runtimes, visited nodes and plan costs averaged over instances that have a strong solution. The actions to isolate faulty lines and to close the circuit-breakers are not included in the reported costs.

The curves characterizing the average runtimes for different numbers of faults are not very smooth. This can be explained by the fact that the runtimes of combinatorial search algorithms are usually heavy-tailed [29], which means that many of the problem instances are much more difficult than the instances on average.

Compared to A*, WA* is often much faster. Although with $W = 2$ the plans could be up to twice as expensive as the optimal ones, in practice, as indicated in Figure 7, the WA* plans differ from optimal only in a small number of cases and only slightly.

In the rest of this section, we focus on the performance of the three heuristics in A*. All algorithm versions are reasonably fast, indicating that computing cost-optimal switching plans is feasible. The empirical data confirms the informedness ranking of the three heuristics mentioned in Section V. The best performing one is the additive heuristic, which is followed by h_S. Even for scenarios with more than 10 faults, A* with h_A needs less than two seconds in average to find a plan. The time of computing h_A could be further reduced by the incremental computation described in Section V.

Recall that h_S optimistically assumes that every isolated region can be resupplied with one close action, whereas h_A computes a more sophisticated lower bound. Hence, the fact that h_A is more informative than h_S indicates that in some cases one single close action of minimal cost is not enough to resupply an isolated region and that in such cases h_A provides a more accurate estimation of the costs.

As suggested, for example, in Figure 6, the larger network does not significantly increase the difficulty of the problem. This can be explained by the fact that the algorithm searches for level 1 plans only: the complexity is exponential in the sum of the sizes of initially isolated regions, which can be arbitrarily smaller than the network size. The only switches that would possibly need to be operated while searching for a level 1 plan are those in $\mathcal{A} = \cup_{r \in \mathcal{I}(\sigma_0)} \mathcal{S}(r)$, and possibly fewer than $|\mathcal{A}|$ circuit breakers. Circuit breaker operations would be needed in those cases when switches in $\mathcal{A}$ can't be operated under load. Since each switch has two possible states, an upper bound on the size of the state space that needs to be explored is in the order of $2^{|\mathcal{A}|}$. This can be much smaller than $2^{|\mathcal{M}|}$, the size of the entire state space, where $\mathcal{M}$ contains all switches in the network.

The data in the three figures indicate that instances tend to grow in difficulty as the number of faulty lines increases (the zigzagging effect comes from the noise in the randomly generated fault scenarios). The growth in difficulty is as expected, being consistent with the brief complexity analysis outlined earlier. As long as the faulty lines remain fairly sparse in the network, increasing their number tends to increase the number of isolated regions (in the initial state) that need to

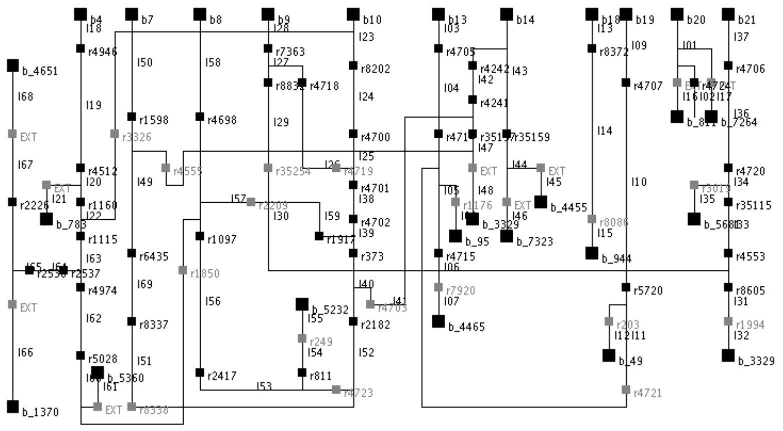


Fig. 4. The topology of part of the small network. Larger boxes are circuit-breakers and smaller boxes are switches. Grey boxes represent open switches.

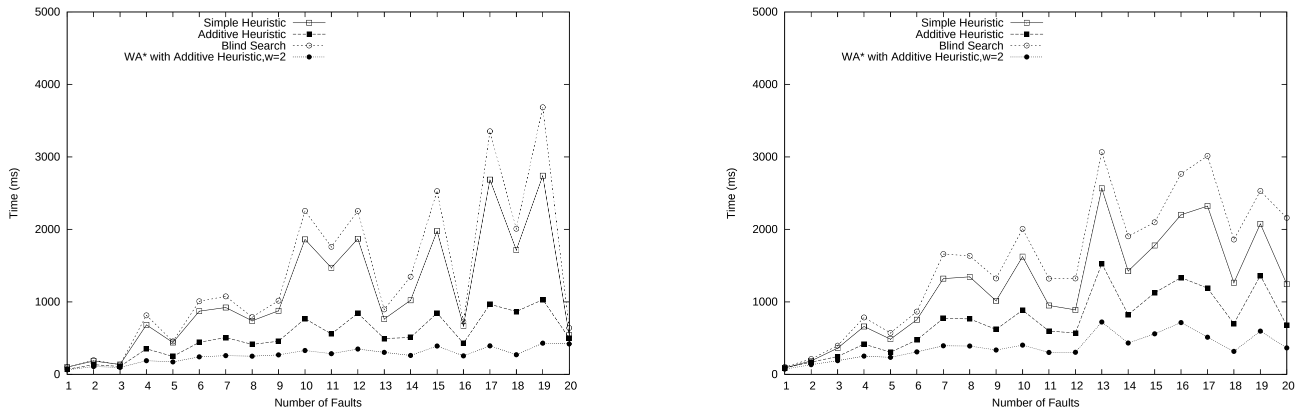


Fig. 5. Average running time for the small network (left) and the large network (right).

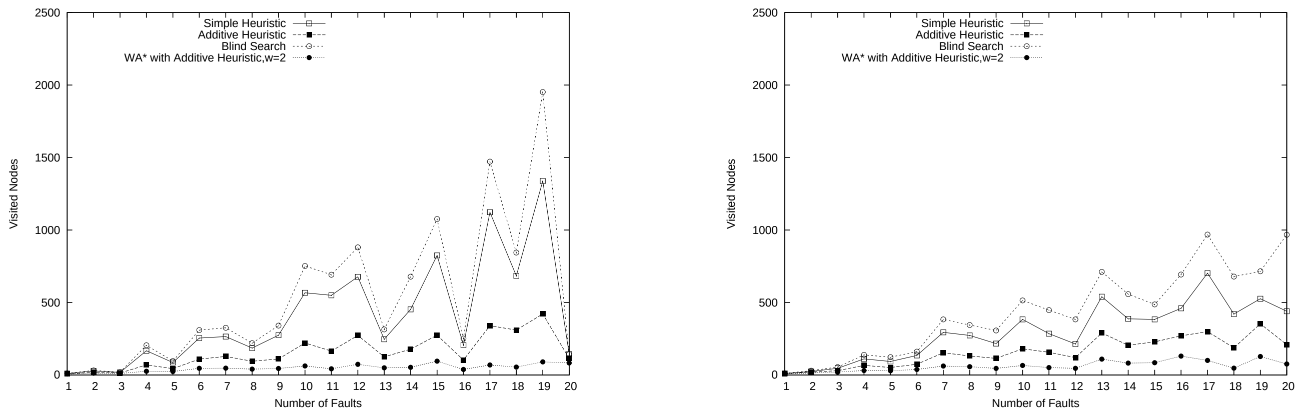
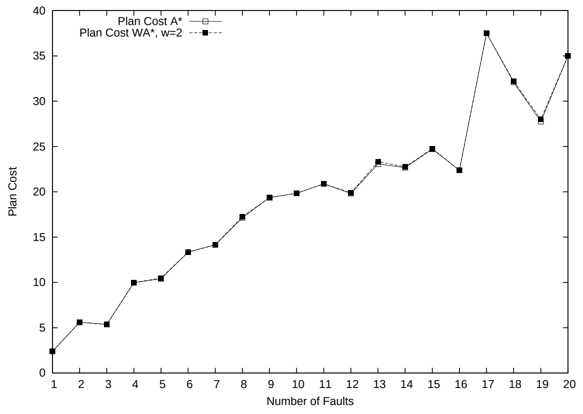


Fig. 6. Average visited nodes for the small network (left) and the large network (right).



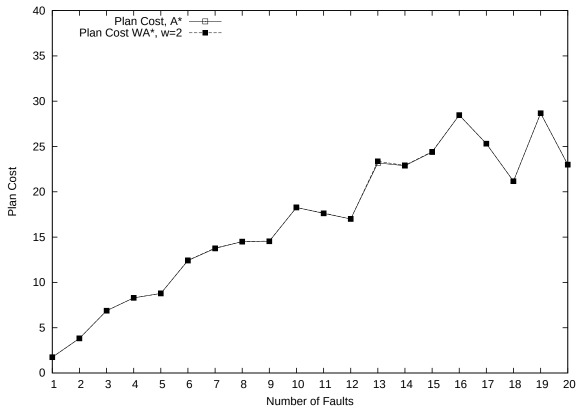


Fig. 7. Average plan cost for the small network (left) and the large network (right).

be re-supplied with power. It should not be concluded that, if the number of faults were further increased, the problem difficulty would always keep increasing. An extreme increase in the number of faulty lines will decrease the number of isolated regions that can be resupplied, which eventually will make the multi-fault problems very easy.

VII. Conclusion

The importance of reconfiguration seems to be even stronger in the next generation of power grids. We have presented a class of algorithms for finding reconfiguration plans that have a guaranteed minimal cost, and provided experimental data to illustrate its scalability to solve large multi-fault supply restoration problems for distribution networks of realistic structure and size. The work is based on systematic informed search algorithms in the A* family. We have also presented variants of these algorithms that find plans that are guaranteed to be only a constant factor more expensive than the optimal ones, and shown that in practice optimal or almost optimal solutions are found with a substantially reduced search effort.

The results of this work may seem to contradict the fact that the reconfiguration problem with capacity constraints is NP-complete [30] and that known algorithms consequently have an exponential time worst-case complexity. However, the exponential worst-cases might show up only with unrealistically complex network topologies and extremely strict capacity constraints.

Topics for future research include finding characterizations of distribution network structure which could guarantee the efficient solvability of reconfiguration problems, and the development of algorithms that cover more of the constraints that are likely to arise during reconfiguration, including the requirement to resupply priority customers quickly.

Another future research direction arises from the combination of remotely and manually controlled switches. The present work addresses the case of remotely controllable switches, in which all the switches can be operated simultaneously and almost instantaneously, and the actual total cost is the sum of the costs of individual switching operations. In the presence of manually controlled switches, the aggregation of individual costs is not as simple. In this case, the total duration of the reconfiguration plan is likely to have cost implications (for example in terms of the duration of the outage). The operation of the manually controllable switches can be carried by one or more field crews, and complex optimization may be required, involving route planning for the crews, decisions about how many crews dispatch, and how to assign the tasks to the crews.

For the Smart Grid scenario, when reconfiguration should take place quickly and fully automatically, the management of uncertainty has to be automated and cannot be delegated to control room operators. Specifically, the reconfiguration procedure should observe uncertainty about fault locations, and generate reconfiguration plans with the highest *expected value*, in terms of the likelihood of success. Also, the reconfiguration plans should be robust in the sense of being unlikely to disrupt power supply to priority customers, even in the case of incorrect initial assumptions about the fault location.

Acknowledgment

We thank the editor, the annonymous reviewers and Stephen Major for their expert feedback. We thank Alban Grastien and Patrik Haslum for their help in preparing the input data.

References

[1] S. Ćurčić, C. S. Özveren, L. Crowe, and P. K. L. Lo, "Electric power distribution network restoration: a survey of papers and a review of the restoration problem," *Electric Power Systems Research*, vol. 35, no. 2, pp. 73–86, 1996.

[2] S. K. Khator and L. C. Leung, "Power distribution planning: A review of models and issues," *IEEE Transactions on Power Systems*, vol. 12, no. 3, pp. 1151–1159, 1997.

[3] R. E. Brown, "Impact of smart grid on distribution system design," in *IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, 2008, pp. 1–4.

[4] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power and Energy Magazine*, vol. 7, pp. 52–62, 2009.

[5] Q. Zhou, D. Shirmohammadi, and W.-H. E. Liu, "Distribution feeder reconfiguration for service restoration and load balancing," *IEEE Transactions on Power Systems*, vol. 12, no. 2, pp. 724–729, 1997.

[6] R. M. Ciric and D. S. Popovic, "Multi-objective distribution network restoration using heuristic approach and mix integer programming method," *International Journal of Electrical Power & Energy Systems*, vol. 22, no. 7, pp. 497– 505, 2000.

[7] S. Toune, H. Fudo, T. Genji, Y. Fukuyama, and Y. Nakanishi, "A reactive tabu search for service restoration in electric power distribution systems," in *IEEE World Congress on Computational Intelligence. IEEE International Conference on Evolutionary Computation*. IEEE, 1998, pp. 763–768.

[8] A. Merlin and H. Back, "Search for minimum-loss operating spanning tree configuration in an urban power distribution system," in *Proceedings of the 5th Power System Computation Conference*, 1975, pp. 1–18.

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

[10] R. S. Rao and S. V. L. Narasimham, "A new heuristic approach for optimal network reconfiguration in distribution systems," *International Journal of Applied Science, Engineering and Technology*, vol. 5, pp. 15–21, 2009.

[11] R. Tamizkar, S. A. M. Javadian, and M.-R. Haghifam, "Distribution system reconfiguration for optimal operation of distributed generation," in *International Conference on Clean Electrical Power (ICCEP 2009)*. IEEE, 2009, pp. 217–222.

[12] A. Ahuja, S. Das, and A. Pahwa, "An AIS-ACO hybrid approach for multi-objective distribution system reconfiguration," *IEEE Transactions on Power Systems*, vol. 22, pp. 1101–1111, 2007.

[13] Y. Kumar, B. Das, and J. Sharma, "Multiobjective, multiconstraint service restoration of electric power distribution system with priority customers," *IEEE Transactions on Power Delivery*, vol. 23, pp. 261–270, 2008.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum-cost paths," *IEEE Transactions on System Sciences and Cybernetics*, vol. SSC-4, no. 2, pp. 100–107, 1968.

[15] R. E. Korf, "Depth-first iterative deepening: an optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[16] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company, 1984.

[17] A. L. Morelato and A. J. Monticelli, "Heuristic search approach to distribution system restoration," *IEEE Transactions on Power Delivery*, vol. 4, no. 4, pp. 2235–2241, 1989.

[18] J. S. Wu, K. L. Tomsovic, and C. S. Chen, "A heuristic search approach to feeder switching operations for overload, faults, unbalanced flow and maintenance," *IEEE Transactions on Power Delivery*, vol. 6, no. 4, pp. 1579–1586, 1991.

[19] V. S. Devi and G. Anandalingam, "Optimal restoration of power supply in large distribution systems in developing countries," *IEEE Transactions on Power Delivery*, vol. 10, no. 1, pp. 430–438, 1995.

[20] V. S. Devi and M. N. Murty, "Stochastic search techniques for post-fault restoration of electrical distribution systems," *Sādhanā*, vol. 25, pp. 45–56, 2000.

[21] Y. Fukuyama, "Reactive tabu search for distribution load transfer operation," *Power Engineering Society Winter Meeting, 2000. IEEE*, vol. 2, pp. 1301–1306, 2000.

[22] A. Furuta and H. Mori, "Application of parallel tabu search-based hierarchical optimization to distribution system service restoration," *Electrical Engineering in Japan*, vol. 164, no. 4, 2008, translated from Denki Gakkai Ronbunshi, Vol. 126-B, No. 1, January 2006, pp. 21–28.

[23] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[24] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial Intelligence*, vol. 1, pp. 193–204, 1970.

[25] D. Peled, "All from one, one for all: on model checking using representatives," in *Proceedings of the 5th International Conference on Computer Aided Verification*, ser. CAV '93. London, UK: Springer-Verlag, 1993, pp. 409–423.

[26] P. M. Costa and M. A. Matos, "Reliability of distribution networks with microgrids," in *Power Tech, 2005 IEEE Russia*, Jun. 2005, pp. 1–7.

[27] E. Carpaneto, G. Chicco, and A. Prunotto, "Reliability of reconfigurable distribution systems including distributed generation," in *International Conference on on Probabilistic Methods Applied to Power Systems (PMAPS)*, Jun. 2006, pp. 1–6.

[28] S. Kennedy, "Reliability evaluation of islanded microgrids with stochastic distributed generation," in *Power Energy Society General Meeting, 2009. PES '09. IEEE*, 2009, pp. 1–8.

[29] H. Chen, C. Gomes, and B. Selman, "Formal models of heavy-tailed behavior in combinatorial search," in *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, ser. Lecture Notes in Computer Science, T. Walsh, Ed., no. 2239. Springer-Verlag, 2001, pp. 408–421.

[30] T. Hadžić, A. Wąsowski, and H. R. Andersen, "Techniques for efficient interactive configuration of distribution networks," in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007, pp. 100–105.

Adi Botea obtained a PhD degree at the University of Alberta, Canada, in 2006, and a MSc degree at the University of Bucharest, Romania, in 1998. Both degrees are in computer science. He holds a research position at IBM Research in Dublin, Ireland. From 2006 to 2011, Adi was a researcher and senior researcher at NICTA. During the same period, he was an adjunct (senior) research fellow at the Australian National University. His research interests include artificial intelligence (AI) areas such as heuristic search, automated planning and path finding.

Jussi Rintanen received a B.Sc./M.Sc degree in computer science and engineering in 1992 and a Ph.D. degree in computer science in 1997, both from the Helsinki University of Technology, Finland. He held research and teaching positions at the universities of Ulm and Freiburg, Germany, from 1997 until 2005. From 2006 until 2011 he was a principal researcher and the leader of the model-based reasoning group at NICTA, Australia. Currently he is an adjunct associate professor at the Australian National University and Griffith University. His research interests include automated decision-making, combinatorial search, automated reasoning, and artificial intelligence, especially the automated supervisory control of complex systems, including the Smart Grid.

Debdeep Banerjee received his BSc degree in Computer and Information science in 2006 from the University of South Australia, and his Masters degree in Information and Communication Technology in 2007 from the Australian National University. Since January 2008 he is a PhD student at the Australian National University. His research interests include automated planning and scheduling, combinatorial search, and automated reasoning.