Sample Algorithms in Multi-start Heuristics for the Switch Allocation Problem

Alexander J. Benavides, Marcus Ritt, and Luciana S. Buriol

Instituto de Informática, Universidade Federal do Rio Grande do Sul - UFRGS
{ajbenavides, mrpritt, buriol}@inf.ufrgs.br

Abstract. We study the problem of allocating switches in electrical distribution networks to improve their reliability. We present a sample construction algorithm and a sample local search for this problem. We compare these approaches with other construction and local search strategies within heuristics that combine them. We present and analyze experimental results, showing that sample approaches are inexpensive and find solutions of good quality.

Keywords: local search, sample algorithms, switch allocation.

1 Introduction

According to Teng and Liu [12], most of the faults of an electrical power system occur in the distribution network. The most common method to improve the reliability of a distribution network is to add redundant lines with switches. Thus, in case of failures, the network topology is altered and the areas without energy supply are reduced. The installation of automatic switches in every line is impracticable due to high costs. Because of that, companies must choose carefully the places to install switches. This combinatorial optimization problem is called the switch allocation problem.

The remainder of the paper is organized as follows. Section 2 explains two related problems: service restoration and switch allocation. It also describes distribution networks using a graph model, and presents a method for network reliability estimation. Section 3 describes the construction algorithms (random, sample, greedy and semi-greedy) and the local search strategies (sample search, first and best improvement). Section 4 shows and discusses computational results. Concluding remarks are given in Section 5.

2 Description of the problems

Fig. 1 shows an example of an electric power distribution network [4]. It shows the network in normal operation. Due to electrical constraints, the basic circuit of an operational distribution network has no cycles. The basic circuit is composed by substations (square nodes), consumers (round nodes), and feeder lines (black lines). Redundant feeder lines with switches (dotted lines) exist to reduce the
time of blackouts in areas affected by failures. In normal conditions, switches of redundant lines are open, while switches on the basic circuit are closed. Because of this, redundant lines are called normally open and basic circuit lines are called normally closed.

2.1 Graph model of distribution networks

We model an electric distribution network as an undirected graph $G = (N, A)$, where the set of nodes $N = N_S \cup N_C$ represents the set of substations ($N_S$) and consumer load points ($N_C$), and the edge set $A = A_{nc} \cup A_{no}$ represents normally closed ($A_{nc}$) and normally open ($A_{no}$) feeder lines. We write $V(G) = N$ for the node set and $E(G) = A$ for the edge set of a graph or subgraph $G$. The presence of a switch on an edge $a \in A$ is indicated by a boolean value $B_a \in \{0, 1\}$.

We represent a solution for the switch allocation problem by a set $A_B \subset A$ of lines that are selected to install new switches ($A_B = \{a\}$, $B_a = 1$).

The sector $S(n)$ corresponding to a node $n \in N$ is defined as the largest connected subgraph of $G$ which contains $n$ and is connected only with basic circuit feeder lines that have no switch installed ($a \in A_{nc}, B_a = 0$). For any edge $a = \{u, v\}$ we define the corresponding sector $S(a) = S(u) \cup S(v) \cup \{\{u, v\}, \{a\}\}$ as the union of the sectors of the nodes that it connects. The frontier of a sector $F(S(n))$ is the set of edges $a \in A$ which are incident to exactly one node in the sector. We define the set of sectors $SS = \{S(n) | n \in N\}$ that contains all the disjoint sectors of nodes $n \in N$.

2.2 The service restoration problem

After a power failure is detected, the network topology must be modified to isolate the failure and to restore the energy supply by alternate feeder lines. The network reconfiguration is the process of opening and closing some switches in the feeder lines to change the topology. Fig. 2 shows an example of this
process. Consider a failure in line \{8,10\}. Without switches, the whole tree under substation 2 would be unattended. When the automatic switches on lines \{2,8\} and \{8,9\} are opened, the failure is isolated in sector \(S(8)\) (in dark gray). Then, sector \(S(9)\) (in light gray) is isolated from the failure but still unattended. When the automatic switch on line \{5,11\} is closed, the service is restored in sector \(S(9)\). The **service restoration problem** consists in choosing which switches must be opened or closed to minimize the unattended area after the isolation of a failure.

This problem becomes complex when it considers electrical constraints. For example, if there exist a loop line that can restore the energy supply to a gray sector, there still exist the possibility that the substation can not support it or that the voltage drops out of allowed limits. This paper considers the service restoration problem as a subproblem of the switch allocation problem.

### 2.3 The switch allocation problem

According to Levitin et al. [9], the number of unattended consumers and the amount of non-supplied energy depend directly on the number and position of the switches in the network. The **switch allocation problem** consists in selecting a set of feeder lines to install \(k\) new automatic switches in a distribution network. The objective is to maximize the reliability, and it is subject to the number of available switches for allocation and to the electrical constraints.

This problem has been studied with different approaches, e.g. simulated annealing approach [2], divide-and-conquer approach [3], genetic algorithm [5], tabu search [6], three state particle swarm optimization [10], ant colony optimization [7].

Many of the mentioned approaches use a simplification to calculate the unattended areas assuming that, for a given set of switches and a failure, the affected nodes are known or easy to compute, estimating reliability with
statistical data or assuming that gray sectors can be restored if there exists a loop line. This disregards the underlying service restoration problem with electrical constraints.

### 2.4 Network Reliability Estimation

We use expected energy non supplied (EENS) \([7]\) to measure the network reliability. The EENS is calculated as

\[
\text{EENS} = \sum_{f \in A_{nc}} \lambda_f r_f \sum_{n \in N_f} P_n \text{ (MWh/year)},
\]

where \(A_{nc}\) is the set of feeder lines that can fail, \(N_f\) is the set of affected nodes by a failure \(f\), \(r_f\) is the average outage time (in hours), \(\lambda_f\) is the average failure rate, and \(P_n\) is the energy normally consumed by node \(n\).

Our approach takes into account the service restoration problem as a subproblem of the switch allocation problem. Therefore, to estimate the reliability of a set of switch locations that represent a solution of the switch allocation problem, we must consider every possible failure, isolate it, maximize the restored area, and calculate the partial EENS.

We use the algorithm in Fig. 3 to estimate the reliability. This algorithm processes all the possible failures in lines of a sector \(S(n)\) together (lines 2-9), to save computing time. First, it simulates a failure in each sector from the sector set \(SS\). The black area is the current sector, so the failure does not need to be expanded and its frontier is known for isolation. Second, it determines the non-served load points with a service restoration algorithm. Third, it calculates the partial EENS of the consumers \(n \in N_f\) affected by the failure \(f\), evaluating it for every feeder line \(a \in E(S(f))\) in the black sector at once (line 7).

Note that frontier feeder lines (normally closed with switches) must still be processed separately (lines 10-17), because they are not part of any sector. The algorithm in Fig. 3 follows a similar process for each line with a failure \(f\). It determines and isolates the black sector \(S(f)\) easily with help of the defined sectors and frontiers (lines 12 and 13). Finally, the algorithm returns the total EENS.

We use an algorithm proposed by Benavides et al. \([1]\) to simulate the service restoration after a failure and to calculate the affected area. This algorithm expands iteratively the supplied area and checks the feasibility of electrical constraints. The considered electrical constraints are lines and substation capacities and acceptable voltage drop. The electrical simulation is computationally very expensive, but electrical constraints are important to reflect a realistic approximation of the attended area.

### 3 Construction and local search algorithms

In this section we explain the construction and local search algorithms proposed to solve the switch allocation problem. Semi-greedy construction, and first and best improvement local searches were originally proposed by Benavides et al. \([1]\).
Reliability Evaluation Algorithm
Input: Distribution Network \( G = (N, A) \)
Set of lines with installed switches \( A_B \).
Output: Estimated reliability EENS.
1: EENS ← 0
2: for \( \forall S \in SS \) do // Sectors
3: Simulate a failure in \( S \).
4: Assume the black area \( S(f) = S \).
5: Isolate the black area by opening the frontier switches \( F(S(f)) = F(S) \).
6: Determine affected nodes \( N_f \) with a service restoration algorithm.
7: EENS_f ← \( \sum_{a \in \{S(f)\}} A_r a \cdot \sum_{u \in S_f} P_u \).
8: EENS ← EENS + EENS_f.
9: end for
10: for \( \forall a \in A_B \) do // Frontier lines
11: Simulate a failure \( f(a) \) in \( a \).
12: Assume the black area \( S(f(a)) = S(a) \).
13: Isolate the black area by opening the frontier switches \( F(S(a)) = F(S(v)) \) \( \{a\} \).
14: Determine affected nodes \( N_f \) with a service restoration algorithm.
15: EENS_f ← \( \lambda f(a) \sum_{u \in S_f} P_u \).
16: EENS ← EENS + EENS_f.
17: end for
18: return EENS.

Fig. 3. Network reliability evaluation by sectors.

Semi-greedy Construction Algorithm
Input: Distribution network \( G = (N, A) \), number of switches \( k \), \( \beta \) randomness.
Output: Set of lines with installed switches \( A_B \).
1: \( A_B \leftarrow \emptyset \)
2: while \( |A_B| < k \) do
3: Candidate List ← \( A \setminus A_B \).
4: Estimate reliability gain of all elements in Candidate List.
5: Restricted Candidate List ← a portion of best elements in Candidate List.
6: \( a \leftarrow \) select randomly a switch location from Restricted Candidate List.
7: \( A_B \leftarrow A_B \cup \{a\} \)
8: end while
9: return \( A_B \).

a. Semi-greedy.

Sample Construction Algorithm
Input: Distribution network \( G = (N, A) \), number of switches \( k \), \( \beta \) sample percentage.
Output: Set of lines with installed switches \( A_B \).
1: \( A_B \leftarrow \emptyset \)
2: while \( |A_B| < k \) do
3: Candidate List ← \( A \setminus A_B \).
4: Sample Candidate List ← sample randomly \( \beta \) percent from Candidate List.
5: Estimate reliability gain of all elements in Sample Candidate List.
6: \( a \leftarrow \) select the best switch location from Sample Candidate List.
7: \( A_B \leftarrow A_B \cup \{a\} \)
8: end while
9: return \( A_B \).

b. Sample.

Fig. 4. Construction algorithms.

First Improvement Local Search Algorithm
Input: Distribution network \( G = (N, A) \), initial solution \( A_{best} \).
Output: Best found solution \( A_{best} \).
1: Estimate reliability of \( A_B \).
2: \( A_{best} \leftarrow A_B \).
3: while stop criterion is not satisfied do
4: \( A_B \leftarrow A_{best} \).
5: for \( \forall a \in A_B \) do // With switch
6: \( A_{best} \leftarrow (A_B \setminus \{a\}) \cup \{b\} \) // Move
7: for \( \forall a \in A_B \) do // Without switch
8: if \( A_{best} < A_{best} \) then
9: \( A_{best} \leftarrow A_{best} \).
10: exit for to line 3 // Line missed in best improvement
11: end if
12: end for
13: end while
16: return \( A_{best} \).

a. First improvement.

Sample Local Search Algorithm
Input: Distribution network \( G = (N, A) \), initial solution \( A_B \), \( \beta \) sample percentage.
Output: Best found solution \( A_{best} \).
1: Estimate reliability of \( A_B \).
2: \( A_{best} \leftarrow A_B \).
3: while stop criterion is not satisfied do
4: \( A_B \leftarrow A_{best} \).
5: for \( \forall a \in A_B \) do // With switch
6: \( A_{best} \leftarrow (A_B \setminus \{a\}) \cup \{b\} \) // Move
7: for \( \forall a \in A_B \) do // Without switch
8: if \( A_{best} < A_{best} \) then
9: \( A_{best} \leftarrow A_{best} \).
10: Estimate reliability of \( A_{best} \).
11: if \( A_{best} < A_{best} \) then
12: \( A_{best} \leftarrow A_{best} \).
13: end if
14: end for
15: end for
16: end while
17: return \( A_{best} \).

b. Sample.

Fig. 5. Local search algorithms.
3.1 Construction algorithms

We use four construction algorithms: random, sample, greedy and semi-greedy. **Random construction** selects \( k \) switches randomly and evaluates the resulting solution. **Greedy construction** builds a feasible solution element by element, evaluating all the elements to select the best each time.

Semi-greedy and sample constructions (depicted in Fig. 4) also build a feasible solution one element at a time. Both use a reduced list of candidate elements to select one and add it to the solution. The difference lies in the way they create that small list. **Semi-greedy construction** (in Fig. 4a) first evaluates every possible element. Then, a portion of \( \alpha \) switches with the highest reliability is kept. And finally, one element is randomly picked from the restricted candidate list. \((\alpha = 0\) selects always the best element, while \(\alpha = 1\) selects randomly between all the elements). **Sample construction** (in Fig. 4b) first selects randomly a portion of \( \beta \) switches. Then, it evaluates the sample candidate list to choose the best. \((\beta = 0\% \) corresponds to a random construction, while \(\beta = 100\% \) corresponds to a greedy construction). When installing \( k \) switches, the computational cost of the greedy and semi-greedy construction algorithms is \( O(k|A|) \), of the sample construction is \( O(k|A|\beta) \), and of the random construction is \( O(1) \).

3.2 Local search algorithms

A local search algorithm iteratively replaces the current solution with a better neighbour. It starts from an initial solution created by a construction algorithm. In this case, it searches in a neighbourhood defined by the relocation of one switch. We used three local search strategies: by sample, first improvement and best improvement.

**First improvement local search** is depicted in Fig. 5a. It searches in the neighbourhood for an improvement of the current solution. When a better solution is found, it becomes the current solution for the next iteration. The search stops when there are no better solutions in the neighbourhood. Finally, the last found solution is returned.

**Best improvement local search** explores all the neighbourhood to select the best for the next iteration, while first improvement accepts the first better solution found and breaks the search out to the next iteration without exploring all the neighbourhood. We obtain a best improvement local search by removing the line 11 from the algorithm in Fig. 5a.

**Sample local search** is depicted in Fig. 5b. It explores the same neighbourhood, but not completely. It samples \( \beta \) percent of the lines with switches \((a \in A_B)\) and \( \beta \) percent of lines without switches \((b \in A \setminus A_B)\) to relocate one switch. Thus, it reduces the size of the explored neighbourhood and the number of reliability estimations. If the algorithm finds a better solution in the sample, it is taken for the next iteration. Finally, it returns the last solution.

Sample neighbourhood exploration is not exhaustive and does not guarantee to find the local minimum. Thus, the stop criterion may be a maximum number
of iterations or a number of iterations without improvement. To guarantee that
the local minimum is reached, we can execute a first or best improvement local
search after the sample local search, or intersperse an exhaustive neighbourhood
search after a number of iterations.

When installing \( k \) switches, the computational cost of each iteration is
\( O(k|A|\beta^2) \) for the sample local search and \( O(k|A|) \) for the other two strategies.

4 Experiments

For our tests we used two instances. The small instance (B4) is very well known
in literature as RBTS Bus 4, introduced by Billinton and Jonnavithula [2]. The
large instance (R6) is the sixth from the REpository of Distribution Systems
(REDS) maintained by Kavasseri and Ababei [8]. R6 is large enough to confirm
results without causing excessively long time process. Table 1 shows details for
these instances.

To complete the necessary information, we followed the adaptation of part
of the RBTS bus 6 by Falaghi et al. [7]. We assume an outage time \( r = 2 \) h,
a resistance \( r = 0.257 \) \( \Omega/km \), a reactance \( x = 0.087 \) \( \Omega/km \), a failure rate
\( \lambda = 0.065 \) \( f/yr/km \), and a capacity \( I_{\text{MAX}} = 500 \) A for every line. Line failure
rates for REDS are calculated with a factor over their resistance as \( \lambda = 0.0696 \) \( r \).

<table>
<thead>
<tr>
<th></th>
<th>RBTS Bus 4</th>
<th>REDS 6th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network instances</td>
<td>B4</td>
<td>R6</td>
</tr>
<tr>
<td>Substations</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Consumers</td>
<td>38</td>
<td>201</td>
</tr>
<tr>
<td>feeder lines</td>
<td>67</td>
<td>201</td>
</tr>
<tr>
<td>loop lines</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>Operation voltage (V)</td>
<td>11000</td>
<td>33600</td>
</tr>
<tr>
<td>Total power demand (kW)</td>
<td>24580</td>
<td>32437</td>
</tr>
<tr>
<td>Consumer power factor *</td>
<td>0.9</td>
<td>0.85</td>
</tr>
<tr>
<td>Consumer demand * (kW)</td>
<td>[415, 1500]</td>
<td>[0, 1211]</td>
</tr>
<tr>
<td>Line resistance (( \Omega ))</td>
<td>[0.1542, 0.2056]</td>
<td>[0.000, 0.137]</td>
</tr>
<tr>
<td>Line reactance (( \Omega ))</td>
<td>[0.0522, 0.0696]</td>
<td>[0.000, 0.254]</td>
</tr>
<tr>
<td>Line failure rates</td>
<td>[0.039, 0.052]</td>
<td>[0.000, 0.013]</td>
</tr>
</tbody>
</table>

* per load point.

We combined construction and local search methods as shown in Table 2.
Sample construction and sample local search use \( \beta = 10\% \) and semi-greedy
construction has \( \alpha = 0.5 \). Stop criterion for sample local search is ten iterations
without improvement. The SplBI combinations execute a best improvement local
search after the sample local search, to guarantee a local minimum. We run tests
to allocate 15 and 20 switches. We repeat each experiment 1000 times for B4,
and 100 times for R6, except Gr-BI which executed once.

All the tests have been executed on an Intel Core 2 processor with a 2.33
GHz clock and 4 GB of main memory, and have been compiled with GNU C++
with the command “g++ *.cpp -Wall -O3 -static”.

Table 2. Combinations of construction and local search algorithms for tests.

<table>
<thead>
<tr>
<th>Construction algorithm</th>
<th>Greedy</th>
<th>Semi-greedy</th>
<th>Random</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>First improvement</td>
<td>SGr-FI</td>
<td>Rnd-FI</td>
<td>Spl-FI</td>
<td></td>
</tr>
<tr>
<td>Best improvement</td>
<td>Gr-BI</td>
<td>SGr-BI</td>
<td>Rnd-BI</td>
<td>Spl-BI</td>
</tr>
<tr>
<td>Sample</td>
<td>SGr-Spl</td>
<td>Rnd-Spl</td>
<td>Spl-Spl</td>
<td></td>
</tr>
<tr>
<td>Sample + Best improvement</td>
<td>SGr-SplBI</td>
<td>Rnd-SplBI</td>
<td>Spl-SplBI</td>
<td></td>
</tr>
</tbody>
</table>

4.1 Experimental Results

We present the results for instance B4 in Table 3 and Figure 6, and for the instance R6 in Table 4 and Figure 7. The tables show the average EENS and the number of reliability estimations used to generate the initial solutions with the construction algorithms. For the final solutions obtained with the local search methods, the tables present the average EENS, the average number of reliability estimations, and the average running time. The tables also present the best solution found by each combination within all the repetitions (column Min.). The last columns compare the number of final solutions that reach (column =GR) or overcome (column <GR) the corresponding greedy solution. The figures compare the average EENS achieved with the required number of reliability estimations. Four points show the average result of the construction algorithms (random, semi-greedy, sample and greedy). Three lines start from each point (except greedy), they outline the average performance of first improvement (FI), best improvement (BI) and sample local search strategies. The three local search strategies show the same behavior for all the test cases, independently of the constructive algorithms.

First, we analyze construction algorithms. Solutions created by a semi-greedy algorithm are better than random solutions in average by 2000 KWh/year (for B4, 1100 for R6), but the required number of reliability estimations increases significantly. A random solution requires only one reliability estimation, while the semi-greedy and the greedy algorithms require more than 900 estimations (for B4, 3000 for R6). Greedy construction generates always the best initial solution at the same cost than semi-greedy, but this solution is usually close to (or is itself) a local minimum, that is undesirable for a multi-start procedure. Solutions created by the sample algorithm are better than random solutions in average by 4600 KWh/year (for B4, 3300 for R6), and they require less than 120 estimations (for B4, 410 for R6). Thus, sample construction creates better solutions than semi-greedy algorithm, and in less than ten percent of the corresponding time. The good cost/benefit of the sample construction algorithm can be seen in the graphs by its proximity to the origin, i.e., low EENS and low number of reliability estimations. Contrarily, semi-greedy construction generates the worst solutions considering its high number of reliability estimations.

Now, we analyze the local search algorithms. The average final solutions of FI and BI are very close, and they yield the best result with all construction algorithms. The biggest difference between FI and BI is 26 MWh/year (semi-greedy for R6 with 20 switches), and it is half of the smallest standard deviation.
<table>
<thead>
<tr>
<th>Algorithm combination</th>
<th>Construction Local search final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gr-BI</td>
<td>EENS</td>
</tr>
<tr>
<td>Rnd-SplBI</td>
<td>1993</td>
</tr>
<tr>
<td>Gr-SplBI</td>
<td>1973</td>
</tr>
<tr>
<td>Gr-Spl</td>
<td>1973</td>
</tr>
<tr>
<td>Rnd-Spl</td>
<td>1973</td>
</tr>
</tbody>
</table>

**Table 3.** Comparison of construction and local search algorithms, instance R6.

<table>
<thead>
<tr>
<th>Algorithm combination</th>
<th>Construction Local search final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gr-BI</td>
<td>EENS</td>
</tr>
<tr>
<td>Rnd-SplBI</td>
<td>1993</td>
</tr>
<tr>
<td>Gr-SplBI</td>
<td>1973</td>
</tr>
<tr>
<td>Gr-Spl</td>
<td>1973</td>
</tr>
<tr>
<td>Rnd-Spl</td>
<td>1973</td>
</tr>
</tbody>
</table>

**Table 4.** Comparison of construction and local search algorithms, instance R6.

**Table 5.** Comparison of construction and local search algorithms, instance B4.
Fig. 6. Average performance for instance B4.
Fig. 7. Average performance for instance R6.
The difference between FI and BI is in their performance over time. The figures show that FI progresses quickly in the beginning, but BI becomes better after some iterations. BI has a stable number of reliability estimations in each iteration along the whole search. FI takes any solution better than current and the number of estimations varies with the iterations. This is an advantage in early iterations because FI finds easily better solutions, but becomes a disadvantage in the late iterations because FI restarts the local search with any small improvement when the number of reliability estimations is almost the same than BI. Thus, FI spends more time than BI in average.

The average final solutions of sample local search are worse than FI and BI. The difference with FI and BI is less than 700 KWh/year (for B4, 400 for B6). Moreover, sample local search was able to find the best solution for instance B4 with 20 switches. The time that it spent is very small, about half the time of the greedy or semi-greedy construction alone. The number of reliability estimations of sample local search is constant in each iteration like BI, but is 100 times smaller because the neighbourhood is restricted randomly to ten percent of switches and ten percent of free lines.

Sample local search is not an exhaustive search in the neighbourhood, i.e., it does not guarantee to find the local minimum, but it finds good results in small time. When a BI is applied after sample local search (SplBI combinations), it reaches the average results than BI or FI alone, but saving at least a quarter of the running time. For instance B4, about half of the final solutions stuck in the greedy solution after FI or BI local search, in particular after sample construction. For instance R6, the local search algorithms do not get stuck in the greedy solution, and exhaustive local search strategies overcome the greedy solution in 89 percent of the cases.

Finally, we analyze the combinations of construction and local search. If we consider each row of Tables 3 and 4 as one multi-start iterated local search, with 1000 iterations (for B4, 100 for R6), and each row with semi-greedy construction as a greedy randomized adaptive search procedure (GRASP) [11], we observe that iterated search processes with FI and BI are effective to reach the best known upper bound for the test cases. But the number of iterations to obtain this results is very high, and the accumulated running time is 1000 times the shown average (for B4, 100 for R6).

A GRASP is as effective as an iterated local search with random initial solutions, but needs less time. Rnd-BI is the combination that finds the biggest number of solutions that overcome the greedy solution. The most expensive combination is Rnd-FI.

The cheapest method for an iterated local search would be the Spl-Spl combination, its execution time is at least twice faster than a greedy or semi-greedy construction algorithm alone. The best method for an iterated local search would be the Spl-SplBI combination, because it is the cheapest combination in terms of execution time that is able to find the best solution. This verifies that a restricted neighbourhood speeds up the construction and the search processes.
5 Concluding Remarks

In this paper, we presented construction and local search methods for the switch allocation problem, with the service restoration problem as a subproblem. The objective is to improve network reliability by decreasing the unattended demand in case of line failures. We presented and compared the combination of four construction algorithms and three local searches strategies. Experimental results show that sample construction and sample local search are very inexpensive and create good and diverse solutions. They also show that semi-greedy construction is expensive and does not generate significative improvements in start solutions. The present work indicates that a more directed local search combined with sample construction might give better results. In future work, we intend to study an iterated search that uses a path relinking between solutions created by sample construction.

Acknowledgements

This work is supported by the project “Metodologia Multiobjetivo para Análise e Melhoria da Confiabilidade de Sistemas de Distribuição de Energia Elétrica” (Auxílio Integrado CNPq, CTEnerg processo 554900/2006-8).

References


