Ant Algorithms for the Optimal Restoration of Distribution Feeders During Cold Load Pickup

Indira Mohanty  Jugal Kalita  Sanjoy Das  Anil Pahwa  Erik Buehler
Department of Computer Science  Department of Electrical & Computer Engineering
University of Colorado  Kansas State University
Colorado Springs, CO 80933  Manhattan, KS 66506

Abstract: The ant colony algorithm is a new technique for combinatorial optimization borrowed from swarm intelligence. This paper outlines an ant colony algorithm to compute the optimal order of restoring sections in a power distribution system. Restoration of distribution feeders after long interruptions creates cold load pickup conditions due to loss of diversity among the loads. The distribution system load may have to be restored step-by-step using sectionalizing switches under such conditions to prevent overheating of substation transformer. The restoration time is dependent on the order in which sections are restored. Results obtained using this method for two test cases are presented including a comparison with the simulated annealing algorithm.

Keywords: Distribution system restoration, cold load pickup, ant colony optimization.

I. INTRODUCTION

COLD LOAD PICKUP

Generally speaking, loads can be classified into two categories, cyclic and non-cyclic. Cyclic loads are those that, during normal operation, turn on and off intermittently. Typically, these cyclic loads are thermostatically controlled and include air conditioners, refrigerators, water heaters etc. Non-cyclic loads, such as lighting are typically turned off and on manually.

In spite of the presence of cyclic loads, the overall power delivered to a distribution system is approximately constant at a given time. This is because the distribution system generally incorporates such a large number of individual cyclic loads that only a small fraction of them are turned on at any time. In other words, under normal operation, a diversity of loads persists. As a result of load diversity, the overall load on the distribution system is fairly evenly distributed, and significantly less than the sum total of its rated load. The installed transformer rating follows this general principle.

The cold load pickup has several stages (Ucak, Pahwa 1994). Although for a transient period, the inrush current is up to 15 times the normal load, this condition lasts for only about 15 seconds. Following this the enduring stage with load equal to 2 to 5 times normal load and it may last for several hours depending on rate of load decay. Loss of diversity due to long interruptions is the cause of the enduring stage and thus the enduring demand on persists until the normal diversity among the load is regained. Under certain conditions all of the system load during cold load pickup cannot be restored simultaneously as overloading of transformers due to enduring load demand may result in excessive transformer heating.

STEP-BY-STEP RESTORATION

Sectionalizing switches can be used on distribution system feeders to restore the load in steps in case total load on the system cannot be restored in one step. The short-term load level and its duration during step-by-step restoration can be determined based on thermal characteristics and loading limitations of the transformer. To model the aggregate load during cold load pickup a delayed exponential model as shown in figure 1 is used (Ucak, Pahwa 1994). Where $S(t)$ is load as a function of time, $T_{out}$ is the outage duration, $S_U$ is the undiversified load, $S_D$ is the diversified load, $\alpha$ is the rate of load decay, and $\Delta$ is the undiversified load duration.

\[ S(t) \]

\[ S_U \]

\[ S_D \]

Figure 1: A model of cold load pickup

It should be noted that because of the serial arrangement of the sections in each feeder, the sectionalizing switch of a section causes supply to be cut off not only from the corresponding section, but from all subsequent sections as well. Therefore, turning on a switch that appears later on in a feeder has no effect unless all previous switches have
already been closed. This imposes a certain restriction on the order in which sections in a feeder can be turned on. We will refer to this as the precedence constraint. For instance, in the distribution system shown in figure 2(a), since sections 1, 2, 3, 4, 5, 6, 7 and 8, 9, 10 are on four different feeders, in the order mentioned, the precedence constraints are 1st \rightarrow 2nd \rightarrow 3rd; 4th \rightarrow 5th; 6th \rightarrow 7th and 8th \rightarrow 9th \rightarrow 10th. In other words section 3 cannot be restored unless sections 1 and 2 are restored.

The sequence in which sectionalizing switches are closed is important since different sequences would result into different restoration times, as undiversified loads of sections are not the same. Restoration time is the time at which the last sectionalizing switch in the sequence is closed. Some of the restoration sequences are not valid because of precedence constraints. For example, in the 10-section system, (1 4 6 2 8 7 9 10 3 5) is a valid sequence whereas (4 3 6 8 1 7 5 9 2 10) is not so, since section 3 appears before 1 and 2 in the sequence.

The restoration time of different sequences can be calculated from published results by Uçak, Pahwa, 1994, which is given by,

\[
\sigma_{x} = \sum_{i=1}^{\infty} U_{[i]} \cdot D_{[i]} \cdot k
\]

where, \( x = MT \) is the maximum allowable transformer loading. The section numbers are shown within square brackets. The set \( R \) contains all the previously restored sections. The undiversified loads and diversified loads for two different distribution systems of Figure 2 are given in Tables 1 and 2. We need to optimize the restoration sequence of sections so that restoration time is minimized. Some of the combinations would give the same restoration time since more than one section can be restored in the first step.

2 THE ANT COLONY APPROACH

It is no surprise that ant colony algorithms based on cooperative artificial ant-like agents (Dorigo, Maniezzo, Colomi, 1996; Dorigo, Gambardella, 1997) were originally proposed as a method for the traveling salesman problem (Garey, Johnson, 1977) where the problem is also one of determining a minimum distance path. But subsequently it has been shown to work well for other difficult permutation problems too, such as quadratic assignment and job shop scheduling (Dorigo, Di Caro 1999; Dorigo, Di Caro, Gambardella, 1999).

DESCRIPTION OF THE ALGORITHM

In each iteration of the algorithm, an ant is allowed to execute a move, leaving behind a pheromone trail for others to follow. Although several attempts at enabling multiple ants trace out paths simultaneously have been successfully implemented such as in a variation called the ant colony systems approach (Dorigo, Gambardella, 1997), where multiple ants have been used to simultaneously traverse different routes in the traveling salesperson problem (see later), such a scheme can easily be construed as only as an improvement of the basic algorithm, where pheromone updating is carried out after multiple ant runs rather than after each. Therefore, ACO is not a population-based method. The extended ACO approach proposed by Das et al. (Das, Mohanty, Yang, 2002) also considers multiple interacting ants to solve Steiner trees. Yet that approach requires very close coordination between these ants. Each ant is used generate different branches of the Steiner tree, and therefore the function of these interacting are non-overlapping. Ultimately these ants merge together to form a single ant in this scheme. The extended ACO too is therefore not a population-based technique. In other words, ACO is very unlike other algorithms based on biological phenomena, such as genetic algorithms, which
necessarily have to maintain a significant population of individuals within each iteration.

In each iteration of the algorithm, an ant produces a candidate solution for the optimization problem. The situation is analogous to nature, where each ant defines a path from the ant nest to a food source. Each ant defines a candidate solution piece-by-piece, probabilistically selecting a component at a time, until an entire solution is obtained. We shall hereafter refer to the process by which an ant assembles a complete solution in a component-wise manner, as the ant’s traversal along a solution. Each solution, or a portion of a full solution, shall be referred to variously, as a path taken by the ant. Each separate component, or path, can be assigned a cost metric, and it is the sum of all the individual components that define the function to be minimized by the ACO.

Because each ant’s traversal yields a single solution of the optimization problem, the ACO approach can be regarded as a search through the solution space of the problem instance. As we will see, the search converges gradually onto regions where potentially good solutions can be found. While traversing a solution, an ant picks components in a stochastic manner. However, to limit the ACO from wasteful iterations that explore unfeasible regions of the search space, the ant’s traversal is guided by the pheromone trail (sometimes, referred to simply as trail). The stronger the pheromone concentration along any path, the more likely an ant is, to include that path in defining a solution. Therefore, ACO is an exploitative algorithm. It seeks solutions from information gathered previously, and performs its search in the vicinity of good solutions. Needless to say, an algorithm that would solely be guided by pheromone concentration, with no probability of using paths with low or zero accumulations would be a greedy strategy. Since solution components are not picked in a greedy manner, ACO is also an exploratory algorithm that samples a wide range of solutions in the solution space. This exploratory-exploitative approach is characteristic of many heuristic based optimization approaches, including evolutionary algorithms, tabu search and particle swarms. In particular, it is also an attribute of the simulated annealing algorithm that serves as the basis of comparing the effectiveness of the proposed algorithm.

In contrast to the basic genetic algorithm or simulated annealing which make no use of common sense heuristics, ant colony algorithms incorporate this feature well within their basic structure. Whenever an ant makes a move, to add a new path to its trajectory, the probabilistic selection of the new path is governed not only by the pheromone level, but also by means of some heuristic that provides a rough measure of the quality of the paths to add. During the initial stages of the algorithm, this factor, called the desirability, plays a very important role. It restricts the exploration to regions in the search space that are most likely to contain better solutions.

During the course of defining a solution, each ant maintains a data structure, called the tabu list. This list is merely the set of all paths that have already been traversed by it. The tabu list is necessary to prevent the ant from including the same path again, producing redundancies, and often, invalid solutions. In most implementations, the solution produced in each iteration can be extracted directly from this set.

The process of pheromone deposition is critical to the success of the algorithm. As opposed to real ant systems where pheromones are deposited on-line, e.g. continuously while the ant is moving, within ACO, the trail intensities is updated at the end, only when a complete solution is defined. Therefore, it is updated only at the end of each iteration. Hence, trail intensity increment is an off-line process in ACO. Furthermore, the trail increment is inversely related to the quality of the solution generated. The reason for this deviation is simply because component paths that constitute a good solution need to be reinforced more than others, and the solution quality can only be determined at the end of the iteration only when it is fully defined.

Closely linked to, and playing an inverse role to pheromone deposition is pheromone evaporation. Pheromones are subject to dissipation even in real ants. In ACO, pheromones are evaporated from the ant trails to prevent them from increasing monotonically to undesirable levels. Evaporation is controlled by a parameter, the evaporation rate that controls how quickly trails are updated. A very slow evaporation rate would make the algorithm converge towards good neighborhoods in the solution space more slowly. But on the other hand, a very high rate of evaporation would result in an undue amount of trail being deposited along the path of any ant, thereby causing subsequent ants to follow closely, if not exactly, the same path. This prevents the algorithm from being exploratory enough, and results in premature convergence. The right rate of evaporation is crucial to the performance of the algorithm, and all paths are subject to the same rate of evaporation regardless of the quality. In order to associate the trail deposition with the evaporation rate, the deposition is controlled by the evaporation rate also, with a higher evaporation rate amounting to more pheromone deposition. The next section introduces two algorithms for computing optimal restoration sequences of any distribution system. These are, the Absolute Switch Position Ant Colony Algorithm (ASP-ACO) and the Relative Switch Position Ant Colony Algorithm (RSP-ACO).

THE ASP-ACO ALGORITHM

Restoring switches with larger undiversified loads let these loads to assume stable diversified levels at an earlier stage of the restoration. This usually results in quicker
overall restoration. Accordingly, we define the desirability $\eta_l$ of picking a switch $l$ for restoration as,

$$\eta_l = \frac{S_{U[l]}}{\sum_m S_{U[m]}}.$$  (2)

The summation in the above equation is carried over all sections. The algorithm requires several trials to compute an optimal sequence. In each trial, an ant produces a valid sequence of sections for restoration in an iterative manner, adding a new section per iteration. Starting from an empty sequence each ant adds new sections to it, once per iteration, while maintaining the precedence constraints. The probability that a section $k$ will be selected, as the $p^{th}$ section to restore is given by,

$$P_{pk} = \frac{(\tau_{pk})^\alpha (\eta_k)^\beta}{\sum (\tau_{pl})^\alpha (\eta_l)^\beta}.$$  (3)

The summation in the denominator is carried out over all sections $l$ that are ready for restoration in iteration $p$. The quantities $\alpha$ and $\beta$ are parameters of the algorithm. The quantity $\tau_{pk}$, the pheromone concentration, simply reflects the effectiveness of restoring section $k$ at the $p^{th}$ position. The pheromone level of each pair of sections is updated at the end of each trial when the ant has generated a valid sequence. If a section $k$ appears in position $p$ in the generated sequence, the pheromone concentration is updated as,

$$\tau_{kp} = (1 - \rho)\tau_{kp} + \rho \Delta \tau_{kp}.$$  (4)

Here the quantity $\Delta \tau_{kp}$ is the pheromone intensity increment, which is given by,

$$\Delta \tau_{kp} = \frac{Q}{T_{total}}.$$  (5)

The quantity $T_{total}$ is the total restoration time computed by summing the individual restoration times of all sections in proper order. The index $\gamma$ and the coefficient $Q$ are algorithm constants. The quantity $\rho$ is the usual evaporation rate. For all entries in the pheromone concentration matrix that were not updated by equations 4 and 5, the evaporation is done in accordance with,

$$\tau_{kp} = (1 - \rho)\tau_{kp}.$$  (6)

THE ASP-ACO ALGORITHM

The algorithm starts by initializing its basic parameters; initial pheromone concentration $\tau_{io}$, pheromone evaporation rate $\rho$, pheromone increment $Q$ and the constant exponents $\alpha$, $\beta$. The desirability $\eta$ is computed in the usual manner shown in equation 2. In this variation of the algorithm instead of starting with an empty sequence of sections, an ant selects the first section randomly and computes its restoration time using equation 1. Then new sections are added to the sequence iteratively based on their probability given by equation 3 and the total restoration time is calculated. The main difference of these two proposed algorithms lies in the configuration of the pheromone concentration matrix. The update rule for the pheromone concentration in this case is as follows,

$$\tau_{mn} = (1 - \rho)\tau_{mn} + \rho \Delta \tau_{mn}.$$  (7)

where, the pheromone increment $\Delta \tau_{mn}$ is given by,

$$\Delta \tau_{mn} = \frac{Q}{T_{total}}.$$  (8)

Here, each pair of sections in the generated sequence gets a reinforcement given by equations 7 and 8 whereas the remaining entries in the matrix subject to uniform evaporation in accordance with,

$$\tau_{mn} = (1 - \rho)\tau_{mn}.$$  (9)

The subscripts in the above equations represent consecutive sections in the generated sequence.

SIMULATED ANNEALING

Simulated annealing (S. Kirkpatrick, C. D. Gellatt, M. P. Vecchi) is a stochastic approach for continuous as well as discrete optimization. Simulated annealing is a relatively simple algorithm, and designing such an algorithm for obtaining the optimal restoration sequences is straightforward. The only aspect of the algorithm that is specific to the problem is the method by which a solution is perturbed. In this case, any given section of switches is perturbed by interchanging the positions of two switches. The switches are picked randomly, while ensuring that after the swapping operation, the validity constraints as described earlier 3 are preserved. As an instance, consider a sequence (1 2 10 4 5 7 8 6 3 11 9 12) of the 12 section problem. This sequence conforms to the aforementioned restrictions. Swapping the positions of switches 8 and 11 yields the new sequence (1 2 10 4 5 7 11 6 3 8 9 12) where validity continues to be maintained. But if switch 7 is picked for swapping with switch 11, instead of switch 8, the sequence (1 2 10 4 5 11 8 6 3 7 9 12) is obtained. This is an invalid sequence as in every valid sequence, the restoration of switch 8 can only follow that of switch 7, since both 7 and 8 belong to the same feeder, and 7 precedes 8. Any move that would create invalid solutions is not done by the algorithm.

IMPLEMENTATION ISSUES

Our programs were implemented entirely in MATLAB. We used the standard MATLAB interpreter to run these programs. All our experiments were carried out in a Pentium processor running Windows NT.
3 RESULTS AND DISCUSSION

We fine-tuned all the parameters of the algorithm empirically, to get sufficiently good results from our limited runs. The data was obtained from the real-world distribution system described earlier (Ucak, Pahwa, 1994; Chavali, Pahwa, Das 2002). The diversified and undiversified loads for the the 10 section and 12 section problems are shown in table 1 and 2. The transformer capacity \( S_{TR} \) was 45.0 MVA. The load decay and the undiversified load duration of each section were equal to 0.5/hr.

Each algorithm was executed 50 times for each of the two problem instances. The parameters for each algorithm were adjusted to obtain the best possible performance. For the ant colony algorithms, the exponents \( \gamma \) and \( \alpha \) were kept at unity. For simulated annealing (SA), the initial temperature was 100. It was cooled to 90% of its value after every 100 iterations.

Figure 3 shows the convergence of the programs. In each case, a variable containing the restoration time of the best sequence obtained by the algorithm was maintained separately. It was updated only if at the end of each iteration, a sequence associated with a lower restoration time than the stored value was obtained. The plot shows this minimum restoration time (y-axis) as a function of iterations (x-axis), averaged over 50 runs. Although the ant colony algorithms converged very quickly, each program was run for a total of 600 iterations to allow the simulated annealing algorithm enough time to converge.

Figure 4 is a similar plots for the distribution system containing 12 sections. It shows similar trends as previously.

Both RSP-ACO and ASP-ACO are ant colony algorithms. The difference between them is the manner in which pheromone reinforcement is carried out. Although both strategies of pheromone reinforcement have similar code and algorithmic complexities, the choice of pheromone reinforcement greatly affected the convergence. RSP-ACO clearly converges faster than ASP-ACO.

The present study was restricted to two relatively small optimization problems. This is because in practically all distribution system installations, the total number of sections rarely exceeds 10 or 12.

Because of their recent nature, ant colony algorithms have been applied thus far, only to simple optimization problems. This research extends the applicability of this class of stochastic optimization algorithms to practical engineering problems. It was found that the method used to reinforce pheromone levels is crucial to the algorithm’s performance. Our results also suggest that designing an appropriate algorithm for any particular problem is very important. This observation should be of great value to future research, particularly since the theoretical convergence properties of these algorithms have not been studied at length.

<table>
<thead>
<tr>
<th>Section Number</th>
<th>Diversified Load (MVA)</th>
<th>Undiversified Load (MVA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.5</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>2.5</td>
<td>4.5</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>5.0</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
<td>3.0</td>
</tr>
<tr>
<td>5</td>
<td>2.5</td>
<td>6.0</td>
</tr>
<tr>
<td>6</td>
<td>2.0</td>
<td>5.6</td>
</tr>
<tr>
<td>7</td>
<td>2.5</td>
<td>6.5</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>9</td>
<td>3.0</td>
<td>9.0</td>
</tr>
<tr>
<td>10</td>
<td>3.0</td>
<td>6.0</td>
</tr>
<tr>
<td>11</td>
<td>1.5</td>
<td>4.8</td>
</tr>
<tr>
<td>12</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Total</td>
<td>27.0</td>
<td>67.9</td>
</tr>
</tbody>
</table>

Table 1: The distribution system with 10 sections.

<table>
<thead>
<tr>
<th>Section Number</th>
<th>Diversified Load (MVA)</th>
<th>Undiversified Load (MVA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.0</td>
<td>9.0</td>
</tr>
<tr>
<td>2</td>
<td>3.0</td>
<td>5.5</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>6.0</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
<td>5.0</td>
</tr>
<tr>
<td>5</td>
<td>3.0</td>
<td>5.0</td>
</tr>
<tr>
<td>6</td>
<td>6.0</td>
<td>9.5</td>
</tr>
<tr>
<td>7</td>
<td>2.5</td>
<td>6.0</td>
</tr>
<tr>
<td>8</td>
<td>3.5</td>
<td>6.5</td>
</tr>
<tr>
<td>9</td>
<td>4.5</td>
<td>11.0</td>
</tr>
<tr>
<td>10</td>
<td>3.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Total</td>
<td>33.0</td>
<td>71.5</td>
</tr>
</tbody>
</table>

Table 2: The distribution system with 12 sections.
REFERENCES


