Abstract

Green power is clean without pollution, such as solar and wind power. It has become a resolution to supplement the deficiency of today’s high cost/danger energy resources. A challenge is to integrate these distributed resources into the power grid. The high cost and voltage drop of running long transmission lines motivate the way to minimize the length by serializing their connections to load centers. This introduces a risk that the loss of a transmission line section can disconnect multiple green power resources. Our previous study proposed an idea of a fault tolerance network, in which every power resource has at least two independent connections to the load centers. Preliminary studies showed that fault tolerance can be achieved with reasonable extra transmission line expenses. The objective of this paper is to develop an artificial intelligence algorithm to identify possible Steiner points into the network. The new network intends to further reduce the cost without compromising the reliability.

Keywords: Green Power; Steiner Point; Fault Tolerance; Optimization

1. Introduction

Green power resources, including solar and wind power, produce clean and renewable energy without pollutants and do not require extensive safety planning required of traditional power plants such as coal and nuclear. As of the end of 2011, the United States now has the 2nd most installed capacity of wind energy on the planet [1]. The trend of increasing more green power is expected to continue as the Department of Energy attempts to have 20% of the country’s energy produced by wind by year 2030 [2].

Nevertheless, an untapped supply of potential wind energy is located within the Midwest portion of the country, where transmission capacity is inadequate due to the limited number of lines available. In other words, many wind farms will be harnessed at locations that are not close to the existing power grid. As a consequence, more transmission lines are needed to connect these wind farms to the power grid. One estimate has put the cost of each transmission line at $1.5M/mile [3]. Thus, substantial cost savings can be accrued by minimizing the length of lines.

For this optimization problem, a solution is to serialize the connections of the wind power locations to the load centers. This will not just shorten the length, but can also reduce the voltage drop due to line losses. The downside is the loss of a transmission line section will likely disconnect multiple green power resources. Therefore, in [4] we proposed an idea of developing a fault tolerance network, in which every load center has at least two independent...
connections to the power grid and its renewable resources. The purpose is to ensure that the loss of a line section does not result in the loss of power from green power sources to any loads. There is a major distinction comparing this effort to existing fault tolerance studies on wireless sensor networks and network security [5] [6] [7]. The power sources and loads in our study are treated as different entities for connection, but no such concern in those studies.

In [8], we proved this optimization together with fault tolerance to be a NP-hard problem. A 3-step algorithm was developed to find an acceptable solution. Step one constructs a minimum spanning tree (MST) to connect all power sources and load centers. The MST offers a good solution but is not fault tolerant, so the next step tackles fault tolerance by adding extra connections. The final step performs edge refactoring to further optimize the solution without sacrificing the reliability requirements.

The objective of this paper is to develop an artificial intelligence (AI) algorithm to introduce Steiner points into the fault tolerant power grid network if possible. Adding a Steiner point to three given vertices yields the shortest distance to connect them. This concept has been applied to building Steiner trees for optimizing VLSI circuits. It is our expectation that the new network with Steiner points will further reduce the cost without compromising fault tolerance. Because Steiner problems are NP-complete problems [9], our AI algorithm takes advantage of swarm intelligence using ants to go through the learning process to determine good Steiner locations. Experiments are conducted to analyze the potential cost saving in comparison with the current means.

Transmission line routing needs to satisfy many environmental, engineering, and legal constraints, as noted in [10]. This study ignores issues of right-of-ways, cable sizing, line losses, etc. The major focus is on the learning algorithm, which considers mainly green power sources without a mixture with traditional power plants.

The rest of this paper is organized as below. Section 2 summarizes our previously developed stepwise algorithm. Section 3 introduces a learning approach to allocate Steiner points to further optimize the system. Experiments are discussed in Section 4 to justify the saving by the new technique. Finally, Section 5 concludes this study and discusses future works.

2. Summary of the Stepwise Algorithm

Our stepwise algorithm tackled two different entities: power sources and load centers. It differs from that of k-edge-connectivity problems [11] [12] [13], which ensure a graph remains connected for same entities with the removal of fewer than k edges. The steps of the algorithm are summarized as follows.

2.1. Minimum spanning tree (MST):

The sources and loads are modeled to be two kinds of vertices s and d. MST [14] is applied to find the shortest edges to connect them together to form a tree structure. For better performance, Delaunay triangulation [15] can carry out the reduction of the amount of edges for length sorting, before the Kruskal’s algorithm [14] greedily selects the shortest edge and iteratively combines disjoint sets. Figure 1 shows an example of MST that connects 4 sources and 11 loads on top of the Delaunay triangulation.

2.2. Extra edge addition:

Some loads in the MST may have only one path to a source. They are defined as singletons and are vulnerable to an edge broken down in the path. In Figure 1, we can identify a set of singletons $S = \{d_1, d_2, d_4, d_5, d_8, d_{10}\}$. To achieve fault tolerance, this step is to eliminate all singletons. The elimination process will select a singleton and add an edge to it to form another independent path to a source. This process will repeat until no more singletons exist.

The search for another independent path is done by using sets. We define a robust set $R$ as excluding the singletons from the universal set $U$ that contains all vertices, i.e., $R = U - S$. The creation of $R$ can take advantage of the depth-first traversal approach. By picking a source to traverse the graph, all loads between this source and
another source are guaranteed to have at least two independent paths to connect to the sources. These vertices are thus the elements of $R$. The goal of the elimination process is to grow $R$ to be the same as $U$. This can be done by exploiting the elements in $R$ to eliminate singletons. Basically, when an added edge connects the selected singleton to a vertex in $R$, all loads in the newly created independent path become fault tolerant and can be moved from set $S$ to $R$. By greediness, in $R$ the closest vertex to the singleton is chosen. Figure 2 shows the result of adding in extra edges.

2.3. Edge Refactoring:

This step adjusts the edge connections and removes superfluous edges to further improve the result. There are two iterative sub procedures and they maintain all vertices to stay in the robust set $R$. In other words, all vertices will stay robust without a change.

- Triangle inequality:

By the theory of triangle inequality [14], one side of a triangle has the length shorter than the sum of the other two sides. Accordingly, the idea is to look for two adjacently connected edges to be replaced by the unconnected side. Figure 3 shows the improved result of Figure 2. First, edge $<d_1, d_2>$ replaces edges $<d_3, d_2>$ and $<d_3, d_1>$. Next, edge $<d_1, d_3>$ replaces edges $<s_2, d_1>$ and $<s_2, d_3>$.

- Edges reduction and rearrangement:

By graph theory, the number of edges connecting to a vertex is the degree of that vertex. If the degree of a load center is high, i.e. degree $\geq 3$, some edge(s) may be superfluous and can be eliminated. Figure 4 shows the edge $<d_7, s_3>$ in Figure 3 is unnecessary and can be removed.

The procedure of triangle inequality tackles two adjacently connected edges. Optimization may also be performed on two non-adjacent edges, if the sum of their lengths is longer than the length of a new edge that directly connects the two. This new edge can substitute the two old edges, as long as all vertices stay in the robust set $R$. In Figure 4, edges $<d_3, s_1>$ and $<d_1, d_2>$ are replaced with edge $<d_2, d_3>$. The result yields a shorter length without sacrificing fault tolerance.

3. A Learning Algorithm for Determining Steiner Points

This section discusses the purpose of Steiner points and the rules to add them into the fault tolerant system for further improvement. Since finding the best set of Steiner points is a hard problem, a learning algorithm using swarm intelligence is developed to quickly identify a good set of the points.

3.1. Steiner points:

For three vertices on the Euclidean space, the shortest distance can be adding a Steiner point [9] to connect them. The Steiner point is at the location where every two vertices out of the three connecting to it form an angle of 120 degree. If no such point is available, the two shorter edges that connect the three vertices will be the shortest distance. For example, Figure 5 enhances Figure 4 by having a Steiner point added to connect to $d_6$, $d_8$ and $d_9$.

The addition of Steiner points must not jeopardize fault tolerance. Due to the identity difference between power sources and loads, we examine two separate rules as below to possibly add the points.

- For degree(source) $\geq 2$, up to degree(source)-1 Steiner points may be added.
- For degree(load) $\geq 3$, up to degree(load)-2 Steiner points may be added.
Note that Steiner points are not definitely available when the rules are met. The reason can be either mismatching the 120 degree criterion or violating fault tolerance. On the other hand, Steiner points will not be considered if none of the rules are met. Figure 6 shows a counter example to the second rule. The degree of load $d_{11}$ is only 2, i.e., not $\geq 3$. By adding a Steiner point $s_p$, $d_{11}$ becomes vulnerable to power shortage for the single line loss of $<d_{11}, s_p>$.

Our allocation for Steiner points exploits the coordinate/gradient descent technique to iteratively adjust their locations in turns. Each repetition will bring a Steiner point closer to its perfect 120 degree location if available. Figure 7 shows an example of adding and allocating two Steiner points for a four vertices system. These vertices are all sources to temporarily disregard the consideration of fault tolerance. For this system, the bottom two sources meet the first rule. Our technique first places two Steiner points to the same locations as the two respective sources and then begins the descent iterations. The analyzed Steiner point stays unmoved, if the new 120 degree location cannot be found. The middle image reveals the revolving results for the iterations. The final outcome is displayed as the image on the right.

### 3.2. The learning algorithm:

In [16], one of the authors investigated routing of transmission lines using Steiner trees. The algorithm only tackled tree structures and concerned no fault tolerance. We observed that a fault tolerant power grid network is very likely to form a graph instead of being just a tree. The previous approach lacked adaptability to support that. This motivates our development of this learning algorithm to accommodate new circumstances and optimize the result.

In [17], Dorigo and Gambardella solved the traveling salesmen problem (TSP) using the idea of ant colonies. Our learning algorithm adopts the same swarm intelligence approach. Without reinventing the wheel, the modeling of Steiner points is deduced to the TSP problem. This allows us to directly apply the existing heuristic functions and pheromone trail updating equations with only minor changes. Moreover, TSP solves a graph problem. A successful mapping to our problem proves the ability to handle the graph issue of fault tolerance networks.

For TSP, each ant picks a random city, travels all cities once, and gets back to the starting point. For our problem, Steiner points are treated as cities. In 3.1, the two rules help identifying the utmost number of Steiner points that can be available. By the same manner, each ant processes this number of Steiner points exactly once. If some of the points should not be existent for a reason, we can tell because they stay at the same locations as their corresponding sources or loads without descent. The actual set of Steiner points can simply exclude the non-existent ones.

The choice for the next city to visit in TSP considers both the distance and the strength of the pheromone. The one with shorter distance and stronger scent in between is more favorable to be traversed, and other cities are still occasionally given a chance. Similarly, our selection from Steiner point $r$ to the next point $s$ also makes use of distance, pheromone and probability for decisions, as formulated in Eqs. (1) and (2) same as in [17]. It is important to note that Steiner points that are close in distance may mutually affect each other during coordinate descent, e.g., the two added Steiner points in Figure 7. Hence, the processing order on them has an impact on the final result.

\[
S = \begin{cases} 
\arg \max_{s \in A} \left[ \pi(r, u) \cdot q(r, u) \right]^\beta & \text{if } q \leq q_0 \\
0 & \text{otherwise}
\end{cases}
\]

\[
p_k(r, s) = \begin{cases} 
\frac{[\eta(r, s)]^\beta}{\sum_{s' \in A} [\eta(r, s')]^\beta} & \text{if } s \in M_k \\
0 & \text{otherwise}
\end{cases}
\]

, where $\pi(r, u)$ is the amount of pheromone trail placed by ants between Steiner points $r$ and $u$, $\eta(r, u)$ is a heuristic function, returning the inverse of the distance between $r$ and $u$, $\beta$ is a parameter which weighs the relative importance of pheromone trail and distance, $q$ is a random value with uniform probability in $[0,1]$, $q_0 (0 \leq q_0 \leq 1)$ is a probability parameter, $M_k$ is the working memory of ant $k$, and $S$ is a random variable selected according to the probability distribution of $p_k(r, s)$, which ant $k$ chooses to move from Steiner point $r$ to point $s$.

Pheromone trail is updated locally and globally like TSP. Local trail updating avoids a processing sequence between two Steiner points to be constantly chosen by all ants. Once chosen, the scent $\pi(r, s)$ decreases slightly to
reduce the chance for other ants to pick the same one. Global trail updating rewards the complete processing order for all the points that yields the best solution within the iteration. The scent $\varphi(r, s)$ on the trail of the entire processing order is increased to attract other ants to follow. The updating formulas, are the same as in [17], shown in Eqs. (3) and (4), respectively.

**Local updating:**

$$\tau(r, s) \leftarrow (1 - \alpha) \cdot \tau(r, s) + \alpha \cdot \tau_0$$  \hspace{1cm} (3)

where $\alpha$ is a parameter about ratio and $\tau_0$ is assigned as a small value $(n - L_{MST})^{-1}$ for scent dissipation. The value $n$ is the total number of Steiner points and $L_{MST}$ is the MST length for the system.

**Global updating:**

$$\varphi(r, s) \leftarrow (1 - \alpha) \cdot \varphi(r, s) + \alpha \cdot \Delta \varphi(r, s)$$  \hspace{1cm} (4)

$\Delta \varphi(r, s) = (\text{shortest length})^{-1}$. The shortest length is the best yielded solution of the iteration, and the inverse gets bigger when a better solution is found. This gives a positive reinforcement.

4. Experiments

The learning algorithm has been implemented and tested to have a good performance. In this study, we are interested in understanding the potential on length saving after applying Steiner points. One experiment will account for different network sizes, ranging from 100 to 800 vertices. Because a power grid network consists of two types of entities: power sources and loads, another experiment exercises different ratios between them. For both experiments, 10 ants are running to perform swarm intelligence. The parameters are assigned as $\beta = 2$, $\alpha = 0.1$, and $q_0 = 0.9$ for the four equations in Section 3.2.

Figure 8 shows the result of the first experiment. The number of entities is set to be even for both types, regardless of the network size. The percentage of saving is collected after 1000 iterations for each run of the ants and the displayed result is an average of 100 runs. It can be seen that adding Steiner points gives a consistent result on length reduction around 2.3%. The network size does not have a bias on the saving.

The second experiment evaluates the impact caused by different number of entities of the two types. A network size of 400 vertices is used as the base. We increase the ratio on the number of power sources from 10% up to 90% and the variation on saving is depicted in Figure 9.

The result shows a network with a higher ratio of source nodes tends to yield a stronger saving. This can be explained from the two rules discussed in Section 3.1. To maintain fault tolerance, source nodes are likely to incorporate a Steiner point, when the degree is only $\geq 2$. Power loads however require the degree to be $\geq 3$. This is good news as green power resources can be abundant in number in the future. To offer a robust network and simultaneously save cost on transmission lines, smartly adding Steiner points can undoubtedly offer a good optimization solution. Figure 10 gives an optimization result for a network of size 200 with same amount of entities for both types. Many Steiner points can be clearly observed.
5. Conclusions and future work

Green power resources have become prominent and will play an important role in the near future. It is expected to see the growth of clean power resources to be in abundance. Unfortunately, the poor locality and the robustness on their connections can be an issue. Our past fault tolerance algorithm was able to strengthen a power grid network to prevent load centers from suffering power loss caused by the disconnection of one single transmission line. However, fault tolerance brings up a higher expense on the connections. In this study, we develop a swarm intelligence learning algorithm to further introduce Steiner points to the network. Experiments show a dependable result on the line length reduction to save cost. More excitingly, the saving percentage can be achieved even higher with more power sources being existent in the network.

The learning algorithm addresses a number of difficulties. It extends the scope to handle graph structures besides trees. The addition of Steiner points takes into account different types of entities. It tackles the addition sequence of the points, because the order affects the result. The allocation of the Steiner points takes advantage of the coordinate descent approach to iteratively find the correct location. More importantly, the algorithm does not compromise the fault tolerance status.

Future work in this area includes the consideration of terrains. This work considers space to have a uniform cost, but in reality installing transmission lines across an open plain is cheaper than across a mountainous region. Besides, since the luxury of designing a new network topology from scratch is rare, there is a need to examine how to augment an existing network to make it fault tolerant.

References