Multi-UAV path planning methodology for postdisaster building damage surveying - Supplementary Material

Ryosuke Nagasawa[1], **Erick Mas**[*2], **Luis Moya**[2,3], **and Shunichi Koshimura**[2]

[1]Mox-Motion, Tokyo, 161-0033, JAPAN
[2]International Research Institute of Disaster Science, Tohoku University, Sendai, 980-8572, JAPAN
[3]Japan-Peru Center for Earthquake Engineering and Disaster Mitigation (CISMID), Universidad Nacional de Ingeniería, Lima, PERU
[*]Corresponding author: mas@irides.tohoku.ac.jp

ABSTRACT

N/A

Basic Concepts of Methods Used

Traveling salesman problem (TSP)

The objective of the TSP is to find a route among several cities by which a salesman can visit each once (Hamilton circuit) but that also has the minimal total path length[2]. The MTSP is an extension of the TSP that requires minimization of the total path length of multiple routes, where all cities must be visited only once and by only one salesman. Moreover, the path lengths of the multiple paths must be uniform[4]. This combinatorial optimization problem is NP-hard, meaning that it cannot be fully solved in polynomial time. The amount of time needed to obtain the optimal solution by calculating all possible paths would be on the order of $\mathcal{O}(n!)$. If the number of cities is greater than 20, it cannot be solved in a practical amount of time. A famous MTSP solver, CONCORDE, provides the exact solution. This solver uses the branch-and-cut and branch-and-bound methods[1]. Recently, studies of heuristic algorithms for approximate solutions with guaranteed precision have become very popular. Several solution methods using various techniques have been proposed, such as the nearest neighbor (NN) method, the insertion method, genetic algorithms, the annealing method, and ant colony optimization. In this study, the NN method was applied to find the initial solution, and the 2-opt and 1.5-opt methods were applied to improve that initial solution[3]. Each method is introduced in the following discussion.

Nearest neighbor (NN) method

The NN method proceeds as shown below:

1. Select city A as the starting location.
2. Select an unvisited city that is closest to city A.
3. Connect city A and city B and move to the latter.
4. Return to step 2 to select an unvisited city, if any. Otherwise, proceed to step 5.
5. Link the current city to the starting city.

The amount of calculation required for this method is on the order of $\mathcal{O}(n \log n)$, and its guaranteed precision is $\mathcal{O}(\log n)$. Because paths with smaller costs are selected first, the selected paths will include those of the optimal solution. Therefore, much improvement can be expected. However, because paths with greater costs are left to the later part of the process, the solution precision is not good.

2-opt method

In the 2-opt method, the improvement shown below is added.

1. Select two paths from among all paths.

2. If exchanging these two paths makes the total path length shorter, then make that exchange.

3. Once this process has been performed for all possible combinations of paths, the method is complete.

As shown in Figure 1, the 2-opt method works well in improving a route if two paths on that route mutually cross. This method might require iterative improvement of exponential order in the worst case and might yield solutions with extremely poor precision. However, this does not occur frequently. In practice, this method enables good improvement.

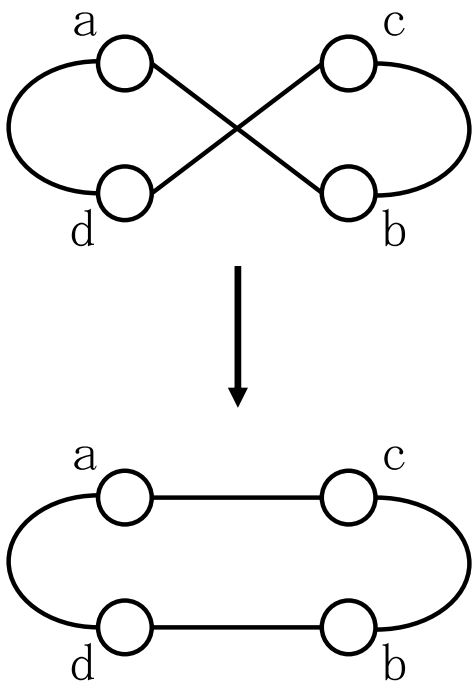

Figure 1. Illustration of the 2-opt method.

5-opt method

The 1.5-opt method provides an improved way to cope with local optimization problems, as shown in Figure 2, that the 2-opt method cannot solve.

1. Select an appropriate city a and one path.

2. If the insertion of city a into the path shortens the path length, then apply that insertion.

3. Repeat the steps above for all possible combinations of one city and one path to complete the operation.

This method is effective for a long selected path. It works well in reducing the lengths of the long paths that tend to appear in the final stage of the NN method.

A* algorithm

The A* algorithm is used to find a path between two points in a graph search problem. It performs a search operation using a heuristic function $h(n)$, which functions as an index for the search. Let n be the halfway point on the shortest path, let $\hat{g}(n)$ be the cost from the starting point to the current point, and let $\hat{h}(n)$ be the cost from the current point to the goal point. Then, the cost of the shortest route, $\hat{f}(n)$, is written as presented in Equation 1 below:

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \tag{1}$$

However, because both $\hat{g}(n)$ and $\hat{h}(n)$ are unknown, estimates of these functions, $g(n)$ and $h(n)$, are used instead. Although several functions are available for $h(n)$ for different purposes, the Euclidean norm is the most generally applicable one. The algorithm progresses as presented below.

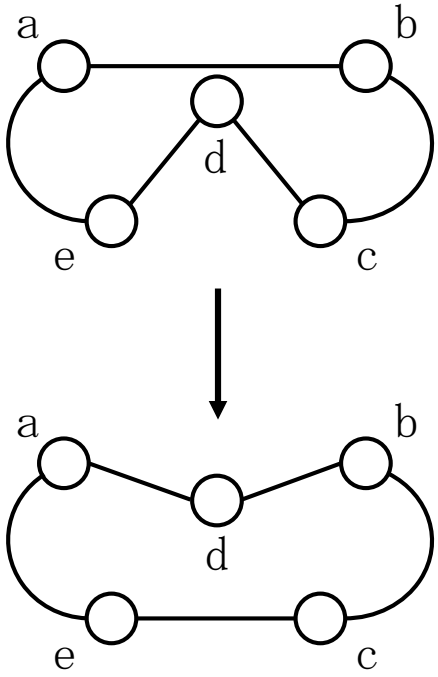

Figure 2. Illustration of the 1.5-opt method.

1. Prepare the starting node (S), the goal node (G), and the *OPEN* and *CLOSE* lists.

2. Add S to the *OPEN* list. At this time, $g(S) = 0$ and $f(S) = h(S)$.

3. Among the nodes in the *OPEN* list, select a node n for which $f(n)$ is the smallest.

4. If $n \in G$, then the search operation terminates. Otherwise, store n in the *CLOSE* list.

5. Calculate $f'(m) = g(n) + COST(n,m) + h(m)$ for all nodes m neighboring n. Here, $COST(n,m)$ is the cost of moving from n to m, and $g(n)$ is given by $g(n) = f(n) - h(n)$.

6. Record n as the parent of m and perform the process below depending on the state of m.

    - If m is contained in neither the *OPEN* list nor the *CLOSE* list, then designate $f(m) = f'(m)$ and add m to the *OPEN* list.

    - If m appears only in the *OPEN* list and if $f'(m) < f(m)$, then designate $f(m) = f'(m)$.

    - If m appears only in the *CLOSE* list and if $f(m) < f(m)$, then designate $f(m) = f(m)$ and store m in the *OPEN* list.

7. Repeat the steps above, beginning from step 3.

8. After the search operation, the shortest route traveling from the parents starting from G is found.

If the heuristic function $h(n)$ is not used, then this algorithm is the same as the Dijkstra algorithm. Although these two algorithms both incur a high calculation load, the amount of calculation may be reduced through the proper selection of $h(n)$.

References

1. Applegate, D., Bixby, R., Chv*á*tal, V., Cook, W. CONCORDE TSP Solver, 2001.
2. Kubo, M. Invitation to Traveling Salesman Problem (I). *[O]perations research as a management science [r]esearch*, 39, 25–31, 1994. (In Japanese)
3. Sakagami, T., Yoshizawa, M., Ohta, Y., Oyamaguchi, M. On the approximation algorithm for the Traveling Salesman Problem. *Res. Rep. Fac. Eng. Mie Univ.*, 25, 81–96, 2000. (In Japanese)
4. Watanabe, H., Ono, T., Matsunaga, A., Kanagawa, A., Takahashi, H. Multiple Traveling Salesman Problems Using the Fuzzy c-means Clustering. *Journal of Japan Society for Fuzzy Theory and Systems*, 13(1), 199–126, 2001.