Path-Planning Module Using Contraction Hierarchy NAITO-Rescue 2016 (Japan)

Kazuo Takayanagi[1], Shunki Takami[1], Yuki Miyamoto[1], Masahiro Yamamoto[1], Yoshiyuki Kozuka[1], Nobuhiro Ito[1], Kazunori Iwata[2]
rescue-2016@maslab.aitech.ac.jp

[1] Department of Information Science, Aichi Institute of Technology, Aichi, Japan
[2] Department of Business Administration, Aichi University, Aichi, Japan

Abstract. The Agent Development Framework (ADF) [1] contains modules for solving key research problems. In this paper, we focus on the route search problem, which is one of these research problem. We propose a contraction hierarchy algorithm that considers faults, and implemented a new PathPlanning modules. Further, we describe the implementation of each module.

Keywords: Rescue Simulation, Path Planning, Contraction Hierarchy Algorithm, Agent Development Framework

1 Introduction

The Agent Development Framework(ADF) contains research problems to be addressed in the RoboCupRescue Simulation (RCRS). The path finding problem is one of them. In the path finding problem, agents are required to move to a destination using the shortest path while avoiding blockades. Further, agents should calculate this path within a short time. In this paper, we solve the path finding problem using precomputation. The path finding algorithms that use precomputation can be divided into highway-node routing algorithms and contraction hierarchy algorithms. The contraction hierarchy algorithm calculation time is shorter than that of the highway-node routing algorithm for large networks. Therefore, we selected the contraction hierarchy algorithm because the RCRS map has large network. We also describe the design of the modules that implement our proposed method. In addition, we describe the design of the TargetSelector modules that also implemented our some idea. Note that the contraction hierarchy algorithm was used in the team description paper (TDP) of the Agent Simulation Competition. The remainder of this paper is organized as follows. In Section 2, we describe our adaptation of the contraction hierarchy algorithm. In Section 3, we describe the TargetSelector module designs. Finally, in Section 4, we provide our conclusions and directions for future work.

2 Adaptation of the Contraction Hierarchy Algorithm

In this section, to resolve path search within the RCRS, we describe an adaptation of the contraction hierarchy algorithm. We first briefly outline this algorithm, including its key definitions and importance calculations. We explain our proposed method, and then present our experimental results and evaluation.

2.1 Contraction Hierarchy Algorithm

The contraction hierarchy algorithm, developed by Geisberger et al. in 2008, is divided into two stages: precomputation and shortest path calculation[2]. The precomputation stage constructs a hierarchy that stores the path nodes. The order of these nodes is determined by their relative importance. In this paper, we use edge difference to compute importance. More specifically, we calculate the importance of each node in the first graph, and the lowest importance node is stored in the hierarchy. In this case, the hierarchy and the node directly correspond to one another. When all nodes are stored in the hierarchy, the precomputation stage terminates. The shortest path search stage calculates the shortest path based on the graph calculated in the precomputation stage. This is the approach that the contraction hierarchy algorithm uses to search for the shortest path.

2.2 Key Definitions

The key definitions are shown in Tables.1 and 2.

Table 1. Symbol definitions

Key	Explanation
G	The weighted graph
V	The set of nodes
E	The edge connecting two nodes
C	A cost function
B	A fault incidence function

Table 2. Term definitions

Name	Definition
Graph	We denote a road network as a weighted graph G, which is defined as $G = (V, E, C)$
Path	A path from v_1 to v_k is denoted as a sequence of nodes $P = (v_1, v_2, \ldots, v_k) \in V \times V \times \cdots \times V$ with v_i adjacent and connected to v_{i+1} by $e_{i,i+1} \in E$ for $1 \leq i < k$
Cost of Path	The cost of path $(v_1, v_2, ..., v_k)$ is as follows: $c_{1,2} + c_{2,3} + c_{3,4} + ... + c_{(k-1),k} (2 \leq k \leq n)$
Shortest Path	The shortest path from v_s to v_g is path $P = (v_1, v_2, \ldots, v_n)$ $(with\ v_1 = v_s\ and\ v_n = v_g)$ that, over all possible n, minimizes $\sum_{i=1}^{n-1} c_{i,i+1}$; the cost of the shortest path is called *a minimum cost for the path*
Shortcut	New edges $e_{x,y}$ that represent the shortest path $(v_x, ..., v_y)$
Agents	An agent moves similarly to a vehicle from a start node to a destination node using a map.
Fault Information	Fault information denotes information about blocked roads that an agent cannot pass through.
Detour Path	The next-best shortest path when there is a fault on the shortest path.

2.3 Importance

As mentioned earlier, importance is measured using edge difference. The value of the edge difference of node v is the difference between the number of edges connected to node v and the number of shortcuts belonging to the same level as the node. In Fig. 1, we show an example of the edge differences. In the figure, solid lines indicate edges, while dotted lines show shortcuts that are at the same level as the given node. Further, the numbers on each edge or shortcut represents cost.

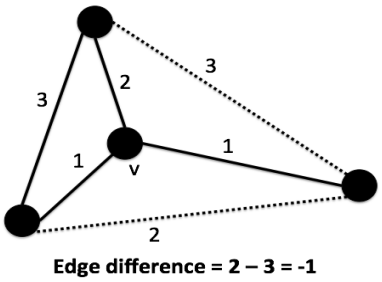

Fig. 1. Edge differences

2.4 Considering Fault Incidences

We introduced fault incidence into the contraction hierarchy algorithm; thus, in this section, we explain our extended contraction hierarchy algorithm. Specifically, there are two paths ((v_x, v_a, v_y) and (v_x, v_b, v_y)), which together become a shortcut, as shown in Fig. 2. In node $v(m, n)$ of Fig. 2 , m represents cost and n represents fault incidence. Further, if three or more paths exist, we randomly choose two paths and the third path becomes the preparatory path.

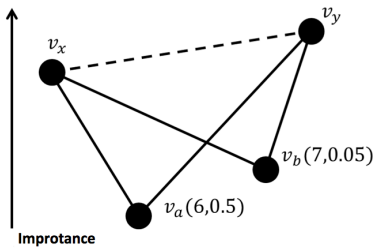

Fig. 2. Shortcut and detour paths

In the RCRS, when a failure incident occurs along the shortest path, we must then consider a detour path. Here, we choose a path in which the predicted distance considers the detour path as the shortest. We calculate the cost $c_{x,y}$ of the shortcut that connects nodes v_x and v_y using fault incidence to determine $c_{x,y}$. Moreover, it is necessary to compare the cost of the resulting path to other paths. In Fig. 2, when path α and path β are set to (v_x, v_a, v_y) and (v_x, v_b, v_y),

respectively, the costs are defined as c_α and c_β. In addition, fault incidences of edges $e_{x,a}$ and $e_{a,y}$ are denoted by $b_{x,a}$ and $b_{a,y}$, respectively. Therefore, if fault incidence b_α occurs on path α, b_α is defined as shown below.

$$b_\alpha = 1 - (1 - b_{x,a})(1 - b_{a,y})$$

We define fault incidence b_β for path β similarly.

A shortcut is composed of two or more edges; however, we would like to calculate the path while keeping the shortcut wherever possible. This is a feature of the contraction hierarchy algorithm. Hence, we perform the calculation without deploying the shortcut. Therefore, it is necessary to calculate the cost and fault incidence of the edge from the shortcut cost and fault incidence. In this case, costs and fault incidences of the two edges are approximately calculated as an equivalence. Therefore, costs $c_{x,a}$ and $c_{a,y}$ and fault incidences $b_{x,a}$ and $b_{a,y}$ of edges $e_{x,a}$ and $e_{a,y}$ of path α are defined below.

$$c_{x,a} = c_{a,y} = \frac{c_\alpha}{2}, b_{x,a} = b_{a,y} = 1 - \sqrt{1 - b_\alpha}$$

Costs and fault incidences of the edges of path β are calculated similarly.

Using the above equations, we propose the following equation to calculate the predicted distance $s_\alpha(x, y)$ and $s_\beta(x, y)$ of paths α and β, respectively, given the fault incidence on the path from x to y.

$$s_\alpha(x,y)=\frac{1}{1-b_\alpha}\{c_\alpha+(1-b_\alpha)c_\alpha+(1-\sqrt{1-b_\alpha})c_\beta+\sqrt{1-b_\alpha}(1-\sqrt{1-b_\alpha})(c_\beta+\frac{c_\alpha}{2})\} \quad (1)$$

Here, if the fault incidence is low, the value of 1 increases. Note that $(1-b_\alpha)c_\alpha$ is the expected value of the case in which path α can be traversed. Further, $(1 - \sqrt{1 - b_\alpha})c_\beta$ is the expected value of the case in which the agent traverses path β when a fault occurs on edge $e_{x,a}$. Finally, $\sqrt{1 - b_\alpha}(1 - \sqrt{1 - b_\alpha})(c_\beta + \frac{c_\alpha}{2})$ is the expected value of the agent traversing path β and turning back when a fault occurs on edge $e_{a,y}$.

In the extended algorithm, the optimal path is selected as the smaller one by comparing the predicted moving distance values calculated in (1). If three or more paths exist, the lowest cost one is selected as the optimal path. Next, we randomly select a path from among the spare paths. We compare it with the optimum route and using the path costs calculated by (1). This calculation continues until there are no more spare paths.

2.5 Proposed Method

The process flow of our proposed method is as follows.

Precomputation

1. Calculate the importance of the current node, and then add it to the hierarchy.
2. Remove the lowest important node from the hierarchy.
3. Add the shortcut to the input graph.
4. If a shortcut that connects the nodes of the new shortcut already exists, use the one with the lowest cost.
5. Repeat steps 1 through 4 until two nodes are left in the hierarchy.

Shortest Path Search

1. Generate a list of paths to the nodes connected to the start point (the start side list).
2. Generate a list of paths to the nodes connected to the destination(the destination side list).
3. Calculate the cost of nodes connected by edges in a shortcut from the start point and search this path.
4. If a path that connects to the destination is found, output this path; however, if the path is the shortcut, expand it to the path, output the path, and exit.
5. Add the path to the start side list. This path is the path to a node connected by an edge that contains the shortcut to the selected node from the start point side; however , this path does not contain the start point.
6. Repeat, as much as possible, steps 3 through 6 as required. While performing these steps, if the same path containing a node of the search target in the start side list in step 3 exists, the path of that node that has lowest cost is selected.
7. Calculate the cost of nodes connected by edges that contain a shortcut from the destination and search this path.
8. If a path that connects to the start point is found, output the path; however, if the path is the shortcut, expand it to the path, output the path, and exit.
9. Add the path to the destination side list. This path is the path to a node that is connected by an edge that contains a shortcut to the selected node from the destination side; however, it does not contain the start point.
10. If there is a node of the search target that leads to the last point of the path of the start side list, the path that adds the destination side list to the start side list becomes a candidate for the optimum route.
11. Repeat, as required, steps 7 through 10. While performing these steps, if the same path of a node of the search target of the start side list in step 7 exists, the path of that node with the lowest cost is selected.
12. If two candidate paths exist for the optimal path, the lowest cost path is selected as the optimal path.
13. If the optimal path contains shortcuts, expand it to the path.
14. Output the path and exit.

2.6 Experiments

Experimental Methodology

We compared our proposed method to Dijkstra 's algorithm [3] and the original contraction hierarchy algorithm using the following metrics:

- Moving distance
- Number of faults discovered
- Number of nodes the search process traverses (number of hops).

The precomputed data of our proposed method and the contraction hierarchy algorithm as well as the map data and its fault incidences in graph from were all given in advance. Further, in our simulation, we assume our experiments take place in an environment after a disaster, hence we consider it appropriate

to perform the precomputations before the disaster. The moving distance and number of faults discovered per number of hops yield the results for Dijkstra's algorithm when failures do not occur. Further, the number of faults discovered was the number of faults encountered per 100m.

Experiment Maps

We used a 1:25,000 scale map data of all areas in Japan. These maps are published by the Geographical Survey Institute [4] and the data are expressed in G-XML, which is a Japanese Industrial Standards format. We used the road information, which consists of nodes and edges. Road node V in weighted graph G depicts an intersection. Road edges E in weighted graph G, are roads, each of which connects two road nodes. The cost function C is the length of each edge. Blocked road rate function B was created using city liquefaction hazard maps [5]. We used two areas in our experiments, as summarized in Table 3. For each are, 10,000 experiments were performed. In each experiment, the start and destination nodes of an agent were randomly selected.

Table 3. Two sample areas

Area No.	Area	$\|V\|$	$\|E\|$
1	Atsuta ward	1,760	2,609
2	Minato ward	4,892	7,127

Table 4. Results of our experiments

	The moving distance		The number of faults discovered	The number of hops	
	Avg.	S.D.	count	Avg.	S.D.
Area 1: Atsuta ward					
Dijkstra's	1.66080	4.85629	0.20313	1.5489	0.6884
Contraction hierarchy	1.50747	4.68011	0.21674	1.5501	0.6899
Our proposed method	1.36081	0.79169	0.03127	1.3369	0.7360
Area 2: Minato ward					
Dijkstra's	1.45009	1.03418	0.26991	1.5127	0.5092
Contraction hierarchy	1.39170	0.39416	0.27670	1.5131	0.5091
Our proposed method	1.26117	0.33277	0.02286	1.1587	0.3493

Results and Discussion

The moving distance, number of encountered faults, and number of hops of the proposed method were less than those of the other existing methods in an environments where faults occurred. The results of the existing methods contained edges with high fault incidences because such methods do not consider the fault incidence of edges. Our proposed method calculates a path while avoiding edges with high fault incidences, i.e., its results do not contain many edges with high

fault incidences. Therefore, we conclude that using it is possible for our proposed method to calculate a more efficient path.

3 Module Design

In this section, we describe the modules to improve the ADF, which are CHPathPlanning, NAITOVictimSelector, NAITOBuildingSelector and NAITOBlockadeSelector. We explain each module in detail.

3.1 CHPathPlanning

We create a CHPathPlanning module as a new PathPlanning module. CHPathPlanning implements the proposed method described in Section 2. In the implementation, we define the area of the RCRS as a node. Further, the fault incidence is determined using the area and building distribution around the area node.

3.2 NAITOBuildingSelector

We create the NAITOBuildingSelector module as a new TargetSelector module of the FireBrigade (FB). The NAITOBuildingSelector performs the combined clustering and allocation method to determine the responsible area of the agent. Clustering is done using the k-means algorithm [6] and the Hungarian algorithm [7] is used for allocation. The behavior of the NAITOBuildingSelector depends on the presence or absence of fire.

- If there is no fire, the agent patrols the fixed responsible area and searches for civilians and fire.
- If there is a fire, the agent uses available information to predict the location of the center of the fire in the map and cluster area to extinguish it and keep it from spreading.

3.3 NAITOBlockadeSelector

We create the NAITOBlockadeSelector module as a new TargetSelector module of PoliceForce (PF). The NAITOBlockadeSelector is the module of the TargetSelector that targets a Blockade. It performs clustering and allocation to determine the responsible area of the agent using k-means for clustering and Hungarian algorithm for allocation. The behavior of the NAITOBuildingSelector is divided into two modes.

- Mode1 : The agent clears the Blockade on a Civilian path. However, the agent does not traverse the path once the Blockade has been removed.
- Mode2 : The PoliceForce processes tasks in the following order.
  $rescuerequests > tasksinhighimportanceareas > othertasks$
  As for the requests of other agents, the Agent prioritizes the agents that are close.

3.4 NAITOVictimSelector

We create the NAITOVictimSelector module as a new TargetSelector module of AmbulanceTeam (AT). NAITOVictimSelector is a module of TargetSelector that targets Civilians and Agents. The NAITOVictimSelector performs clustering and allocation to determine the responsible area of the agent using k-means for clustering and the Hungarian algorithm for allocation. The Agent gives the highest priority to the rescue of the AmbulanceTeam. The priority of the other agents is calculated using the following inequalities.

$$The distance between the nearest Agent * 2 < distance of Civilian$$

$$The distance agent * 2 < distance of FB - PF ? AT : FB \cdot PF$$

4 Conclusions

In this paper, we focused on the path planning problem and proposed a new method for solving it. We also proposed four new modules. In the future, we plan to implement the modules described in this TDP. If possible, we also plan to compare our TargetSelector with other TargetSelectors to implement our TargetSelector more effectively.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP 16K00310 and 26330166.

References

1. Kazuo Takayanagi, Shunki Takami, Nobuhiro Ito, Kazunori Iwata. 2015. RCRS-ADF. https://github.com/RCRS-ADF/RCRS-ADF
2. Geisberger, R., Sanders, P., Schultes, D., & Delling, D. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. Experimental Algorithms, 319-333.
3. Dijkstra, E.W. 1959. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269-271.
4. Geospatial Information Authority of Japan. 2003. Gsi:1/25,000 map information of japan. http://www.gsi. go.jp (in Japanese).
5. Nagoya City Hall. 2004. Earthquake map of nagoya city (in japanese). http://www.city.nagoya.jp/kurashi/category/20-2-5-6-0-0-0-0-0-0.html
6. MacQueen, J. B. 1967. Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. 281-297
7. Harold W. Kuhn. 1955. The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly. 83-97. Kuhn's original publication.