B3–6

A FAST ALGORITHM FOR FINDING BETTER ROUTES BY AI SEARCH TECHNIQUES

Takahiro Ikeda, Min-Yao Hsu, Hiroshi Imai
Department of Information Science, University of Tokyo, Hongo, Tokyo 113, Japan

Shigeki Nishimura, Hiroshi Shimoura, Takeo Hashimoto, Kenji Tenmoku, and Kunihiko Mitoh
Sumitomo Electric Industries, Ltd., Osaka, Japan

Abstract

The shortest path problem is one of the most fundamental problems applicable in various fields, and has close relation to route navigation systems. This paper surveys algorithms for the two-terminal shortest path problem and proposes bidirectional A* algorithm based on a new approach. This algorithm is suitable for finding not only the shortest route but also better routes. The efficiency and the property of these algorithms are discussed through experiments applying them to an actual road network.

1 Introduction

Presenting the route minimizing the necessary time to the destination is one of the most fundamental function required for route navigation systems. From the difficulty of keeping information on appropriate routes for all pairs of the origin and the destination and the possibility of using dynamic information such as traffic flow, this problem must be solved at every request by the user. In this way, it is important to produce more effective method to search appropriate route on a road network.

This subject is generalized as the two-terminal shortest path problem on graphs, which is the problem to find the shortest path from s to t on a directed graph $G = (V, E)$ such that the length of an edge (u, v) in E is $l(u, v)$ where the origin s and the destination t are given. Because of wide range of its application, this kind of problem has been studied in various fields for long time. For example, the Dijkstra method is a traditional algorithm for this problem. Besides, the A* algorithm [2][6] and the bidirectional search algorithms [3][4] are well-known algorithm based on AI (Artificial Intelligence) search techniques.

This paper surveys these algorithms from the point of view of applying them to the two-terminal shortest path problem on the road network, and proposes bidirectional A* algorithm based on a new approach based on the technique translating the A* algorithm into the Dijkstra method. This algorithm is suitable for finding not only the shortest route but also better routes, namely suboptimal routes as the candidates for alternative routes containing paths not the shortest but less number of turns, for instance. The efficiency and the property of these algorithms are discussed through experiments using actual road data.

2 Preliminary Works

2.1 The Dijkstra Method

The Dijkstra method is a basic algorithm to solve the shortest path problem. Although the original algorithm by Dijkstra is for the single-origin all-destination problem, it is applicable for two-terminal problem with little modification. The following is the outline of the Dijkstra method for two-terminal problem.

1. Let S be the empty set, and $p_s(v)$, the potential of a vertex v, be $+\infty$ except for the origin s. Let $p_s(s)$ be zero.
2. Find the vertex v_0 which has the minimum potential in $V - S$ and add v_0 to S. If v_0 equals to t, then halt.
3. For all vertices v such that (v_0, v) is in E, if $p_s(v_0) + l(v_0, v)$ is less than $p_s(v)$, replace the path from s to v with the path from s to v_0 and the edge (v_0, v), and let $p_s(v)$ be $p_s(v_0) + l(v_0, v)$.
4. Go to step 2.

On this algorithm, each vertex keeps the shortest path from s among have been searched and its potential denotes the length of this path. A vertex is added to S if its current shortest path from s is shorter than all other paths kept by vertices. This indicates S consists of vertices such that the actual shortest paths from s to them has been already decided. In this way, Dijkstra method finds the shortest path from s to a vertex in order of the path length from s to the vertex. This suggests the searched area with this algorithm is in a circle with s as the center if G is homogeneous. Dijkstra method has the disadvantage of searching in all directions regardless of the place of t for two-terminal shortest path problem.

0-7803-2105-7/94/$4.00 © 1994 IEEE

2.2 The A* Algorithm

The A* Algorithm finds the shortest path from s to t more efficiently when a heuristic estimate of the shortest path length from each vertex to t, which must be less than the actual shortest path length, is given [2][6]. Let $h_s(v)$ denote this estimate for a vertex v. Then the outline of the A* Algorithm is described as follows:

1. Let S be the empty set, and $p_s(v)$, the potential of a vertex v, be $+\infty$ except for the origin s. Let $p_s(s)$ be zero.
2. Find the vertex v_0 which has the minimum value of $p_s(v) + h_s(v)$ in $V - S$ and add v_0 to S. If v_0 equals to t, then halt.
3. For all vertices v such that (v_0, v) is in E, if $p_s(v_0)+l(v_0, v)$ is less than $p_s(v)$, replace the path from s to v with the path from s to v_0 and the edge (v_0, v), and let $p_s(v)$ be $p_s(v_0)+l(v_0, v)$, and remove v from S if v is in S.
4. Go to step 2.

On this algorithm, $p_s(v)$ also denotes the temporary shortest path length from s to v, and $p_s(v) + h_s(v)$ denotes the temporary estimate for the shortest path length from s to t via v. Hence A* algorithm searches vertices considered to be on the shortest path from s to t preferentially. This implies searched vertices distribute toward t with an appropriate estimator. If estimator tells the actual shortest path length to t, the algorithm never searches vertices off the shortest path from s to t.

The assumption $h_s(v)$ is less than the actual shortest path length from v to t is necessary to guarantee the finally obtained path is the shortest in all paths from s to t. The algorithm using an estimator without this assumption is called A algorithm. A* algorithm is defined as the special type of of A algorithm properly.

In the A* algorithm, the shortest path from s is not always fixed for each vertex in S, that is, shorter paths may be found in future search. This is the reason why the vertex is removed from S in step 3 when its potential is renewed. This ineffective operation inducing the increase of searches can be eliminated if the estimator is dual feasible as the following definition.

Definition 1 *The estimator for the shortest path to t, h_s, is dual feasible if and only if h_s satisfies the following condition for each edge (u, v) in E:*

$$l(u, v) + h_s(v) \geq h_s(u) \ . \qquad (1)$$

If the graph is a road network, Euclid distance between a vertex and t is utilized as the dual feasible estimate for the shortest path from the vertex to t.

2.3 The Bidirectional Dijkstra Method

Previous two methods are unidirectional search algorithms searching vertices with the origin as the center. In these algorithms, the destination plays a minor role than the origin. On the other hand, bidirectional search algorithms utilize both the origin and the destination uniformly by searching alternatively from the origin side and from the destination side. In the following, a forward search denotes a search from the origin side and a backward search denotes a search from the destination side for convenience.

The bidirectional Dijkstra method uses the Dijkstra method both for forward and backward searches [3][4]. The outline of the bidirectional Dijkstra method is described as follows.

1. Let S and T be the empty sets, and the potential of a vertex v, $p_s(v)$ for s and $p_t(v)$ for t, be $+\infty$ except for s and t respectively. Let $p_s(s)$ and $p_t(t)$ be zero.
2. Find the vertex v_0 which has the minimum potential for s in $V - S$ and add v_0 to S. If v_0 is in T, then go to step 7.
3. For all vertices v such that (v_0, v) is in E, if $p_s(v_0)+l(v_0, v)$ is less than $p_s(v)$, replace the path from s to v with the path from s to v_0 and the edge (v_0, v), and let $p_s(v)$ be $p_s(v_0) + l(v_0, v)$.
4. Find the vertex v_0 which has the minimum potential for t in $V - T$ and add v_0 to T. If v_0 is in S, then go to step 7.
5. For all vertices v such that (v, v_0) is in E, if $l(v, v_0)$ $+p_t(v_0)$ is less than $p_t(v)$, replace the path from v to t with the edge (v, v_0) and the path from v_0 to t, and let $p_t(v)$ be $l(v, v_0) + p_t(v_0)$.
6. Go to step 2.
7. Find the edge (u, v) minimizing $p_s(u) + l(u, v) + p_t(v)$ such that u is in S and v is in T. The shortest path from s to t consists of the path from s to u and the edge (u, v) and the path from v to t if $p_s(u) + l(u, v) + p_t(v)$ is less than $p_s(v_0) + p_t(v_0)$, otherwise it consists of the path from s to v_0 and the path from v_0 to t.

After forward and backward searches reach the same vertex, the shortest path from s to t is obtained by simple post-processing such as step 7. This is due to the property of the Dijkstra method that it decides the shortest path from s to a vertex in order of the path length from s to the vertex. If G is homogeneous, the number of searched vertices is approximately half of that with unidirectional Dijkstra method since searched vertices distributes in two circles with two terminals as the centers in this case.

3 A New Approach for a Simple Bidirectional A* Algorithm

It is natural to use A* algorithm for both forward search and backward search to reduce the searched area. Assume that for each vertex v a heuristic estimate for the

shortest path from v to t, $h_s(v)$, and from s to v, $h_t(v)$, are given. The algorithm using h_s as the estimator for the forward search and h_t as the estimator for the backward search has been proposed [1][5]. This method corresponds to applying the orthodox A* algorithm independently both from s and t. Although this approach is natural, the algorithm is complex because it cannot specify the shortest path from s to t when searched areas from both sides overlap each other. This means the shortest path from s to t does not always pass the overlapped point and the algorithm cannot stop searching after reaching the vertex have been searched from another side. This is due to the fact that the algorithm utilizes independent estimators for two inner A* algorithms and does not change if both estimators are dual feasible.

In order to avoid such a demerit, this paper proposes a new bidirectional A* algorithm based on another approach based on the following technique translating the A* algorithm into the Dijkstra algorithm in the case that each estimator is dual feasible.

Theorem 1 *The Dijkstra method using the edge length l' modified as follows is equivalent to the A* algorithm using the original edge length l with dual feasible estimator h_s:*

$$l'(u,v) = l(u,v) + h_s(v) - h_s(u) \ . \tag{2}$$

Proof: Since $l'(u,v)$ is not negative from dual feasibility of h_s, it is possible to use $l'(u,v)$ for the length of an edge (u,v) in the Dijkstra method. Let P be the temporary path from s to a vertex v. Then the following formula is satisfied for the potential of v in the Dijkstra method:

$$\begin{aligned} p_s(v) &= \sum_{(u,u')\in P} l'(u,u') \\ &= \sum_{(u,u')\in P} l(u,u') + h_s(v) - h_s(s) \ . \end{aligned}$$

Notice that $h_s(s)$ is a constant and $\sum_{(u,u')\in P} l(u,u')$ denotes the potential of v in the A* algorithm. This indicates the Dijkstra method using modified edge lengths is equivalent to the A* algorithm. □

This theorem indicates the the A* algorithm can be transformed into the Dijkstra method by using special edge length given by (2) if h_s is dual feasible. This means bidirectional A* algorithm is realized by applying the bidirectional Dijkstra method using modified edge lengths by (2). However, this method has a problem that backward search does not changed to A* algorithm because (2) does not contain h_t, the estimator for the shortest path from s.

To apply A* algorithm using h_t to the backward search, the length of a edge (u,v) must be changed as follows:

$$l'(u,v) = l(u,v) + h_t(u) - h_t(v) \ . \tag{3}$$

Hence it is natural to adopt the following l' as the new edge length for translating both forward and backward searches into A* algorithm:

$$\begin{aligned} l'(u,v) = l(u,v) &+ \frac{1}{2}(h_s(v) - h_s(u)) \\ &+ \frac{1}{2}(h_t(u) - h_t(v)) \ . \end{aligned} \tag{4}$$

This l' is always not negative because of dual feasibility of h_s and h_t. This modification of edge lengths corresponds to utilizing $(1/2)(h_s(v) - h_t(v))$ as the estimate for the shortest path from v to t and $(1/2)(h_t(v) - h_s(v))$ as the estimate for the shortest path from s to v. These new estimators satisfy the following inequalities whenever $h_s(v)$ and $h_t(v)$ have meaningful value, that is not negative value, for each vertex v:

$$h_s(t) \geq \frac{1}{2}(h_s(v) - h_t(v)) \ , \tag{5}$$

$$h_t(v) \geq \frac{1}{2}(h_t(v) - h_s(v)) \ . \tag{6}$$

Each inequality guarantees the corresponding new estimator does not exceed the actual shortest path length for which it estimates. In this way, the following theorem is established.

Theorem 2 *The bidirectional Dijkstra method using the edge length l' defined as (4) is equivalent to a bidirectional A* algorithm using the original edge length l utilizing $(1/2)(h_s(v) - h_t(v))$ as the estimate for the shortest path from v to t and $(1/2)(h_t(v) - h_s(v))$ as the estimate for the shortest path from s to v.*

Inequalities (5) and (6) indicate new estimators are inferior than original ones with regard to their performance as estimators. However this approach has the advantage of realizing a simple bidirectional A* algorithm, which can specify the shortest path with less operations when searched area from both sides overlap each other.

This algorithm is suitable for finding better routes moreover. Better routes are regarded as suboptimal routes which are the candidates for alternative routes. Suppose that better paths are defined as the paths whose length are longer than the shortest path length by at most α where α is some constant. These better paths can be substitutes for better routes. Although it is possible to find such better paths easily by deciding the shortest paths both from s and t for all vertices, this method has the disadvantage of the cost for calculation. With the bidirectional A* algorithm in this paper, better paths are obtained by resuming the forward search until finding the path whose length is larger than the shortest path by α. Because each vertex in the forward and backward searched areas knows the shortest path from the corresponding terminal at that point, all better paths are in these areas. Then searches in these bounded areas, which are rather small due to the directionality of A* algorithm, are required for finding better paths.

4 Experiments on a Road Network

In this section, the efficiency of algorithms described in this paper is investigated based on experiments applying

Table 1: The number of searched nodes in the forward search and backward search of the shortest path from Machida to Tokorozawa.

Algorithm	Forward	Backward
Unidirectional Dijkstra	16177	0
Unidirectional A*	6303	0
Bidirectional Dijkstra	2559	2587
Bidirectional A*	1604	1706

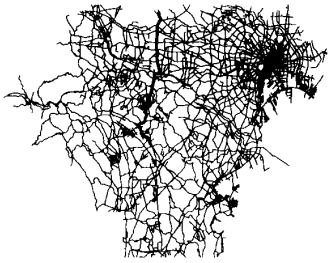
them to the actual road network of 160 kilometers times 80 kilometers region in Tokyo metropolitan area. As the length of each edge, the necessary time to pass the edge is utilized in experiments rather than the distance along the edge. Accordingly, Euclid distance to t and from s divided by the maximum speed 105 kilometers per hour is used as h_s and h_t respectively.

4.1 The Efficiencies of Various Algorithms

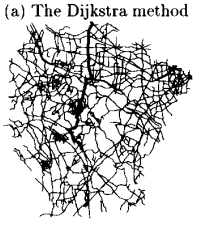
The first experiment is for the comparison of algorithms described in this paper. The experiment to find the shortest path from Machida to Tokorozawa has been performed with the following four type of algorithms: the Dijkstra method, the A* algorithm, the bidirectional Dijkstra method, and the bidirectional A* algorithm based on Theorem 2. Both terminals are selected in the suburbs where the network is almost homogeneous to avoid the influence of the network topology. Table 1 shows the number of searched nodes for each case. Figure 1 illustrates the distribution of searched edges. The bold line denotes the shortest path from Machida, the lower terminal, to Tokorozawa, the upper terminal.

Figure 1 directly shows the property of each algorithm. For the Dijkstra method, searched edges distributes in all directions from the origin. On the other hand, A* algorithm does not search deeper on the opposite direction to the destination. The number of searched nodes with A* algorithm is about 40 percent of that with the Dijkstra method from Table 1. Whether this percentage is regarded as small or large is an arguable question. In this case, the performance of the estimator is not excellent because the estimator always calculates the necessary time to the destination using the maximum speed. Although this is inevitable for guaranteeing the shortestness of the obtained path, it is no problem to use stronger estimator in practice because only few ways are to be passed with maximum speed.

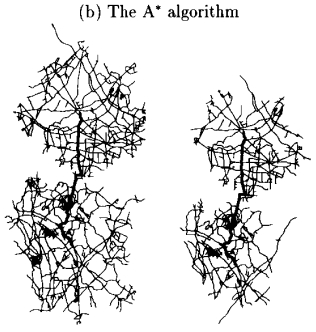
Table 2 shows the number of searched nodes and the path length, that is, the necessary time along the path, with A* algorithm using a k times stronger estimate $k \cdot h_s(v)$ for a vertex v. where k is in the range from one to four. The algorithm is the A* algorithm if k equals to one, and is the A algorithm otherwise. The result indicates the obtained path length is the shortest until k is less than or equals to two. For the case k equals to two, the number of searched nodes is less than half of that for the original case. Figure 2 illustrates the



(a) The Dijkstra method



(b) The A* algorithm



(c) The bidirectional Dijkstra method (d) The bidirectional A* algorithm

Figure 1: The distribution of searched edges on finding the shortest path from Machida to Tokorozawa.

Figure 2: The distribution of searched edges with A* algorithm using two times stronger estimator.

Table 2: The number of searched nodes and the path length with A* algorithm using k times stronger estimator.

k	The Number of Nodes	The Path Length
1.0	6303	2902
1.5	4072	2902
2.0	2485	2902
2.5	1078	2906
3.0	450	2966
3.5	163	3057
4.0	120	3035

distribution searched edges for the case k equals to two.

Again on Table 1 and Figure 1, the ability of bidirectional search algorithms is conspicuous. The bidirectional Dijkstra method only searches approximately a third of nodes compared with the unidirectional Dijkstra method. The bidirectional A* algorithm moreover reduces searches towards opposite directions and makes the number of searched nodes about 60 percent of that with the bidirectional Dijkstra method.

In the following section, the property of bidirectional A* algorithm is investigated with an experiment for more general case.

4.2 The Property of the Bidirectional A* Algorithm

The second experiment is for investigation of the bidirectional A* algorithm based on Theorem 2. In this experiment, the formula translating the edge length is defined as follows where k_s and k_t are parameters:

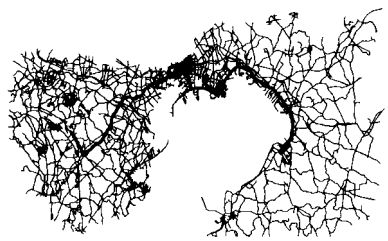
$$l'(u,v) = l(u,v) + k_s(h_s(v) - h_s(u)) + k_t(h_t(u) - h_t(v)) \ . \qquad (7)$$

By changing the combination of k_s and k_t, this algorithm is transformed into various type of bidirectional search algorithms. The experiment has been performed for the following four combinations of (k_s, k_t): (0, 0), (1, 0), (0, 1), (1/2, 1/2). The case (k_s, k_t) equals to (1/2, 1/2) corresponds to the bidirectional A* algorithm based on Theorem 2. Each estimator in the forward search and the backward search of this bidirectional A* algorithm does not equal to h_s and h_t respectively. The aim of introducing the parameters k_s and k_t is to observe the case this restriction is removed for the one estimator at the sacrifice of the other estimator.

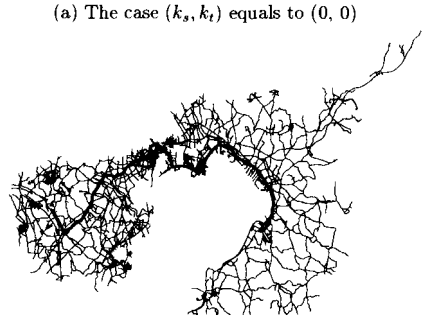
The experiment has been performed with the origin Machida and the destination Ichihara. They are selected as the two location between which the Tokyo bay lies in order to examine the effect of the A* algorithm for the case the shortest path is far away from the straight line. Table 3 shows the number of searched nodes and Figure 3 illustrates the distribution of searched edges as the result of this experiment. The bold line denotes the shortest path from Machida, the left terminal, to Ichihara, the right terminal.

Table 3: The number of searched nodes in the forward search and backward search of the shortest path from Machida to Ichihara.

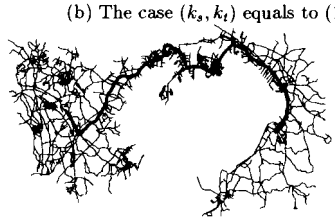
(k_s, k_t)	Forward	Backward
(0, 0)	7110	7064
(1, 0)	5707	5944
(0, 1)	4642	4436
(1/2, 1/2)	4898	4831



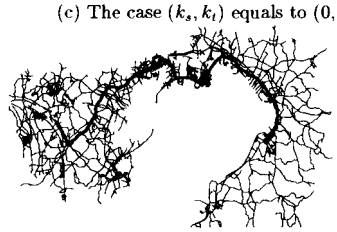
(a) The case (k_s, k_t) equals to (0, 0)



(b) The case (k_s, k_t) equals to (1, 0)



(c) The case (k_s, k_t) equals to (0, 1)



(d) The case (k_s, k_t) equals to (1/2, 1/2)

Figure 3: The distribution of searched edges on finding the shortest path from Machida to Ichihara.

The algorithm for the case (k_s, k_t) equals to (0, 0) is the bidirectional Dijkstra method because edge lengths are not modified in this case. Figure 3(a) shows the uniform distribution of searched nodes as the standard case.

The algorithm for the case (k_s, k_t) equals to (1, 0) corresponds to a kind of bidirectional A* algorithm utilizing $h_s(v)$ as the estimate in the forward search and $-h_s(v)$ as that in the backward search for a vertex v. The backward search of this algorithm is meaningless A* algorithm, while the forward search of this algorithm is complete A* algorithm, because $-h_s(v)$ denotes the distance between t and v and has no information on the relation between s and v required for the estimator in the backward search. The farther away from t a vertex is, the smaller the estimate for the shortest path length from s to a vertex v with this eccentric estimator is. This implies a vertex farther away from t is searched preferentially in the backward search of this algorithm. Notice that each edge length is measured by its necessary time in this experiment. Since it costs less time for the same distance by more speedy way, vertices along the speedy way tend to be searched earlier in this case. This tendency also appears in the forward search for the case (k_s, k_t) equals to (0, 1), where the corresponding algorithm utilizes $-h_t(v)$ as the estimate in the forward search and $h_t(v)$ as that in the backward search for a vertex v.

The demerit of this tendency is observed in the northeast part of Figure 3(b). The backward searched edges spreads along the expressway in the direction opposite to s. The forward searched area is smaller than that of Figure 3(a) with the bidirectional Dijkstra method. This means A* algorithm is effective under disadvantageous condition that the shortest path gets out of the straight line between two terminals.

The property of that vertices along the speedy way are searched earlier produces good results for the case (k_s, k_t) equals to (0, 1), where the forward search has this property. The forward searched area in the left side of Figure 3(c) occupies smaller region than that of Figure 3(a) with the bidirectional Dijkstra method. This is due to the fact that expressways run radially from the center of Tokyo in the Tokyo metropolitan area. Hence this phenomenon is noted for most cases that Tokyo locates between two terminals. In consequence of this phenomenon, this algorithm reduces the number of searched edges compared with the previous algorithm.

For the case (k_s, k_t) equals to (1/2, 1/2), the algorithm is the bidirectional A* algorithm based on Theorem 2. This algorithm is, as it were, the average of previous two algorithms. Although the directionality as the A* algorithm is weaker than the corresponding searches of the two algorithms, the effect of different estimator is also weaker. With regard to the number of searched edges, this algorithm is in a middle of the two. The algorithm with (k_s, k_t) equal to (0, 1) is better in this case because of the positive effect of different estimator. The bidirectional A* algorithm based on Theorem 2 is regarded as a stable algorithm less influenced by such effect.

5 Conclusion

In this paper, algorithms to solve the two-terminal shortest path problem such as the Dijkstra method, the A* algorithm, and the bidirectional Dijkstra method have been surveyed and the bidirectional A* algorithm based on a new approach translating the A* algorithm into the Dijkstra method has been proposed for the case estimators are dual feasible. This algorithm is fit for finding not only the shortest route but also better routes. The efficiency and the property of these algorithms has been discussed through experiments using actual road data.

In the experiments, the necessary time to pass an edge is utilized as the length of the edge rather than the distance along the edge. Euclid distance between two points divided by the maximum speed is used for the estimate in the necessary time between them. The result of experiments indicates A* algorithm has an effect on such heterogeneous graph and its directionality toward the destination can be improved with stronger estimator obtained by multiplying a constant to the original estimator.

The result of experiments also shows the bidirectional A* algorithm proposed in this paper is more effective than other algorithms. This algorithm is stable compared with the bidirectional A* algorithm utilizing only one estimator because it is less influenced by the effect that vertices along the speedy way are searched earlier.

The research concerning the algorithm for the road network constructed on hierarchical structure, and the problem to predict the traffic flow in the road network are to be made in the future.

References

[1] D.Champeaus, "Bidirectional Search Again," *J. ACM 30*, 1983, pp.22–32.

[2] P.E.Hart, N.J.Nillson, and B.Rafael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Sys. Sci. and Cyb. SSC-4*, 1968, pp.100–107.

[3] T.Hiraga, Y.Koseki, Y.Kajitani, and A.Takahashi, "An Improved Bidirectional Search Algorithm for the 2 Terminal Shortest Path," *The 6th Karuizawa Workshop on Circuits and Systems*, 1993, pp.249–254 (in Japanese).

[4] M.Luby, and P.Ragde, "A Bidirectional Shortest-Path Algorithm With Good Average-Case Behavior," *Proc. 12th International Colloquium on Automata, Languages and Programing, LNCS 194*, 1985, pp.394–403.

[5] I.Pohl, "Bi-Directional Search," *Machine Intelligence vol.6*, pp.127–140, 1971.

[6] Y.Shirai and J.Tsuji, *Artificial Intelligence*, Iwanami Course: Information Science vol.22, Iwanami, Tokyo, 1982 (in Japanese).