A Method for the Shortest Path Search by Extended Dijkstra Algorithm

Masato Noto
Kanagawa University
Yokohama, Japan 221-8686

Hiroaki Sato
Fujitsu Program Laboratories
Yokohama, Japan 222-0033

Abstract

The Dijkstra method is a well-known algorithm for finding the optimum path in shortest path search problems. With that method, however, the time required to find the optimum path becomes remarkably long when the search scope is broad, so the Djikstra method is not suitable for real-time problems. In this paper, we propose a method for obtaining, in a short time, a path that is as close as possible to the path obtained by the Dijkstra method (the optimum path). The new method extends the conventional Dijkstra method so as to obtain a solution to a problem given within a specified time, such as path search in a car navigation system. The effectiveness of that extended method is described through use of simulations.

1 Introduction

The path search problem involves finding the optimum path between the present location and the destination under given conditions. Currently, these problems arise in networks such as the highway system, railroads, and communication networks, and cover a wide range of applications. In particular, car navigation systems have exhibited explosive growth in popularity due to recent advances in science and technology. The actual route found in car navigation is not necessarily the optimum solution because of hardware restrictions and the calculation time required for the path search. A number of path search methods have been proposed for use in car navigation systems, but while all of those methods have their respective merits and demerits, no exceptionally good method has yet been established. In this paper, we propose a method of path search that is based on the Dijkstra method, which can obtain the optimum solution, but extends that method so as to obtain a path that is as close as possible to the optimum solution in a shorter time.

This paper organized as follows.

In section 2, we introduce the main algorithms that are employed for the shortest path search problem.

In section 3, we propose a new algorithm obtained by extending the Dijkstra method, which is capable of yielding the shortest path, and present the results of comparative experiments to discuss the effectiveness of the proposed method.

In section 4, we identify problems and issues to be addressed in future work to further improve the extended algorithm and summarize our work.

2 Shortest Path Search

2.1 Features of Various Path Search Methods

Here, we describe the respective features of the Dijkstra method, the A* algorithm, and genetic algorithms, which are the main types of algorithms that are currently being used or studied for use in path search problems.

- The Dijkstra method [1]

  This is an algorithm for finding the optimum path. Because this algorithm searches for the minimum-cost path among all paths in order, beginning from the starting point, the search region expands concentrically. This method thus has the disadvantages of poor search efficiency and a long search time when the distance to the destination is large.

0-7803-6583-6/00/$10.00 © 2000 IEEE

- The A* algorithm

  This algorithm is based on the approach of the Dijkstra method, but eliminates fruitless searches by considering the distance to the destination.

- Genetic algorithms [2, 3, 4]

  These algorithms imitate a number of processes that are seen in natural evolution. In the case of path search, a path is selected and regarded as a gene, and the solution path is obtained by performing calculations that emulate genetic crossing, spontaneous mutations, and so on. The search time is determined by the number of genes that are produced, so this method has the advantage of not being affected by the network size. A problem with this approach, however, is that the number of generations necessarily becomes large as the optimum solution is approached.

2.2 Dijkstra Algorithm

In this section, we describe the algorithm of the Dijkstra method. First, we consider road intersections as nodes, the roads that connect nodes as paths (routes), and the length and ease of transit of a path as the movement cost (cost), as shown in Fig. 1.

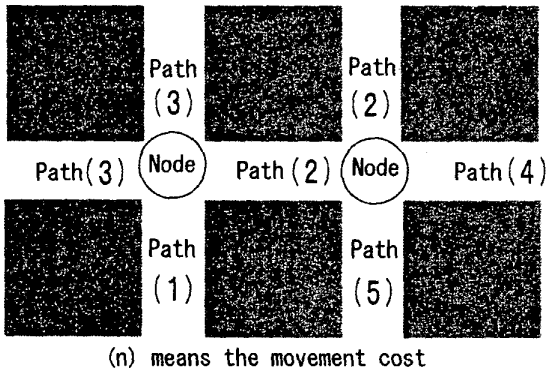

Figure 1: Route diagram

The algorithm is shown below.

Algorithm

- Step 0: Mark the starting point.
- Step 1: Calculate the movement cost for movement from the starting point to each node connected to the starting point and mark the node for which that value is the smallest.
- Step 2: Calculate the movement cost for movement between the starting point and each node connected to the marked node and mark the node for which that value is the smallest.
- Step 3: Repeat Step 2 until the destination is marked.

The value obtained here is the minimum cost of movement to the destination. Also, by storing the previous node in memory when marking a node, it is possible to obtain the shortest path to the destination.

3 Extended Dijkstra Algorithm

3.1 Basic Algorithm

In the conventional Dijkstra method, the search region generally expands concentrically, so many nodes are checked for each search and the search time is long. Because of that feature, this method has the disadvantage that the solution cannot be obtained in real time if the distance from the starting point to the destination is long. Therefore, we propose a new algorithm in which the Dijkstra method is applied from both directions, which is to say beginning from the starting point and also beginning from the destination, and the conventional algorithm is extended so as to make it possible to obtain a solution to the given problem within a specified time. In this way, the concentric expansion of the search region is restricted and the number of nodes to be searched can be reduced. (See Fig. 2.)

In the following, the new algorithm, which is an extension of the conventional Dijkstra method, is shown.

Algorithm

- Step 0: Mark the starting point and the destination.

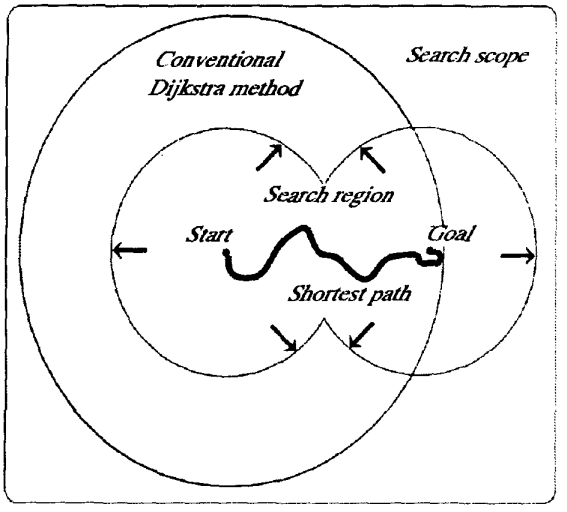

Figure 2: Comparison of the conventional Dijkstra method and the extended Dijkstra method

- Step 1: Calculate the movement cost for movement from the starting point to each node connected to the starting point and for movement from the destination to each node connected to the destination, and mark the nodes for which those values are the smallest.
- Step 2: Mark the nodes that have the smallest movement cost from the unmarked nodes that are connected to the marked nodes to either the starting point or the destination.
- Step 3: Repeat Step 2 until the search from the starting point and the search from the destination overlap and then end.

When conducting a path search with this extended algorithm, some time is required if the search scope becomes too wide, because this algorithm is based on the Dijkstra method. Therefore, if the search time or the search scope exceeds a certain value, the search is ended at that point, and another approach in which the remaining search region is approximated and a genetic algorithm is applied to that region is considered. Here, due to space restrictions, we report only the simulated comparison of the conventional Dijkstra method and the extended Dijkstra method.

3.2 Simulation

We simulated a comparison of the conventional Dijkstra method and the extended Dijkstra method proposed here on a Pentium 500MHz (256 Mbyte memory) computer, compiled with gcc. The search times, numbers of nodes checked, and the mean costs that were obtained in the simulation as the comparison results are presented in Table 1.

We created a 2000 × 2000 node grid pattern of paths to serve as the search region. The path movement costs were assigned with a uniform random distribution in five levels, 1 to 5, with the lower movement cost values representing easier movement. Also, because it is possible for the path to be monotonic if the starting point and the destination are on a straight line, a square is constructed at the center of the search region grid, and opposite corner points of the square serve as the starting point and the destination (Fig. 3). The distance between the starting point and the destination can be changed by changing the size of the square. Under these conditions, 10 simulation runs were executed with random changes in the movement cost values for each run. Although 25 paths for which the distance between the starting point and the destination ranged from 10 to 450 were actually simulated, only 17 of those are presented in Table 1.

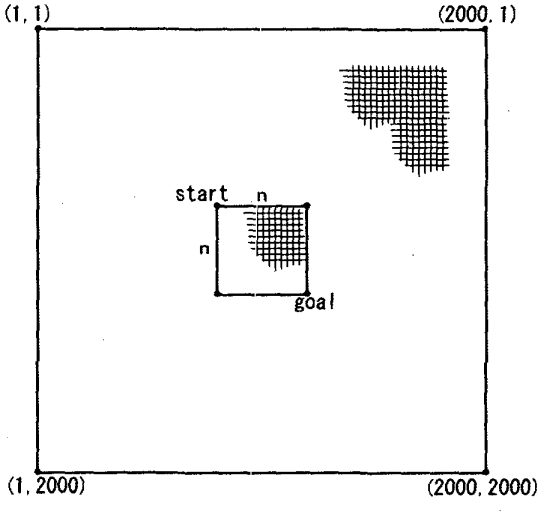

Figure 3: The search region for the simulation

Table 1: Simulation results

Distance	Conventional Dijkstra method			Extended Dijkstra method		
	Search time (s) (time1)	Number of node (count1)	Mean cost (cost1)	Search time (s) (time2)	Number of node (count2)	Mean cost (cost2)
10	0.03	645	38.1	0.01	305	38.1
20	0.25	2671	74.5	0.05	1252	74.8
30	0.93	6132	110.0	0.18	2929	110.3
40	2.34	11106	147.0	0.44	5289	147.3
50	4.43	16744	178.7	0.84	8079	178.9
60	7.65	23832	211.2	1.46	11518	211.4
70	12.10	32133	244.6	2.31	15528	244.9
80	18.63	42496	280.4	3.50	20348	280.5
90	26.89	54011	316.0	5.11	26022	316.1
100	37.36	66890	350.7	7.18	32455	350.8
150	127.06	148891	521.0	24.82	72558	521.6
200	311.55	267064	694.2	61.18	130238	694.6
250	616.41	415855	863.0	122.62	203524	863.2
300	1087.29	598716	1031.9	218.37	293502	1031.9
350	1771.71	816690	1203.1	358.87	401059	1203.3
400	2692.92	1062340	1370.7	552.75	523268	1370.9
450	3926.72	1344920	1540.2	811.82	663627	1540.4

The simulation results of Table 1 are shown in graph form in Fig. 4.

3.3 Discussion

The data in Table 1 is presented in a form that facilitates comparison of the conventional Dijkstra method and the extended Dijkstra method in Table 2.

First, we consider the search time (the "Time" in the table). With the extended Dijkstra method, the number of nodes checked (the "Count" in the table) is reduced by about one-half compared to the conventional Dijkstra method. By reducing the number of nodes checked, the search time was shortened to about 1/5 the original time. Next, we consider the movement cost (the "Cost" in the table). The conventional Dijkstra method necessarily gives both the optimum solution and the minimum movement cost. With the extended Dijkstra method, in all of the simulations, a difference in movement cost between the conventional Dijkstra method and the extended Dijkstra method was seen in 44 of the 250 runs with a probability of about 18%. However, as can be seen by comparing the "Mean cost" columns of Table 1, even when the cost difference appears, value of the difference in movement cost is only about 1 or 2, so we know that a near-optimum solution is obtained.

Table 2: Simulation results

Distance	Conventional Dijkstra method / Extended Dijkstra method		
	Time (%)	Count (%)	Cost (%)
10	12.82	47.27	100.00
20	17.58	46.88	99.60
30	18.81	47.77	99.73
40	18.69	47.63	99.80
50	19.00	48.25	99.89
60	19.02	48.33	99.91
70	19.06	48.32	99.86
80	18.79	47.88	99.96
90	19.00	48.18	99.97
100	19.21	48.52	99.97
150	19.54	48.73	99.88
200	19.64	48.77	99.94
250	19.89	48.94	99.98
300	20.08	49.02	100.00
350	20.26	49.11	99.98
400	20.53	49.26	99.99
450	20.67	49.34	99.99

4 Conclusion

We have proposed a shortest path search method that is an extension of the conventional method (the Dijkstra method) and confirmed its effectiveness by means of simulation. The results show that the search time can be greatly reduced without such a large affect on

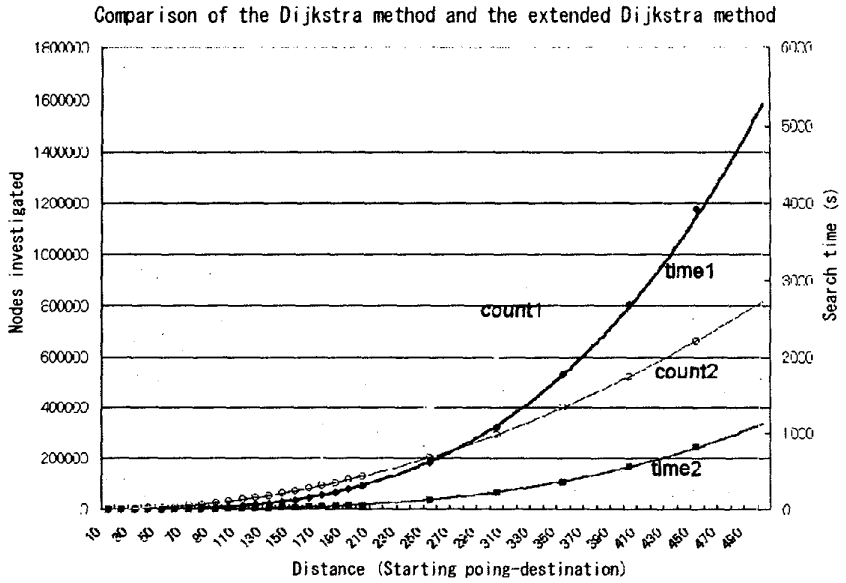

Figure 4: The simulation results in graph form

the movement cost, so that a near-optimum solution can be obtained rapidly. The new method can thus be applied to car navigation systems and other such path search problems that require a real-time solution. Nevertheless, although the new method allows for a faster solution than does the conventional Dijkstra method, this method is based on the Dijkstra method, and so problem of long search time remains for problems that have a broad search scope. As can be seen from Table 1, a 200 × 200 search takes about one minute, so practical use is probably limited to problems of this scope. As an issues to be addressed in future work, because genetic algorithms are an effective means of obtaining solutions rapidly, we can expect to achieve an even more effective method by improving and extending the method reported here by simulating use together with a genetic algorithm and comparison with conventional genetic algorithms. It is also necessary to conduct simulations in which the results of this work are applied to actual maps.

Acknowledgments

This work is partially supported by the Grant-in-Aid for Encouragement of Young Scientists: No. 12780246 for the first author.

References

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Elsevier North–Holland, 1976.

[2] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, 1989.

[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Ed., Springer–Verlag, 1996.