A Study of Point-to-Point Routing Problem in Mazes

Der-Cherng Liaw, Chien-Chih Kuo, Hung-Tse Lee

***Abstract*— In this paper, a three-step design is proposed to solve the point-to-point problem in a given maze. First, a well-known "depth-first" algorithm is adopted to find all the possible connection among grids in the maze with a graph structure. The obtained graph will then be simplified by deleting both of non-intersection and non-end grids. Such a step will largely decrease the computation load for constructing the path between the starting grid and the desired destination in the given maze. Finally, based on simplified graph the Dijkstra's algorithm is employed to find the shortest path between the starting grid and the destination. Numerical results of three typical examples are obtained to demonstrate the success of the proposed design.**

***Index Terms*— Maze solving, route searching, shortest path.**

I. INTRODUCTION

In the recent years, the study of automatic robots has attracted lots of attention. Those designs usually are equipped with many expensive sensors to fetch the necessary information from unfamiliar space (e.g., [1]-[3]). Among the existing researches, self-driving vehicle is one of giant mobile robots and its development might lead the trend of future technology [4]. In addition, many companies have developed their own robot for several years, which may bring considerable benefits on business trade. One of the main tasks of automatic robots is to build the overall road map of the interesting environment. It means that the surrounding environment to the robot is like a maze.

The topic of maze-like problem has been studied for more than forty years (e.g., [5]-[7]). In fact, in the recent years there are many micro-mouse robot competitions held around the world every year. Some of examples can also be found in the video link given in [8]-[10]. Competitors are requested to work for the improvement of the efficiency for path planning (e.g., [11]-[15]). Among the existing literatures, there are several famous algorithms (e.g., [16]-[19]) have been proposed without prior information of the maze such as Trémaux's algorithm, random mouse, wall follower, Pledge, breadth-first, and depth-first method. Among those algorithms, wall follower methods and random mouse scheme are the easiest ones for implementation. However, their efficiencies are not so good. In addition, both Trémaux's and Pledge algorithm cannot meet the requirements of travelling all feasible paths in the maze. Thus, in the previous study we have modified the depth-first algorithm and applied it to the path searching task in an unknown place [24].

Another task of automatic robots might be the design of traveling around the maze. One of the examples is that robot could get the shortest route from any starting point to a desired end point. There are several existing algorithms coming from graph theory have been used to find the shortest paths such as Dijkstra's algorithm, A* algorithm, Bellman-Ford algorithm and Floyd-Warshall algorithm [25]-[28]. Among those existing studies, best routes selection between Dijkstra and Floyd-Warshall algorithm were compared in [28] and a performance comparison study have been analyzed in [26]. In addition, a computational load analysis of Dijkstra, A*, and Floyd-Warshall algorithms has been studied in [29].

It is observed from those micro-mouse robot competitions (e.g., [8]-[10]) that those designers for the micro-mouse have tried very hard to find the path from the starting point to the set ending point and increase the speed of the robot to minimize the elapsed time. In general, they do not concern whether the chosen path is the shortest one since the whole graph of the maze has not been built. Here, we extend our previous study [24] to propose a different approach by finding the connecting graph among all the points of the given maze. The shortest path will then be created from the graph.

The paper is organized as follows. First, both depth-first searching algorithm and Dijkstra's searching method will be recalled in Section II. It is followed by the discussions of the main design. Numerical simulations of three typical micro-mouse robot competitions will be given in Section IV to demonstrate the success of the proposed scheme. Finally, conclusions are given in Section V to highlight the contributions.

II. PRELIMINARIES

In this section, we will briefly recall depth-first algorithm (e.g., [20]-[21]) and Dijkstra's algorithm [30]. Those two algorithms will then be employed in Section III to construct a cascade-type searching scheme for the maze which is considered in micro-mouse contest. Details are given as follows.

A. Definition of Maze

Before possible path finding, the rule and coordinate of the maze solving problem should be first defined clearly. According to the IEEE standard "Micro-Mouse Competition Rules," the maze is composed of 18cm x 18cm unit squares arranged as 16 x 16 units as shown in Fig. 1 [9]. The wall of each unit of the maze is 5cm high and 1.2 cm thick. In addition, the starting point of the maze is located at one of the four corners and is bounded by the walls on three sides. Besides, the start line of the game is located between the first and second squares. It means that the timer of the game starts to work at the time for the mouse exiting the corner square. The destination of the game is defined as the place which it has

D.-C. Liaw, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.
C.-C. Kuo, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.,
H.-T. Lee, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C

four units of square and is at the center of the maze. According to the standard, the destination is assumed to have only one entry. The objective of the competition is to find the path from the starting point to the destination point with minimum time of completing the whole procedure.

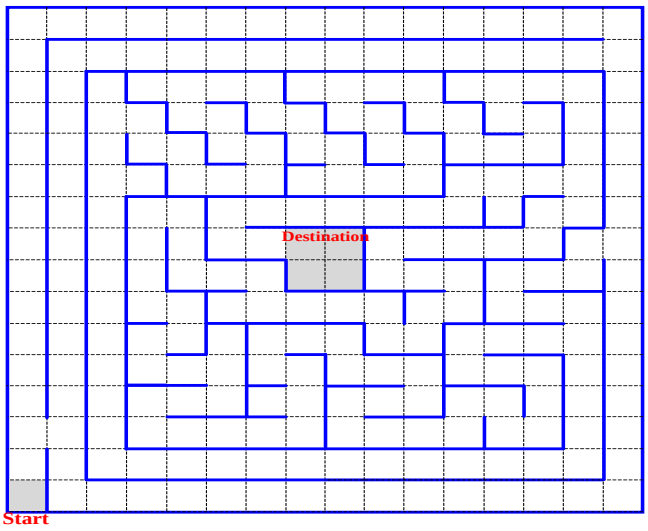


Fig. 1. Standard maze in micromouse contest [9]

To facilitate the analysis, we define the global maze coordinate as shown in Fig. 2. Here, *X*-axis is defined as positive in the right and *Y*-axis is defined as positive upward, respectively. If the robot starts at the node (x_0,y_0), then the next position can be calculated by the rotation angle of the wheel. In addition, there will be many sensors, like ultrasonic emitter, radar array or camera, to sense the environmental obstacle which is wall in the maze.

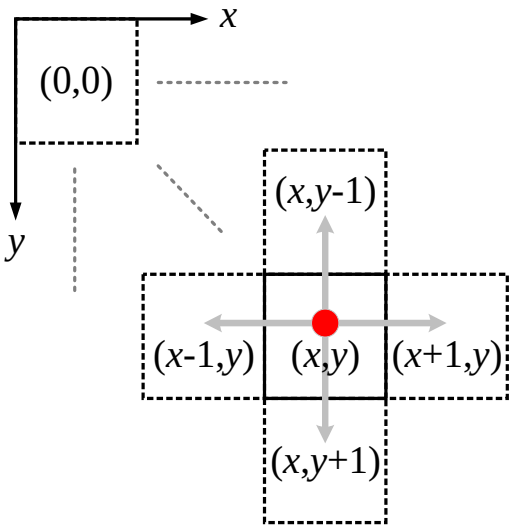
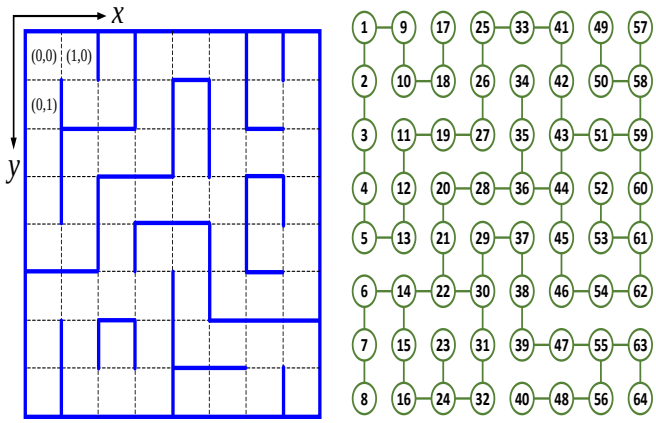


Fig. 2. Definition of coordinate in a maze

In general, the topology of the maze is usually converted into a graph. For instance, an example of maze is shown in Fig. 3, which has size of 8x8. The original topology of maze is depicted in Fig. 3(a), while the corresponding graph is defined in Fig. 3(b). Here, each vertex and edge of Fig 3(b), respectively, represents the corresponding square and the traversable path between two neighboring squares in Fig. 3(a).



(a) (b)

Fig. 3. A 8x8 maze: (a) The original maze, (b) The corresponding graph representation

B. Depth-First Searching Algorithm

Next, we briefly recall an algorithm from [20] for finding all possible connections in a given maze. In the sequel, we use "grid" to stand for the "square" for the basic element of maze defined in IEEE standard.

The main purpose of the depth-first algorithm is to help robot to generate the unknown maze automatically. Details of the corresponding steps are listed as follows:

Step 1. Place the robot on the starting point *v* of the maze and record the environment information collected by sensors.

Step 2. Calculate the current position of the robot in terms of the maze coordinate system defined in Section II.A and push the starting point *v* in the stack *N*.

Step 3. If the stack *N* is empty, the searching process is finished. Otherwise, pop a grid *w* from stack *N*.

Step 4. If *w* is a visited grid, go to previous **Step 3**. Otherwise, robot will visit grid *w* and record current position and update the map of maze.

Step 5. Use sensors to find traversable unvisited neighbors of *w* and push those unvisited neighboring grids to the stack *N* in sequence.

Step 6. If the current grid *w* is an end grid and the stack *N* is not empty, the robot will back to the previous intersection grid and pop a grid *w* from stack *N* and go to **Step 4**. Otherwise, go to **Step 3**.

C. Dijkstra's Algorithm

Now, we recall Dijkstra's algorithm which is used to solve the single-source node based shortest path problem with weighted directed graph. After many years of evolution, the algorithm is not the same as its original version. The current version is to choose one vertex as source node and find the shortest path from the source node to any others in the graph. Details of the corresponding steps are recalled from [30] as listed below:

Step 1. Mark the nodes which have not been visited and Create an unvisited set *Q* by all unvisited nodes.

Step 2. Assign an initial distance value to each node. The

initial distance value for the initial node is set to 0 while all other nodes have the value of infinity. Set the initial node as current node.

Step 3. Find all unvisited neighbors of current node and calculate their estimated distances between the current node and the neighbor. Then compare the newly calculated distance with the previous assigned value and update the distance value as the smaller one.

Step 4. After all of the unvisited neighbors of the current node are calculated, the current node is marked as visited node and is deleted from the unvisited set *Q*.

Step 5. If the destination has been visited or the smallest distance value among the nodes in the unvisited set is infinity, then stop and the algorithm is finished.

Step 6. Otherwise, select the unvisited node that is marked with the smallest estimated distance and set it as the new "current node." Then go to **Step 3**.

III. A THREE STEP DESIGN

In the design of the robot for micro-mouse contest, people might consider to find the whole structure of the maze and convert into a graph as depicted in Fig. 3. Then construct a searching scheme for obtaining the shortest path from the starting point to the desired destination. It is known that the time complexity of calculation for shortest path will increase as the size of maze becomes larger. To relax such a burden, in this study, we propose to insert a graph simplification scheme to reduce the size of the grid before executing shortest path searching. It is achieved by deleting the unnecessary vertex such as non-intersection and non-end vertex. An example is given in Fig. 4. The original graph is given on the right-top of Fig. 4. First step is to delete those non-intersection and non-end vertex as depicted on left-top of Fig. 4. The second step is to reform the edge with the length as given in right-bottom of Fig. 4. In the final step of graph simplification, we can re-name all remaining vertices as shown on the left-bottom of Fig. 4.

Based on the discussions above and the recall in Section II, we can now propose a three step design for solving the point-to-point problem in a maze. It is consisted of depth-first algorithm, graph simplification and Dijkstra's algorithm. The flow chart of the proposed scheme is described in Fig 5.

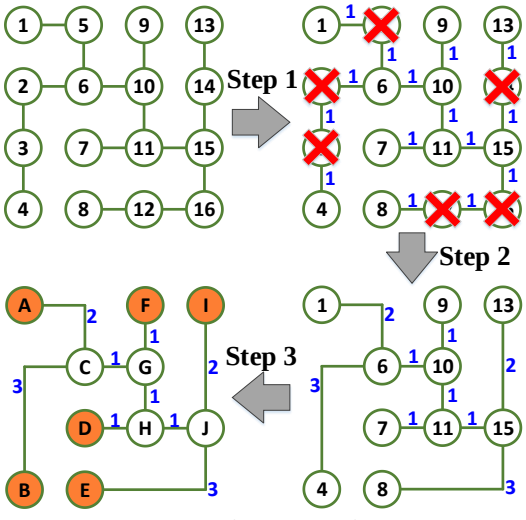


Fig. 4. Step of graph simplification

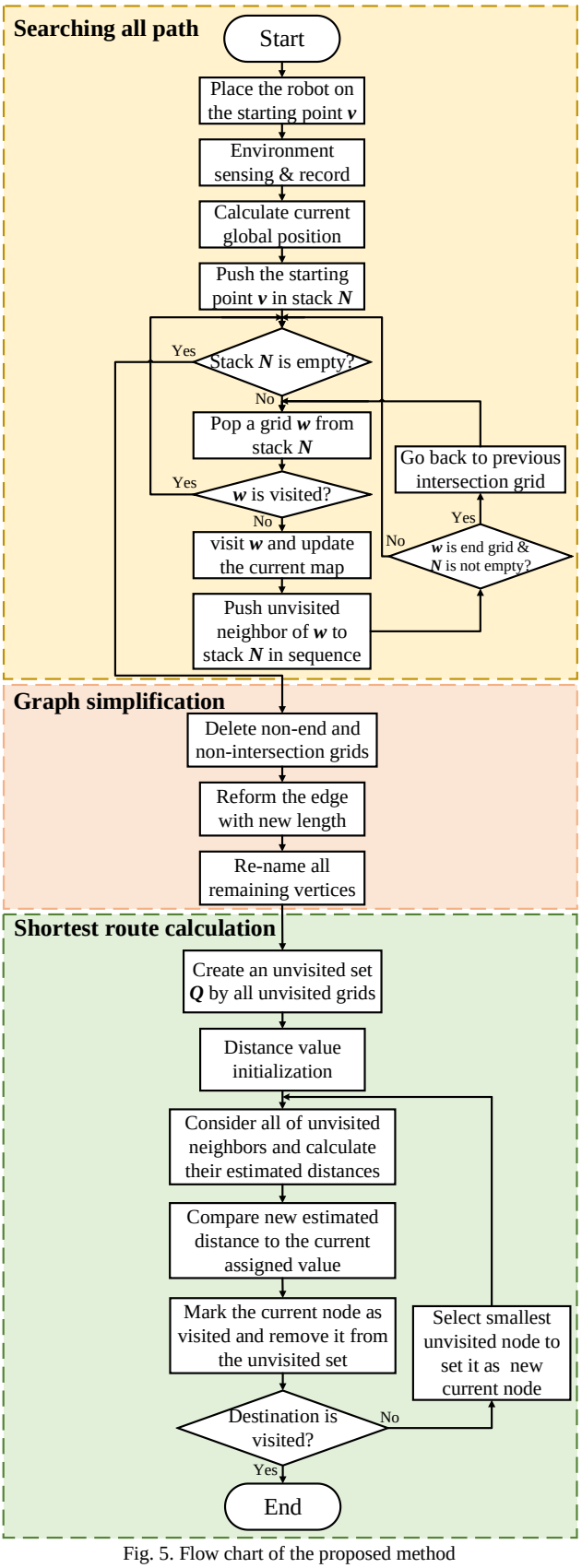


Fig. 5. Flow chart of the proposed method

IV. NUMERICAL SIMULATIONS

In the following, two types of numerical simulations will be given to verify the performance of the proposed design presented in Section III. The first one is to verify the functions of the proposed three-step design and the second one is to present the results for three typical examples used in micro-mouse contest. Details are given as follows. For

simplicity and without loss of generality, in the following study we only consider the maze with size of 8x8.

A. Verification of Proposed Scheme

First, we consider the verification for the functions of the proposed scheme. An example is given in Fig. 6. As shown in Fig. 6, all the connection among vertices can be found after eight iterations of executing depth-first algorithm. The corresponding graph can then be constructed as depicted on the left part of Fig. 7.

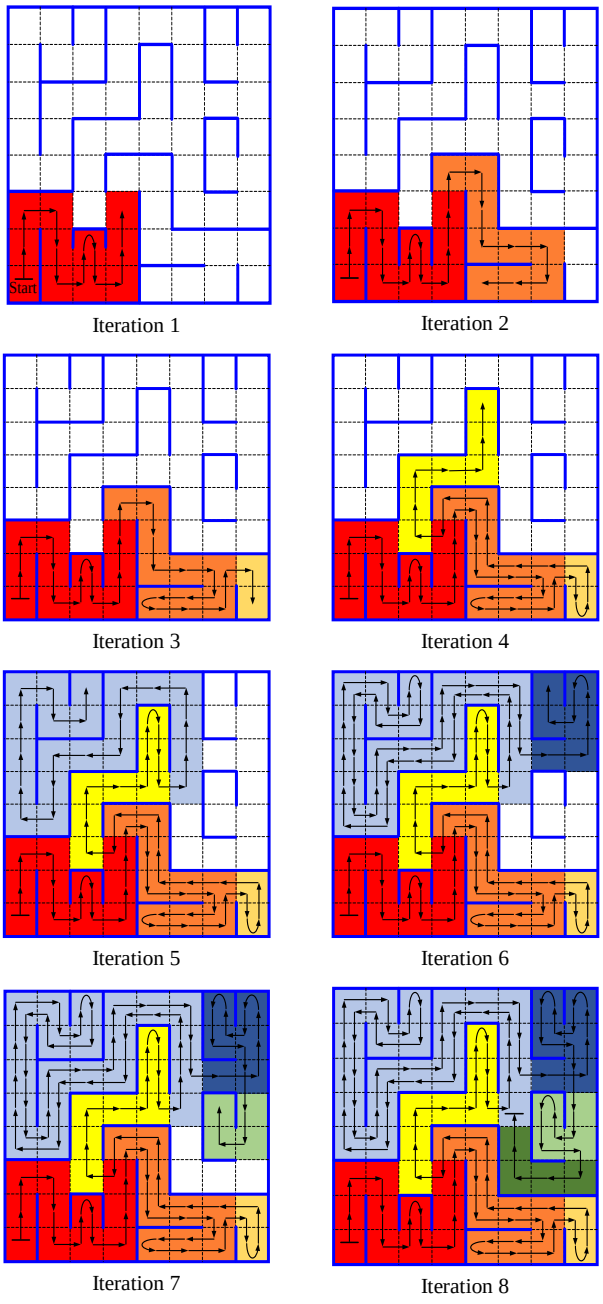


Fig. 6. Procedure of path searching

Next, a simplified graph can be obtained by deleting non-intersection and non-end vertex and re-naming the remaining vertices. The result is shown on the right part of Fig. 7.

Finally, based on simplified graph we can now construct the shortest path for point-to-point routing problem by using Dijkstra's algorithm. The results are shown in TABLE I.

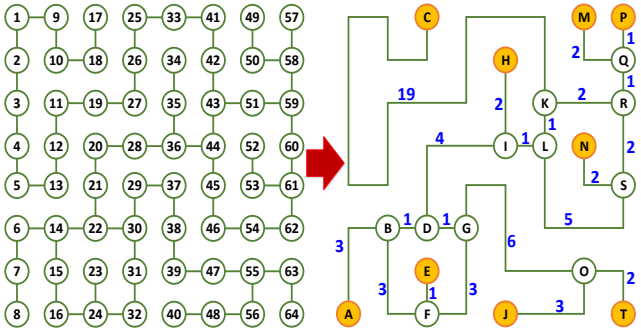


Fig. 7. Graph simplification

TABLE I. THE SHORTEST DISTANCE WEIGHT VALUE

	A	C	E	H	J	M	N	P	T
A	0	29	7	10	14	15	16	14	13
C	29	0	30	23	35	24	25	23	34
E	7	30	0	11	13	16	17	15	12
H	10	23	11	0	16	9	10	8	15
J	14	35	13	16	0	21	22	20	5
M	15	24	16	9	21	0	7	3	20
N	16	25	17	10	22	7	0	6	21
P	14	23	15	8	20	3	6	0	19
T	13	34	12	15	5	20	21	19	0

B. Case Study of Typical Examples

Now, we consider the application of the proposed scheme to three typical examples used in micro-mouse contest. In order to test the performance of the proposed scheme, in the following we will use Visual C++ to construct mazes with inner loops for testing the feasibility of the proposed scheme. Details are given below.

Case 1. Starting point to open end point [31]

The first example is to consider the problem of the starting point to one open end point as shown in Fig. 8. Here, the starting point is assumed to be at (0,8) and the destination is at (2,0) of the *X*-*Y* coordinate. Follow the procedure presented in Section III, we can obtain the simplified graph with twenty vertices as shown in Fig. 8(b). The twenty vertices are also named as A to T with the distance between two vertices shown on the corresponding edge. Hence, the shortest path problem becomes the problem for finding the shortest problem between vertices A and C. By applying the Dijkstra's algorithm, we then have the shortest path depicted as green line in Fig. 8(a). According to the performance evaluation given in TABLE II, there are four direct routes from vertex A to vertex C. The suggested path shown in Fig. 8(a) is truly the one with shortest distance.

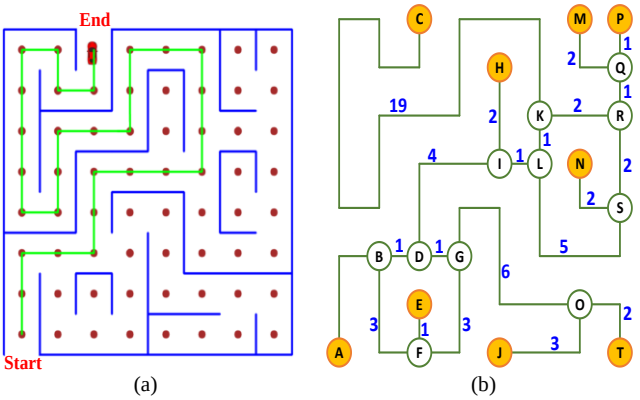
Case 2. Starting point to the middle of maze

The second example is to consider the problem of the starting point to the middle of maze as shown in Fig. 9. Here,

the starting point will be the same as the one for Case 1 while the destination is assumed to be at (4,3) of the *X*-*Y* coordinate. Similarly, we can obtain the simplified graph with twenty-nine vertices as shown in Fig. 9(b). The twenty-nine vertices are also named as A to γ with the distance between two vertices shown on the corresponding edge. Hence, the shortest path problem becomes the problem for finding the shortest problem between vertices A and O. The shortest path can then be obtained by using the Dijkstra's algorithm as depicted in green line of Fig. 10(a). As given in TABLE II, there are two direct routes from vertex A to vertex O. The suggested path shown in Fig. 9(a) is truly the one with shortest distance.

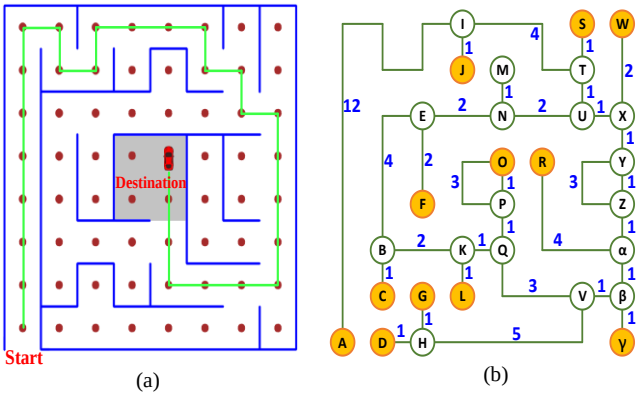
Case 3. Starting point to the closed end point

The third example is to consider the problem of the starting point to one closed end point as shown in Fig. 10 with the starting point being at (0,0) and the destination being at (7,0) of the *X*-*Y* coordinate. A simplified graph can then be obtained with twenty-five vertices as shown in Fig. 10(b) with the name of A to Y and the distance between two vertices shown on the corresponding edge. The shortest path problem becomes the problem for finding the shortest problem between vertices A and W. By using the Dijkstra's algorithm, the shortest path can then be obtained as depicted in green line of Fig. 10(a). According to TABLE II, there are two direct routes from vertex A to vertex W. The suggested path shown in Fig. 9(a) is found to be the one with shortest distance.



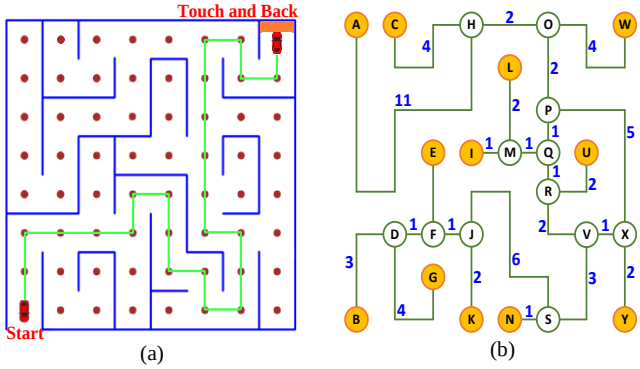
(a) (b)

Fig. 8. Case 1. Starting point to open end point [31]



(a) (b)

Fig. 9. Case 2. Starting point to the middle of maze (e.g., [8]-[10])



(a) (b)

Fig. 10. Case 3. Starting point to the closed end point [32]

TABLE II. PERFORMANCE EVALUATION

	Route searching	Length
Case 1	A→B→D→I→L→K→C	**29**
	A→B→F→G→D→I→L→K→C	35
	A→B→D→I→L→S→R→K→C	37
	A→B→F→G→D→I→L→S→R→K→C	43
Case 2	A→I→T→U→X→Y→Z→α→β→V→Q→P→O	**27**
	A→I→T→U→N→E→B→K→Q→P→O	29
Case 3	B→D→F→J→S→V→R→Q→P→O→W	**24**
	B→D→F→J→S→V→X→P→O→W	26

V. Conclusions

In this paper, we have proposed a three-step design for finding the shortest path from one point to the desired destination. Numerical results have also been obtained to demonstrate the success of the proposed scheme. In this study, we only concern the length of the edge between one vertex and the other. Possible direction change of robot running is neglected. To facilitate the design, a stack can be added in the program to record all possible direction changes for the practical implementation.

References

[1] Leonard, John J., and Hugh F. Durrant-Whyte, "Directed sonar sensing for mobile robot navigation," *Springer Science & Business Media*, 2012, vol. 175.

[2] Tsai, C. C., Huang, H. C., & Chan, C. K., "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Transactions on Industrial Electronics*, 2011, vol. 58, no.10, pp. 4813-4821.

[3] R. del-Hoyo-Alonso, N. Medrano-Marques and B. Martin-del-Brio, "A simple approach to robot navigation based on cooperative neural networks," *IEEE 2002 28th Annual Conference of the Industrial Electronics Society (IECON)*, 2002, pp. 2421-2426.

[4] Dolgov, D., Thrun, S., Montemerlo, M., & Diebel, J., "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, 2010, vol. 29, no. 5, pp. 485-501.

[5] L. Wyard-Scott and Q. -. M. Meng, "A potential maze solving algorithm for a micromouse robot," *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings*, 1995, pp. 614-618.

[6] S. Mishra and P. Bande, "Maze solving algorithms for micro mouse," *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, 2008, pp. 86-93.

[7] B. Gupta and S. Sehgal, "Survey on techniques used in autonomous maze solving robot," *2014 5th International Conference - Confluence*

The Next Generation Information Technology Summit (*Confluence*), 2004, pp. 323-328.
[8] https://www.youtube.com/watch?v=NqdZ9wbXt8k
[9] https://www.youtube.com/watch?v=cYCMcdjEetY
[10] https://www.youtube.com/watch?v=rZsHmy251UI
[11] Cai, J., Wan, X., Huo, M., & Wu, J., "An algorithm of micromouse maze solving," In *2010 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 1995-2000.
[12] Willardson, D. M., "Analysis of micromouse maze solving algorithm," *Learning from Data*, 2001.
[13] Sharma, M., & Robeonics, K., "Algorithms for Micro-mouse," In *2009 International Conference on Future Computer and Communication*, 2009, pp. 581-585.
[14] Chen, N., "A vision-guided autonomous vehicle: an alternative micromouse competition," *IEEE Transactions on Education*, 2009, pp. 253-258.
[15] Saman, A. B. S., & Abdramane, I., "Solving a reconfigurable maze using hybrid wall follower algorithm," 2013.
[16] K. Lutvica, J. Velagić, N. Kadić, N. Osmić, G. Džampo, and H. Muminović, "Remote path planning and motion control of mobile robot within indoor maze environment," *2014 IEEE International Symposium on Intelligent Control* (*ISIC*), 2014, pp. 1596-1601.
[17] R. Kumar, P. Jitoko, S. Kumar, K. Pillay, P. Prakash, A. Sagar, R. Singh, U. Mehta, "Maze solving robot with automated obstacle avoidance," *Procedia Computer Science*, 2017, vol. 105, pp. 57-61.
[18] M. O. A. Aqel, A. Issa, M. Khdair, M. ElHabbash, M. AbuBaker, and M. Massoud, "Intelligent maze solving robot based on image processing and graph theory algorithms," 2*017 International Conference on Promising Electronic Technologies* (*ICPET*), 2017, pp. 48-53.
[19] Y. Yu, G. Pan, Y. Gong, K. Xu, N. Zheng, W. Hua, X. Zheng, and Z. Wu, "Intelligence-augmented rat cyborgs in maze solving," *PLoS ONE*, 2016.
[20] TARJAN, Robert, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, 1972, vol. 1, no. 2, pp. 146-160.
[21] KORF, Richard E, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial intelligence*, 1985, vol. 27, no. 1, pp. 97-109.
[22] X. Liu, "A comparative study of A-star algorithms for search and rescue in perfect maze," *2011 International Conference on Electric Information and Control Engineering*, 2011.
[23] S. Mahmud, U. Sarker, M. Islam, and H. Sarwar, "A greedy approach in path selection for DFS based maze-map discovery algorithm for an autonomous robot," *2012 15th International Conference on Computer and Information Technology* (*ICCIT*), 2012, pp. 546-550.
[24] C.-M. Wu, D.-C. Liaw and H.-T. Lee, "A Method for Finding the Routes of Mazes," *2018 International Automatic Control Conference* (*CACS*), Taoyuan, 2018, pp. 1-4.
[25] Dijktra, E. W., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, 1959, vol. 1, pp. 269-271.
[26] B. Popa and D. Popescu, "Analysis of algorithms for shortest path problem in parallel," *2016 17th International Carpathian Control Conference* (*ICCC*), Tatranska Lomnica, 2016, pp. 613-617.
[27] J. C. Dela Cruz, G. V. Magwili, J. P. E. Mundo, G. P. B. Gregorio, M. L. L. Lamoca and J. A. Villaseñor, "Items-mapping and route optimization in a grocery store using Dijkstra's, Bellman-Ford and Floyd-Warshall Algorithms," *2016 IEEE Region 10 Conference* (*TENCON*), Singapore, 2016, pp. 243-246.
[28] Risald, A. E. Mirino and Suyoto, "Best routes selection using Dijkstra and Floyd-Warshall algorithm," *2017 11th International Conference on Information & Communication Technology and System* (*ICTS*), Surabaya, 2017, pp. 155-158.
[29] M. A. Djojo and K. Karyono, "Computational load analysis of Dijkstra, A*, and Floyd-Warshall algorithms in mesh network," *2013 International Conference on Robotics*, Biomimetics, Intelligent Computational Systems, Jogjakarta, 2013, pp. 104-108.
[30] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
[31] https://www.youtube.com/watch?v=xskRE5MVRn4
[32] https://www.youtube.com/watch?v=IngelKjmecg

Der-Cherng Liaw, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.

C.-C. Kuo, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.

H.-T. Lee, Institute of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.