A Review of Various Maze Solving Algorithms Based on Graph Theory

Navin Kumar[1] Sandeep Kaur[2]
[1]M. Tech Student [2]Assistant Professor
[1,2]Department of Computer Science Engineering
[1,2]Sri Sai College of Engineering & Technology, Manawala, India

Abstract— Solving a maze using computers is a complex though enticing task as one needs to come up with an algorithm that is effective in terms of time withal space for finding the shortest path. There is an ample number of graph algorithms to solve this problem, each one surpasses the other in performance aspects. This paper reviews different graph algorithms for maze solving along with their performance. The image of the maze is first processed by using GD library[1] functions in PHP[2], eventually, converting it into a graph data structure. A particular graph search algorithm is subsequently applied to find the shortest path between the start and the end node which is then mapped back to the image highlighting the solution. An "8X8" maze is used here for studying the performance of all graph based algorithms.

Key words: Maze Solving, Graph Search Algorithms, Artificial Intelligence, Image Processing

I. Introduction

The maze solving problem is defined as finding a path from starting point to the ending point in a way that the path so found must be the shortest one. Figure 1 exhibit the solution for the given maze in the red line. The left one is the input maze image while the right one is the solution image of the maze reckoned by the computer program. The crucial thing to note here is that there exist multiple paths in this maze however the algorithm will give the one with minimum path length i.e. number of edges, therefore, discarding any other solution which is of greater path length.

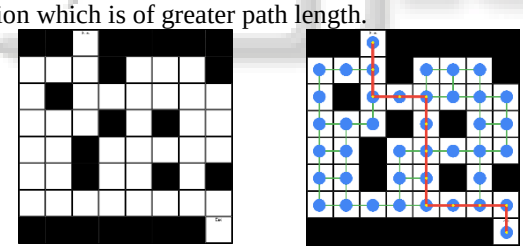

Fig. 1:

A. Conversion from Image to the graph data structure

The prime focus of this paper is to ponder on the performance of the graph algorithms hence the image of maze studied is an uncomplicated 8X8 maze where each building block is of 100x100 pixels. This will make image processing effortless as it is only needed to consider the pixel intensity of the centre of each block to indicate whether it corresponds to a path or not. This needs to be done for all 64 blocks by using the GD Library function imagecolorat ($image, $x, $y) where $x is the horizontal distance from the origin of the image, $y is the vertical distance and the $image is a reference variable for the image in the memory. The pixel intensities which are close to white will be treated as nodes of the graph while the black ones are considered as a block.

The next task is to find the adjacency list of nodes for each vertex which implies the edges in the graph as formed. There are only four nodes which can be adjacent to any vertex i.e. north, south, east and west, just as indicated in figure 2. The thin green lines are edges signifying the adjacent nodes.

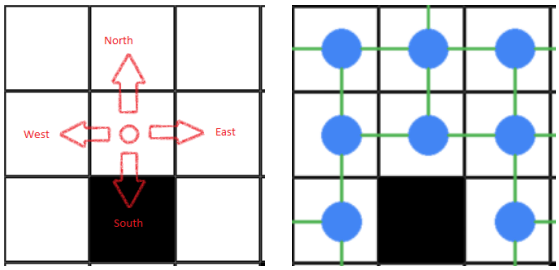

Fig. 2:

B. Discovering the solution using graph search Algorithms

The image processing work consequently reduces the problem to an undirected graph for which several graph search algorithms are available nevertheless for any algorithm ensued result will fall into following three groups i.e.

1) There exist no path

For this case, the algorithm will terminate after some time thereafter implying the nonexistence of any possible path. In such a situation, the whole search space will be explored, hence the time taken will supposedly be more.

2) There exist only one path

Since here only one solution to the problem exists so there will not be any notion of minimum length, consequently, after exploration of the entire maze, it will map the path so found it back to the image.

3) There exist multiple paths

This is a little complicated as there will be multiple solutions possible to the maze. This can be a point of concern when using an algorithm such as depth-first search because it will take one direction and go deep down until it hits a dead end or the goal node. If it gets the goal node, it will return it as a solution, undoubtedly, it will be a solution but is not guaranteed to be of the minimum length.

II. Literature Survey

This section covers the study of the related work which is presently done in the area of maze solving a problem or some other domains analogous to the maze problem.

M.O.A. Aqel et al. (2017) [3] has proposed the algorithm of breadth-first search to find the solution to the maze problem. It follows the approach which is quite similar to flood fill where it will expand nodes of the graph in such a fashion that it divides the node into different levels. The nodes which are at level one are queued first and are visited in a FIFO manner. This approach works great where the maze size is limited hence making the search space limited. The time taken will be proportional to the number of nodes in a graph thus with increasing maze size performance will take a downturn.

N.Hazim et al. (2016) [4] implemented the A* algorithm for a similar problem for finding the minimum path in the game playing strategies. This approach is far better than any of the blind searches like breadth or depth-first search since it encompasses the straight line distance as a heuristic for the expansion of the nodes. The performance of this technique will perhaps improve as at every expansion it gives an idea of how close is the goal node form the current frontier. The A* algorithm when closely studied turn out to be an enhancement of breadth-first search by using some heuristics, indeed, A* will take one direction and will explore it until it hits the end but it will explore the part of the search space which is more close to the goal node. This makes the A* search well than any of blind search algorithms when search space is very large. All in all, it gives a better result when the size of the maze is large and gives an acceptable result in small and medium size maze.

Y. Murata et al. (2014) [5] suggested the segregating of the maze into small blocks of n x n where the value of n is provided by the user based on the prediction that it will yield an improved set of results. The blocks so obtained are now explored as an individual maze thereafter the blocks are joined back to get the final solution of the maze. The results are surprisingly better than the A* algorithm provided the value of n passed by the user divides the maze adequately into small blocks. In an essence, this approach is the implementation of A* in an intelligent way by ignoring the search areas that will not yield anything contributing to the final solution.

B. Gupta et al. (2014) [6] presented and analysed different maze solving algorithms after converting the maze into a graph data structure using image processing techniques. It also compared the algorithms which don't use a graph data structure for finding the solution such as the wall follower approach which turns out to be expensive in time and space complexity when compared with graph based algorithms. Its results concluded that Lee's algorithm based on breadth-first search is culminating every other algorithm which it was compared with in that survey paper. The only drawback of such an algorithm is that the space complexity is very high as it incorporates the nodes which don't play any role in the solution of the maze.

S. Tjiharjadi et al. (2017) [7] proposed the method for finding the shortest path for a robot which uses the graph search algorithms for finding the solution to the problem. It concluded that robot can only find the shortest path only after the entire exploration of the maze, eventually, applying the A* or flood fill algorithm. The results produced from the study stated that the flood fill algorithm was good only for a small size maze as the search space size is enlarged the results tend to degrade. The A* algorithm performed well in case of a large search space. The time required by the robot was also combined in this study.

M. Karova et al. (2016) [8] proposed a method of converting the search space into the graph data structure by using image processing techniques thereafter applying a simple breadth-first search algorithm for finding the shortest path. Although, the main focus was on conversion of the search space as seen by a robot the approach followed was a blind search which might not give the best results. The further enhancement can be done by using informed search techniques like depth limited search and best first search. The breadth-first search performed well in a small search space.

M. Pond (2017) [9] has discussed the maze solving problem using different search algorithms. It first demonstrated the depth-first search limitation about taking one direction in a maze declaring the first path as a solution which will lead to the goal node. This limitation of depth-first search makes it unsuitable for maze solving problem where the path needed should be of minimum length. The Dijkstra's algorithm is used which produced a solution which is optimal in terms of time and space. The Dijkstra algorithm is based on a greedy approach which takes an edge of minimum weight while the expansion of the node. Since the graph here is an undirected as well as an unweighted graph, therefore, Dijkstra's will behave the the same as breadth-first search. It also demonstrated A* algorithm's ability to find the solution in minimum time as compared to any of the graph search algorithm.

S.J. Russell (2018) [10] has discussed various strategies which can be applied to the graph data structures for finding the minimum path in a maze. There are strategies like iterative deepening, depth limited search and some enhanced informed search strategies which can be a point of the interest for research when applied on a maze problem. A bidirectional search can give very good results but the crucial problem with such a search is trying to figure out the point where two searches will meet can be a challenging task to implement. It can give a run for their money to traditional graph search algorithms.

III. GRAPH ALGORITHMS

This section canvasses various graph search algorithms used for maze solving. Any algorithm which is implemented will fall in two categories i.e. Uninformed and Informed search strategies [10]. The two groups are discussed in underneath.

A. Uninformed Search Strategies

An uninformed search strategy is one which doesn't have any additional information about the problem other than the problem definition and the solution intended. The uninformed search is also known as the blind search as it only knows about its current state and all states which are reachable from that state. These mainly include the following described algorithms.

1) Breadth-First Search

The breadth-first search algorithm explores the search space in a way that at any node it looks for the adjacent nodes and queue them up, thenceforth it expands each node by de-queuing them from the queue and adding the nodes in the queue which are adjacent to one which is de-queued. This procedure is repeated until the queue becomes empty signifying that the search space is explored completely. The breadth-first search has a unique property that it divides the nodes and labels them a number which is equal to the distance of the particular node from the start node, moreover, that number label is the minimum distance from the start node [11]. This is exhibited in Figure 3.

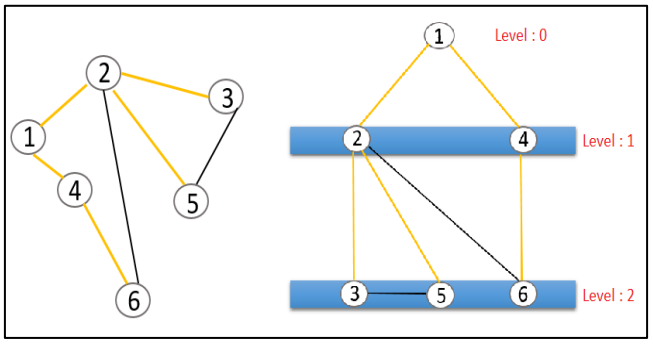

Fig. 3:

The given graph is on the left side of the figure and the right one is the level traversal of the same graph obtained by the breadth-first search. The yellow edges are the one which was taken in the traversal and the black are the one which was ignored. The crucial aspect of such a traversal is that the level number which is mentioned in the figure is the actual minimum distance from the starting node 1 if the graph is a connected one. The edges of the graph are denoted by m and nodes by n, then the time complexity of breadth-first search turns out to be O(m+n) i.e. for the worst case provided that the graph residing in the memory is in adjacency list representation. The above algorithm performs well in case of a small maze as the search space is limited but for a maze of large size, the performance will degrade as it would then need to explore a large number of nodes. The advantage of using it for maze solving is mainly as it is complete and will give a minimum path guaranteed.

2) Depth-First Search

The principle behind the depth-first search is to go deep in the direction where the first node is found at the frontier. The possible outcome of the above procedure will be either to hit the goal node or to hit a dead end. While former will be a solution to a problem which might not be of a minimum length the later will cause the algorithm to backtrack and find another frontier node to repeat the same process unless goal node is found or there isn't any node left to be explored. Such an algorithm might seem like a convenient for maze solving problem, however, the solution provided comes with a condition that some part of search space might not be inspected at all. This will make the obtained solution sub-optimal and will not be of any use. The solution can be optimal by a close adjustment i.e. to continue the search till all the nodes are explored, eventually taking the minimum one at the termination. The time complexity of the depth-first search algorithm is also O(m+n) where m and n are the usual notations. The depth-first search makes it suitable for a maze which is having only one path as a solution. The acclaimed topological sort is an application of the depth-first search where the nodes are arranged according to their chronological order [12].

3) Iterative deepening depth-first search

Iterative deepening search is based on a depth-first search with the idea of exploring the graph in an iterative manner by increasing the depth limit by one at each iteration. It combines the benefits of breadth-first and depth-first search, thus, this approach supersedes the traditional depth-first search where the maze is of large size.

B. Informed Search Strategies

The informed search strategies use some additional problem-specific information perhaps finding the solution in an efficient manner. Beneath described are some of the informed search strategies.

1) A Algorithm*

The A* algorithm unlike any uninformed search tries to explore in a direction which is more close to the goal node. This is done by using some heuristic which should be consistent as well as admissible in contemplation with the problem. The A* expands a node (n) for which the following function gives the minimum value.

$$f(n) = g(n) + h(n)$$

Here the function g(n) is the cost involved to travel from start to the current node and h(n) is the heuristic function which is problem specific, therefore will vary from problem to problem. In maze problem the heuristic used is the straight line distance from the goal node. With the additional heuristic information, A* surpasses every uninformed search thereby making it suitable for the maze problem.

IV. Table of Comparison

Algorithm	Performance	Complete	Optimal
Breadth-First Search	High	Yes	Yes
Depth –First Search	Low	No	No
Iterative Depth-first	Medium	Yes	Yes
A* Search	Very High	Yes	Yes

V. Conclusion

The graph search algorithms are far superior to the other naive search techniques like wall follower approach. The uninformed search algorithms perform better than the informed search algorithms in a large search space. In particular, A* algorithm is widely used for mazes solving and other analogous problems as it uses straight-line heuristic which cut down the time by a significant amount.

References

[1] The PHP Group. PHP: GD – Manual. February 3, 2019, http://php.net/manual/en/book.image.php.
[2] The PHP Group. PHP: What is PHP? – Manual. February 3, 2019, http://php.net/manual/en/intro-whatis.php.
[3] 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," 2017 International Conference on Promising Electronic Technologies (ICPET), Deir El-Balah, Palestine, 16-17 October 2017,pp. 49-53.
[4] N.Hazim, S.S.M. Al-Dabbagh, and M.A.S. Naser, "Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm," Journal of Computer and Communications, January 2016, pp.15-25.
[5] Y.Murata and Y.Mitani, "A Fast and Shorter Path Finding Method for Maze Images by Image Processing

Techniques and Graph Theory," Journal of Image and Graphics, Volume 2, No.1, June 2014, pp.89-93.
[6] 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), 25-26 September 2014
[7] S. Tjiharjadi, M. Chandra Wijaya, and E. Setiawan, "Optimization Maze Robot Using A* and Flood Fill Algorithm," International Journal of Mechanical Engineering and Robotics Research Vol. 6, No. 5, September 2017
[8] M. Karova, I. Penev, M. Todorova, H.Bobev, and N. Kalcheva, "Graph Construction Algorithm for finding the Shortest Path in a Maze," International Conference on Computer Systems and Technologies - CompSysTech'16, June 2016
[9] M. Pond, (2017, Feburary 24). "Maze Solving - Computerphile [Video file]". Retrieved from https://www.youtube.com/watch?v=rop0W4QDOUI
[10] S.J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Third Edition, Pearson India, pp. 64-108, 2018
[11] N. Garg, (2008, September 24). "Lecture - 25 Data Structures for Graphs [Video file]". Retrieved from https://www.youtube.com/watch?v=hk5rQs7TQ7E
[12] T.H. Coremen, C.E. Leiserson, R.L. Rivest and C. Stein, "Introduction to Algorithms", Third edition, Prentice Hall of India, pp. 587-748, 2009.

