International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem

Navin Kumar[1], Sandeep Kaur[2]

[1,2]*Department of Computer Science & Engineering, Sri Sai College of Engineering & Technology, Manawala, Amritsar, India*

Abstract - *The intriguing problem of solving a maze comes under the territory of algorithms and artificial intelligence. The maze solving using computers is quite of interest for many researchers, hence, there had been many previous attempts to come up with a solution which is optimum in terms of time and space. Some of the best performing algorithms suitable for the problem are breadth-first search, A* algorithm, best-first search and many others which ultimately are the enhancement of these basic algorithms. The images are converted into graph data structures after which an algorithm is applied eventually pointing the trace of the solution on the maze image. This paper is an attempt to do the same by implementing the bidirectional version of these well-known algorithms and study their performance with the former. The bidirectional approach is indeed capable of providing improved results at an expense of space. The vital part of the approach is to find the meeting point of the two bidirectional searches which will be guaranteed to meet if there exists any solution.*

Key Words: **Maze Solving, Bidirectional Search, Artificial Intelligence, Image Processing**

1. 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 of the shortest length. The process involves three main modules, first involving the acquisition of the actual image, conversion of the image to a graph data structure and the last but not least finding the shortest path to the maze using graph search algorithms[1] and mapping the found solution back on the image.

Fig. 1 exhibit the entire process, the left one is the input maze image "8X8" in the given figure. The middle image represents the conversion to the graph data structure. The right one is the actual solution which is mapped on the image in the form of a gradient trace using the GD library [2] in PHP [3]. The crucial thing to note here is that there exist multiple paths in this maze however the algorithm should give the one with minimum path length i.e. number of edges, therefore, discarding any other solution which is of greater path length.

The bidirectional searching technique used in this implementation is quite different from traditional algorithms like breadth-first search (BFS) or A* algorithm [4]. Unlike the one directional search, the bidirectional search will run two simultaneous instances of the chosen algorithm one from start to end and other from end to start. The search will stop once a meeting point is found for two instances, like the one shown in the fig.1 in green color, hence, announcing that a solution to the problem is found.

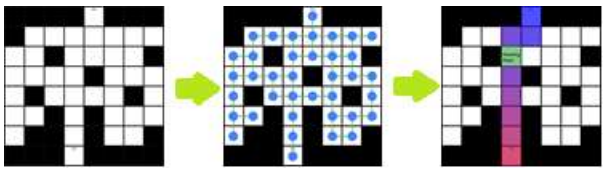


Fig.1 The Maze process solution using Bidirectional Search Technique.

In this paper the two types of mazes are considered for calibrating the performance of the proposed method of bidirectional search technique, these are "8X8" as shown in Fig.1 where the minimum possible path length can be 7 hops and "16X16" which is shown in Fig. 2 having a large search space with the minimum possible path length as 15 hops. In this paper both informed and uninformed searching techniques are compared with their counterpart bidirectional search technique and also all the result set are compiled to find the best algorithm suitable for maze solving problem.

2. RELATED WORK

M.O.A. Aqel et al. (2017) [5] has proposed the algorithm of BFS to find the solution to the maze problem. The algorithm expands the nodes level by level i.e. the nodes which are adjacent to the currently considered nodes are all expanded first before going to the nodes which are at next level, this is done by using a queue and hence is done in 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. The BFS is guaranteed to give the path of minimum length as it divides the nodes into different level groups as shown in Fig. 2.

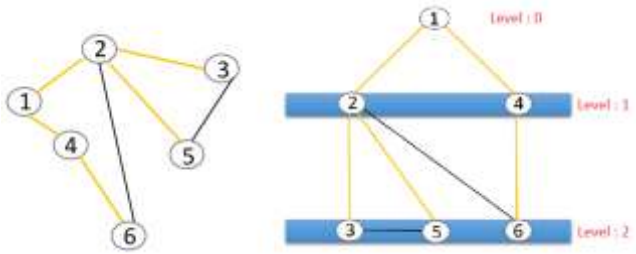


Fig. 2 Level Traversal of BFS for a given graph.

N.Hazim et al. (2016) [6] implemented the A* algorithm for a problem analogous to maze solving problem. The performance of this technique will perhaps improve as at every expansion it considers a node which is more close to the goal node form the current frontier. 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) [7] proposed a solution which involved 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. The individual block goes through the selected graph search algorithm eventually combining the result to get the shortest path. In essence, this approach is the implementation of A* in a way where the performance of the approach depends upon the values of n predicted by the user.

B. Gupta et al. (2014) [8] surveyed different maze solving algorithms for maze solving robot. It considered the algorithms which are related to flood fill as the robot cannot see the entire maze at once. Lee's algorithm based on BFS is culminating every other algorithm which was considered in the survey. The drawback of such an algorithm is its space complexity.

S. Tjiharjadi et al. (2017) [9] proposed the method for maze solving robot. 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 comparison doesn't give a complete picture for a maze of smaller size ("5X5") because in a small search space the overheads in A* algorithms are always highlighted in comparison with every uninformed search. In order to comprehend mazes of different sizes should be considered before concluding a result.

3. METHODOLOGY

This section gives a detailed explanation of the methods used to convert the image to a graph data structure, the working of bidirectional search technique and the entire flow of the method followed.

A. Image to graph Conversion

1) Detect the image size for "8X8" or "16X16" maze.

2) For each block get the intensity at the center, an intensity close to white indicates a node in the graph while the black one is ignored.

3) For each node so obtained, the adjacent neighbor is checked. If an adjacent block is white, an edge is taken between two nodes. Since the diagonal move is not enabled so for each node only three adjacent nodes can be there.

4) The nodes and edges obtained are then converted into a graph data structure suing adjacency list representation.

B. Bidirectional Search Algorithm

The bidirectional search works just like the normal version of a specific algorithm except for the fact that the termination of the algorithm happens for the following two conditions

1) All nodes of the graph are visited

This condition is true under the circumstances where no path exists from start to the end node. The two searches will terminate when there are no more nodes left to be visited. On an average, the two searches will expand half the number of nodes individually, if it's a bidirectional BFS, however, same cannot be said for the A* as the expansion is dependent upon heuristic values.

2) Meeting point is found

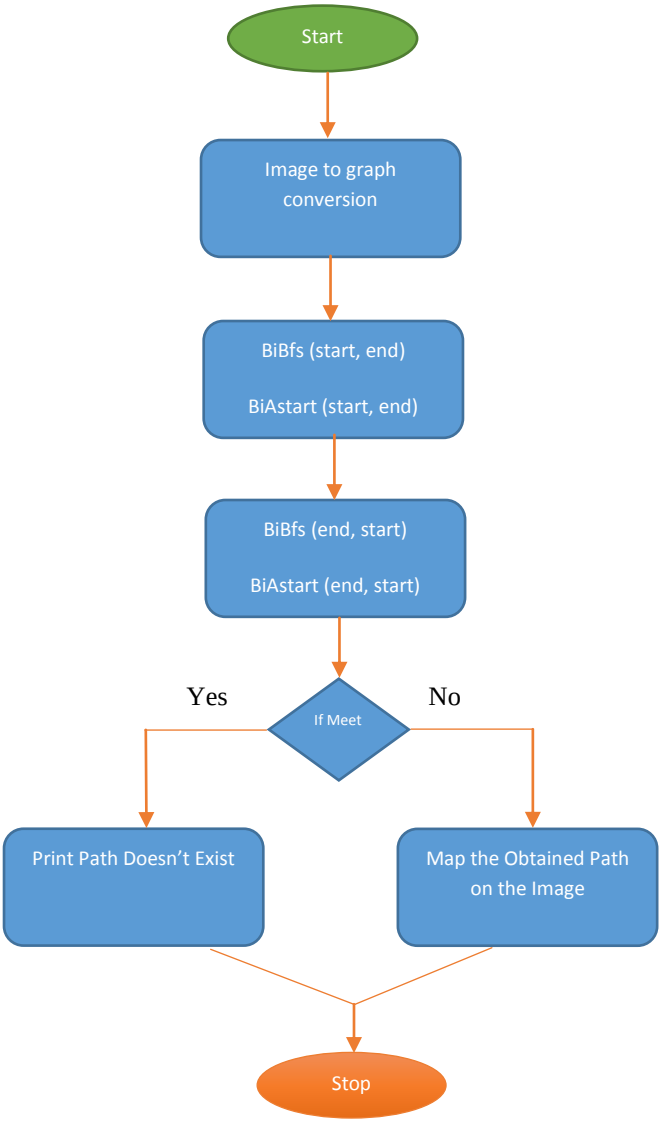
This condition indicates that the solution does exist, hence, two searches have a meeting point. The meeting point so found is guaranteed to be part of the minimum solution in case of A* as the expansion is done on the basis of heuristic values. The BFS, on the other hand, may or may not lead to a solution of shortest path length, however, this can be achieved by modifying the algorithm to repeat the procedure further until the path length of a pre specified length is found which can be different depending upon the maze size.

C. Heuristics Used For Informed Searches

The heuristic used for A* should be admissible and consistent in contemplation of the problem studied. Here the maze is considered is made from equal size blocks so as a result a simple straight line heuristic is used which can be calculated by the following formula. Where xg, yg are the coordinates of the goal node and x1, y1 are the coordinates of the required node for which heuristic value is to be calculated. The other heuristics can also be used like Manhattan heuristic which takes the minimum horizontal and vertical block moves from the goal node. The straight line heuristic is used as it is easy to calculate hence the overhead is less.

$$h1=\sqrt{(x_G - x_1)^2 + (y_G - y_1)^2}$$

D. Flow Chart of Procedure



4. RESULTS AND DISCUSSION

The results obtained after implementation were analyzed and compared with the previously implemented algorithms, this section is all about the same and its objective findings.

The figures shown underneath represents various solution given by various algorithms, no doubt algorithms took their own specific path for finding solutions like in Fig.3 A* went in a direction in which it finds the minimum heuristic while the BFS naively expanded the nodes level by level. Although paths for various algorithms are different as per their nature of expansion, however, the path length obtained in all of them are minimum i.e. 31 hops for the given maze.

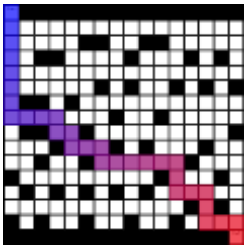


Fig.3 BFS Algorithm

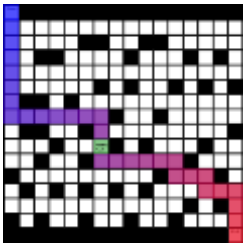


Fig.4 Bidirectional BFS Algorithm

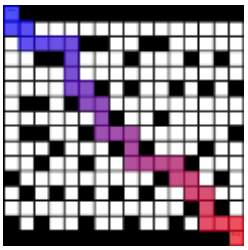


Fig .5 A* Algorithm

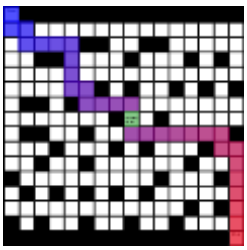


Fig .6 Bidirectional A* Algorithm

A. Comparison between BFS and Bidirectional BFS

1) For "8X8" Maze

The traditional BFS performed better than its bidirectional counterpart, the small search space is the main cause as there will be more overhead of finding the meeting point in a small search space has led to the increased time complexity.

Fig. 7 is the plot representing the time taken by BFS and bidirectional BFS, signifying how bidirectional technique is no better than BFS as there will be much more overhead for the other calculations involved for finding the meeting condition.

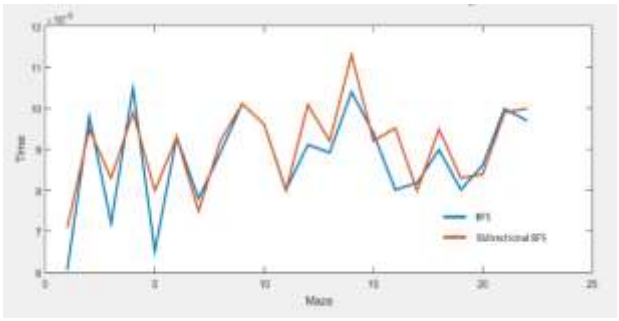


Fig 7. BFS and Bidirectional BFS for "8X8" Maze

For some mazes where the shortest path was of length greater than 10 hops, the bidirectional search did perform well over BFS as for both the instances there were more nodes to be expanded for each frontier.

2) For "16X16" Maze

The bidirectional BFS performed better in terms of time complexity when compared with its counterpart BFS. The story switched as we moved to "16X16" maze because of the expansion of the search space the benefits of bidirectional benefits can be seen in Fig 8.

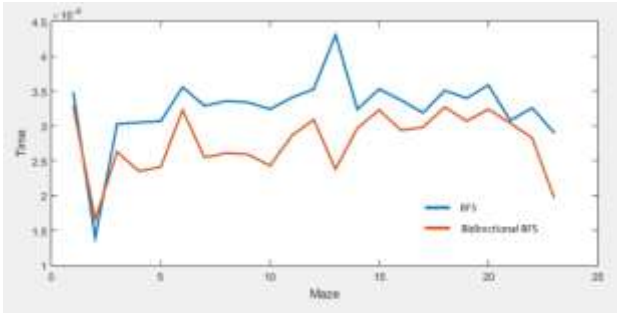


Fig 8. BFS and Bidirectional BFS for "16X16" Maze

B. Comparison between A and Bidirectional A* Algorithm*

1) For "8X8" Maze

On average, the bidirectional A* approach failed to perform well when compared with its counterpart. The overhead of checking for a meeting point in a small search space gave the advantage to a simple A* search which eventually surpassed it for most of the studied mazes.

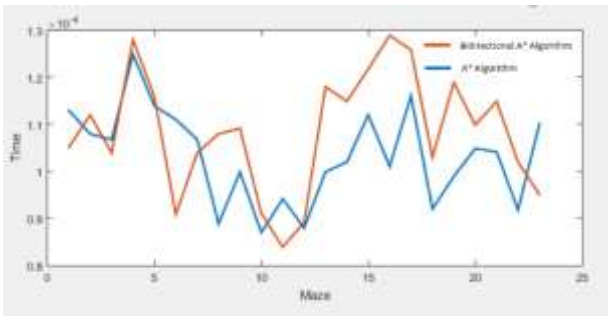


Fig 9. A* and Bidirectional A* Algorithm for "8X8" Maze

2) For "16X16" Maze

Fig 10 clearly marked the bidirectional A* best for the maze of large search space. The "16X16" maze's large search space made the bidirectional technique suitable as it can be clearly seen in the figure. The bidirectional search technique terminated as soon as the meeting point is found and since the algorithm used here is A*, the corresponding path is guaranteed to be the minimum one.

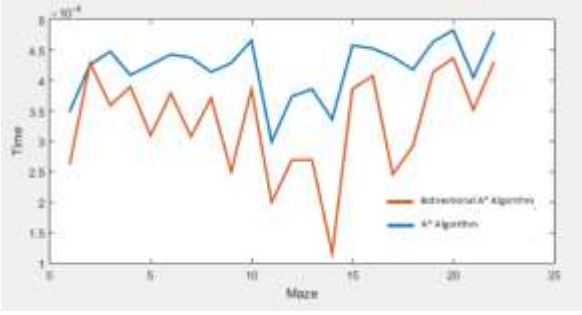


Fig 10. A* and Bidirectional A* Algorithm for "16X16" Maze

C. Finding the best Algorithm for different mazes

1) For "8X8" Maze

The Fig. 11 is showing the comparison of all the algorithm applied on "8X8" maze for a subset chosen randomly from a set of few hundreds of mazes. The BFS algorithm turns out to be a better solution for an average number of mazes.

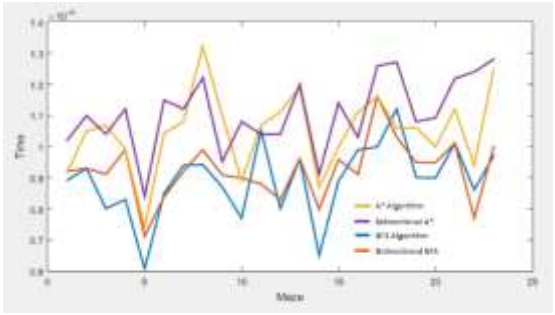


Fig 11. Comparison of all the Algorithms for "8X8" Maze

2) For "16X16" Maze

Fig. 12 shows the randomly selected results for all the algorithms practiced on hundreds of "16X16" mazes. The bidirectional BFS clearly outdid itself for almost every maze when compared with all the other algorithms.

The bidirectional A* algorithm did perform well as compared to the A* as expected from the previous observations but was not able to compete with other bidirectional algorithms i.e. BFS as it needs to evaluate every adjacent node at every expansion so as to select a minimum one.

The sequence of the algorithms according to the time taken to get the solution from worst to best can be written as A*, bidirectional A*, BFS algorithm and bidirectional BFS algorithm.

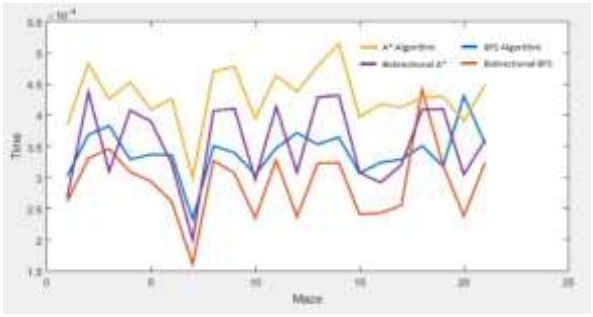


Fig 12. Comparison of all the Algorithms for "16X16" Maze

5. CONCLUSIONS

In this paper, the various algorithms used for maze solving problem were studied and compared with the bidirectional approached which was proposed as an enhancement to find the solution of minimum path length.

The bidirectional searching techniques for various algorithms performed better than any of their equivalent traditional techniques. The enhancement comes at the expense of using extra memory thereby making these algorithms out of place.

The bidirectional BFS in particular outperformed for various mazes which were considered for this research.

The informed searching algorithms like A * also fall behind its corresponding bidirectional version withal overhead considered of calculating the heuristic values and checking of the meeting condition.

The algorithm like depth-first search [10] and its enhanced versions are not considered in this paper as the solution required should be of minimum hops while the depth-first search will give a solution that might not be of shortest length.

In conclusion, the bidirectional BFS performed better for a maze of larger search space i.e. "16X16", however, for a maze of smaller size i.e. "8X8" the BFS outdid in terms of time complexity.

REFERENCES

[1] 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.

[2] J. Valade, T. Ballad and B. Ballad, "PHP & MySQL Web Development All-in-One Desk Reference For Dummies", Third edition, Wiley Publishing Inc., pp. 449-459, 2008.

[3] L. Ullman, "PHP and MySQL Web Development", CA: Peachpit Press, pp. 193-241,2016.

[4] S.J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Third Edition, Pearson India, pp. 64-108, 2018

[5] 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.

[6] 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.

[7] 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.

[8] 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

[9] 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

[10] E. Rich and K. Knight, "Artificial intelligence", New Delhi: Tata McGraw-Hill, pp. 307-326,2000.

BIOGRAPHIES

Navin Kumar received his B.Tech Degree in computer science and engineering from Guru Nanak Dev University, Amritsar. Worked for Accenture Mumbai, thereby, acquiring work experience of a year in the field of IT. He has also worked as an Assistant Professor at DAV College, Amritsar for 3 years, additionally qualified GATE 2019, GATE 2018 and UGC NET for Assistant Professor in 2018. He is currently pursuing M.Tech in Computer science and Engineering at Sri Sai College of Engineering & Technology, Manawala, Amritsar. His main research interests include Algorithms, Theoretical Computer Science and Artificial intelligence.

Sandeep Kaur received her B.Tech and M.Tech in Computer science and Engineering from Punjab Technical University, Kapurthala. She is currently working as an Assistant Professor at Sri Sai College of Engineering & Technology, Manawala, Amritsar. Her main research space is Image processing and Software Engineering.