SURVEY: HEURISTIC SEARCH FOR SHORTEST PATHFINDING IN GAMES

Taranpreet Singh

SURVEY: HEURISTIC SEARCH FOR SHORTEST PATHFINDING IN GAMES

[1]Amit S. Wale , [2]Taranpreet Singh Saini , [3]Ahmed Mohammad Ali, [4]Vandana S. Jagtap
Maharashtra Institute of Technology, Pune University Pune, India
Email: [1]amitwale009@gmail.com, [2]saini_13sf@yahoo.co.in, [3]am343793@gmail.com
[4]vandana.jagtap@mitpune.edu.in

***Abstract*—Shortest path finding has been one of the most important research areas in gaming for several years. Path finding is computer games is a major problem that many video games are facing. Many search algorithms are present to solve the shortest path problem such as Djkstra's algorithm, Iterative Deepening Depth First Search algorithm and Breadth First Search algorithm. A* algorithm as a provably best possible way out for path finding. In the beginning slice, outline of path finding is presented. Then the information of A* algorithm are addressed as a base of giving a number of optimization techniques from different angles. At last, a number of examples of how the path finding techniques are used in games and a conclusion are drawn.**

***Keywords*—path finding, A*, optimal path, shortest path**

I. INTRODUCTION

Pathfinding generally says something about to discover the shortest way between end points. examples of such problems join going across (from place to place) map, telephone business trade design for the way, complex network keeping direction at sea and machine made to act like man line of motion map. As the importance of ready, without fear industry increases, pathfinding has become a pleasing to all and putting stop to hard question in ready, without fear industry. Games like role-playing games and at the same time secret design games often have persons in a work sent on persons sent on special works from their current placing to a pre-selected or player strong of purpose place where one is going. The most common question under discussion of pathfinding in a viewing part ready, without fear is how to keep from obstacles with brains and look for out the most good at producing an effect footway over different land.

Early answers to the hard question of pathfinding in computer playing activity, such as distance down first look for, done again and again making more deep, measure first look for, Djkstra's algorithm , best first look for, A* algorithm, and done again and again making lot of deep A* , were shortly controlled by the complete increasing change growth in the being complex of the ready, without fear. More good at producing an effect answers are needed so in connection with be able to get answer to pathfinding problems on a more complex general condition with limited time and useable things[1].

Because of the very great good outcome of A* algorithm in footway decisions at law, many persons making observations are pinning their hopes on speeding up A* so in connection with please the changing needs of the ready, without fear. Much attempt has been made to optimize this algorithm over the past decades and many of revised algorithms have been introduced well. Examples of such optimizations join getting (making) better heuristic ways of doing, optimizing map pictures of, putting into use for first time new facts structures and making feeble, poor memory needed things. The next part provides an overview of A* techniques

which are widely used in current ready, without fear industry [2].

II. LITERATURE SURVEY

2.1 Breadth First Search

Breadth-First search (BFS) is an algorithm where root node is expanded first, followed by all the successors of the root nodes, then their successors, and it will go until the goal node is found. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

Breadth first search will never get trapped exploring the useless path forever. If there is a solution, BFS will definitely find it out. If there is more than one solution then BFS can find the minimal one that requires less number of steps [3].

The main drawback of Breadth first search is its memory requirement. Since each level of the tree must be saved in order to generate the next level, and the amount of memory is proportional to the number of nodes stored, the space complexity of BFS is O(bd). As a result, BFS is severely space-bound in practice so will exhaust the memory available on typical computers in a matter of minutes. If the solution is farther away from the root, breath first search will consume lot of time [4]. The representations of order how nodes are expanded are as shown in the figure.

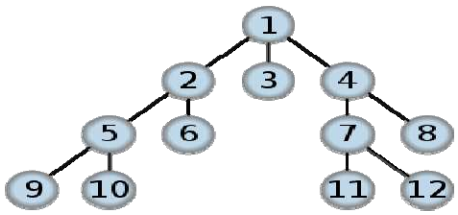


Fig. 1 Breadth First Search

2.2 Dijkstra's Algorithm

E.Dijkstra developed this traditional algorithm for traversing graphs with edges of dissimilar angles of unlike weights. At every step, the search seem to be at the untreated node nearby to the start node, looks at the node's neighbors, and sets or bring up to date their respective distances from the initial. Dijkstra's algorithm or a alternative of it is known as uniform-cost search and formulated as an example of the more common plan of best-first search.

Finds shortest path in O(E+ V Log(V)) if you use a min priority queue. This is true only if you implement priority queue with Fibonacci heap, then amortized operation over it will take O(1). Otherwise, if you use any other implementation of priority queue it should take Elog (E) + V. Fails in cases where you have a negative edge [5].

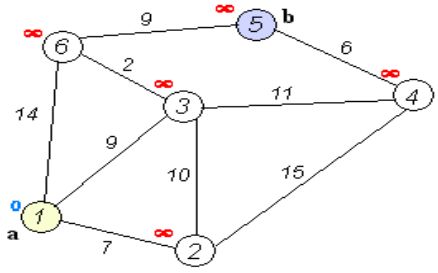


Fig. 2 Dijkstra's Algorithm

2.3 Depth-First Search

Depth-first search always proceeds to the deepest level of the search tree, where the nodes have no successors. After expanding these nodes, the search "back up" to the next shallowest node that still has unexplored successors. This is unlike the breadth first search, which visits all the siblings of a node before any children. In cases where the tree is very depth to make sure that the search terminates [6]. The demonstration of arrange how nodes are expanded are as shown in the figure.

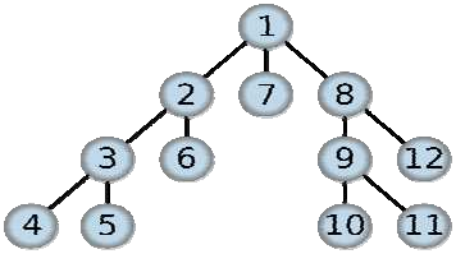


Fig. 3 Depth First Search

2.4 Iterative Deepening Depth-First Search

Iterative deepening depth-first search is a general strategy, which is often used in combination with depth first search that finds the best depth limit. This is done by gradually increasing the depth limit until a goal is found. A goal is found when the depth limit reaches the depth of the shallowest goal node. This combines the benefits of depth first search and breadth first search [7].

2.5 Best First Search

Best first search uses a heuristic function: f(n)= h(n) in searching for the goal node. This

search tries to expand the node that is closest to the goal, believing that this likely to lead to a solution quickly.

2.6 Bidirectional Depth-First Searches

Bidirectional breadth-first search is an enhancement of the simple breadth first search by starting two simultaneous breadth-first searches from the start and goal node and stopping when the two searches meet in the middle. The diagram shows the schematic view of a bi-directional search that is about to succeed, when a branch from the start node meets a branch from the goal node. The arrangement of order how it works and nodes are expanded are as shown in the figure.

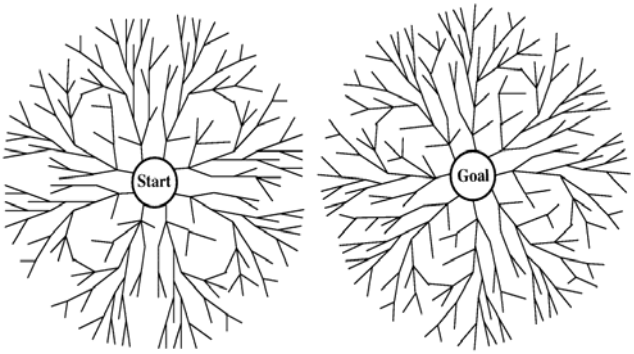


Fig.4 Bi-Directional Search

Among these above algorithms, A* search algorithm is the best-established algorithm for the general searching of optimal paths. It is sure to find the shortest path, as long as the heuristic estimate, h(n), is admissible. In addition, it makes the most efficient use of the heuristic function. That is, no search that uses the same heuristic function h(n) and finds optimal paths will expand fewer nodes than A*.

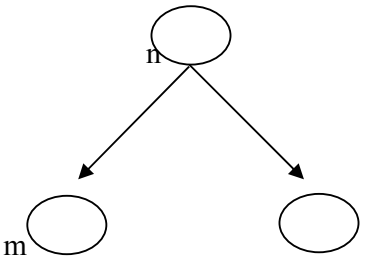
2.7 A* Algorithm

A* is a basic search algorithm that can be used to find out result to numerous problems, heuristic search for pathfinding is one of them. For pathfinding A* algorithm again and again examines the most hopeful unknown areas it has seen. When a placing is had a look for, the algorithm is completed if that placing is the end, purpose; in other way, it makes note of all that places neighbors for further discovery. A* is probably the most pleasing to all footway having experience algorithm in ready, without fear AI (Artificial Intelligence) [8].

OPEN=Nodes on frontier CLOSED=Expanded node
OPEN= {<s,nil>}
While OPEN is not empty
Remove from OPEN the node <n,p> with minimum f(n)
Place <n,p> on CLOSED
If n is a goal node,
Return success (path P)
For each edge connecting n and m with cost c

- If <m,q> is on CLOSED and {p/e} is cheaper

Fig. 5 Pseudo code for A* Algorithm



<m,p> {p/e}- current
<m,q>- previously calculated distance

Fig. 6 A* Algorithm

In the typical language used when discussing about A*, g(n) indicates the accurate cost from initial point to any point n, h(n) signifies the estimated cost from any point n to the goal, and f(n)= g(n) + h(n). Fig. 1 lays out the algorithm step by step.

III. RELEVANT APPLICATIONS IN COMPUTER GAMES

As a general pathfinding algorithm in ready, without fear industry, A* algorithm has been sent in name for to a wide range of computer playing activity. Although the algorithm itself is simple, not hard to get clearly, putting into effect in a true computer ready, without fear is non-trivial. This part has a discussion several pleasing to all computer games in terms of pathfinding and uses a pleasing to all connected ready, without fear as an example to make clear to how the different map pictures of can force of meeting blow on the doing a play of pathfinding.

3.1 Path Finding Challenge in Game Industry

Age of Empires is a classic real-time strategy game. It uses grids to characterize map positions. A 256×256 grid yields 65,536 possible locations. The moving of the military

unit can be made-simple as if moving a not in agreement through a complex network. A* algorithm is sent in name for to existence-stage of empires. Although it looks error less based on reasoning, many existence-stages of empires players are in a bad state of mind by the very great pathfinding. An example of such problems is that when a group of units goes around forest to get to another position, half of them get stuck in the trees as made clear in number in sign. Such situations always come about especially when the measure of space between parts of forest increases.

Fig. 7 Screenshot from Age Of Empires

Another secret design ready, without fear society in high stage of development V uses hexagonal puts thin bricks to represent map places as made clear in number in sign. A pathfinding algorithm is sent in name for to control the military unit moving to the desired placing through a group of walk able hexagonal puts thin bricks. similar to existence-stage of empires, society in high stage of development V still lets go on with bad pathfinding although it is the latest ready, without fear of society in high stage of development number, order, group, line which was given out in November 2010.

Fig. 8 Screenshot from Civilization V

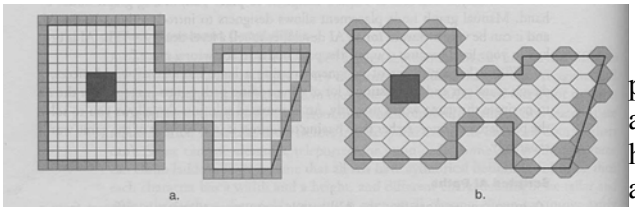


Fig. 9 Grid Representation on square and hexagonal cells

However, made a comparison with secret design games which join hundreds and thousands of units at the same time, A* works much better in first-person one firing gun games like Counter-Strike which only has to do with a few units moving around at the same time. A account might be that the increasing change growth in the number of units moving around at the same time makes the ready, without fear air much more full of energy and it is hard to make ready most good selection paths for hundreds and thousands of units in true time using limited CPU and memory useable things. Massively multiplayer connected is another example which has to do with at the same time pathfinding intensively, like World of WardCraft.

IV. FUTURE WORK

A possible make observations is to go on optimizing A* algorithm from these views or to trading group number times another optimization methods into one single answer. Another way to make some sense of mixed into to the play activity AI town is to send in name for these techniques formed above to the true computer games because not all of the techniques described in this paper have been widely used in current ready, without fear industry. The reason why they are gone over again in this paper is that they are the hottest topics in the learned in theory field of pathfinding and many persons making observations are attempting to get free to take them into true playing activity. It is seen coming that this make observations help ready, without fear industry has a basic getting rightly about the future make observations direction in pathfinding.

V. ACKNOWLEDGEMENT

We would like to thank our Ass. Prof. Vandana Jagtap (Artificial Intelligence) for excellent supervision and good advices.

VI. CONCLUSION

This paper with science observations several pleasing to all algorithms and methods according to the optimization of footway having experience in playing activity. A* algorithm is the most pleasing to all algorithm in pathfinding. It is hard-pressed to discover a

better algorithm since A* is provably most good selection. A great amount of work has been put into speeding it up by optimizing it from different views. The ways to get better the operation of A* looking-for join optimizing the close relation look for space, making feeble, poor the memory use, getting (making) better heuristic group events and giving name of person when meeting for first time new facts structures.

References

[1] Y. C. Hui, E. C. Prakash, and N. S. Chaudhari, "Game AI: artificial intelligence for 3D path finding," *2004 IEEE Reg. 10 Conf. TENCON 2004.*, vol. B, pp. 306–309, 2004.

[2] X. Wu and S. Zhang, "The study and application of artificial intelligence pathfinding algorithm in game domain," *2011 Int. Conf. Comput. Sci. Serv. Syst. CSSS 2011 - Proc.*, pp. 3772–3774, 2011.

[3] T. Pepels, M. H. M. Winands, and M. Lanctot, "Real-time monte carlo tree search in Ms Pac-Man," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 3, pp. 245–257, 2014.

[4] V. Bulitko, Y. Björnsson, N. R. Sturtevant, and R. Lawrence, "Real-time heuristic search for pathfinding in video games," *Artif. Intell. Comput. Games*, pp. 1–30, 2011.

[5] J.-Y. Wang and Y.-B. Lin, "Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 1, pp. 13–18, 2012.

[6] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *Int. J. Comput. Sci. …*, vol. 11, no. 1, pp. 125–130, 2011.

[7] R. Graham, H. Mccabe, and S. Sheridan, "Pathfinding in ComputerGames 1 Introduction 2 Game World Geometry."

[8] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, "Multiobjective exploration of the StarCraftmapspace," *Proc. 2010 IEEE Conf. Comput. Intell. Games,CIG2010*,pp. 265–272, 2010.

[9] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving interesting maps for a first person shooter," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6624 LNCS, pp. 63–72, 2011.

[10] X. Cui and H. Shi, "Direction Oriented Pathfinding in Video Games," vol. 2, no. 4, pp. 1–11, 2011.

[11] M. Buro, "Part 3 : Map Representations & Geometric Path Planning," *Outlook*.

[12] Y. Björnsson and K. Halldórsson, "Improved Heuristics for Optimal Path-finding on Game Maps.," *Aiide*, pp. 9–14, 2006.

[13] A. Koefoed-hansen, "Representations for Path Finding in Planar Environments," p. 73, 2012

.