Literature Review

Matin Horri
horri031@umn.edu
University of Minnesota
Minneapolis, Minnesota, USA

Evelyn Junaid
junai015@umn.edu
University of Minnesota
Minneapolis, Minnesota, USA

1 Introduction

Pathfinding algorithms continue to be a relevant topic in Artificial Intelligence today. With applications ranging from multi-player interactive video games to GPS, there is always a demand for improvement. The Snake Game is a single player game in which the goal is to control the movement of a snake, and collect food in the map. Although usually played by a human, the Snake Game can be implemented with a controller that utilizes a pathfinding algorithm to move the snake toward the food. Two popular uninformed search algorithms, Breadth-First Search (BFS) and Depth-First Search (DFS) have a long history in solving single agent pathfinding games. Additionally, informed search algorithm A Star Search (A*) introduces a heuristic evaluation function that can enhance path finding abilities across metrics such as time, space, and path length. In our project we will compare and contrast DFS, BFS, and A* in the Snake Game based on four parameters: time complexity, space complexity, optimality, and completeness. An algorithm will be considered optimal if the solution it returns is also the best solution. If an algorithm finds a solution in a finite amount of time when at least one solution exists, it will be complete. Our project aims to design an agent for different levels of the game, including easy, normal, and professional. A plethora of experiments to evaluate the performance of these algorithms in various environments have been conducted throughout A.I. history. This review will explore such experiments and their findings.

2 Approach

DFS explores a state space that begins at a root and explores as deep as possible before returning to the root to explore other nodes[3]. DFS has been known to perform suboptimally in large and complex state spaces. The time complexity of DFS is

$$O(b^d)$$

and the space complexity is

$$O(b * d)$$

where b represents the maximum branching factor and d represents the maximum depth[10]. The linear space complexity of DFS is a major advantage of the algorithm that makes it worth exploring.

3 BFS

BFS explores a search tree level by level, expanding all nodes in a level before moving on to the subsequent level. BFS is always guaranteed to find an optimal solution, has a space and time complexity of

$$O(b^d)$$

, where b is the maximum branching factor and d is the maximum depth[10]. Since BFS is guaranteed to find an optimal solution, it will be interesting to see how it compares to A* in the Snake Game.

4 A*

A* has been found to be an effective pathfinding algorithm in video games. The A* algorithms can effectively guide characters through game worlds and handle dynamic environments[7]. Considering that the Snake Game has a dynamic environment and consists of the snake navigating the map, A* may be a good choice for the game. When the game world is small, either BFS or DFS are sufficient for pathfinding as they do not use large amounts of time or memory in small environments[7]. The same is not true as the environment increases in size and complexity. The slowest component of A* is searching for which move the algorithm will take. This is done by examining each possible move and executing whichever one returns the lowest value of the heuristic function. Using a binary heap instead of a queue is an effective way of speeding up the search process[7]. A* tends to be the preferred pathfinding algorithm because it outperforms uninformed search algorithms in efficiency and adaptability due to its heuristic evaluation[5].

5 Comparing Search Algorithms in the Snake Game

In the article written by Appaji, BFS, DFS, and A* were used as search algorithms in the Snake Game. In the first experiment each algorithm was run for 120 seconds. The results showed that when the controller was using A* the snake ate the highest number of fruits (92), followed by BFS (73), and DFS (22)[3]. For the second experiment, the fruits were

placed at fixed coordinates and the algorithms were all run for a set amount of time. Under these conditions, the snake ate the same amount of fruit for all three algorithms but BFS took 104s, DFS took 203, and A* took 82. The researchers concluded that BFS was more successful than DFS and a human but not as successful as A*[3]. In a similar experiment conducted by Kong, five different search algorithms were compared in solving the Snake Game. Best-First Search, A*, A* with forward checking, Almighty Move, and Random Move. A* with forward checking is similar to A* except that it considers what state the snake will be in after the fruit is reached for each potential move as part of its path evaluation[6]. The researcher found that A* with forward checking outperformed A* in that it produced the highest score while taking fewer steps. A sequential implementation of BFS followed by A* with forward search then by Almighty Move was suggested by Sharma et al. Through trial and error the authors concluded that for an AI bot to achieve the highest number of scores with the least amount of moves, the bot should move using BFS for first 4 moves, A* with forward search for next 34 moves, and Almighty with 62 moves[11]. The authors discovered that the AI Bot can be utilized to train players in the field of "Electronic Sports"[11]. Although this discovery will not be realized in this project, the findings are impressive.

6 Comparing BFS, DFS, and A* in Various Environments

The comparison of BFS, DFS, and A* can be analyzed in many different environments. For example, in the article written by Banerjee, BFS and DFS are compared based on optimality, completeness, and time and space complexity in a simulated restaurant environment. The environment contains custom start and goal nodes, tables, and a mobile serving robot[4]. Both BFS and DFS were able to find a path to the goal node but BFS had lower time and space complexity compared to DFS[4]. In an experiment conducted by Mahmud et al., the ability of three variants of DFS are examined to determine their effectiveness at discovering a map on an unknown maze. Their implementation of DFS with a Greedy Approach performed better in terms of number of movements than classic DFS[8]. The Greedy Approach works better than the Simple DFS Algorithm because it prioritizes finding the shortest path in the maze. By first going to unchecked very short dead-ends, followed by unchecked front adjacent cells, and then unchecked cells on the right or left, the Greedy Approach helps to minimize the movements and rotations required to navigate through the maze, resulting in a more efficient path.This suggests that although classic DFS is often suboptimal, it can improve with modifications to the algorithm.

In the article written by Permana et al., BFS, A*, and Dijkstra's pathfinding algorithms are used to solve the Maze Runner game. The Maze Runner game consists of an NPC who is trying to reach a goal state and has to navigate through a labyrinth of blocks created by players.The performance of each algorithm was measured by time, path length, and number of blocks played. The authors found that all three algorithms find the optimal path length, with A* being the overall best due to short searching times and smallest number of blocks[9]. Similarly to Permana et al., Iloh used a maze game to analyze the performance of DFS, BFS, and A*. The author was able to conclude that each algorithm can locate the shortest path length. It was found that DFS takes a long time to reach the goal. The author also concluded that A* is the most effective pathfinding algorithm, taking the shortest amount of time to reach the goal[**?**].

Alhassan et al. compare the performance of BFS, DFS, and A* for the game of Bloxorz. Bloxorz is a single agent path-finding problem in which a block must be moved in one of four directions (up, down, left, right) to reach a goal square. It is a 3-D block sliding game with a map of 1x1 boxes arranged in a special shape. The navigation of the block from its initial state to its goal state will be similar to the snake game, such that there are also four directions the snake can move in to reach the goal state. The authors consistently found that BFS and A* are able to solve the game in the number of optimal moves, while DFS was not always able to do so. Depending on the starting box and ending box, DFS can perform better than BFS and A* but, this is generally not the case. In conclusion, the authors note that BFS always finds the optimal path and that A* performance is strongly dependent on the heuristic function used. Additionally, because A* uses tree search, it expands more nodes than BFS and DFS, using more memory[2].

Shi assessed the differences between BFS, DFS, and A* in solving the infamous eight puzzle. Consistent with the findings of other articles previously discussed, A* was found to be the most effective algorithm in solving this puzzle[12]. A* outperformed BFS and DFS in elapsed time and number of search steps. In this specific puzzle, DFS performed close to ten times worse than BFS, and BFS performed four times worse than A*. It appears that in puzzles similar to the eight puzzle, DFS does not find the optimal solution, although it does find a solution.

7 Conclusion

Pathfinding algorithms play an important role in the field of Artificial Intelligence, and the Snake Game is an excellent platform for evaluating the performance of different search algorithms, including DFS, BFS, and A*. Evidently the consensus is that with a thoughtful heuristic evaluation, A*

performs optimally compared to BFS and DFS. While DFS has linear space complexity, it may not be suitable for larger and complex environments. We can modify it and use a greedy variation of DFS to improve the speed. BFS is guaranteed to find an optimal solution, but its time and space complexity may become impractical in certain scenarios. A* has emerged as the most efficient algorithm due to its heuristic evaluation, and overall efficiency. Based on the knowledge gained from this literature review, it is probable that DFS is best suited for the easy mode, BFS for the normal mode, and A* for the professional mode in our game.

References

[1]]Iloh [n. d.]. A COMPREHENSIVE AND COMPARATIVE STUDY OF DFS, BFS, AND A* SEARCH ALGORITHMS IN A SOLVING THE MAZE TRANSVERSAL PROBLEM. 2 ([n. d.]). https://ijssass.com/index.php/ijssass/article/view/54

[2] Tahani Q. Alhassan, Shefaa S. Omar, and Lamiaa A. Elrefaei. 2019. Game of Bloxorz Solving Agent Using Informed and Uninformed Search Strategies. *Procedia Computer Science* 163 (2019), 391–399. https://doi.org/10.1016/j.procs.2019.12.121 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence.

[3] Naga Sai Dattu Appaji. 2020. Comparison of Searching Algorithms in AI Against Human Agents in Snake Game. *International Journal of Computer Science and Mobile Computing* (2020).

[4] Ramin Banerjee, Chakraborty and Satti. 2018. Space Efficient Linear Time Algorithms for BFS, DFS, and Applications. *Theory of Computing Systems* 62, 1 (2018).

[5] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. 2021. A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science* 179 (2021), 507–514. https://doi.org/10.1016/j.procs.2021.01.034 5th International Conference on Computer Science and Computational Intelligence 2020.

[6] Shu Kong. 2013. Automated Snake Game Solvers via AI Search Algorithms. *IEEE* 5, 3 (2013).

[7] Daohong Liu. 2023. Research of the Path Finding Algorithm A* in Video Games. *Highlights in Science, Engineering and Technology* 39 (2023).

[8] Md. Sazzad Mahmud, Ujjal Sarker, Md. Monirul Islam, and Hasan Sarwar. 2012. A Greedy Approach in Path Selection for DFS Based Maze-map Discovery Algorithm for an autonomous robot. In *2012 15th International Conference on Computer and Information Technology (ICCIT)*. 546–550. https://doi.org/10.1109/ICCITechn.2012.6509798

[9] Arifitama Syahputra Permana, Bintoro. 2018. Comparative Analysis of Pathfinding Algorithms A*, Dijkstra, and BFS on Maze Runner Game. *International Journal Of Information System Technology* 1, 2 (2018).

[10] S Pooja, S Chethan, and C V Arjun. 2016. Analyzing uninformed search strategy algorithms in state space search. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. 97–102. https://doi.org/10.1109/ICGTSPICC.2016.7955277

[11] Shubham Sharma, Saurabh Mishra, Nachiket Deodhar, Akshay Katageri, and Parth Sagar. 2019. Solving The Classic Snake Game Using AI. In *2019 IEEE Pune Section International Conference (PuneCon)*. 1–4. https://doi.org/10.1109/PuneCon46936.2019.9105796

[12] Hua Shi. 2011. Searching algorithms implementation and comparison of Eight-puzzle problem. In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Vol. 2. 1203–1206. https://doi.org/10.1109/ICCSNT.2011.6182175