WRITING ASSIGNMENT 5

Yahya Alhinai
alhin010@umn.edu
CSCI4511W

December 4, 2019

1 Introduction

This paper is presenting several algorithms that are intended to automate the snake game. The goal of comparing several algorithms is to find the best way pros and cons for each algorithm and ultimately finding an optimal or near optimal in every stage of of the snake game.

In order to find the best path to take in each stage the snake is in, an efficient and fast search techniques needed to be formulated and developed to specifically solve such a problem where it is so much time and resources in order to investigate every possible path in each state which might be impossible for sequential processing computer if it's given a large snake with high resolution. For instance, if the Snake game is 300x300 grid, it would take over 140 CPU-years[3].

2 Literature Review

In the paper "Near Optimal Hierarchical Path-Finding", the authors have developed several methods that overcomes the restrictions A* search faces [1]. Those restrictions are mainly seen when the problem gets large and as the size of search space increases, the number of paths that could be taken increase exponentially. Therefore, path-finding can result in serious performance bottlenecks. This paper presented Hierarchical Path-Finding A* (HPA*) which reduces the possible paths taken by disregarding the distance that is not near optimal distances. Near optimal distance is defined by the user where it shows in the paper that

"HPA* is shown to be up to 10 times faster, while finding paths that are within 1% of optimal". Leading for this algorithm to work under constraints of limited memory and CPU resources.

Multi-agent path finding (MAPF) is an important computational algorithm that is introduced by the authors Adi Botea¸ Pavel Surynek [2]. The main purpose of MAPF algorithm is to navigate agent from its location to their target location while avoiding collisions and deadlocks. This is methods and strategies shown in this paper are helpful for the project because in the snake game not only you need to get from one location to another, but also you have to avoid collisions with the tail of the snake as a rule for the game. Fundamentally, MAPF is an improvement over multi-agent path finding focuses on undirected graphs. MAPF works on directed graphs which is ideal in my case as snake game is consider to be semi- directed graphs.

Several papers have approached this problem by using an evolutionary algorithm (EA) [9]. In the paper "Snake Game AI", the authors have proposed the idea about dynamically adjusting the weights according to the length of the snake. As the snake get taller the collision probability increases; therefore, the weight has to change to tune in performance. Since the algorithm presented is an evolutionary-base algorithm, the paper has identified good crossover operator to make generation-to-generation evolution the best it could that I might use in my project.

The paper "Path planning with modified A star algorithm" introduces several modifications and improvements of A star algorithm [4]. These modifications are focused primarily on path optimality and the time it takes to find a path. Even though the paper has specified those improvements to work on mobile robot based on a grid map, the methodology can be implemented for similar problems such as the snake game. Several interesting findings have been articulated in the paper. First, it was shown that the fastest algorithm to find a path is JPS A* and it was shown that it works the fastest in various environments. Though, the resulted path would be less optimal than the original A* algorithm would have produced. Another finding that was noted in the paper is the best algorithm to find the optimal or near optimal path is Basic Theta* even in environments with low symmetry. The cost of Basic Theta* finding an optimal path comes with time insufficiency in comparison to other A* algorithms family.

The paper “A force-directed maze router” investigate a new routing algorithm [6]. The algorithm is utilizes force-directed placement and maze searching techniques in order to full functions. The algorithm is optimized to for maze routing in the since that it is able to route around complex layouts with various obstructions.The algorithm presented is an improvement of maze algorithm which requires a large memory consumption. This algorithm uses less memory. This is done by investigating only the agent part of the maze. The part of algorithm that optimized routing might be used in the snake game if time allows.

3 Related Work

Genetic algorithm (GA) is a possible candidate that I will consider to solve the snake game [8]. The basic functionally of genetic algorithm can be sum in two operations. The first process is the selection of individuals to be moved to the next generation. The second process is manipulation the selected individuals to form the next generation by crossover and mutation techniques As the authors indicates in the paper “the selection mechanism determines which individuals are chosen for mating (reproduction) and how many offspring each selected individual produces” [8]. This means that the main principle of this algorithm is the better an individual; the higher is its chance of being parent. Making the produce of each new generation be better than its successor. In this algorithm, it’s important to balance between exploration and exploration within the mechanism of the selection. In solving the snake game, The GA algorithm start set initial random cities and then evaluate how fit each city current set of cities. Followed by selecting parent cities for next generation and cross over these parents’ cities to create new set of cities. Evaluate the fitness of search city and then decide if it is the optimal or near optimal path. If it is then terminating the algorithm and if it’s not, then keep producing new generation of cities. The downside of GA algorithm is that it takes a lot of time to reach a to a good-enough result. As well as it does not produce an optimal solution for the snake game because the fact it’s randomness nature in moving from on generation to another.

Another possible candidate to solve snake game is lin-kernighan algorithm (LH). This algorithm starts by having a set of cites described in paper [5] by a tour. LH algorithm refining the path that links cities by swapping pairs of sub-tours of cities to make a new tour and thus getting closer to the optimal solution each time. It works by switching two or three cities to make the tour shorter. With each step, LH decides how many paths between cities need to be switched to find a shorter tour. It keeps refining and switching between two/three cites until there is no further improvement that can be made to any of the cities. By then, it can be concluded that LH algorithm have reached an optimal solution. LH algorithm is the best out of the three algorithms presented in this paper. Because the fact it's relatively fast in reaching to a result. Also, usually the results produced by this algorithm are usually optimal. The only downside is that it requires some sort of symmetry to the problem.

The last algorithm presented in solving snake game is African Buffalo Optimization Algorithm (ABO). First of all, the authors of this paper [7] have chosen to represent population-based stochastic optimization technique as African buffalos as the ABO algorithm was inspiration from their behavior. ABO start choosing a start city for each of the buffalos and randomly locate them in those cities. Then it Updates the buffalo fitness by updating their locations as they follow the current best path with the shortest heuristic value. The determine the local minimal path and compare it to the overall minimal path. Using heuristic to construct new path by adding cities that the buffalos have not visited. Repeats this process until the local path cannot be minimized further. Once reaching the minimal local path the algorithm spits out the results. The only downside for ABO algorithm is that it takes an exponential time to reach to the solution. This is due to the repetitiveness of updating the path for each node each time something changes.

References

[1] A. Botea, M. Müller, and J. Schaeffer. Near optimal hierarchical path-finding. *Journal of game development*, 1(1):7–28, 2004.

[2] A. Botea and P. Surynek. Multi-agent path finding on strongly biconnected digraphs. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] K. K. Burusco Goñi. *Methodological approach to conformational search a study case: cyclodextrins*. Universitat Autònoma de Barcelona,, 2009.

[4] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.

[5] K. Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[6] F. Mo, A. Tabbara, and R. K. Brayton. A force-directed maze router. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 404–407. IEEE Press, 2001.

[7] J. B. Odili and M. N. Mohmad Kahar. Solving the traveling salesman's problem using the african buffalo optimization. *Computational intelligence and neuroscience*, 2016:3, 2016.

[8] N. M. Razali, J. Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

[9] J.-F. Yeh, P.-H. Su, S.-H. Huang, and T.-C. Chiang. Snake game ai: Movement rating functions and evolutionary algorithm-based optimization. In *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 256–261. IEEE, 2016.