IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AI ALGORITHMS FOR PACMAN

[1]Pendem Swetha, [2]A.Mary Sowjanya
[1]Student, [2]Associate Professor
[1] Department of Computer Science and Systems Engineering, [1]Andhra University College of Engineering(A),Visakhapatnam, Andhra Pradesh, India.

Abstract: This project is about developing pacman game with AI..The Game Pac-Man is a very challengingvideo game that can be useful in conducting AI(Artificial Intelligence) research. Here, the reason we have implemented various AI algorithms for pacman game is that it helps us to study AI by using visualizations through which we can understand AI more effectively. The main aim is to build an intelligent pacman agent which finds optimal paths through the maze to find a particular goal such as a particular food position, escaping from ghosts. For that, we have implemented AI search algorithms like Depth first search, Breadth first search, A*search, Uniform cost search. We have also implemented multi-agents like Reflex agent, Minimax agent, Alpha-beta agent .Through these multiagent algorithms, we can make pacman to react from its environmental conditions and escape from ghosts to get high score. We have also done the visualization part of the above AI algorithms by which anyone can learn and understand AI algorithms easily. For visualisation of algorithms, we have used python libraries matplotlib and NetworkX(used to draw graphs for the states explored).

Index Terms **- Pacman, maze, DFS, BFS, UCS, A*Search, Reflexagent, Minimaxagent, alpha-betapuning, visualization, Networkx, matplotlib.**

I. INTRODUCTION

Pac-man is a maze arcade game developed and published by Namco in 1980. The player supervise Pac-man, who must eat all the dots inside an surrounded maze while keeping away from the four colored ghosts. Eating huge flashing dots called power pellets changes the ghosts to turn blue, allowing Pac-Man to consume them for bonus points. Pac-Man was a critical and commercial victory, and has an commercial and cultural legacy. The game is supreme and influential, and it is often listed as one of the extraordinary video games of all time.

II. RELATED WORK

Veenus Chhabra et al. [1] expressed the brief explanation of Game theory. through this we showed some characteristics of game. The algorithm is developed for using smaller indices so that there is less complication for guessing the tack number and then determining as we follow color indexing through number and assign. Also, we can also execute the game to begin from initial by not only getting “6” at dice but also with “1”. We can also restrict that to “two” only. As we get continuous “three” “6”, the turn is altered without any move of token.

César Villacís et al. [2] proposed a system how to optimize an educational video game named Tic-Tac-Toe by means of semiotics analysis, in order to stimulate logical and spatial reasoning of children. The main issue in this model has been to design a mathematical model that was implemented with Artificial Intelligence algorithms and a graphical user interface including Semiotics, applied to an incremental methodology with the aim of producing an enjoyable and interactive environment., this examination mingle theories about stimulating cognitive growth of children; game design prototypes.

Sebastian Thrun et al. [3] proposed a system called NeuroChess, an approach for learning to play chess from the final outcomes of games. Firstly, instruction time is bounded. This is particularly the case if only the final outcomes are considered. Secondly, with each move of TO-learning NeuroChess loses data. It is therefore unclear that a TD- like approach will ever, for example, develop good chess openings .\ It has been well conceded that the resulting cost in chess is determined by the time it takes to make a move.

Sergey Karakovskiy et al. [4] proposed a system describes the Mario AI benchmark, a game-based benchmark for reinforcement learning algorithms and game AI techniques developed by the authors. The researchers and students from round the world have contributed diverse solutions to try to beat the benchmark. The paper synopsis these contributions, gives an survey of the state-of-art in Mario-playing AIs, and chronicles the growth of the benchmark.

Koller et al. [5] proposed a graphical representation for non-cooperative games—*multi-agent influence diagrams* (MAIDs). The basic elements in the MAID representation are *variables*, allowing an explicit representation of dependence, or relevance, relationships among variables and showed strategic relevance can be used to decompose large games into a set of interacting smaller games, which can be solved in sequenceand also show that this decomposition can lead to substantial savings in the computational cost of finding Nash equilibria in these games.

Stone et al. [6] proposed the Distributed artificial intelligence (DAI) has existed as a subfield of artificial intelligence for less than two decades. DAI deals with systems that consist of multiple independent entities that interact in a domain. Traditionally, DAI is divided into two sub-disciplines: Distributed Problem Solving (DPS) focuses on the information management aspects of systems with several components working together towards a common goal.

Shoham et al.[7] explained the Game theory and social choice theory have traditionally been the domain of economics. With growing interest, computer scientists are increasingly using them as tools. The authors refrain from providing a limited definition and emphasize the openness and interdisciplinary nature of multi-agent systems.

Elmasry et al.[8] gave, among others, an implementation of breadth-first search (BFS) and depth-first search (DFS) in a graph on n vertices and m edges, taking \(O(m+n)\) time using O(n) and \(O(n \lg \lg n)\) bits of space, respectively an improvement of the naive implementation \(O(n \lg n)\)(We use \(\lg \) to denote the logarithm to the base 2.) bits. We continue this line of work with a focus on space.

III. METHODOLOGY

Algorithms:

The structure include implementation of AI search algorithms and Adversial search(Multiagents).
The following are the search algorithms performed:

1. Depth First Search
2. Breadth First Search
3. Uniform Cost Search
4. A* Search

The following are the adversial search algorithms that are performed:

1. Reflex Agent
2. Minimax Algorithm
3. Alpha-beta Pruning

SEARCH ALGORITHMS:

3.1 Uninformed Search Algorithms:

Uninformed search is a class of common search algorithms which control inbrute force-way. Uninformed search algorithms do not have added instruction about state orsearch space other than how to move the tree, so it is also called blind search.

3.1.1 DFS:

Depth-first search is a circular algorithm for traversing a tree or graph data structure. It is called the depth-first search because it begins from the source node and go along with each path to its greatest depth node before operating to the following path. DFS utilize a stack data structure for its execution. The procedure of the DFS algorithm is related to the BFS algorithm.

Time Complexity: Time complexity of DFS will be similar to the node traversed by the algorithm. It is specifiedby:

$$T(n)= 1+ n^2+ n^3 +\ldots\ldots+ n^m=O(n^m) \quad (1)$$

Where, m= maximum depth of any node and this can be much big than d (Shallowest solutiondepth).

Space Complexity: DFS algorithm require to reserve only single path from the root node, hence space complexity ofDFS is equivalent to the size of the fringe set, which is

$$O(bm). \quad (2)$$

3.1.2 BFS

Breadth-first search is the most usual search strategy for traversing a tree or graph. This algorithm searches breadth wise in a tree or graph, so it is called breadth-first search.BFS algorithm begins searching from the source node of the tree and enlarge all beneficiary node at the present level before moving to nodes of next level. The breadth-first search algorithm is an sample of a general-graph search algorithm. Breadth-first search is performed using FIFO queue data structure.

Time Complexity: Time Complexity of BFS algorithm can be acquire by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at each position .

$$T(b) = 1+b^2+b^3+ \quad + b^d = O(b^d) \tag{3}$$

Space Complexity**:** Space complexity of BFS algorithm is specified by the Memory size of frontier which is $O(b^d)$. (4)

3.1.3 UNIFORM COST SEARCH:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm approach into play when a dissimilar cost is available for each margin. The primary aim of the uniform-cost search is to discover a path to the aim node which has the lowest cumulative cost. Uniform-cost search enlarge nodes according to their path costs form the root node. It can be used to resolve any graph/tree where the optimal cost is in demand. A uniform-cost searchalgorithm is executed by the priority queue. It offers highest priority to the lowest cumulative cost. Uniform cost search is equal to BFS algorithm if the path cost of all edges is the identical.

Time Complexity: Let C* is Cost of the optimal solution, and ε is each step to get near to the aim node. Thenthe number of steps is = C*/ε+1. Here we have taken +1, as we start from begin 0 and end toC*/ε.

Hence, the worst-case time complexity of Uniform-cost search is$\mathbf{O(b^{1 + [C^*/\varepsilon]})}$/. (5)

Space Complexity: The same logic is for space complexity so, the worst-case space complexity of Uniform-costsearch is $O(b^{1 + [C^*/\varepsilon]})$. (6)

3.2 Informed Search Algorithms

3.2.1 A* SEARCH ALGORITHM:

A* search is the most frequently known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the begin state g(n). It has collaborate features of UCS and greedy best-first search, by which it resolve the problem efficiently. A* search algorithm discover the direct path through the search space using the heuristic function. This search algorithm enlarge less search tree and provides optimal result quick. A* algorithm is close to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we utilize search heuristic as well as the cost to reach the node. Hence we can merge both costs as following, and this sum is called as a **fitness number**. f(n)=g(n)+h(n)

f(n)=estimated cost of the cheapest solution

g(n)=cost to reach node n from start state

h(n)=cost to reach from node n to goal node

At each point in the search space, only those node is enlarge which have the small value of f(n), and the algorithm terminates when the goal node is found.

Time Complexity: The time complexity of A* search algorithm depends on heuristic function,and the number of nodes enlarge is exponential to the depth of solution d. So the timecomplexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **O(b^d)** (7)

3.3 Adversial Search Algorithms*:*

3.3.1 REFLEX AGENT:

We produce here a reflex agent which pick out at its turn a random action from the legal ones. Note that this is non identical from the random search agent, since a reflex agent does not construct a series of actions, but select one action and performs it.This reflex agent selects its present action based only on its current perception. The ReflexAgent computes the results of the states reachable with these steps and choose the states that outcome into the state with the highest score. In case more states have the highest score, it will select randomly one.

3.3.2 MINIMAX ALGORITHM:

In case the world where the agent plan of action ahead includes other agents which plan against it, adversarial search can be used. One agent is called MAX and the other one MIN. Utility(s; p) offers the _nal numeric value for a game that close in terminal state s for participant p. For example, in chess the standards can be +1, 0, ½. The game tree is a tree where the nodes are game states and the edges are moves. Optimal decisions in games must give the finest move for MAX in the beginning state, then MAX's moves in all the states proceed from each possible response by MIN, and so on. Minimax value make sure optimal strategy for MAX.

3.3.3 ALPHA BETA PRUNING:

In order to limit the number of game states from the game tree, alpha-beta pruning canbe registered, where α = the value of the best (highest value) choice there is so far at any choice point across thepath for MAX

β = the value of the best (lowest-value) choice there is so far at any choice point across the path for MIN

Table 1. Comparison of search algorithms for Medium maze

	DFS	BFS	UCS	A*
Total Cost	130	68	68	68
Nodes expanded	144	268	269	221
Score	380	442	442	442

Table 2. Comparison of search algorithms for tiny maze

	DFS	BFS	UCS	A*
Total Cost	10	8	8	8
Nodes expanded	14	16	15	14
Score	500	502	502	502

Table 3. Comparison of search algorithms for big maze

	DFS	BFS	UCS	A*
Total Cost	210	210	210	210
Nodes expanded	390	618	620	549
Score	300	300	300	300

IV. RESULTS

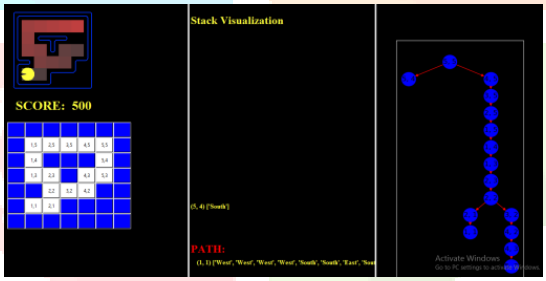

fig 4.1 DFS

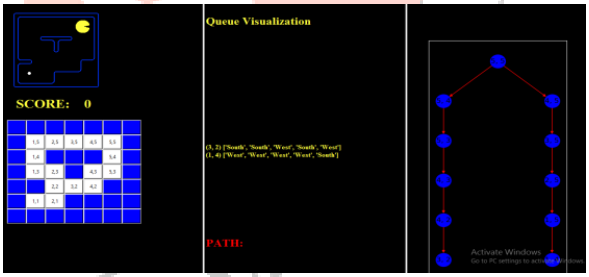

fig 4.2 BFS

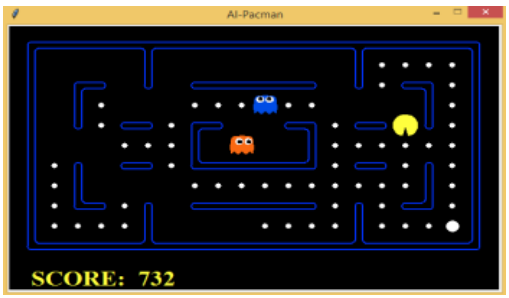

fig 4.3 Minimax agent

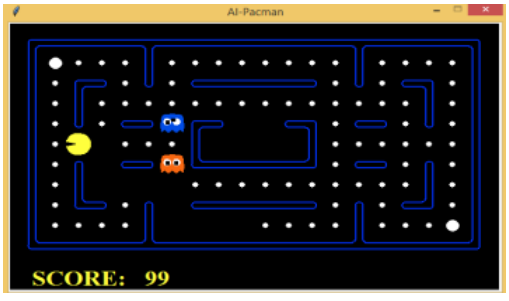

fig 4.4 AlphaBeta agent

V. CONCLUSION

We have completed executing search algorithms Depth First Search,Breadth FirstSearch,Uniform Cost Search,A* Search.and also executed Reflex Agent,MiniMax Agent, AlphaBeta Agent. In search algorithms, DFS doesn't give the best solution because it does not provide least cost solutions. BFS provides least cost solutions with respect to effort by the pacman in reaching the food but the expanded nodes are very large(268) which takes a lot of time to find the solution which is best and when we compare UCS with BFS and DFS, the time and costs are relatively high which does not give us the best solution. So, A* is the best as it finds out the sum of cost of reaching nodes and the cost of reaching the goal node from that particular node which is (x+y).We have compared the performance of the search algorithms in terms of nodes expanded, cost, and time taken.

REFERENCES

[1] Chhabra, Veenus and Kuldeep Tomar. "Artificial Intelligence : Game Techniques Ludo-A Case Study." (2015).

[2] Villacis Silva, Cesar & Fuertes, Walter & Santillán Trujillo, Mónica & Aules, Hernán & Tacuri, Ana & Zambrano, Margarita & Salguero, Edgar. (2016). On the Development of Strategic Games based on a Semiotic Analysis: A Case Study of an Optimized Tic-Tac-Toe. 425-432. 10.5220/0005772904250432.

[3] Learning To Play the Game of Chess - Sebastian Thrun**,** University of Bonn, Departmentof Computer Science III, Romerstr. 164, 0-53117 Bonn, Germany.

[4] S. Karakovskiy and J. Togelius, "The Mario AI Benchmark and Competitions," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 55-67, March 2012, doi: 10.1109/TCIAIG.2012.2188528.

[5] D. Koller and B. Milch (2003): Multi-agent influence diagrams for representing and solvinggames.

[6] P. Stone and M. Veloso (2000): Multiagent systems: a survey from a machine learningperspective.

[7] Y. Shoham and K. Leyton-Brown (2008): Multiagent systems: algorithmic, game theoretic,and logical foundations,Cambridge University Press.

[8] Banerjee, Niranka & Chakraborty, Sankardeep & Raman, Venkatesh. (2016). Improved Space Efficient Algorithms for BFS, DFS and Applications.