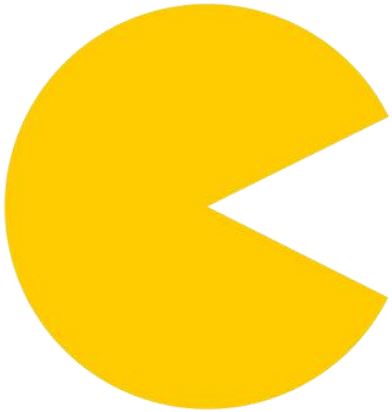
Université Saint-Joseph de Beyrouth
Faculté d'ingénierie
École supérieure d'ingénieurs de Beyrouth

Artificial Intelligence



Presented by:

Miriam Chebib

Carl Haddad

Jovan Aoukar

Presented to:

Dr. Youssef Bakouny

PAC-MAN is one of the most well-known video games in the world. We can use a problem-solving agent to automatically find optimal paths through its maze world, taking into account both reaching specific locations (e.g., finding all the corners) and eating all the dots in as few steps as possible. To create this intelligent agent, we used several uninformed search algorithms (e.g., DFS, BFS, UCS) that are given no information about the problem other than its definition. However, while some of these algorithms can solve any problem, none of them can do so efficiently. As a result, we introduced informed search algorithms (e.g., A*), which, with some guidance, can perform quite well.

DFS:

Depth First Search is an uninformed search strategy which means that strategies have no additional information about the states other than the ones provided in the problem definition. DFS always expands the deepest, recently generated, node in the search tree (LIFO). After implementing the algorithm, the results are:

Maze Size	Score	Cost	Nodes Expanded
Tiny Maze	500	10	15
Medium Maze	380	130	146
Big Maze	300	210	390

With this algorithm, pacman did not traverse al the search nodes to arrive at the goal node since all the search went as far down one path before turning around and trying another.

A downside was that DFS doesn't necessarily find the shortest path to a node, it only finds the leftmost solution in the search tree regardless of the depth or cost of the node.

BFS:

Breadth First Search is a level by level search algorithm. Before proceeding to the next level, all nodes in a given level are expanded. It begins by expanding the shallowest node in the search tree (FIFO). After implementing the algorithm, the results are:

Maze Size	Score	Cost	Nodes Expanded
Tiny Maze	502	8	15
Medium Maze	442	68	269
Big Maze	300	210	620

BFS will find the shortest path between the starting point and any other reachable node.

The downside of BFS is that the number of expanded nodes is relatively large compared to DFS, which requires a long time to find the optimal solution.

UCS:

Uniform-cost search (UCS) expands the node with lowest path cost. When a separate cost is provided for each edge, this algorithm is used. Finding the path with the lowest cumulative cost to the goal node is the main objective of the uniform-cost search.

Maze Size	Score	Cost	Nodes Expanded
Tiny Maze	502	8	15
Medium Maze	442	68	269
Big Maze	300	210	620

We conclude that BFS and UCS have the same results concerning the Score, Cost, and number of Nodes Expanded since the path cost of all edges is the same.

Agent	Score	Cost	Nodes Expanded
StayWestAgent	418	68719479864	108
StayEastAgent	646	1	186

As for the mediumDottedMaze, the StayEastSearchAgent presents a cost equal to 1 because of its exponential cost function: f(x) = 0.5^x
For the mediumScrayMaze, the StayWestSearchAgent presents a cost equal to 17183894840 because of its exponential cost function: f(x) = 2^x

The downside to UCS is that it only considers the cost of the path, and not the number of steps involved in the search.

A* Search

To assess nodes, A* Search –Informed Search Algorithm- combines g(n), the cost to get to the node, and h(n), the cost to travel from the node to the goal (heuristic function): f(n) = g(n) + h(n). Using the Manhattan heuristic:

Maze Size	Score	Cost	Nodes Expanded
Tiny Maze	502	8	14
Medium Maze	442	68	222
Big Maze	300	210	549

To better illustrate the difference between UCS and A*, we tried the Open Maze problem:

Search	Score	Cost	Nodes Expanded
UCS	456	54	682
A*	456	54	535

We can deduce that A* minimized the number of expanded nodes.

The downside to A* is that every action should have a fixed cost, and it is dependent on the accuracy of the heuristic function used.

Finding All the Corners:

In this problem, the goal is to eat all four of the food dots located in the corners, instead of finding the shortest path. Using BFS:

Corners Size	Score	Cost	Nodes Expanded
Tiny Corners	512	28	252
Medium Corners	434	106	1966
Big Corners	378	162	7949

Corners Problem:

The same problem as above but using A*:

Corners Size	Score	Cost	Nodes Expanded
Tiny Corners	512	28	63
Medium Corners	434	106	459
Big Corners	378	162	1037

We can conclude that A* search is the most optimal search algorithm if the heuristic function is used correctly.