Lab Report- Exploring Search Strategies through Pacman

emil.haglund@umu.se

January 2021

1 Aim

The aim of this lab is to explore and compare different search algorithms commonly used in artificial intelligence applications. Experimentation with the search algorithms will occur in a simulated world playing Pacman.

The Pacman agent will find paths through his maze world, to reach a particular location while avoiding ghosts and collecting food efficiently. Search algorithms including depth first search (DPS), breadth first search (BFS), uniform cost search (UCS) and a- star search (A* Search) will be applied to Pacman scenarios.

2 Background and Implementation

This section will provide a brief introduction to the search algorithms implemented in this lab. In addition, specifics on the code implementation namely data structures used will also be provided.

2.1 Depth First Search (DFS)

In DFS, the algorithm starts at the root node and explores as far as possible along a given path until reaching a dead end. At this point the algorithm backtracks on the same path to find nodes to traverse. The DFS algorithm was implemented using the data structure stack.

2.2 Breadth First Search (BFS)

In BFS, the algorithms starts at the root node and explores all of the neighboring nodes before moving on to nodes at the next depth level. The BFS algorithm was implemented using the data structure queue.

2.3 Uniform Cost Search (UCS)

UCS can be applied when moving to different states infers variable costs for the agent. The cost is decided by a cost function which can be selected dependent on the problem. The goal of the UCS algorithm is to find the path to the goal node which has the lowest cumulative cost. UCS chooses the node with the lowest cost as the next node for exploration. It is thus

equivalent to BFS algorithm if the path cost of all edges is the same. The UCS algorithm was implemented using the data structure priority queue.

2.4 A-star search (A*)

A* search is an informed search algorithm in contrast to the previous 3 search algorithm which are uninformed. In addition to a cost function, A* search uses a heuristic function to find the most promising path. The heuristic function produces an estimate of how close the agent is to the goal state. The A* algorithm was implemented using the data structure priority queue.

3 Method

The search algorithms were implemented in accordance with the instructions provided in question 1-4 in the assignments for the Berkeley AI Pacman project. See: https://github.com/jspacco/pac3man/searc

All 4 search algorithms: DFS, BFS, UCS A*-search were compared on their performance in the tinyMaze, mediumMaze and the BigMaze. Evaluated metrics are expanded nodes and path cost. The A* utilized the manhattan distance to the end state as its heuristic function.

4 Results

4.1 Comparison of search algorithms

In the following tables the nodes expanded and path cost in 3 mazes of varying sizes are presented. All mazes are empty, that is they have no food or ghosts present.

4.1.1 tinyMaze

	DFS	**BFS**	**UCS**	**A***
Nodes expanded	15	15	15	14
Path cost	10	8	8	8

Table 1: Figure 1- Nodes expanded and path cost for tinyMaze experiments

4.1.2 mediumMaze

	DFS	**BFS**	**UCS**	**A***
Nodes expanded	146	269	269	258
Path cost	130	68	68	68

Table 2: Figure 2- Nodes expanded and path cost for mediumMaze experiments

4.1.3 bigMaze

	DFS	BFS	UCS	A*
Nodes expanded	390	620	620	582
Path cost	210	210	210	210

Table 3: Figure 3- Nodes expanded and path cost for bigMaze experiments

4.2 Question 3- Varying the cost function

By changing the cost function, we can encourage Pacman to find different paths. In the following subsections both agents use uniform cost search but with different cost functions operating on different maps.

4.2.1 mediumDottedMaze

The pacman agent using the cost function StayEast succesfully reached the end state and ate all the food along the way. Average score is a function of the path cost.

- Nodes expanded: 186
- Path cost: 1
- Average score: 646

4.2.2 mediumScaryMaze

The pacman agent using the cost function StayWest succesfully reached the end state and avoided all ghosts along the way.

- Nodes expanded: 108
- Path cost: 68719479864
- Average score: 418

5 Discussion

All search algorithms successfully traversed the mazes from start to end state however they did not always choose the same path and explored different sections of the map in the process of selecting their proposed path. This conclusion can be drawn by inspecting the paths and specific nodes explored by the agent but it can also be inferred by looking at the results in section 4.1. DFS can take less time to reach the end goal than BFS as BFS traverses one depth level at a time while DFS can quickly find paths to reach the end goal through a series of depth levels. This can be seen in the results in Table 2 where DFS expanded 146 nodes compared to 269 for BFS. BFS will find the shortest path while DFS is not guranteed to do so. This can also be seen in Table 2 where the path cost for DFS was 130 and for BFS was 68. In a larger, more complex search problem than in this lab the number of nodes explored has consequences for both the computational time and memory required. Finding

a path with a low cumulative cost is important where an efficient or even optimal solution is required.

Both UCS and A* search like BFS found optimal solutions in terms of path cost. A* star search using the Manhattan distance to the end state as its heuristic function managed to find the optimal path whilst expanding fewer nodes than both BFS and UCS in all 3 test cases. This is an expected result as the thought behind informed search methods like a* search is to use domain knowledge to efficiently find a solution.

By using a cost function to inform the path of the pacman agent, the agent successfully achieved the objectives of eating all the food items in the mediumDottedMaze and avoiding the ghosts in the mediumScaryMaze. For solving more advanced tasks a more sophisticated cost function could be implemented using the same principles and algorithms as in this lab.