LINGI2261: Artificial Intelligence

Solving Problems with Informed Search

Jiayue Xue
Sanae Abdelouassaa

24th October 2018

CONTENTS

CHAPTER 1

A^* VERSUS UNIFORM-COST SEARCH

1.1 Question 1

Give a consistent heuristic for this problem. Prove that it is consistent. Also prove that it is admissible.

Answer: By definition , h is the heuristic function that estimates the cost of the optimal path from node n to the goal and it is consistent only if for each pair of node n and its successor $n^{'}$ obtained by action a, the estimated cost of reaching the goal from n is no greater than the total cost of getting to $n^{'}$ from n and the estimated cost of reaching the goal from $n^{'}$:

$$h(n) \leq c(n, a, n^{'}) + h(n^{'}). \tag{1.1}$$

Let $h(n)$ be the cost of the optimal path from n to the goal.
If $h(n) = 0$, then n is the goal and therefore $h(n) \leq h(n^{'}$.
If n is k steps away from the goal, there must exist some successors $n^{'}$ of n generated by some action a that is on the optimal path from n to the goal (via action a). In that, $n^{'}$ is $k1$ steps away from the goal.Therefore, we have the conclusion: $h(n) \leq c(n, a, n^{'}) + h(n^{'})$.

1.2 Question 2

Show on the left maze the states (board positions) that are visited during an execution of a uniform-cost graph search. We assume that when different states in the fringe have the smallest value, the algorithm chooses the state

with the smallest coordinate (i, j) ($(0, 0)$ being the bottom left position, i being the horizontal index and j the vertical one) using a lexicographical order.

Answer: The Uniform cost search aims to explore first the node with the lowest cost and evaluate the cost for each available node so g(n) = path cost(n) = sum of individual edge costs to reach the current node.
In the figure 1.1, we assume that the cost from n to $n + 1$ is 1 because we choose the state with the smallest coordinate (i, j) when we have the same value for two different states.

Figure 1.1: Question 2

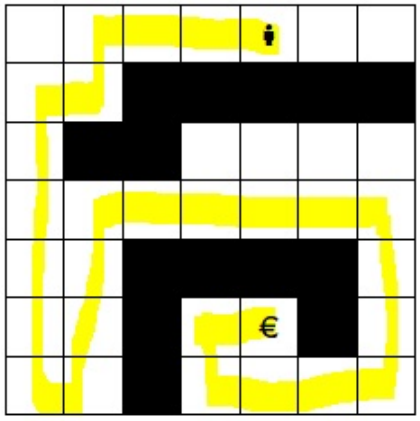
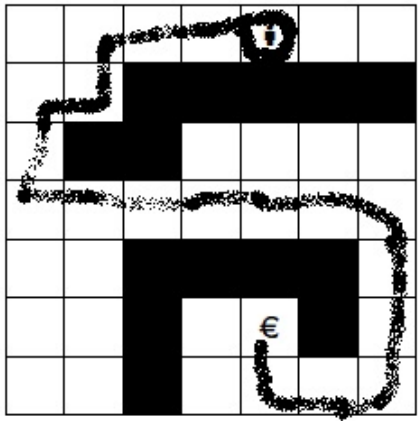


Figure 1.2: Question 3



1.3 Question 3

Show on the right maze the board positions visited by $A?$ graph search with a manhattan distance heuristic (ignoring walls). A state is visited when it is selected in the fringe and expanded. When several states have the smallest path cost, this uniform-cost search visits them in the same lexicographical order as the one used for uniform-cost graph search.

Answer: A^* algorithm is an informed search algorithm that chooses the optimal path through the current node and the path cost is defined by the function $f(n) = g(n) + h(n)$, where $g(n)$ is the exact cost starting from initial state to node n and $h(n)$ is the estimated cost from node n to the goal.
In the figure 1.2, we assume that the cost from n to $n + 1$ is 1 because we apply the algorithm by using Manhattan distance heuristic and simply ignore the walls defined by black positions.

CHAPTER 2

PACMEN PROBLEM

2.1 Question 1

Model the Pacmen problem as a search problem.

Answer:

1. States S: describles the locations of pacmen, foods and borders in the current time slot.
2. Initial state S_0: specifies how the game is set up at the start
3. Action a: defines a legal move
4. Transition model $RESULT(S, a)$: defines the result of a given action from the current state
5. Goal test $TEST(S)$: returns true when the game is over and false otherwise
6. Path cost function $COST(S)$: defines the final numeric value for a game that ends in terminal state

2.2 Question 2

What is the maximum branching factor for this problem considering there are k pacmen in the maze ?

Answer: Since the pacmen can move to the same postion, they will not influence each other. In that, the maximum branching factor for this problem is $four$.

2.3 Question 3

Give an admissible heuristic for a case of one pacman and n foods. Prove that it is admissible. What it is complexity ?

Answer: In this case, our heuristic is defined by the Manhattan Distance between the pair of pacman and food whose Manhattan Distance is the shortest among every possible combination of one pacman and one food. Since the Manhattan Distance of the selected pair of pacman and food is the shortest and we do not take walls into account, the number of the actual moves required to eat the food is strictly equal or larger than our estimated number, which means our heuristic is admissible.
The relative error is defined as $\epsilon \equiv (h^* - h)/h^*$. Since the step cost is constant in this game, the time complexity is $O(b^{\epsilon d})$, where d is the solution depth.

2.4 Question 4

Give an admissible heuristic for a case of k pacmen and n foods. Prove that it is admissible. What it is complexity ?

Answer: In this case, since every pacman must take one step, we make some changes to the aformentioned heuristic: we scan all the pacmen's position, and first one pair of pacman and food following the former standard. After one pacman take one step, it is removed from the pacmen list and our selection continues among the remaining pacmen list. As is mentioned before, the Manhattan Distance do not take walls into account and we choose the minimum possible pair in each step, our heuristic is still admissible.
The relative error is defined as $\epsilon \equiv (h^* - h)/h^*$. Since the step cost is constant in this game, the time complexity is $O(b^{\epsilon d})$, where d is the solution depth.

2.5 Experiments

As is shown in the experiment results, the number of explored nodes is always smaller with astar-graph-search, so is the the computation time. The reason for this phenomenon is that the astar-graph-search adopts heuristic

function which efficently provide useful information and prune unpromising candidates. However, in the last experiment, astar-graph-search takes more moves to reach the goal than breadth-first-graph-search does.

Table 2.1: Instance 1

Property / Algorithm	Time(s)	Explored nodes	Moves
astar-graph-search	0.016	11	9
breadth-first-graph-search	0.026	32	9

Table 2.2: Instance 2

Property / Algorithm	Time(s)	Explored nodes	Moves
astar-graph-search	0.034	27	19
breadth-first-graph-search	0.038	27	19

Table 2.3: Instance 3

Property / Algorithm	Time(s)	Explored nodes	Moves
astar-graph-search	0.071	45	25
breadth-first-graph-search	0.089	69	25

Table 2.4: Instance 4

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.035	28	13
breadth-first-graph-search	0.096	56	13

Table 2.5: Instance 5

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.036	26	13
breadth-first-graph-search	0.044	43	13

Table 2.6: Instance 6

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.064	41	25
breadth-first-graph-search	0.078	67	25

Table 2.7: Instance 7

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.108	78	25
breadth-first-graph-search	0.183	100	25

Table 2.8: Instance 8

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.223	157	33
breadth-first-graph-search	0.353	216	33

Table 2.9: Instance 9

Algorithm \ Property	Time(s)	Explored nodes	Moves
astar-graph-search	0.571	171	13
breadth-first-graph-search	3.207	1129	13

Table 2.10: Instance 10

Property / Algorithm	Time(s)	Explored nodes	Moves
astar-graph-search	4.104	562	12
breadth-first-graph-search	53.253	8761	8