BLG 435E - Artificial Intelligence Homework 1

Şahin Akkaya

November 27, 2020

1 Problem 1 - PEAS Description of Agents

1.a A drone that carries packets

- **Performance measure:** Number of packets carried successfully (without any damage) in a day
- **Environment:** Other drones, birds, tall buildings
- **Actuators:** Propellers, motors
- **Sensors:** Camera, GPS, tilt sensors, velocity sensors

Utility-based agent is the best architecture because the agent tries to carry the packets without any *damage* so it will try to maximize its utility.

1.b A robot that sings songs

- **Performance measure:** Time passed until baby sleeps
- **Environment:** Baby, other objects in the room
- **Actuators:** Speakers
- **Sensors:** Camera, microphone

The best architecture for this agent type is **Goal-based agent** because the agent tries to reach the goal state, which is defined as ”baby sleeps”.

1.c An agent that detects anomalies

- **Performance measure:** Number of anomalies detected correctly in a day
- **Environment:** Airport, people
- **Actuators:** Alarm system
- **Sensors:** Camera, X-Ray

Model-based agent is best architecture for this environment because the anomalies are *series of things* that differs from standard. So the agent needs to keep track of previous states.

1.d An agent that classifies emails

- **Performance measure:** Percentage of emails that classified correctly
- **Environment:** Mailbox
- **Actuators:** Marking emails as spam
- **Sensors:** NLP tools, contents of the email

Best architecture for this agent type is **Simple-reflex agent** because the actions are deterministic and the world is fully observable. So the agent can decide which action to perform now with a lookup table.

Task Environment	**Observable**	**Deterministic**	**Episodic**	**Static**	**Discrete**	**Agents**
Packet carrying drone	Partially observable	Stochastic	Sequential	Dynamic	Continuous	Multi agent
Singing robot	Partially observable	Stochastic	Sequential	Static	Continuous	Single agent
Anomaly detector	Partially observable	Stochastic	Sequential	Dynamic	Continuous	Multi agent
Emails classifier	Fully observable	Deterministic	Episodic	Static	Discrete	Single agent

2 Problem 2 - Admissible but Inconsistent Heuristics

2.a What values of $h(B)$ make h an admissible heuristic function?

An admissible heuristic function should satisfy 2 properties:

- It never overestimates the actual path cost
- $h(n) \geq 0$ for all n and $h(G) = 0$

The actual path cost from node B to G is 9. So if $0 \leq h(B) \leq 9$ then we can say h is admissible because it will satisfy above properties.

2.b What values of $h(B)$ make h a consistent heuristic function?

A heuristic is consistent if for all n, $h(n) \leq c(n, a, n') + h(n')$ where n' is neighbour of n and $c(n, a, n')$ is step cost to from n to n' when action a is performed. So if we write two inequality and so it together:

- $h(S) \leq c(S, a, B) + h(B)$ $5 \leq h(B)$
- $h(B) \leq c(B, a', C) + h(C)$ $h(B) \leq 6$

If $5 \leq h(B) \leq 6$ then h is consistent heuristic because all the other nodes are also satisfying the criteria.

2.c What values of $h(B)$ make it so that when you perform graph version of A^* on this graph, you first expand node S, then node A, then node B in order?

A^* algorithm uses a priority queue and expands cheapest nodes in the queue first. It decides the cost of node n using actual cost from starting node to node n, $g(n)$, and estimated cost from node n to goal node $h(n)$.

$$f(n) = g(n) + h(n) \tag{1}$$

In the initial state S is the only node in the queue and it will be expanded. Nodes A, B, and C will be added to the queue. In order to expand A next, $f(A)$ should be less than or equal to $f(B)$ and $f(C)$.

$$\begin{aligned} f(A) &\leq f(B) \qquad f(A) \leq f(C) \\ 7 &\leq 1 + h(B) \qquad 7 \leq 9 \\ 6 &\geq h(B) \end{aligned}$$

After node A is expanded the nodes in the priority queue will be B, C and D and their cost will be $1+h(B)$, $4+5=9$ and $7+2=9$ respectively. In order to expand B next $1+h(B)$ should be smaller than 9.

$$\begin{aligned} 1+h(B) &\leq 9 \\ h(B) &\leq 8 \end{aligned}$$

By combining these two inequalities we get

$$6 \leq h(B) \leq 8 \tag{2}$$

3 Problem 3 - Problem Solving with Search Algorithms

3.a Formulating the problem

In this problem agents are trying to reach their goals from their initial positions by performing actions that are available to them.

- **Initial State:** The initial positions of the agents
- **Actions:** The available actions for the agents are moving left, right, up, down and staying at the same position.
- **Goal Test:** The desired positions of the agents
- **Step cost:** Uniform step costs for each action, which is 1
- **States:** All the possible permutations of agent locations in the maze

3.b Running uninformed search algorithms

The only difference between Depth first search and Breadth first search algorithm is the data structure that they use as frontier. So I wrote a function and call it with appropriate arguments:

```python
def bfs(problem):
    return uninformed_search(problem, frontier=Queue)

def dfs(problem):
    return uninformed_search(problem, frontier=Stack)
```

Listing 1: Example code from external file.

`Stack` is just a regular Python `list` and `Queue` is `collections.deque` with a small modification. Just by looking at this we can understand that in depth first search the last node added will be popped first so it will go as deep as it can whereas in breadth first search nodes will be popped from the queue in the order they added so it will perform level by level search. You can find comparison of the algorithms with input_0.txt

	BFS	**DFS**
Num of steps	9	189
Num of nodes generated	213,869	2,117
Num of nodes expanded	23,893	228
Max num of nodes kept in the memory	19,669	1020
Running time (s)	3.094	0.038

Breadth first search algorithm checks all the states at the current level, hence it will need to create and check every node in each level making it use more memory and spend more time but it finds the optimal solution.

Depth first search algorithm doesn't need to store all the nodes in the current depth in memory hence requires less memory and since the maze is small the chances that it will encounter a solution is high. So it will find the solution faster than breadth first search but it is not optimal.

Both of the algorithms couldn't find solutions for input_3.txt and input_4.txt because there are too much possibilities and trying to find a solution without any extra information is expensive in terms of memory and time.

3.c Running informed search algorithms

My heuristic function is maximum of Manhattan distances of the agents from their goals.

- It is admissible because even if there is no wall, the actual cost of moving one point to another won't be less than the Manhattan distance of the two points.
- It is consistent because the step cost is 1 and no such neighbouring states exists such that $abs(h(n) - h(n')) > 1$, making f nondecreasing.

A^* **Algorithm**	**input_0.txt**	**input_3.txt**	**input_4.txt**
Num of steps	9	6	36
Num of nodes generated	487	1,026	4,499,749
Num of nodes expanded	45	7	144,965
Max num of nodes kept in the memory	247	1,869	229,353
Running time (s)	0.035	0.039	76.890

As it can be seen from the table, $A*$ algorithm outperforms uninformed search algorithms with input_0.txt and it finds solutions for input_3.txt and input_4.txt thanks to the information it has. Based on the new information, h it will pick cheapest nodes first and the nodes with higher costs is less likely to be expanded.

I've compared 2 different tie breaker: select newest node open or select the one which has less path cost, g. The latter one found solution for input_4.txt in 100 seconds whereas the first one did in 76 seconds so I kept the first one.

4 Conclusion

I have learned how to implement various searching algorithms that we saw in the class and experimented them with different inputs. I tried to implement everything to make it look like pseudo code and Python's flexibility helped me a lot. I've also used 3rd party libraries to visualize to result, which also helped me while debugging. In conclusion, I'm very happy to had and complete this assignment.