Colton Avila

CS531

5 April 2021

Vacuum Cleaning Agent – Assignment 1

Abstract:

Three agent programs were explored in this paper: a memory-less deterministic reflex agent, a randomized reflex agent, and a memory-based deterministic reflex agent. State memory, randomized actions, and different environments were all topics explored in this experiment. These program agents were allowed five actions: to go forward, turn right, turn left, suck dirt, and turn off. They each had three percepts: a wall sensor, a dirt sensor, and a home sensor. The activation of these percepts determined the actions performed by each agent program. The memory-based deterministic reflex agent cleaned the most squares in the empty room, 100, in 249 steps. The randomized reflex agents performed the worst on average only cleaning 6 squares in the empty room in 27 steps. Based on our measure of performance (number of squares cleaned vs. number of steps) the simple reflex agents performed the best with the highest performance rating of 0.48. This experiment showed how important knowledge of the environment is when constructing agent programs.

Introduction:

The main goal behind this work was to highlight the importance of resource constraints and the resulting performance of agents. Memory-less deterministic reflex agents were designed to only interpret what the percepts feed into the agent program. Random reflex agents were designed similarly – to only take in percept information without having any state memory - but during the decision process it randomly chose different actions based on sensor readings. The determinist model-based reflex agent with a small amount of state memory was able to not only take in percept information but turn off and on a few state variables. These agent programs were modelled in a 10 x 10 room and set to run based off a set of if-then statements. Turning, sucking, checking if they were home, and moving were all moves that these program agents could implement. State memory, randomized actions, and the complications of a limited set of percept information all contributed to the difference in results from these program agents.

Methods:

Each of the agent programs were designed to maximize the number of clean cells in the least number of steps. The simple memory-less deterministic reflex agent was designed with the most basic concepts in mind. Since this agent could only make decisions based off its percepts, it was designed to move forward after it cleaned a tile and take a right any time it perceived a wall. Below is the list of rules the memory-less deterministic reflex agent acted upon. These are listed in order of observation; the first rule to be found true was acted upon.

1. If Home Sensor & Wall Sensor & Dirt Sensor → Clean Square
2. If Dirt Sensor & Home Sensor → Move Forward
3. If Dirt Sensor → Clean Floor
4. If Home Sensor & Wall Sensor → Power Off
5. If Wall Sensor → Turn Right
6. If No Sensors Activated → Move Forward

The randomized reflex agent that chose actions at random based on sensor reading had a similar set of instructions. This agent was ran in this environment 50 different times to explore the different paths the agent could take. However, this one differed that when certain inputs were recorded a set of actions with different probabilities could be implemented. Below is the list of rules by order of observation with the first rule to be found in the list was the one that was acted upon.

1. If Home Sensor & Wall Sensor & Dirt Sensor → Clean Square
2. If Dirt Sensor & Home Sensor → Move Forward
3. If Dirt Sensor → Clean Floor
4. If Home Sensor & Wall Sensor → Power Off
5. If Wall Sensor → 50% Chance Turn Right, 50% Chance Turn Left
6. If No Sensors Activated → 10% Chance Turn Left, 40% Turn Right, 50% Chance Go Forward

The deterministic model-based reflex agent had a similar set of rules as the previous two agent programs. However, since it was able to have a certain amount of state memory three rules were included. It kept track of if it had just turned right at a wall (TR), if it turned left at a wall (TL), and if it was in a currently turning position (CT). Below is the list of rules by order of observation with the first rule to be found true on the list being the one that is acted upon. The "Turned Right" state variable is automatically set to true since the first turn of the machine will be to the right.

1. If Home Sensor & Wall Sensor & Dirt Sensor → Clean Floor
2. If Dirt Sensor & Home Sensor → Move Forward
3. If TR & Dirt Sensor & CT → Turn Right, TR = False, TL = True, CT = False
4. If TL & Dirt Sensor & CT → Turn Left, TL = False, TR = True, CT = False
5. If Dirt Sensor → Clean Floor
6. If Wall Sensor & TL → Turn Left, CT = True
7. If Wall Sensor & TR → Turn Right, CT = True
8. If No Sensors Activated → Move forward

These 3 state variables allowed for the agent to turn across two squares and not simply turn on one. The first room these agents moved around in was empty with the second having walls with a single door between the rooms. Below are two ascii figures of the rooms, respectively.

```
[[9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]]
```

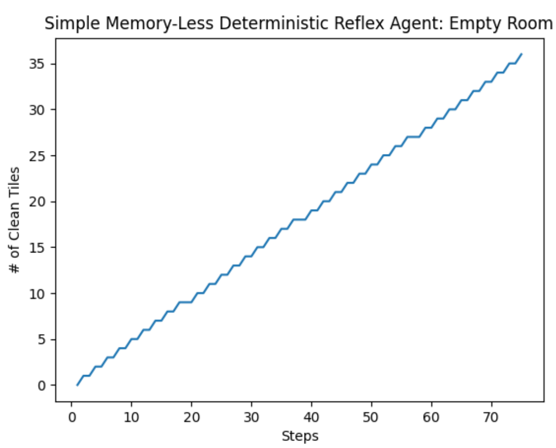
```
[[9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 9. 9. 0. 9. 9. 9. 9. 9. 0. 9. 9. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 0. 0. 0. 0. 0. 9. 0. 0. 0. 0. 0. 9.]
 [9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]]
```

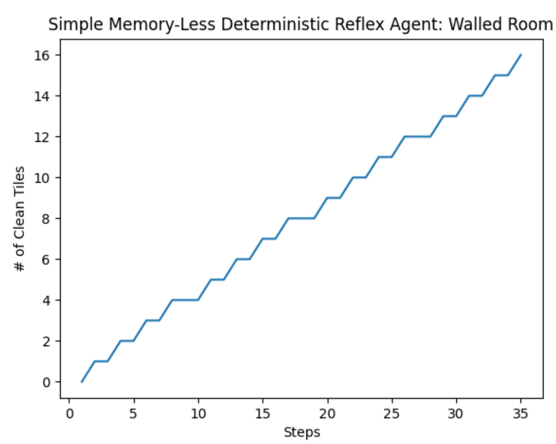
Results:

Below is a table showing the type of agent, the environment, their performance (number of clean squares vs. number of steps), the total number of squares cleaned, and the number of steps taken.

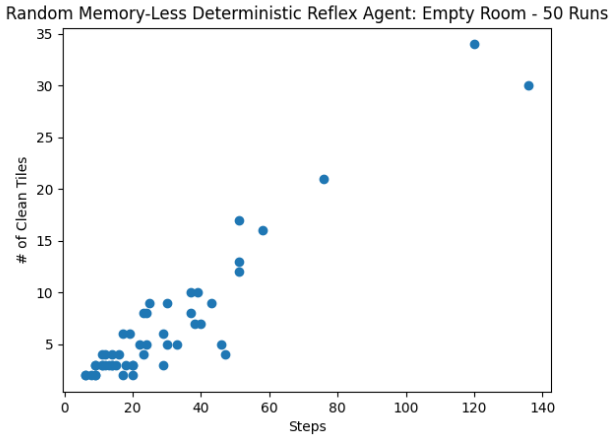
Agent Type	Environment	Performance	Squares Cleaned	Steps Taken
Simple Reflex	Empty Room	0.48	36	75
Simple Reflex	Walled Room	0.46	16	35
Random Reflex	Empty Room	0.24	6	248
Random Reflex	Walled Room	0.25	7	Inf.
Memory Reflex	Empty Room	0.4	100	27
Memory Reflex	Walled Room	0.072	36	23

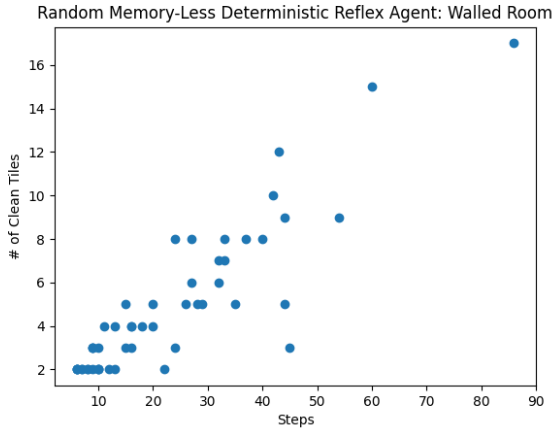
The simple reflex agents performed the best, with the deterministic model-based reflex agent following, and the randomized reflex agent performing the worst. In the graphs below we see the results of the runs. The simple reflex agent moved and cleaned the environment quickly, but it was unable to clean all squares.



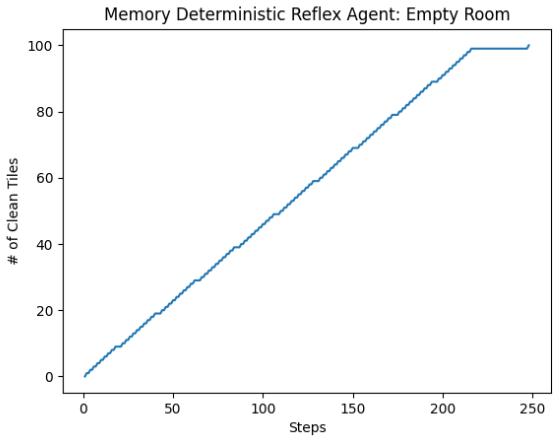


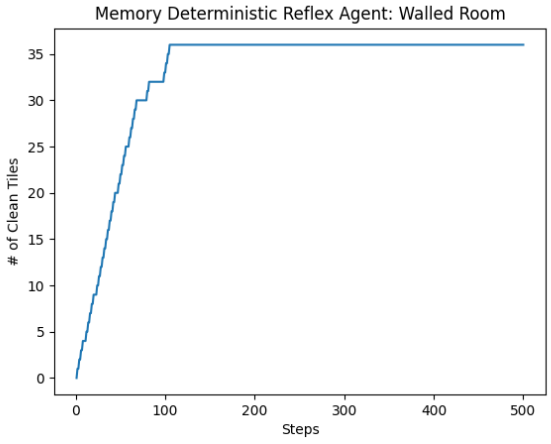
We see that on average the randomized agent did not make it very far before returning home in both environments. On some occasions it cleaned over 25 in the empty room and 12 in the walled room, but these cases are outliers.





The deterministic model with state memory cleaned all 100 squares in the empty room, but since it had to travel back to the home square, the performance measure decreased. We see that this model also failed in the walled environment after ~ 100 steps. It became stuck in the top-left room of the environment and was unable to return home.





Discussion:

The best possible performance achieved by the simple reflex agent was 36 squares in the empty room and 16 in the walled room. Since it lacked the ability to retain state information, it did not know how to turn without hitting a wall. This meant that the agent would simply move around the perimeter of the environment, or the first square area in the case of the walled room.

The random agent performed poorly. This could be remedied by perfecting the probabilities of the actions and a large enough sample size. If we ran this agent program 1,000,000 times, we may see that it does a perfect run. We did not see it clean 90% of the room in 50 trials. The cost of randomness is that when the agent program may be in a column or row

that is dirty, it might turn back to clean tiles or simply sit in one spot without moving for quite some time. The benefits however allow this agent to turn and move without the need of detecting walls and allows it to find new parts of the environment that are dirty.

The memory-based deterministic agent performed excellently in the empty environment. It was able to clean all 100 tiles in 248. However, this agent fails in the walled room. It passes through the door on the left side of the environment and becomes trapped in the top-left room. By increasing the amount of state memory this agent would be able to check if it crossed the threshold between rooms and possibly implement a new program function.

Deterministic agents excel in performance when the environment is fully known. Random agents would perform better in a room where unknown obstacles are present. An ideal agent that would deal with unknown obstacles would implement a random action when presented with a situation where the state variables and percept information present an unknown situation to the program agent.

I learned a lot from this experiment. I was surprised at how efficient I could make the memory-based agent clean the empty room using only three state variables. In this same vein I was surprised how much of an impact the walls made on this agent. I expected different results from the randomized agent, but with actions based on probabilities and such a low sample size I was not surprised by the results. It is exciting to think about how efficient a mixed randomized / memory-based agent might perform in situations like these.