Artificial Intelligence CS 531: Programming Assignment 1

Vivswan Shitole and Roli Khanna

Oregon State University

Abstract. This is the documentation of the first programming assignment in CS 531 Artificial Intelligence (Winter 2019) class. We introduce the problem with the expected goal and performance measure. This is followed by a formal description of the different agents developed to solve the problem (optimize expected performance). Next we describe the experimental setup involving a description of the environments and their representations. Then we present the obtained results and plots. Finally we conclude with a discussion of the follow-up questions asked in the problem statement.

Keywords: Artificial Intelligence · Vacuum cleaner · Intelligent Agents.

1 Introduction

A summary of the problem statement is as follows: The vacuum cleaner environment consists of 10 X 10 grid, which needs to be cleaned. An agent can receive only 3 percepts from the environment: (a) presense/absence of wall, (b) presence/absence of dirt and (c) presence/absence of home cell. This is so because the agent is equipped with only 3 sensors: (a) a wall sensor which is equal to 1 if the agent has a wall right in the front and 0 otherwise, (b) a dirt sensor which is equal to 1 if the current square contains dirt, and (c) a home sensor which is equal to 1 if the agent is home (the starting location). The agent can take one of the five actions at a time: go forward, turn right by 90 degrees, turn left by 90 degrees, suck up dirt, and turn off.

The aim of the assignment is to design and implement 3 different vacuum-cleaning agents whose goal is to clean the environment floor:

- A simple memory-less deterministic reflex agent.
- A randomized reflex agent that can choose actions randomly based on sensor readings.
- A deterministic model-based reflex agent with a small amount (2 to 3 bits) of memory that represents its "state." The actions are based on its current state bits as well as the current percept.

The performance is measured by the number of clean cells versus the number of actions taken by the agent. The more the number of clean cells achieved by the lesser the number of actions, the better is the agent's performance.

2 Description of Agents

2.1 Simple Reflex Agent

The simple reflex agent is a memory-less deterministic reflex agent. It uses only the current percept with a fixed set of if-then rules to decide the action to take. It does not base its decisions on the percept sequence since its memory-less and has no way of maintaining a track of the percept sequence using an internal state. Moreover, since the if-then rules are fixed, its actions are deterministic: it has a fixed mapping from the current percept to the action to be taken.

For the given problem statement, our proposed agent function for the simple reflex agent can be described as follows: If the current cell is dirty, the agent sucks the dirt. If the current cell is clean, the agent moves forward unless there is a wall in front. In case of a wall, the agent always turns right. Thus the agent traverses (and cleans) only the boundary cells (cells adjacent to the wall). We later argue (in section 3) that this is the optimal agent function for a simple reflex agent in the given environment.

The agent program for the specified agent function is as follows:

Algorithm 1 Pseudocode for Simple Reflex Agent

```
Input: current percept {dirt, wall, home}
if dirt then
    action = suck
else if wall then
    action = turn right
else
    action = forward
return action
```

2.2 Random Reflex Agent

The random reflex agent is a memory-less non-deterministic reflex agent. It is similar to the simple reflex agent in the sense that its memory-less (no internal state). Hence it uses only the current percept as input and not the percept sequence. But its does not have a fixed percept to action mapping. Instead, for a given percept, it has a probability distribution over the set of possible actions. Hence its actions are stochastic, with each action having a certain probability of being chosen.

For the given problem statement, our proposed agent function for the random reflex agent can be described as follows: The agent selects each of the 5 actions with an equal probability of 0.2. Hence it has equal probability to select any of the 5 actions, irrespective of the percept. We later argue (in section 3) that

a better agent function can be obtained by modifying (tuning) the probability distribution over actions.

The agent program for the specified agent function is as follows:

Algorithm 2 Pseudocode for Random Reflex Agent (Untuned)

```
Input: current percept {dirt, wall, home}
x = generateRandomNumber(0,10)
if x < 3 then
    action = turn right
else if x > 2 and x < 5 then
    action = turn left
else if x > 4 and x < 7 then
    action = suck
else if x > 6 and x < 9 then
    action = forward
else
    action = turn off
return action
```

2.3 Model Based Agent

The model-based agent is a deterministic agent with a memory to maintain an internal state. The internal state can be used to model the percept sequence and the effect of agent's actions on the sequence. Hence the model-based agent uses the current percept as well as the current internal state as the inputs to its agent program. The agent program is a fixed set of if-then rules which decide the action to take. Hence the actions are deterministic, but they are conditioned on both the current percept and the current internal state. Finally, the internal state is updated depending on the next percept generated by the environment after the agent has taken the chosen action.

For the given problem statement, our proposed agent function for the model based agent can be described as follows: The agent maintains an internal state using a memory of 3 bits (bit0, bit1, bit2). The first two bits (bit0 and bit1) are used to maintain a track of the direction the agent is currently facing (00: North, 01: East, 10: South, 11: West). The third bit (bit2) is used as a flag to enable the agent to take a single step in the east direction whenever the agent encounters a wall. Hence the agent cleans the first column of the grid by going in the north direction until it reaches the top wall. Then it takes a single step in the east and cleans the adjacent column by going in the south direction, until it reaches the bottom wall. It continues this zig-zag motion, cleaning all the cells of the open grid.

The agent program for the specified agent function is as follows:

Algorithm 3 Pseudocode for Model Based Agent

Input: current percept {dirt, wall, home}
Initial state: [False, False, False]

```
if dirt then
    action = suck
else
    if wall then
        if not state[0] and not state[1] then
            action = turn right
            state[0] = False
            state[1] = True
            if state[0] and not state[1] then
                action = turn left
                state[0] = True
                state[1] = True
            else
                action = turn right
            state[2] = True
        else
            if state[2] then
                action = forward
                state[2] = False
            else
                if not state[0] and state[1] then
                    action = turn right
                    state[0] = True
                    state[1] = False
                else if state[0] and state[1] then
                    action = turn left
                    state[0] = False
                    state[1] = False
                else
                    action = forward
```

3 Experiments and Inferences

3.1 Environment: 10X10 grid

This is the first kind of environment to be used to gauge the performance of the 3 agents. It is in the form of a 10X10 grid. The grid is open in the sense the walls exist only at the outer boundary of the 10X10 square. At time t=0, all the 100 cells of the grid are dirty. The agent starts from the bottom-left cell of the grid, facing upwards.

An analysis of the performance of the 3 agents in this environment is given as follows:

Simple Reflex Agent In our experiment of a simple reflex agent with an open grid, the agent simply marks and cleans the periphery of the grid. This occurs primarily because the agent is memory-less and maintains no internal state. Thus it has no way of differentiating between cells except when it runs into a wall. Moreover, it doesn't have the motivation or logic to turn when certain cells are already visited. Hence the optimal behavior it can achieve is cleaning all the cells adjacent to the walls. The performance graph for a simple agent plateaus after 75 iterations / action steps.

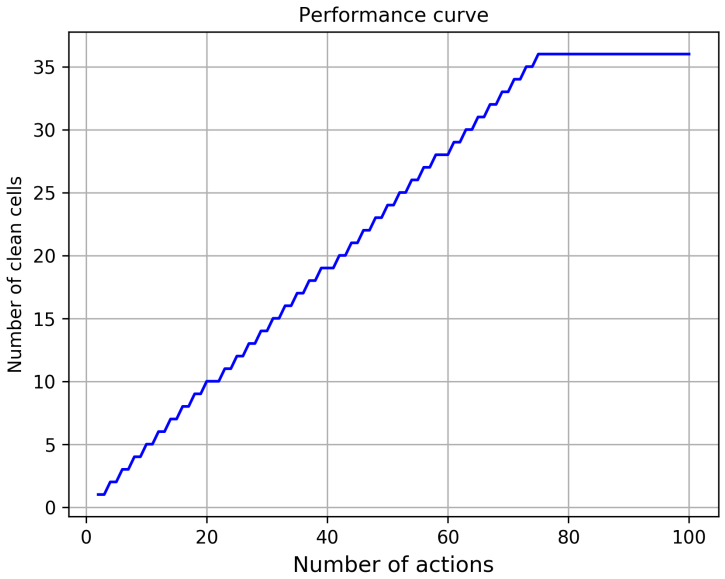

Fig. 1: Performance curve for Simple Reflex Agent.

Random Reflex Agent The experiments of the random reflex agent, in contrast, fare better than in the case of the simple reflex agent. This is primarily because the actions taken by the agent are probabilistic and random in nature, it might and will eventually cover cells which it has not cleaned previously. One major drawback of this approach, we find, is that revisits cells unnecessarily, due to lack of internal state.

To improve the performance of the random agent, we can tune the probabilities associated with each action the vacuum cleaner takes: such as increasing the probability associated with the 'suck' action, and reducing the probability associated with the 'turn off' action. We observe a note a significant improvement in performance with the tuned probability values. The probability distribution was modelled to account for the 'suck' and 'forward' actions to each be 40%,

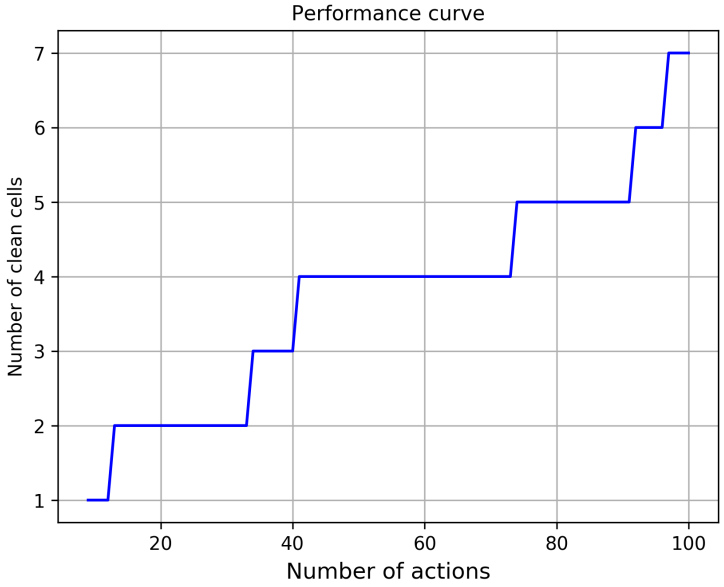

Fig. 2: Performance curve for Random Reflex Agent: 100 iterations

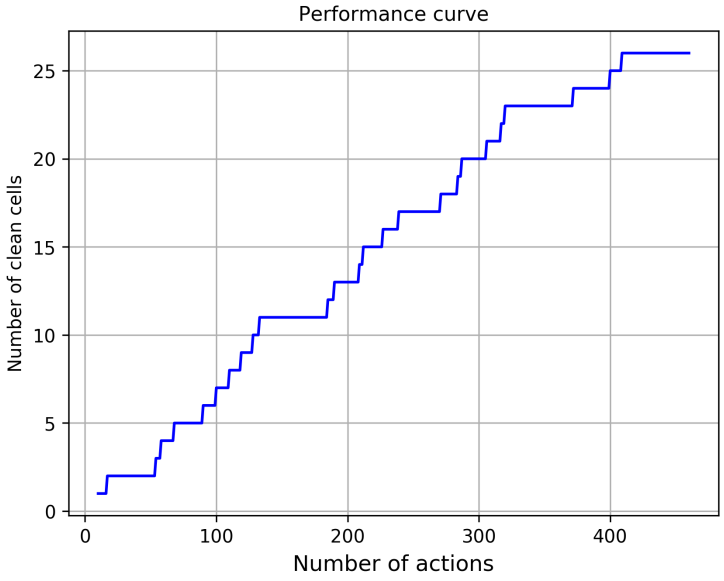

Fig. 3: Performance curve for Random Reflex Agent: 90% of the grid: 460 iterations

while the 'turn left' and 'turn right' actions were each allocated 10% probabilities each. On experimenting with this model, the agent cleans 14% of the cells, compared to the un-tuned recorded value of 7%. When allowed to completely clean the entire grid, the agent takes nearly 4900 (untuned) iterations to successfully operate.

We note that the performance of the random agent fares stochastically for 100 iterations, as a result of which it is difficult to compare it with the simple agent. However, once allowed to run till it cleans 90% of the cells, we observe that it takes 498 iterations to complete its operation, which is significantly better than the simple agent.

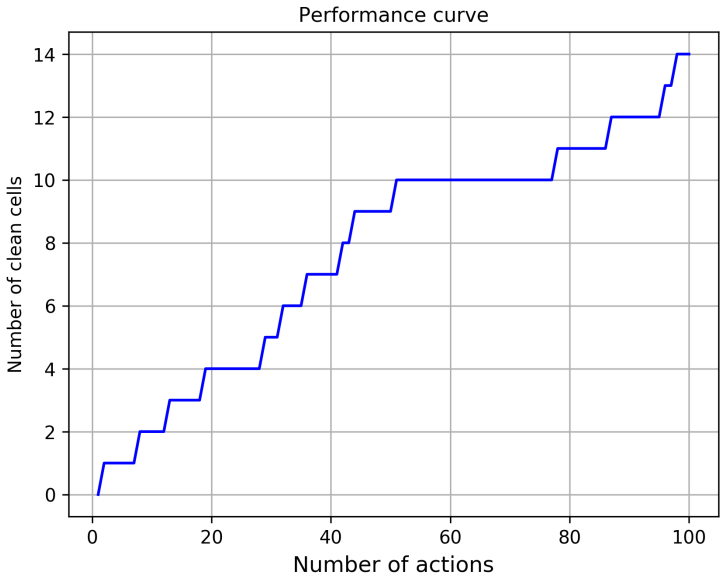

Fig. 4: Performance curve for Random Reflex (tuned) Agent: 100 iterations

On an average, the agent takes 1600 iterations (after tuning the probabilities) to clean the grid entirely. The 45 best performing instances are listed in Table 1.

Model based Agent Owing to the maintenance of an internal state consisting of three bits, the model based agent finds itself at advantage compared to the other two models. It is specifically programmed to turn at the cells where it encounters 'roadblocks' or 'walls', and keeps track of the direction it needs to head in. This embedding of knowledge lets the agent navigate better through the environment.

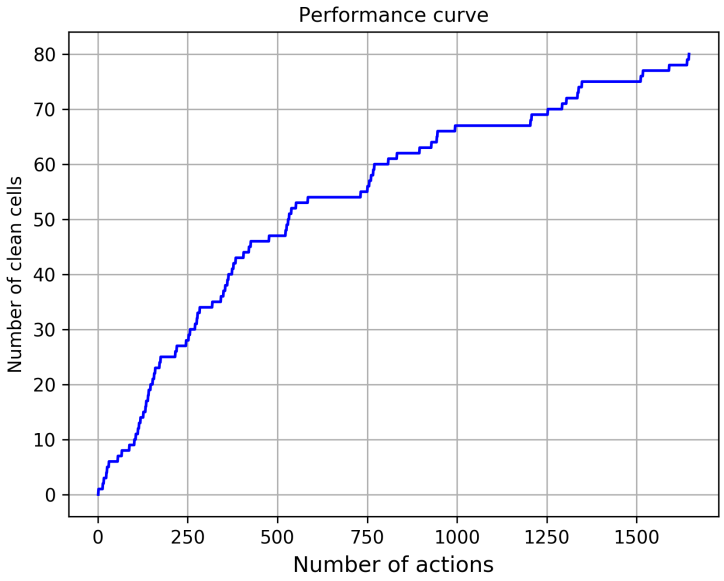

Fig. 5: Performance curve for Random Reflex (tuned) Agent till it cleans 100% of the grid: 1646 iterations

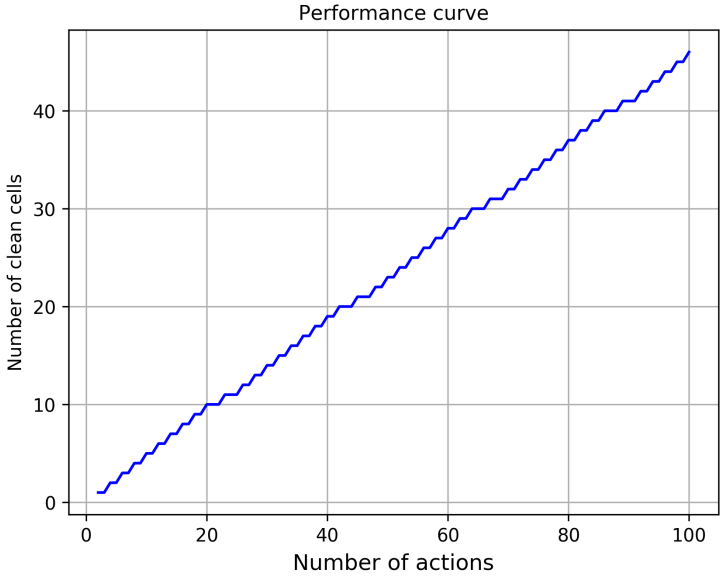

Fig. 6: Performance curve for Model based Agent: 100 iterations

Table 1: Random Reflex Agent Performance of 45 best trials

Instance	Number of actions
Instance 1	1314
Instance 2	1363
Instance 3	1414
Instance 4	1425
Instance 5	1436
Instance 6	1469
Instance 7	1477
Instance 8	1495
Instance 9	1520
Instance 10	1522
Instance 11	1550
Instance 12	1556
Instance 13	1571
Instance 14	1608
Instance 15	1615
Instance 16	1616
Instance 17	1622
Instance 18	1642
Instance 19	1644
Instance 20	1665
Instance 21	1666
Instance 22	1676
Instance 23	1684
Instance 24	1691
Instance 25	1716
Instance 26	1725
Instance 27	1726
Instance 28	1740
Instance 29	1755
Instance 30	1760
Instance 31	1776
Instance 32	1777
Instance 33	1806
Instance 34	1815
Instance 35	1818
Instance 36	1851
Instance 37	1860
Instance 38	1865
Instance 39	1866
Instance 40	1869
Instance 41	1870
Instance 42	1870
Instance 43	1894
Instance 44	1895
Instance 45	1904

We observe that the model based agent cleans nearly 60% of the cells in 100 iterations, which is a significant improvement in performance. Once allowed to run till it cleans 90% of the cells, the agent takes 198 iterations to completion, which we find is again a significant improvement.

To further improve the model based agent, in the case of infinite resources and consequently infinite memory, each location the agent visits and remains pending can be maintained as an internal of the agent. By doing so, the agent can be learn actionable insights as well as perform more efficiently, as opposed to working strictly on the design of if-then rules for each cell.

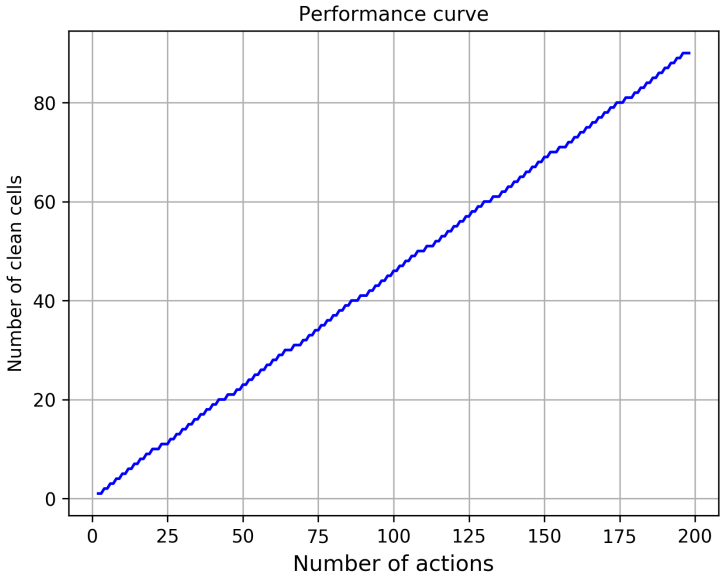

Fig. 7: Performance curve for Model based Agent till it cleans 90% of the grid: 198 iterations

3.2 Environment: 10X10 grid, divided into 4 rooms

This is the second kind of environment to be used to gauge the performance of the 3 agents. It is in the form of a 10X10 grid but the grid is divided into 4 rooms, each of size 5X5. Thus, in addition to the boundary walls, it has a vertical wall at the horizontal center and a horizontal wall at the vertical center. There are four doors, one between each pair of adjacent rooms to allow agent to enter a different room. Since there is no constrain on the position of these doors, we position them intelligently according to the nature of the agent such that it facilitates the agent to clean as many cells as possible. At time t=0, all the 100

cells of the grid are dirty. The agent starts from the bottom-left cell of the grid, facing upwards.

An analysis of the performance of the 3 agents in this environment is given as follows:

Simple Reflex Agent For the case of simple reflex agent in the 4-room grid, in absence of doors, the agent would just go along the wall of one of the 4 rooms. Hence, the optimal performance would be achieved when the agent cleans all the boundary cells of all 4 rooms. This was achieved by intelligently placing the 4 doors near center of the grid, so that the agent spirals along all the boundaries of all the rooms.

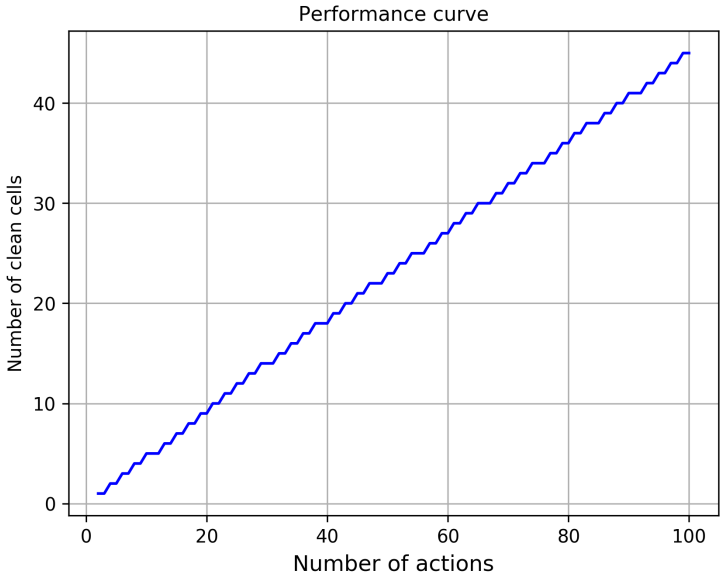

Fig. 8: Performance curve for Simple Reflex Agent: 100 iterations

The performance graph is increasing as long as the agent is cleaning new cells along the room boundaries. After that, it has cleaned the maximum number of cells possible for the simple reflex agent. Hence the performance curve plateau after increasing steadily for 143 iterations.

Random Reflex Agent The experiments of the random reflex agent, again, fare better than in the case of the simple reflex agent. The agent selects one of the 5 actions stochastically, hence eventually covering all cells which it has not cleaned previously. Though the agent's performance is far from optimal with an

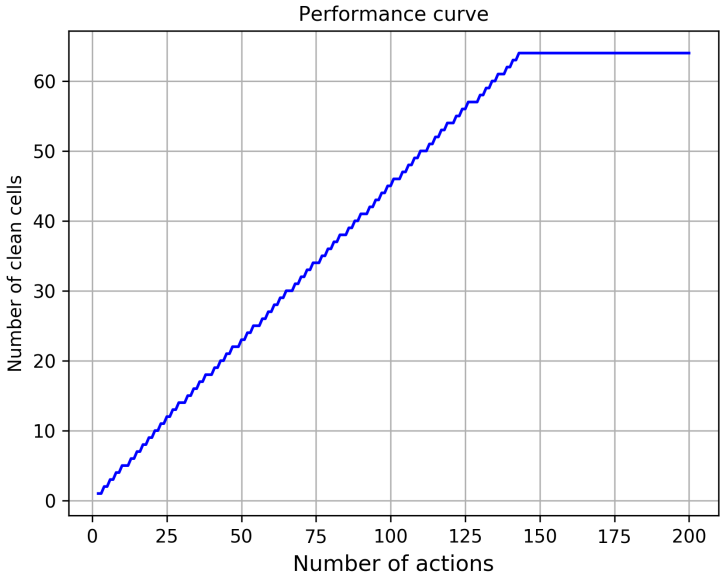

Fig. 9: Performance curve for Simple Reflex Agent: 200 iterations

increased unpredictability in its behaviour due to the room walls, it is able to clean all the cells given enough time / action steps.

To improve the performance of the random agent, we can tune the probabilities associated with each action the vacuum cleaner takes: such as increasing the probability associated with the 'suck' action, and reducing the probability associated with the 'turn off' action.

We observe that the performance curve is an increasing curve but the curvature is concave. This is expected since when acting randomly, the probability of visiting a dirty (unvisited) cell decreases. We see that the random agent cleans 90% of the cells in around 7500 iterations on average.

Model based Agent The performance of the model based agent is the best among the three agents. Even though the random agent is able to clean all the cells of the 4-room grid, it takes way more action steps to achieve that. Our proposed model based agent is able to clean 80% of the cells, but it takes far less action steps than the random agent. This is because the model based agent takes far greater number of rational actions than the random agent by virtue of its internal state.

We observe that the model based agent cleans 80% of the cells in 280 action steps. The performance curve is increasing, except for some plateaus in between. This is because the agent's policy is such that it ends up circling withing a room for some time before it encounters the door to the other room.

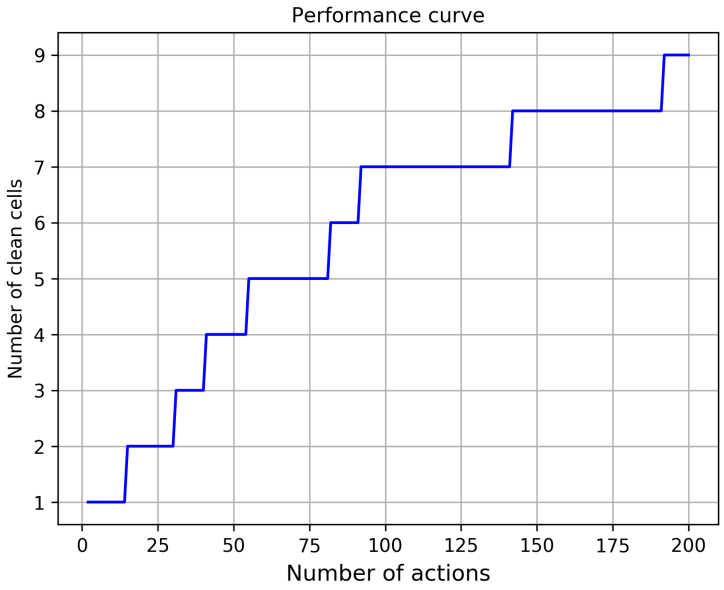

Fig. 10: Performance curve for Random Reflex Agent: 200 iterations

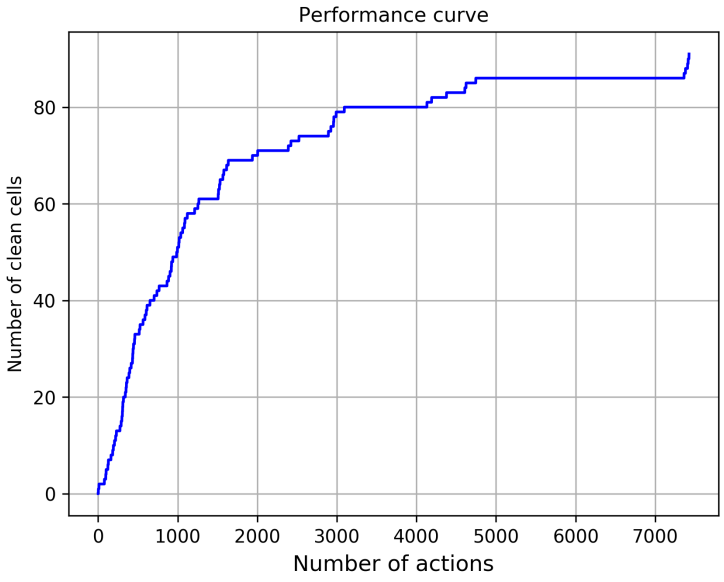

Fig. 11: Performance curve for Random Reflex Agent to clean 90% of the rooms

Fig. 12: Performance curve for Model Agent to clean 80% of the rooms

4 Conclusion and Discussion

We developed 3 different agents and analyzed their performance on 2 different environments. We found that the simple reflex agent is able to clean a very limited number of cells. The random agent is able to clean all the cells in both the environments given long enough time. But the performance is poor as it takes way too many actions to clean all the cells. The model based agent performes the best among the 3 agents, requiring the least number of actions to clean most number of cells. Its able to clean all the cells in the open grid environment and 80% of the cells in the 4-room environment.

The surprising things we learnt from this assignment are that:

- Random actions can be improve the performance majority of the time
- Increasing the memory for internal state can improve the performance exponentially.

We also propose that the policies undertaken by the agent can be learnt with time in a given setting. The use of intelligent decision mapping, instead of brute-forced if-then rules, would better equip us to tackle complicated problem spaces such as environments with polygon obstacles.