AI-531 Vacuum Cleaner Agent

Naren Digambar Khake
934451067

Devashree Sanjay Bhavsar
934452998

Abstract

In the field of artificial intelligence, intelligent agents play a crucial role in performing specific tasks autonomously. These agents are imbued with the capacity to perceive their surroundings through sensors, process information, and act upon it using actuators. At their core, intelligent agents are driven by a set of programmed behaviors that enable them to adapt, make decisions, and execute tasks with varying degrees of independence and complexity. Using intelligent agents, tasks like vacuum cleaning can be done efficiently. This paper aims to develop three different vacuum-cleaning agents using artificial intelligence techniques. The first agent is a simple memory-less deterministic reflex agent, which makes decisions solely based on current sensory information without considering past experiences. The second agent is a randomized reflex agent that incorporates randomness in its decision-making process, choosing actions randomly based on sensor readings. The third agent is a deterministic model-based reflex agent with a small amount of memory. At last, we will be showcasing analyzed outputs of our implementation.

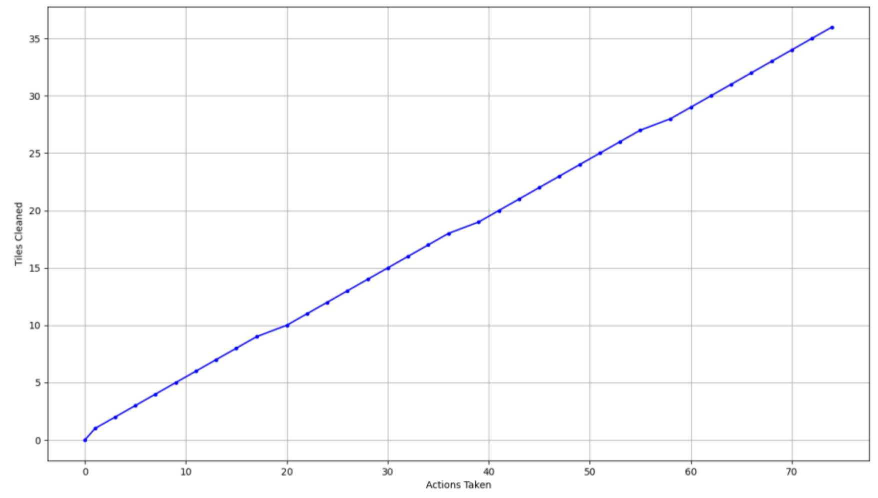
1. Introduction:

In this experiment we examine how different vacuum cleaner agents perform in cleaning a 10x10 grid environment. The goal of our study is to understand how different agent architectures and environmental factors affect cleaning performance. We aim to compare three types of agents: a simple reflex agent without memory, a randomized reflex agent that can choose actions randomly, and a deterministic model-based reflex agent with a small amount of memory. We have conducted experiments in two different grid environments, the first environment is an empty grid of 10x10, while the second environment consists of four rooms separated by doors on a 10x10 grid. We will be considering the following sets of actions - move, turn right, turn left, clean, and turn off. The agent can sense only 3 things – Wall ahead, Tile Dirty, Is at Home Tile. Considering these conditions we have implemented the agent.

2. Results:

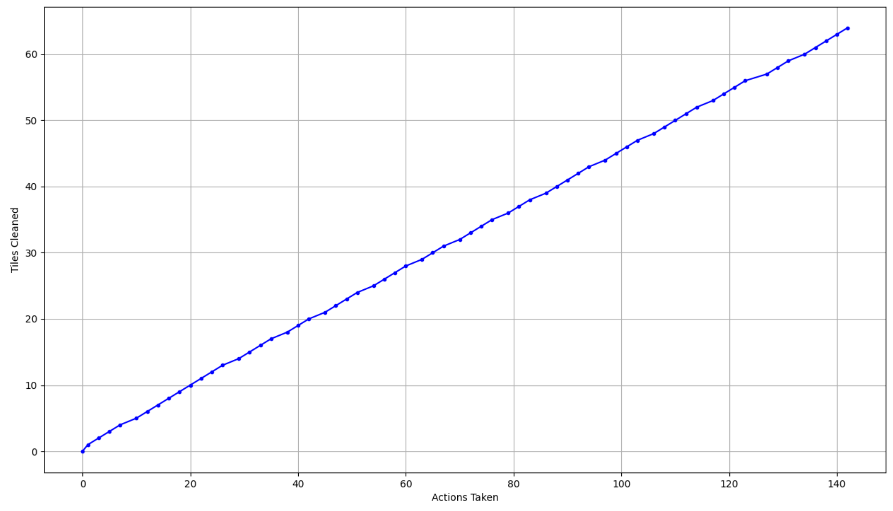
2.1 Simple Reflex Agent With no walls:

Total cleaned tiles = 36 and Total action taken = 78.



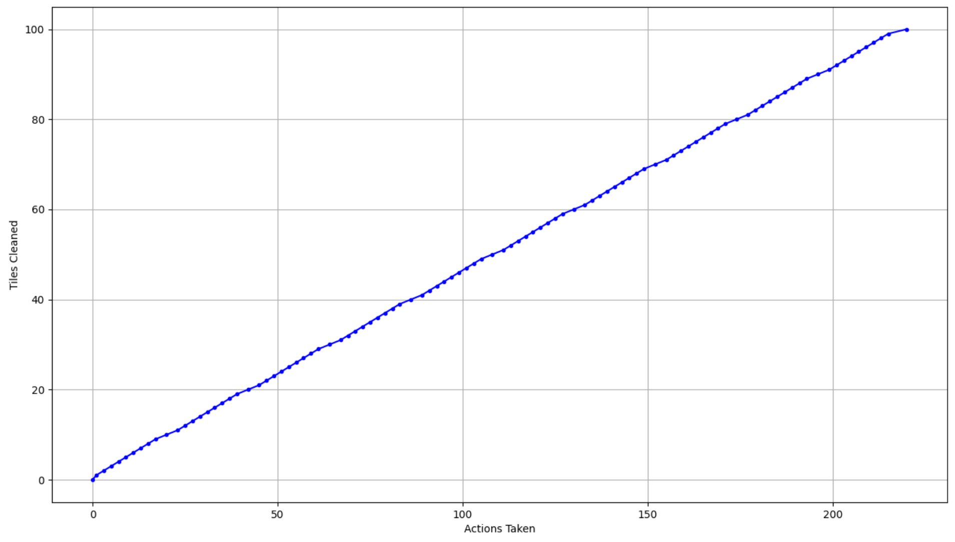
2.2 Simple Reflex Agent With 4 rooms and walls:

Total cleaned tiles = 64 and Total action taken = 146.

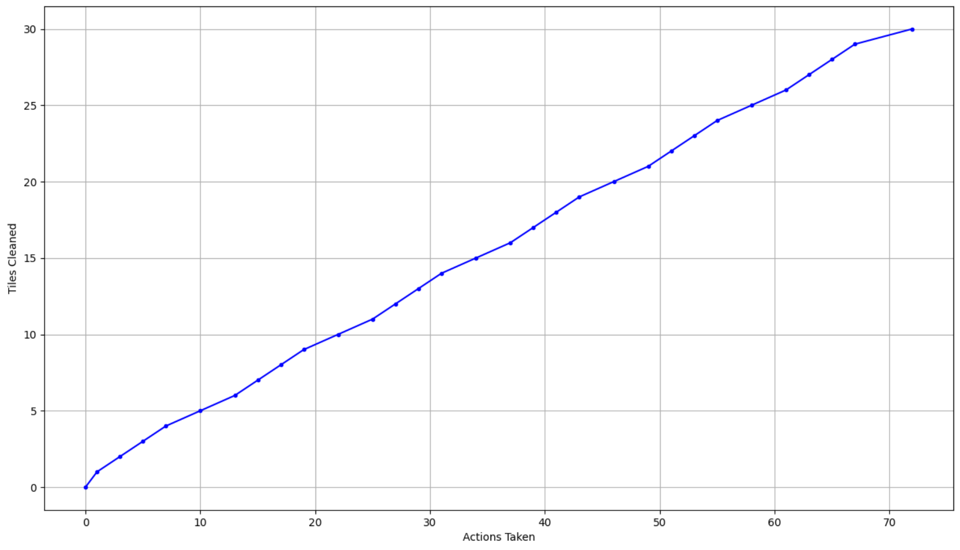


2.3 Memory based agent with no wall:

Total cleaned tiles = 100 and Total action taken = 232.

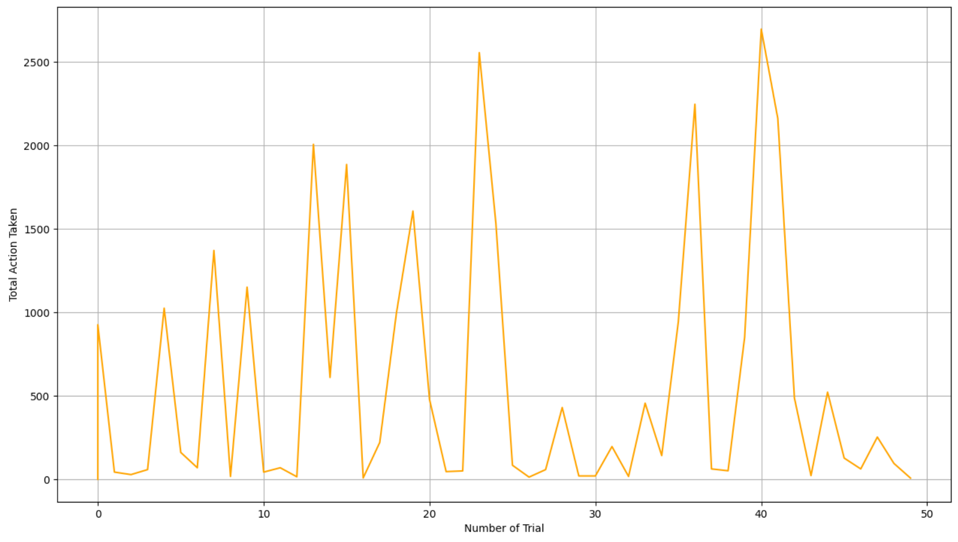


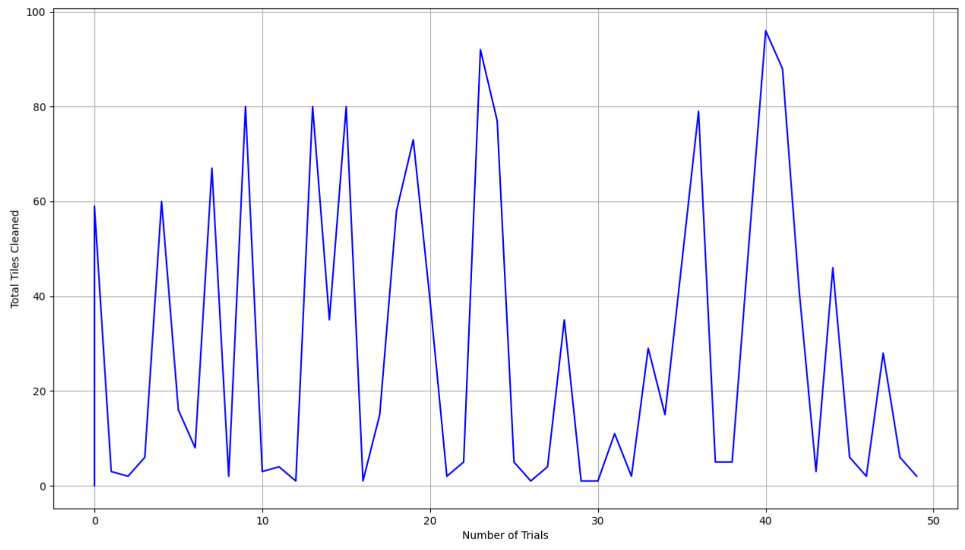
2.4 Memory based with 4 rooms and walls:



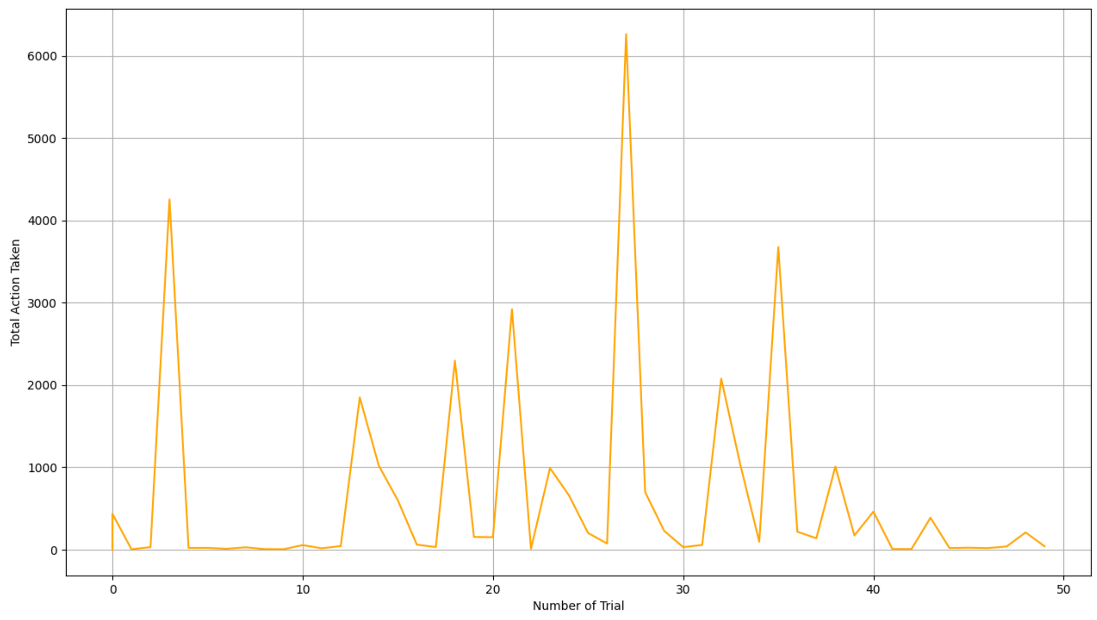
2.5 Random Simple Agent (original) with no walls (50 trials):

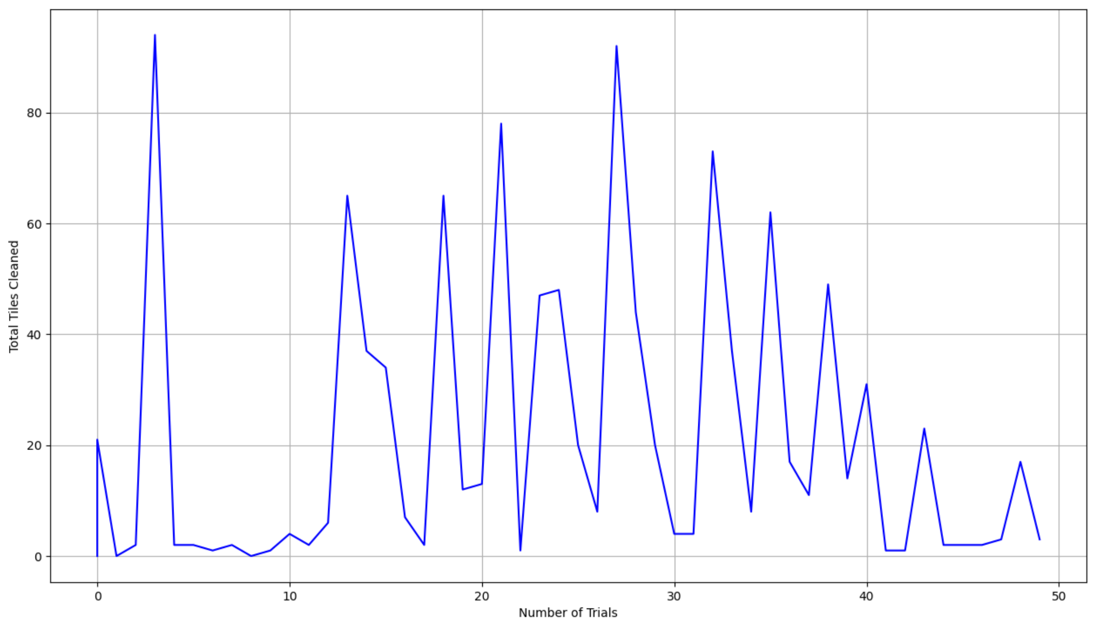
This is the random agent where the vacuum cleaner turns off when it reaches the home tile.





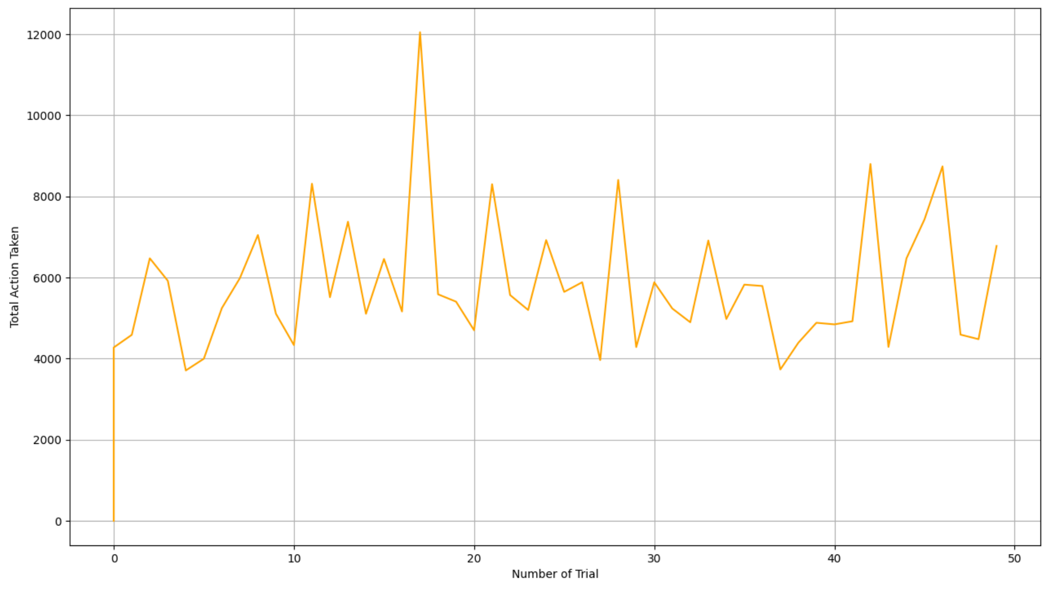
2.6 Random Simple Agent (original) with 4 rooms and walls (50 trials):



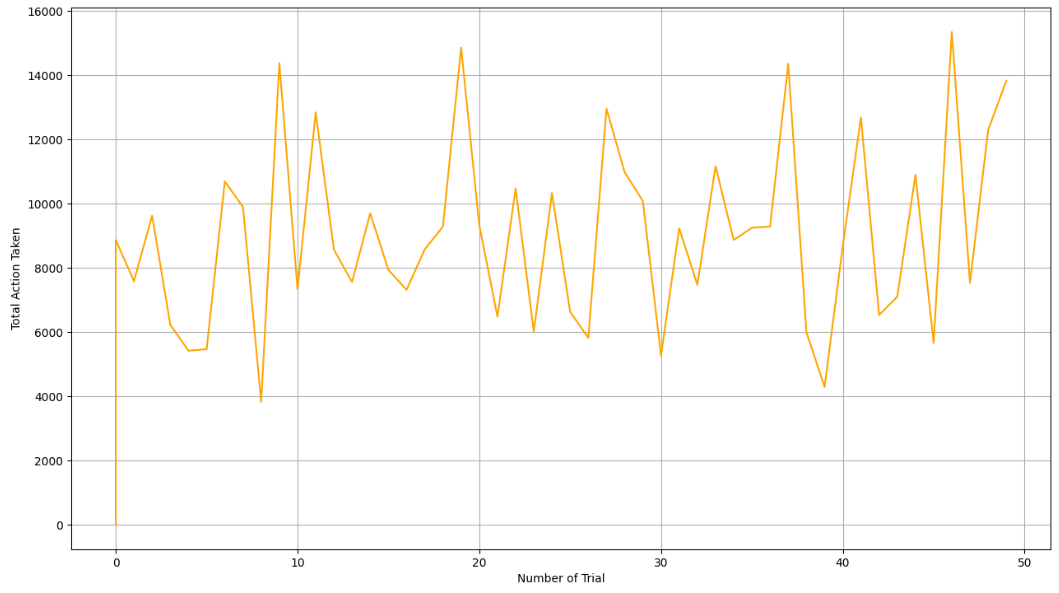


2.7 Random Simple Agent (modified) with no walls (50 trials):

This random agent will go on till it cleans all the tiles in the environment.



2.8 Random Simple Agent (original) with rooms and walls (50 trials):



In the above results and graph we can see that the deterministic agents mostly have a linear graph. This makes sense as it turns only when there is a wall. At wall we need to take an extra action than what we usually need to take. Where as in memory based model, the agent takes 2 extra action when it reaches the end and take a U-turn to go other way. In case of randomized agent, the graph is not exactly linear. This is because it is not deterministic and we cant tell what the next action is.

When we run 50 trials for the random in both environment, we obsereve that in the original random case the agent mostly performs very less action for most of the trials. There are some trials where the number of action is very large. As there are less action, the number of tiled cleaned in this case is also very less which can be seen in the graph.

In case of the modified random agent, It does not stop till all tiles are cleaned. Hence, the number of action needed in this case is significantly large than the original model. It almost reaches 12000 actions in some trials but it guarantees that all tiles are cleaned. When we add walls to this setup, the number of trials increase even more as the walls restrict it to randomly move. It needs to enter through door and randomly getting there is less likely, which causes the increase in number of actions. The average of 45 best trials for cleaning all the tiles is 10547 actions.

3. Discussion:

3.1 Describe the idea behind the design of each of your agents. Use diagrams and English as appropriate.

a. **Simple Deterministic Agent:**

For this agent, we decided to start from the initial position, the sensor checks for wall first and makes a right turn if the tile is dirty. If it hits a wall and the tile is clean it turns left. As these are input from the sensor, we are not using extra state memory for the simple agent. This approach is helpful with the environment with walls. We are assuming that the vacuum cleaner can only turn off at home tile. So if we did not sense the wall and dirt and then decide the turn, it might get stuck in a loop. In our case, if it gets stuck in a loop inside some room, it will trace back the path (it does not keep track of past percepts, but just the order of turns is reversed. So, it can always come back to the home tile and turn off. The trade off if that there are many actions in this kind of agent. It depends on the designer if they want the vacuum cleaner to wander forever or take extra action and come back home.

b. <u>Random Simple Agent:</u>

For the random agent, we used two approaches. In the first approach, we can only turn off the vacuum cleaner when it reaches the home tile (irrespective of how many tiles it cleans). In the second approach, the agent or vacuum cleaner stops cleaning when it's cleaned all the tiles.

We designed it in such a way that, if it senses a wall, it will choose a random action between turn left, turn right, clean. Whereas if it does not sense a wall, it will choose between 4 actions (left turn, right turn, move ahead, clean). It will perform the clean action irrespective of the tile being clean or dirty (can be modified to avoid cleaning unnecessarily). So, we cannot determine what the vacuum cleaner will do.

c. <u>Memory-based Reflex Agent:</u>

For the memory based agent, we used 2 memory bits for the state. The first bit counted the number of right turns made by the agent and the second bit for number of left turns made. After the agent makes 2 turns the bit resets to 0. So this logic helps us to make the vacuum cleaner to move in zigzag pattern. This helps us to clean the entire room in no wall environment. Other than that, it is mostly similar to a simple reflex based agent.

3.2 Describe all your agents as if-then rules.

a. <u>Simple Deterministic Agent:</u>

```
While VacuumOn:
    If senseWall:
            If Dirty then TurnRight
            Else TurnLeft
    If isHome Then Vacuum off
    If Dirty Then Clean
    Else MoveAhead
```

b. Random Simple Agent:

```
While All Tiles are Clean:
    If senseWall then choose random between (left,right,clean)
    Else choose random between (left,right,clean,move ahead)
    If isHome then Vacuum off
```

c. Memory-based Reflex Agent:

```
While Vacuum on or actions <10000:
    If senseWall:
        If Dirty:
            If [no. of right turn]==2 then LeftTurn
            Else RightTurn
        Else:
            If [no. of left turn]==2 then RightTurn
            Else LeftTurn
    If isHome then Vacuum Off
    If Dirty then Clean
    Else MoveAhead
```

3.3 What is the best possible performance achievable by the simple reflex agent in the two environments? What prevents it from achieving the goal of cleaning the room perfectly in each case?

In environment with not walls, the simple agent cleans all the tiles along the edge of the environment. The result we get is, Total cleaned tiles = 36 and Total action taken = 78. If we consider the Performance standard to be Total Cleaned / Total action, then we get performance = 0.46.

In environment with 4 rooms and doors, it depends on how the walls are placed and especially on where the doors are located. If we place the door in the optimal position, the simple agent cleans the edges of every room and returns home. Total cleaned tiles = 64 and

Total action taken = 146. Hence, performance is 0.43. It cleaned more than the earlier environment. But if we randomly place the doors and the walls, it would not give such optimal performance and will also clean less tiles than the earlier environment.
As this is a deterministic agent, it performs action depending on what the percept is. So if the tile is dirty it cleans. If the tile is clean, it moves forward. If it hits a wall, it turns. In this case it will hit the wall only 4 times in environment with no wall. Hence cleaning only the edges. Similar case with environment with wall and 4 rooms. It will only turn when it encounters a wall, hence cleaning only the edges of the room. If the door is not placed in corner of the room's wall, it will not be able to exit the room. Hence placement of door is very essential in this case.

3.4 How well does the random agent perform? Tune any parameters of the random agent to improve its performance. Give a table showing the number of actions it took to clean 90% of the room for each trial. What is the average of these numbers for the best 45 trials? What are the costs and benefits of randomness in agent design?

The random agent did not perform very well with the original design. It hardly cleaned all the tiles and took a lot of action. When the parameter of it not turning off at the home tile is changed, it is able to clean all the tiles. Below is a table for number of action needed for 90% of the tiles cleaned.

Trial	Actions
1	2031
2	2193
3	2259
4	2992
5	2985
6	2188
7	2529
8	2456
9	3484
10	2698
11	2296
12	2910
13	3162
14	2686
15	2325
16	3039
17	2809

18	2871
19	1927
20	2395
21	2188
22	2328
23	3065
24	2998
25	2600
26	1985
27	2557
28	2903
29	2160
30	2904
31	3468
32	3106
33	2474
34	4100
35	2433
36	2501
37	3097
38	2500
39	2242
40	3411
41	3018
42	2286
43	1724
44	3170
45	2609
46	1590
47	2451
48	3188
49	3390
50	2239

The average number to Action for 45 best trial is 2557 actions. The cost of randomness is that it takes a lot of actions to get the work done, but the benefit is that it is able to clean all the tiles.

3.5 How does the memory-based deterministic agent perform in the two environments? Is it able to clean the room perfectly? How long does it take? Can the agent be improved with more memory?

The memory-based model performs great in the environment with no walls. It can clean all the tiles. Total cleaned tiles = 100 and Total action taken = 232. Hence the performance is 0.43. Which is almost same as simple reflex agent but in this case, we are able to clean all the tiles. In case of environment with 4 rooms and walls, it is not able to clean all the tiles. It can clean one room completely, but with the current room design it was not able to clean the entire environment. Total cleaned tiles = 30 and Total action taken = 80. Hence the performance is 0.375. It depends on the room design how the agent will perform.

The agent can be improved with more memory, where it can store the location for the door and eventually notice if the room is cleaned and then proceed to the door for the next door. There can be multiple approach how to model it so that it can clean all the tile in all kind of environment.

3.6 What are the tradeoffs between the random and deterministic agents? How would you design better agents for more complex environments, say with polygonal obstacles?

The tradeoff between random and deterministic agent is that random is able to clean all the tiles eventually, but it takes a lot of actions to do it. Hence the performance is very bad but it is able to clean all time over the period of time. An example would be cleaning the house at night with no time restriction. In case of deterministic agent, we get good performance, but there are chances that it cleans all the tiles is less. There can be cases that it does not clean at all and it will go in a loop.

We can design a model for complex design depending on the environment. If there are polygonal obstacles, the agent can sense the angle it interacts with the wall and determine the angle of turn. Another approach can be to map the environment before hand and tag the location with some marker. So this is one time investment. So later on it can always follow the optimal path. Another approach will be to clean in rows. Turn around when the wall is hit and like that clean the room. If there are obstacles, the model can be designed to go around it. It really depends on what the user wants and how the environment is designed.

3.7 What did you learn from this experiment? Were you surprised by anything?

We learned that every agent has it pros and cons. And there is scope of improvement in each agent. Some give better performance but don’t do the job and vice versa. We understood that it depends on the environment and the situation. No one agent will always be the best in all situation. It may perform great in one case but fail miserably in other.

Also the order in which we place the if then statement matters a lot. It can change the performance and outcome of the agent. We learned a lot about different types of agent and would like to expand the project further to add GUI and interactive menu. Also try to show the movement of the vacuum cleaner for better understanding. Maybe even improve the Memory based agent by including more memory bits and changing the order of if statements.

The thing that surprised us was that when running the random agent for 90 tile being cleaned the average actions needed was 2557 whereas in the same environment with the same agent, but running for all 100 tiles cleaned, the average was 10547. This is a significant increase considering that only extra 10 tiles needed to be cleaned. When checked for 80 tiles it was around1572.