Comparisons of Vacuum Cleaning Agents - CS 531

Christopher Mendez, Rupika Dikkala

ABSTRACT

There are many different ways to approach programming an agent to navigate an environment. In this assignment, we implemented three different kinds of agents, a simple agent, a random agent and a limited memory agent across two environments. We compare and contrast the agents performances to each other, as well as to themselves across environments. We also explore ways the agents could have been improved, either by environment or expanded capabilities.

INTRODUCTION

As Artificial Intelligence (AI) systems are composed of an agent and its environment [1], when comparing the building blocks of different kinds of AI, it makes sense to vary the agents and the environments. In this paper, we compare the performance of three different vacuum cleaning agents across two environments to better understand the implications of different models of agents.

Our simple reflex agent demonstrates the limitation of just relying on sensor input and decision making based on the current state. Our random agent shows potential thought some inefficiency and a need to tune various parameters to achieve better performance. Our memory-based deterministic agent shows that relying on a little bit of memory can provide the agent enough context to improve its performance. Some of the agents limitations are shown well in comparing their performance across environments, and how overfitting a model to the first environment, which has no obstructions, can lead to worse performance in an environment with some obstructions.

DESCRIPTION OF AGENTS

Simple Reflex Agent

The simple reflex agent uses sensor input to make a decision based on the current state of the square it is on. The logic it follows is:

- **If** on dirty space, **then** clean
- **If** at home, **then** turn off
- **If** facing a wall, **then** turn right
- **Else** move forward

Random Agent

The random agent uses similar decision points of the simple reflex agent, however it randomizes the choice at those points. The logic it follows is:

- **If** on dirty space, **then** clean
- **If** at home, **then** turn off
- **If** facing a wall **then** 50% left turn, 50% right turn
- **Else** 50% move forward, 50% right turn

Memory-Based Deterministic Agent

The memory agent makes use of three bits of memory: 1) Its current x location in the environment, 2) Its current y location in the environment 3) The direction its currently facing (up, down, left, right).

Based on this input data, as well as its dirt, wall and home sensor,this robot adheres to the following logic:

- **If** on dirty space, **then** clean
- **If** x-location **and** y-location are near a wall **and** robot is facing down, **then** turn right.
- **If** facing wall **and** x-location is an odd number, **then** turn right
- **If** facing wall **and** x-location is an even number **and** robot is facing down, **then** turn left
- **If** at home, **then** turn off
- **If** on clean space **and** not facing a wall, **then** move forward
- **If** y-location is even **and** x-location is near a wall **and** robot is facing right, **then** turn right
- **If** y-location is odd **and** x-location is near a wall **and** robot is facing right, **then** turn left
- **Else** move forward

The goal of this logic is to have the robot move in a zig-zag pattern through the environment from left to right, then once it hits the bottom right corner, it heads in a straight line towards home. The zig zag is achieved by having the robot account for if its on an even or odd number space and near a wall and use that to determine which direction it should turn. By having it turn one direction on even spaces near walls and another on odd number spaces near walls it creates the zig-zag pattern.

EXPERIMENTAL SETUP

The setup for the agents involved two different environments. Each agent was run in each environment. For the simple reflex agent and memory-based deterministic agent, their performance was the same every time and so was just collected once. For the random agent, performance varied so was collected over 50 trials, and will compare the average of the random agent to the other two agents.

Shown in Fig 1, each environment, the main difference was env 1 had walls only on the outside, while env 2 had walls in the middle as well, with a single door between each of the four rooms.

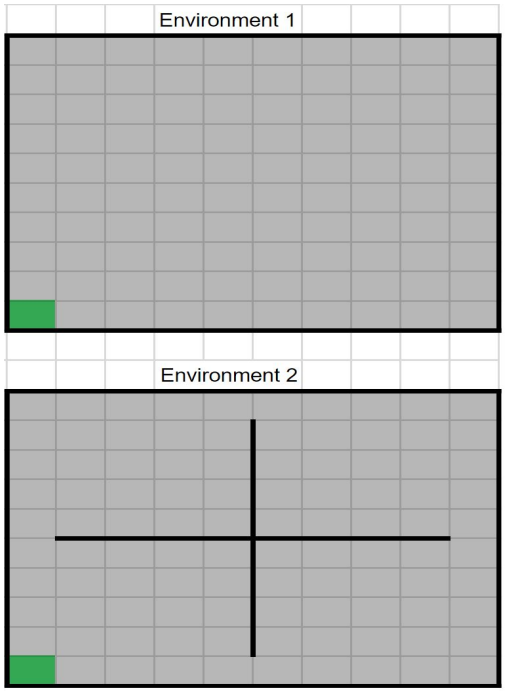

Fig 1. Env 1 and 2 both include a 10x10 grid, a home space in the bottom left and 99 dirty spaces.

RESULTS

Performance of Simple Reflex Agent

The simple reflex agent yielded the second best performance of all the robots. It cleaned up 35% of the cells in the first environment, and about 40% of the cells in the second environment (Table 1). In each case, the robot requires about twice as many actions as the number of cells cleaned. The primary hindrance for the reflex agent is its lack of memory. The best performance we were able to achieve for the first environment was the outer perimeter of the room, because the robot had no memory or spatial awareness.

Agent Performance Across Environments

Agent	Actions Taken	Clean Cells
ENV1 Simple	76	35
ENV1 Random	97	13
ENV1 Memory	228	99
ENV2 Simple	84	39
ENV2 Random	93	9
ENV2 Memory	171	60

Table 1. Each agent's performance across environments 1 and 2. Random in Env1 & Env2 is avg of the best 45 trials

It just knew where to turn when it hit a wall, and did not have any history stored. Cleaning the outer perimeter was the best case for the second environment as well, and we achieved this by strategically placing the doors at the outer edges of each room so that the robot can cover that area. While the robot covered 4 more spaces in the second environment over the first, it is important to note that proportionally, it is the same performance.

Performance of Random Agent

The random agent yielded the most deficient performance of all the robots. The average performance in the first environment was 13 clean cells enlisting 97 actions, and 9 clean cells for 93 actions in the second environment (Table 1). As depicted in Fig 2 and 3, there were some trials with exceptional performance, but the overall averages were very low. We account this to the trials where the robot cleaned 0 cells in either environment. At first, the agent was able to randomly move forward, left, or right from any space it was on. We optimized this logic by giving it only forward or right turns to avoid it from being stuck in a loop - this increased the chance of the robot moving away from home to 50% from a 33% chance. Additionally, we originally gave the robot a chance to turn off when returning home, however that resulted in more action without many additional clean spaces so we made turning off at home a non random action. For both environment 1 and 2, there was only one trial where more than 90% of the room was cleaned as shown in Table 2. In environment 1, 93 cells cleaned for 935 actions - here the actions were 10 times the number of cells cleaned. In environment 2, 101 cells cleaned for 1755 actions - here the actions were 17 times the number of clean cells. The benefits in the random agent design is probability: the agent has a chance (albeit small) of outperforming the simple reflex agent because all the moves are random. However, the performance is inefficient: the reflex agent required only double the actions for the number of clean

cells while the random agent had actions 10 times the number of clean cells in Env1 and 17 times in the Env 2.

Performance of Memory-Based Deterministic Agent

The memory-based deterministic agent performed the best of all agents. It cleans environment 1 perfectly and a majority of environment 2. There was a tradeoff with how we programmed it to perform. Due to the nature of a zig-zag pattern, the performance would suffer in either environment 1 or 2. If we didn't have it go straight home after hitting the bottom right corner, it would zig-zag all the way back in environment 1, causing a large number of unnecessary actions since all the spaces were cleaned. However because we had the agent go straight home from the bottom right corner, in environment 2, that causes it to not perform a zig-zag pattern on the bottom half of the environment.

We think the agent could be improved with more memory. For example, if it had the ability to keep track of how many squares it had cleaned, it could focus on returning home and stop zig-zagging after it had cleaned 99 spaces, allowing it to perform better on both environments.

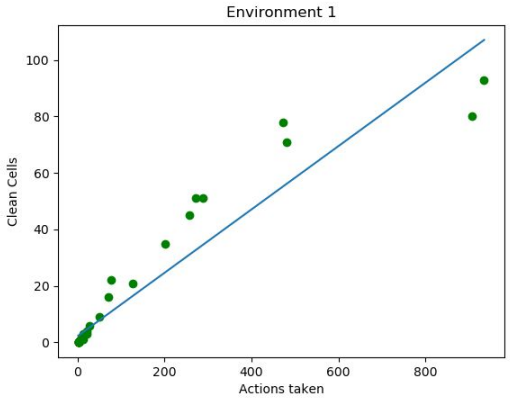

Fig. 2 Random Robot performance over top 45 trials in environment 1

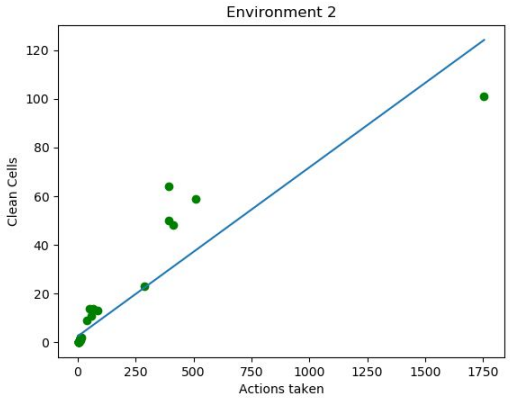

Fig. 3 Random Robot performance over top 45 trials in environment 2

Environment 1 - 90% Clean

Actions Taken	Clean Cells
935	93

Environment 2 - 90% Clean

Actions Taken	Clean Cells
1755	101

Table 2. Displaying the number of trials where the robot cleaned 90% or more of the spaces (90+/100 in our case).

DISCUSSION

Random and deterministic tradeoffs

The random agent is deprived of a location sensor, and randomizing the actions will help it escape infinite loops. However it lacks efficiency because it requires a disproportionately high number of actions to clean a small number of spaces. The deterministic agent has memory which means that it is able to use the full context of its environment to make turn decisions. However if dirt appears after the robot cleaned the space, it won't go back to clean it. In the context of the vacuum world, we would combine both the deterministic and random agent mechanisms in order to maximize performance in complex environments with polygonal obstacles. We would improve the deterministic agent by increasing the size of memory, and incorporate random turn decisions to mitigate the effects of placing obstacles.

Lessons learned

From this project we learned that it was quite difficult to optimize the random robot because it has no memory or access to history. Additionally, we realized that we needed to be strategic about optimizing deterministic robot, which was done by narrowing down the zigzag method and the turning mechanisms. However, we should also be aware that this program is the best case scenario: in real life, the room is not always perfectly empty, and there could be various obstacles (furniture, people, etc). We recognize that there could be cases where the deterministic robot is not be the most efficient, even compared to the non-deterministic agents.

We were surprised at how little the deterministic robot covered in the second environment. With the case of overfitting of the zigzag logic the first environment, it covered a little over half of the second environment thus proving that incorporating walls and other realistic "obstacles" doesn't always yield the most favorable outcome. We were also surprised at how often the random robot didn't clean any spaces in both environments. While we attribute this to the lack of memory, we didn't expect the average clean spaces to be so sparse.

REFERENCES

[1] Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: a modern approach*.