Comparison of Various Types of Simple Automatic Vacuum Cleaning Robots

Michael Anderson

October 12, 2010

Abstract

In this brief report I investigate the performance of different types of automatic vacuum cleaning robots, by modeling them as agents with a computer simulation, in a simple and constrained environment. I first implement a deterministic agent with no memory, and find that it is highly ineffective and is not capable of even mostly cleaning my model rectangular floor. I then implement an agent that can take actions randomly given a certain set of percepts, and find that it is more effective at cleaning the model floor, but requires many actions to do so. Finally, I implement that is deterministic but also has a limited memory, and find that it is both effective at cleaning the simulated floor and extremely efficient.

Contents

1 Introduction

Artificial Intelligence is a field of computer science with far reaching overlaps into many other disciplines, and many different problem spaces. In this report, I look into the relationship between AI and automatic vacuum cleaning robots.

The question I seek to answer is: generally speaking, what type of simple agent is best suited to the task of automatically cleaning a floor? I define the "best suited" agent as the agent that can not only completely or mostly clean a floor, but which can also perform the task quickly and efficiently.

Here I endeavor to answer this question by modeling a simple and constrained environment with a computer simulation. In this environment, agents can only occupy one square at a time of a discrete 10×10 grid of squares. At any given moment, each square can only be completely clean or completely dirty. At the start of each simulation all squares in the grid will be dirty, and agents will be placed in the lower left corner of the grid.

The set of possible percepts that the agent can receive is extremely limited: in any given state it knows whether the square it is standing on is clean or dirty, whether it is standing on the square it began the simulation at (the reason why will be clear in a moment), and whether it is facing a "wall" (the end of the grid). Since each of these three questions has a yes or no answer, the domain of the percept function will have $2^3 = 8$ elements.

The set of actions that agents can take is also restricted. At any given moment agents are facing either north, south, east, or west. At the start of the simulation they will always face north. It is possible for them to change the way they are facing by turning to the right or left. They may also move forward into the next square that they are facing; they can suck, which will make the square that they currently occupy clean, if it is dirty; and they can turn themselves off. The range of the percept function contains only these 5 elements.

Time is discretized in terms of actions. Agents perceive their environment as being in one of the 8 possible states, and then take an action based on those percepts. Each time an agent does this, one unit of "time" elapses. In this environment the time cost of each action is the same.

Now that the constraints of the simulation are defined, the goals of the agent in this simulation can be clarified. The primary goal is to design the percept function of an agent so that it can clean the floor completely, return to its starting square, and shut itself off. The secondary goal is to design an agent that can clean as high a percentage of squares as possible, and can do it as quickly as possible (with the fewest number of actions).

Here I have implemented three agents, each with their own constraints, with the best percept functions I could design. The first agent is a deterministic memoryless agent. "Deterministic" means that the agent always takes the same action given one of the 8 percepts. "Memoryless" means that the agent's action are not affected by its previous actions or percepts, it has no memory of them. The second agent is non-deterministic memoryless agent. "Non-deterministic" means that the agent is capable of randomly performing an action with some probability P associated with that action, given one of the 8 percepts. For

example, if this agent is on a dirty square, not on its starting square, and facing a wall, it could randomly choose to move forward with 20% probability, turn right with 40% probability, or turn left with 40% probability. The third agent is a deterministic agent with three bits of memory. Each bit is initialized to false at the beginning of the simulation, and each time the agent is called upon to take an action it can also consider the values of these three bits. Therefore, this agent's percept function has a domain of $2^3 \times 2^3 = 8 \times 8 = 64$ elements. As the agent takes an action, it can also change the values of any or all of the three bits, giving the range of its percept function $2^3 \times 5 = 8 \times 5 = 40$ elements.

Finally, it is important to note that even though the code for the agents and the code for the simulation exist in the same code file, there is a clear separation between these two objects. The simulation object knows all about the environment, whether every square in it is clear or dirty, and where the agent is located in the environment. The agent objects only know their environment through one of the 8 possible precepts, and internally only use percept functions and memory (in the case of the third agent) to determine an action. All of the code was for this project was written with Python 2.4.3.

2 Deterministic Memoryless Agent

2.1 Design

It turns out that the capabilities of deterministic reflex agents in this simulation are extremely limited.

For one thing, this agent cannot be programmed to return home and shut itself off, because since the agent has no memory, it has no way of knowing whether the simulation just began or whether it has been out cleaning and has returned home. So if this type of agent is programmed to turn itself off when it is on a clean home square, that square is the only square that will be cleaned before it self-terminates, leaving the remaining 99 squares dirty.

Also, there appears to be no way to get this agent off of the $9 \times 4 = 36$ square perimeter of the grid. Once the agent cleans a square and is not facing a wall it can either turn or move forward. If it turns, then it simply spins around in a circle on the same square forever. If it moves forward, then it will continue to do so until it reaches a corner, and then it can turn and reach another corner, and so on. My percept function for this agent is as follows:

On Dirt, Facing Wall, At Home	Action
F,F,F	Forward
F,F,T	Forward
F,T,F	Turn right
F,T,T	Turn right
T,F,F	Suck
T,F,T	Suck
T,T,F	Suck
T,T,T	Suck

Notice that this agent always sucks first when it is on a dirty square, before it attempts to move.

2.2 Results

The given agent will run around the 36 square perimeter of the grid, turning right every time it hits a corner, and sucking as it goes. Since the floor space is 100 squares, this leads to a cleanliness of 36%. Since there is no way to leave the perimeter of the grid with a deterministic memoryless agent in this simulation, this type of agent will never get the floor cleaner than 36%.

Although my code is already submitted and cannot be improved, I realized at write-up time that I could change rule 4 to "F,T,T → Turn off", which would make the agent turn off after going around the perimeter of the grid one time and returning to the home square. This would be an improvement over the percept function above, because the agent following it in my code never turns off. If this improvement is made, the agent will turn off as its 76th action.

3 Non-Deterministic Memoryless Agent

For this section, the goal of the agent is modified slightly. The assumption is made that the agent is capable of getting the floor 100% clean by randomly moving around and sucking up dirt whenever it is encountered. The assumption is also made that it could potentially take many many action to do this, however, and that in fact this goal will be reached asymptotically as an agent stumbles around the grid looking for those "very last few squares." Since the number of actions taken is a part of the performance metric for the agents we are considering, and since this type of agent has no way of knowing when it completely cleans the floor anyway, we instead strive to design an agent that will clean the floor 90% of the way as quickly as possible, and once that occurs the simulation ends.

3.1 Design

This agent should suck whenever it encounters dirt, and it should turn right or left non-deterministically when it encounters a wall. When neither occurs it should move forward most of the time to stay in its current row or column, but should occasionally turn right or left so that it has an opportunity to sometimes leave the edge of the grid, unlike the deterministic memoryless agent. My percept function for this agent is as follows:

On Dirt, Facing Wall, At Home	Action
F,F,F	Forward 90%, Turn right 5%, Turn left 5%
F,F,T	Forward 90%, Turn right 5%, Turn left 5%
F,T,F	Turn right 50%, Turn left 50%
F,T,T	Turn right 50%, Turn left 50%
T,F,F	Suck
T,F,T	Suck
T,T,F	Suck
T,T,T	Suck

In particular, in all situations where the agent could turn, it has an equal probability of turning left or right, since there seems to be no reason for preferring one over the other. When the agent is neither facing a wall or on dirt, it has a 90% chance of moving forward. Since rows and columns in the grid are 10 squares wide, this means it has a good chance of making it to the end of whatever row or column it is in at any given moment.

3.2 Results

Because the agent is non-deterministic, its performance can vary over different trials. So instead of running one trial and possibly getting a result that is a statistical outlier, we instead run the simulation over 50 trials, and look at the average data.

After 50 trials, the average number of actions taken to clean the floor 90% is 683. Considering only the 45 best trials: their range of values lie between 425 and 868, their mean is 653, and their median is 666. Clearly this agent performs its work slowly. It is possible that the percept function given above does not describe the optimal non-deterministic memoryless agent for this task. In particular there may be a superior distribution of probabilities for actions taken in the 1st and 2nd rules. However, it is expected and reasonable to conclude that an agent that randomly stumbles around in a disordered stochastic fashion will take a long time to clean most of a floor. The introduction of randomness allows the agent to wander into additional parts of the grid, but the disorderedness of this wandering can make the agent revisit already clean squares unneccessarily.

4 Deterministic Agent With Memory

Recall that the distinguishing feature of this agent is that it has 3 bits of memory which can be considered at each state to decide on an action, and also altered at each state. It is assumed that at the start of the simulation all 3 bits are reset.

4.1 Design

One possible path that an agent could take that would traverse the entire floor only once would be a zigzag path across the floor. In other words, the agent first goes up the entire first column, then turns, goes forward once, turns again, and goes down the entire second column, then turns, goes forward once, turns again, and goes up the entire third column, etc.

One consideration is that the agent must first turn right twice once it hits the top of the first column, then it must turn left twice once it hits the bottom of the second column, then right at the end of the third column, and so on. In order to make the agent turn the correct direction when it encounters a wall, the first bit of memory —called *left_or_right*— will store turning direction.

Another consideration is that in order to move from one column to the next, an agent needs to turn, move forward, and then immediately turn again. The agent needs to remember that after it turns the first time, it should then take the second and third actions. When the agent needs to take the second action next, the second bit —called *just_hit_wall*— will be set. When the agent needs to take the third action next, the third bit —called *just_just_hit_wall*— will be set. Once the agent sees that it should perform the second step, it resets the second bit, and likewise with the third step and third bit. In this way, the agent can resume normal operation after completing its column-changing maneuver.

Finally, there is one more major case that must be considered in the design of this agent. It would be nice if the agent knew when it was done and that it then needs to go home to shut off. This case can be handled by realizing that the agent will reach the opposite corner of the grid from its starting position right after its traversal and cleaning of the entire grid is nearly complete. That situation will be the only time that it turns to the right and is immediately facing a wall. So if the second bit is set and the agent finds itself facing a wall, set both the second and third bits. This is the only case in which both bits will be set at the same time, and this will indicate to the agent that it should just turn right whenever it encounters a wall, and move forward otherwise, much like the motion of the original deterministic memoryless agent. This will take the agent home along a minimal Manhattan path —the perimeter of the grid.

My percept function for this agent is as follows, where X represents "don't care" to reduce the size of the table:

On Dirt, Facing Wall, At Home, left_or_right, just_hit_wall, just_just_hit_wall	Action
F,F,X,X,F,F	Forward
F,F,X,F,F,T	Turn left, reset just_just_hit_wall
F,F,X,T,F,T	Turn right, reset just_just_hit_wall
F,F,X,X,T,F	Forward, reset just_hit_wall, set just_just_hit_wall
F,F,X,X,T,T	Forward
F,T,F,F,X,T	Turn left, set just_hit_wall
F,T,F,T,X,T	Turn right, set just_hit_wall
F,T,F,F,X,T	Turn right, set left_or_right, set just_hit_wall
F,T,F,T,X,T	Turn left, reset left_or_right, set just_hit_wall
F,T,T,X,X,X	Turn off
T,X,X,X,X,X	Suck

4.2 Results

Out of the three agents modeled, this one performs the best by far. It traverses and cleans the entire floor, only going over uncleaned squares when it has finished and must travel home. It cleans the grid every time (since it is deterministic) with a *mere 228 actions*. This agent does not really waste actions and appears to take an optimum path, and I do not believe an agent could be designed that could perform better in this model, even with more bits of memory. The 3 bits provided are just enough to make it do all of what we could possibly want.

5 Conclusions

To summarize, first recall that there were two primary metrics used to evaluate the agents in this model. Their ability to clean as much of the floor as possible, and their ability to do it in as few actions as possible. The deterministic memoryless agent did poorly by the first metric but did well by the second. It cleaned less than half of the floor but took only 76 actions to do it. The non-deterministic agent did well at the first metric but poorly at the second. It cleaned most of the floor but took on the order of 10 times as many actions as the deterministic agent. The deterministic agent with memory performed well according to both metrics, cleaning the entire floor every time, and doing it in a probably optimal 228 actions every time.

These results are fairly consistent with vacuum cleaning robots in the real world, even though they operate in realtime and continuous space (not necessarily a square space either), under conditions that are much less simple and constrained. Deterministic memoryless robots can easily get stuck in a corner or miss parts of the space. Non-deterministic robots can wander around and eventually clean an entire floor, but it may take them quite some time. Robots with memory, particularly robots that are able to map their environment, are

able to calculate optimum paths for moving around it, and perform extremely well.