CS531 Programming Assignment 1
Vacuum Cleaning Agents
January 19, 2015
Yaonan Zhong, Xiaofei Gao

Abstract: In this project, we design and implement three different vacuum-cleaning agents using if-then rules. The environment consists of a n*m empty rectangular room with each square has a p% chance of containing dirt. There are three sensors: wall, dirt and home. And five actions are availble: go forward, turn right/left, suck up dirt and turn off. We experiment with m=n=10 and p=100%, and compare their performances by evaluating the total number of clean cells as a function of the number of actions taken. We also discuss the idea behind the design of each of our agents.

1. Introduction

The goal of this project is to design and implement three different vacuum-cleaning agents using if-then rules, and explore and discuss the possible factors that lead to their different performances. The three agents include a simple memoryless deterministic reflex agent, a randomized reflex agent, and a deterministic model-based reflex agent. We will give detailed decriptions of these agents and the environment setup in the following sections. The experiment result shows that the simple reflex agent turns out to be of limited intelligence as expected because of its deterministic and memoryless design. The randomized simple agent has an unstable performance with a large variance of the number of clean cells. The deterministic model-based reflex agent has the best performance among the agents. Not only it is able to completely clean the room, but also can shut itself off after it is done.

2. Description of agents

1) Simple memoryless deterministic reflex agent

The simplest kind of agent is the simple reflex agent. It selects actions on the basis of the current percept, ignoring the rest of the percept history. The agent is deterministic such that there is only one possible action selected at each step. Figure 1 shows the structure of such agent in schematic form.

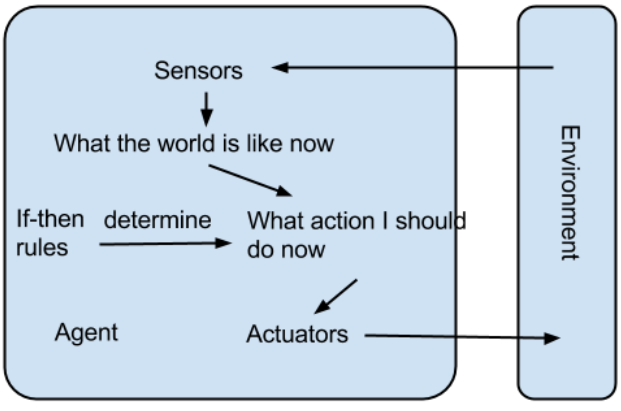


Figure 1 Simple memoryless deterministic agent

2) Randomized reflex agent

The randomized reflex agent is similar with the simple one except that it can choose actions randomly based on sensor readings.

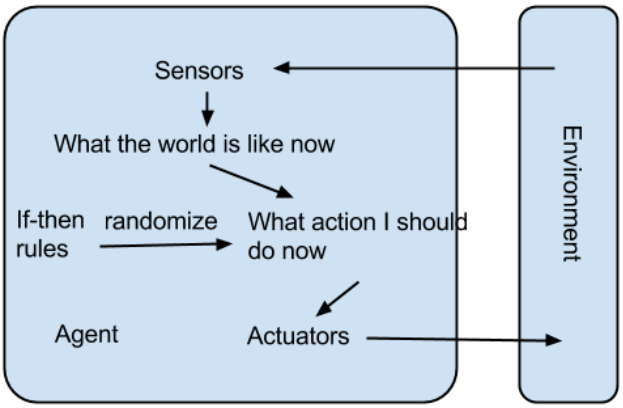


Figure 2 A randomized reflex agent

3) Deterministic model-based reflex agent

A deterministic model-based reflex agent has a small amount (2 to 3 bits) of memory that represents the "state." When executing each action, the agent simultaneously updates the state by setting or resetting these bits according to how you specify them. The actions can be based on its current state bits as well as the current percepts.

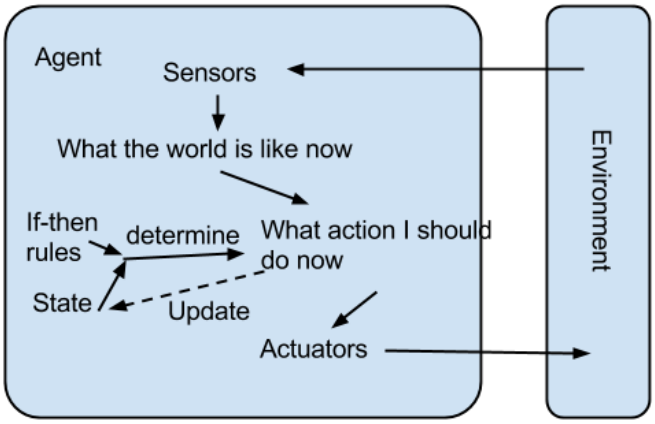


Figure 3 A deterministic model-based reflex agent

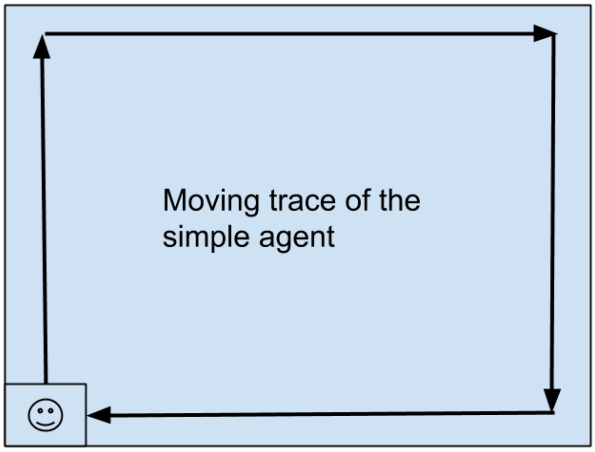
3. If-then rules of agents

---

1) Simple memoryless deterministic reflex agent

```
class SimpleReflexAgent(Agent):
    def run(self, room):
```

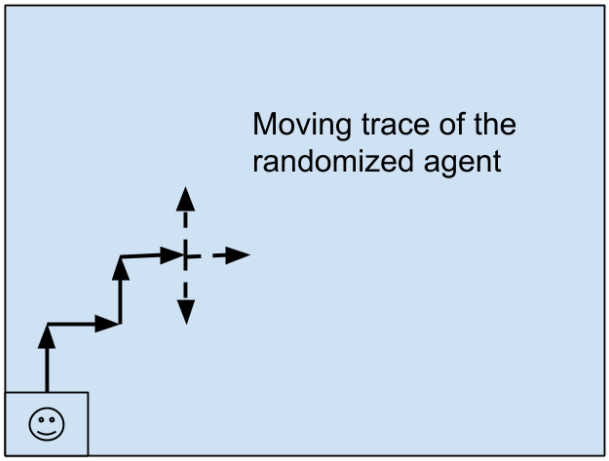
```
        while True:
            if self.dirt_sensor(room):
                self.suck_dirt(room)
            elif not self.wall_sensor(room):
                self.move_forward()
            elif self.home_sensor(room):
                self.turn_off()
            else:
                self.turn_right()
```



---

2) Randomized reflex agent

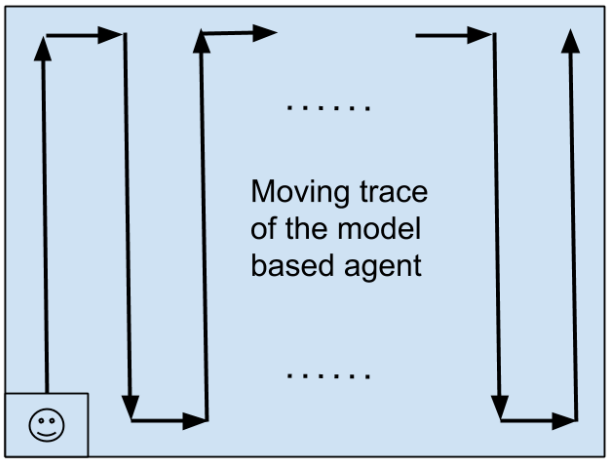
```
class RandomizedReflexAgent(Agent):
    def random_move(self):
        choice = random.choice(range(3))
        if choice == 0:
            self.turn_right()
        elif choice == 1:
            self.turn_left()
        else:
            self.move_forward()

    def run(self, room):
        while True:
            if self.dirt_sensor(room):
                self.suck_dirt(room)
            elif not self.wall_sensor(room):
                # use random action
                self.random_move()
            elif self.home_sensor(room):
                self.turn_off()
            else:
                self.turn_right()
```



---

3) Deterministic model-based reflex agent

```
class ModelBasedReflexAgent(Agent):
    def __init__(self, home, direction):
        Agent.__init__(self, home, direction)
        # inits three states
        self.has_turn = False
```



```
        self.has_right_turn = False
        self.has_move = True

    def run(self, room):
        while True:
            if self.dirt_sensor(room):
                self.suck_dirt(room)
            elif (not self.wall_sensor(room)):
                if self.has_turn:
                    if self.has_move:
                        if self.has_right_turn:
                            self.turn_right()
                        else:
                            self.turn_left()
                        self.has_turn = False
                    else:
                        self.move_forward()
                        self.has_move = True
                else:
                    self.move_forward()
            else:
                if self.has_turn:
                    if self.has_move:
                        if self.has_right_turn:
                            self.turn_right()
                        else:
                            self.turn_left()
                        self.has_turn = False
                    else:
                        self.turn_off()
                else:
                    if not self.has_right_turn:
                        self.turn_right()
                        self.has_right_turn = True
                    else:
                        self.turn_left()
                        self.has_right_turn = False
                    self.has_turn = True
                    self.has_move = False
```

4. Experimental setup

The environment consists of a n*m empty rectangular room with each square has a p% chance of containing dirt. There are three sensors: wall, dirt and home. And five actions are availble: go forward, turn right/left, suck up dirt and turn off. We experiment with m=n=10 and p=100%. The agent always starts in the bottom leftmost corner oriented upwards.

5. Performance

Figure 4 shows the total number of clean cells as a function of the number of actions taken for three agents. For the random agent, we repeat the experiment 50 times and plot the average performance.

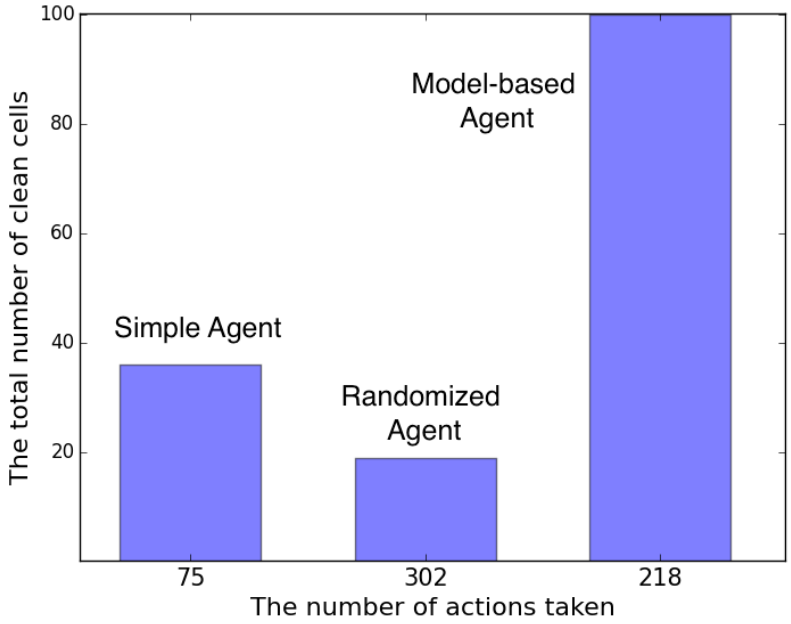


Figure 4 Experiment result of three agents

6. Discussion

The simple agent has the best possible performance among memoryless reflex agents by cleaning 40 cells with 75 actions taken. The limitation of a memoryless reflex agent is that it can only select an action based on the current percept.

The average performance of a randomize agent is 19 clean cells with 302 actions taken. I don't think this is the best possible performance achievable by any random agent, because we can design a random agent with different possibilities of actions. Table 1 shows the number of actions taken for cleaning more than 90% of the room by a random agent. And the average number is 2172. The cost of randomness is that it takes more actions to clean the room, while the benefit is that it can clean more cells than the simple deterministic agent.

Table 1 Total number of actions taken for cleaning more than 90% of the room

clean cells	98	99	99	99	99	average
actions taken	1160	2145	2107	4034	1415	2172

The memory-based deterministic agent outperforms the random agent for that it takes a lower number of actions to clean the room. In our design this number is 218. It can completely clean all the cells and also shut itself off after it is done, while the random one can not guarantee shutting itself off after cleaning most cells.

We use three bits of memory that represents the states in our design. The model-based agent can not be improved further with more memory than this number, because the number of actions taken in our design is the minimum for any agent to cover all the cells, which is illustrated in the moving trace of the agent shown before.

The most challenge part of this project for us is designing the if-then rules for the model-based agent such that it can achieve the best performance. And the most surprising thing is that a random agent can almost clean all the cells, since randomization is usually not rational in a single-agent environment.