CS531 Programming Assignment 1: Vacuum-Cleaning Agents

Michael Lam, Shell Hu
EECS, Oregon State University

October 4, 2012

Abstract

In this assignment you design and implement 3 different vacuum-cleaning agents.

1 Introduction

2 Simple Memoryless Deterministic Reflex Agent

3 Randomized Reflex Agent

4 Deterministic Model-Based Agent

4.1 Design and Algorithm

The requirements of the deterministic model-based reflex agent other than its definition is that it can represent state with only 3 bits of memory, which implies up to 8 states[1].

We want the agent to suck up every dirt. Since the agent has memory, we are able to perform two "consecutive" actions by taking advantage of state

[1]Without loss of generality, we can use one state variable STATE that takes on integers $[0, 7]$ instead of three explicit state bits STATE1, STATE2 and STATE3 because $0 = 000, 1 = 001, 2 = 010, ..., 7 = 111$.

information. This will allow the agent to steer into the middle of the world in contrast to the simple memoryless reflex agent.

Therefore, we devise an algorithm that can sweep across the entire room and suck up every dirt in 8 states or less. We came up with the following algorithm that sweeps all dirt and returns home in at most 7 states:

1. Begin at the home cell.
2. Go north and suck until it hits the south boundary.
3. Go one cell east.
4. Go south and suck until it hits the south boundary.
5. Go one cell east.
6. Repeat steps 2-5 until there is a wall that prevents going one cell east.
7. Traverse the world boundary until it comes home.

To visualize the process, the algorithm does the following sweep pattern across the room:

[Picture]

4.2 Diagram and Rules

We claim that we can construct the algorithm using 7 states by illustrating the following state diagram:

[Diagram]

The following is a formal list of if-then rules used to construct the diagram. It is also coded into the agent as the program:

```
if STATE=0 and DIRT then SUCK
if STATE=0 and NOT WALL then FORWARD
if STATE=0 and WALL then STATE := 1, RIGHT
if STATE=1 and WALL then STATE := 6, RIGHT
if STATE=1 and NOT WALL then STATE := 2, FORWARD
if STATE=2 then STATE := 3, RIGHT
if STATE=3 and DIRT then SUCK
if STATE=3 and NOT WALL then FORWARD
if STATE=3 and WALL then STATE := 4, LEFT
if STATE=4 and WALL then STATE := 6, RIGHT
if STATE=4 and NOT WALL then STATE := 5, FORWARD
if STATE=5 then STATE := 0, LEFT
if STATE=6 and HOME then OFF
if STATE=6 and NOT WALL then FORWARD
if STATE=6 and WALL then RIGHT
```

In plain language, state 0 represents a series of reflex actions for sucking and moving north. When the agent hits the top boundary of the world, states 1-2 assist the agent in moving east one square to prepare sucking and moving south. State 3 (symmetric to state 0) represents a series of reflex actions for sucking and moving south. When the agent hits the bottom boundary of the world, states 4-5 (symmetric to states 1-2) assist the agent in moving east one square to prepare sucking and moving north. The cycle repeats until the agent can no longer move east, as detected in states 1 and 4. If that is the case, then the agent moves to state 6, which is a series of reflex actions to navigate the agent around the edges of the map until it reaches home.

Therefore, we have also verified that the state diagram is equivalent to the algorithm.

4.3 Results and Performance

For any $n > 0$, $m > 0$ and $0 \leq p \leq 1$, the agent is able to completely clean the room and shut off itself properly. The algorithm assumes the home cell is always the bottom-left corner of the world and the world is always a rectangular nxm array of cells with walls only at the boundarie of the rectangle. If the home cell is somewhere else, then it may miss cells to the west and south of its starting location. If the world is not rectangular, then it may make a premature assumption that the world is clean or it may navigate incorrectly such that it misses some cells.

For $n = 10$, $m = 10$ and $p = 1.0$ as in the default case, the agent took 230 steps.

5 Experiments

6 Discussion