CS534 Homework1， September 10, 2018

Ruojun Li

Problem 1

2.3 For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

a. **An agent that senses only partial information about the state cannot be perfectly rational.**
b. **There exist task environments in which no pure reflex agent can behave rationally.**
c. **There exists a task environment in which every agent is rational.**
d. **The input to an agent program is the same as the input to the agent function.**
e. **Every agent function is implementable by some program/machine combination.**
f. **Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.**
g. **It is possible for a given agent to be perfectly rational in two distinct task environments.**
h. **Every agent is rational in an unobservable environment.**
i. **A perfectly rational poker-playing agent never loses.**

A	F	Using the Google Map, the system mostly uses GPS and WiFi signal strength on the street. But in an off-line environment, this system only senses partial information (GPS). In this case, google map is still a rational behavior and convincible example. While the uber using such localization system, is also a partial information task.
B	T	Only the all-observable system can be made as simple pure reflex system. Most AIs such as Alpha go and Auto-car are no pure reflex and rational.
C	T	For example, when the auto-car drive on a desert, there is a no street or map or destination in an agent. So the car may turn right/left, stop, reverse or drive forward. All actions have the same reward.
D	F	The program contains the input history, the function only include present percept.
E	F	Simple reflex agent such as thermostat made by Honeywell, only consider present percept. The program is invariant.
F	T	The possible actions are finite. And finally, the agent will select the correct decision. If the agent has infinite actions, it’s irrational.
G	T	For example, the thermostat agent can work in my kitchen to make it warm and comfortable. Meanwhile, in my laptop to cool down my CPU, GPU and power supply. The task is distinct, but the agent can be the same.
H	F	If the thermostat can only senses the humidity how can it control the temperature.
I	F	For the agent, it plays the card according to the possibility. Since the win possibility is not always 100%, there is still a chance to win the agent. Also the agent cannot decide the poker in his hands.

Problem 2

2.9 Implement a simple reflex agent for the vacuum environment in Exercise 2.8. Run the environment with this agent for all possible initial dirt configurations and agent locations. Record the performance score for each configuration and the overall average score.

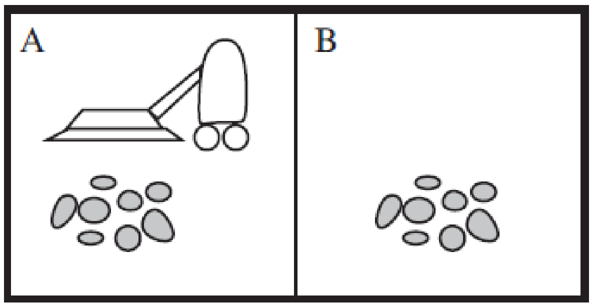
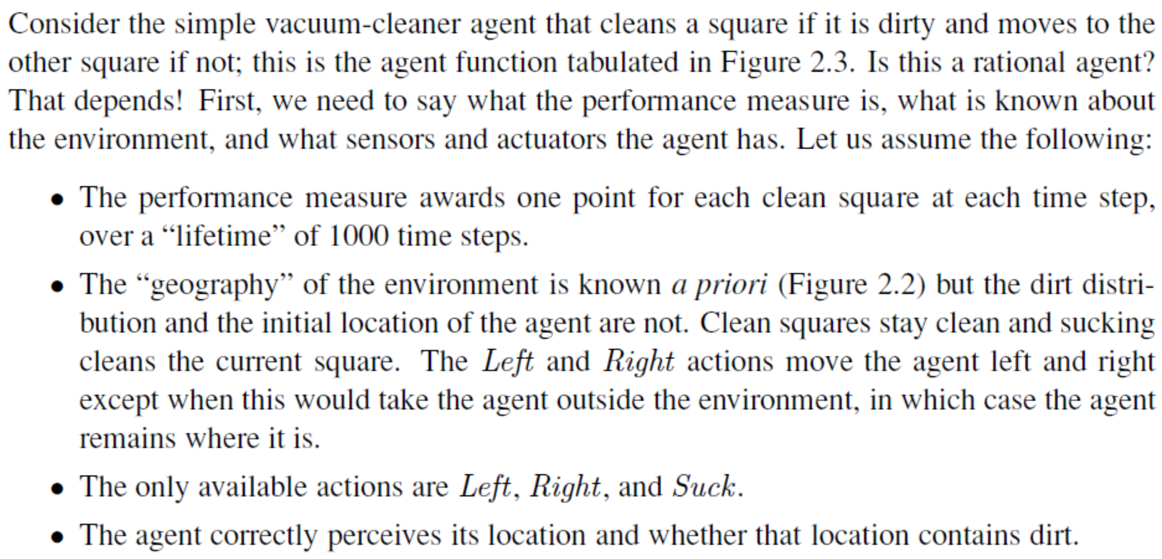


Figure 2.2 A vacuum-cleaner world with just two locations.



Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function tabulated in Figure 2.3. Is this a rational agent? That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has. Let us assume the following:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known *a priori* (Figure 2.2) but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The *Left* and *Right* actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are *Left*, *Right*, and *Suck*.
- The agent correctly perceives its location and whether that location contains dirt.

Simple Reflex Agent: This agent acts based only on the percept, ignoring the rest of the percept history.

Actions: Left, Right, Suck

Environment: These are the two locations for the two-state environment

Configurations:

1. {A: clean, B: clean} 2. {A: clean, B: dirty} 3. {A: dirty, B: clean} 4. {A: dirty, B: dirty}

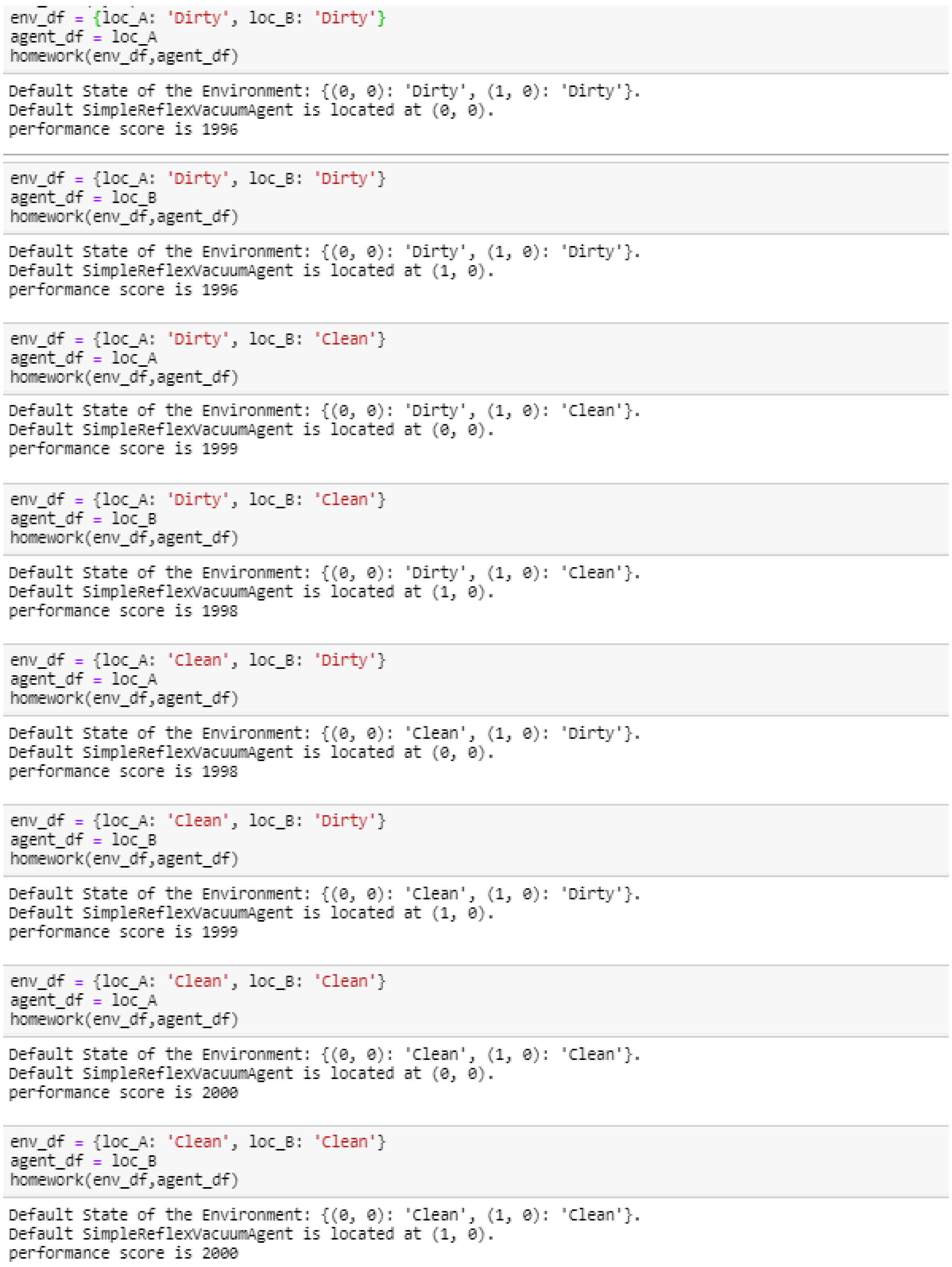
Performance score:

Configurations	Agent location	Performance score after 1000-time-step	Average score
{A: clean, B: clean}	A	2000	1998.25
	B	2000	
{A: clean, B: dirty}	A	1998	
	B	1999	
{A: dirty, B: clean}	A	1999	
	B	1998	
{A: dirty, B: dirty}	A	1996	
	B	1996	

Actions:

Configurations	Agent location	Performance score after 1000-time-step	Average score
{A: clean, B: clean}	A	‘Right’ ’Left’ ‘Right’…. For 1000 times	1998.25
	B	‘Left’ ‘Right’ ‘Left’ … For 1000 times	
{A: clean, B: dirty}	A	1999 ‘Right’ ‘Suck’ ‘Left’ ‘Right’…	
	B	2000 ‘Suck’ ‘Left’ ‘Right’ ‘Left’…	
{A: dirty, B: clean}	A	2000 ‘Suck’ ‘Right’ ‘Left’ ‘Right’ …	
	B	1999 ‘Left’ ‘Suck’ ‘Right’ ‘Left’ …	
{A: dirty, B: dirty}	A	1998 ‘Suck’ ‘Right’ ‘Suck’ ‘Left’ …	
	B	1998 Suck’ ‘Left’ ‘Suck’ ‘Right’	

Python code is attached in the file homework1.ipynb.



```
env_df = {loc_A: 'Dirty', loc_B: 'Dirty'}
agent_df = loc_A
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Dirty', (1, 0): 'Dirty'}.
Default SimpleReflexVacuumAgent is located at (0, 0).
performance score is 1996
```

```
env_df = {loc_A: 'Dirty', loc_B: 'Dirty'}
agent_df = loc_B
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Dirty', (1, 0): 'Dirty'}.
Default SimpleReflexVacuumAgent is located at (1, 0).
performance score is 1996
```

```
env_df = {loc_A: 'Dirty', loc_B: 'Clean'}
agent_df = loc_A
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Dirty', (1, 0): 'Clean'}.
Default SimpleReflexVacuumAgent is located at (0, 0).
performance score is 1999
```

```
env_df = {loc_A: 'Dirty', loc_B: 'Clean'}
agent_df = loc_B
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Dirty', (1, 0): 'Clean'}.
Default SimpleReflexVacuumAgent is located at (1, 0).
performance score is 1998
```

```
env_df = {loc_A: 'Clean', loc_B: 'Dirty'}
agent_df = loc_A
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Clean', (1, 0): 'Dirty'}.
Default SimpleReflexVacuumAgent is located at (0, 0).
performance score is 1998
```

```
env_df = {loc_A: 'Clean', loc_B: 'Dirty'}
agent_df = loc_B
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Clean', (1, 0): 'Dirty'}.
Default SimpleReflexVacuumAgent is located at (1, 0).
performance score is 1999
```

```
env_df = {loc_A: 'Clean', loc_B: 'Clean'}
agent_df = loc_A
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Clean', (1, 0): 'Clean'}.
Default SimpleReflexVacuumAgent is located at (0, 0).
performance score is 2000
```

```
env_df = {loc_A: 'Clean', loc_B: 'Clean'}
agent_df = loc_B
homework(env_df,agent_df)
```

```
Default State of the Environment: {(0, 0): 'Clean', (1, 0): 'Clean'}.
Default SimpleReflexVacuumAgent is located at (1, 0).
performance score is 2000
```

Python code is attached in the file homework1.ipynb.

Problem 3

3.2 Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

a. **Formulate this problem. How large is the state space?**

States:

Direction: 4 {North, East, South, West}

Location: Maze has n blocks.

Actions: Move min (X, distance-to-the facing-wall) blocks

The State Space: 4*n

Initial state:

Robot on {location (0,0), facing (0,1)}

b. **In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?**

States:

Direction: 4 {North, East, South, West}

Location: The m Corridors-blocks in the maze

Direction: 2 direction of {North, East, South, West}

Location: Maze has n-m blocks.

Actions:

The State Space: 2*(n-m) + 4*m= 2*(n + m)

Initial state:

Robot on {location (0,0), facing (0,1)}

c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

No, we don't need to keep the robot's orientation. Each corridors-intersection-blocks has four direction which is related to the robot orientation. We have m corridors, so the state space is m .

d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

1) **The robot can change direction 90 degree perfectly.**

2) The intersection must have four directions to move. Doesn‘t has any three-direction intersection.
3) The maze must be like the squares on the chessboard. And it is vertical to the East direction.

Problem 4

3.3

Suppose two friends live in different cities on a map, such as the Romania map shown

in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city

on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

a. Write a detailed formulation for this search problem. (You will find it helpful to define

some formal notation here.)

State: All possible city-pair P(i, j).(Have road between city i and city j)

Initial state: Two Friends in randomly two city, A and city B

Action: Move to the city near to the current city A.

Goal: Move to the (i, i) at a turn.

b. Let D(i, j) be the straight-line distance between cities I and j. Which of the following heuristic functions are admissible?

3) D(i,j)/2 is admissible. The best condition is the distance between two friends can be sperate in the same size for the moving friends.

c. Are there completely connected maps for which no solution exists?

True, only two cities and one road. The only action in the question 3.3 is moving to the near city. If we also has a action ’stay’. All connected maps have a solution.

d. Are there maps in which all solutions require one friend to visit the same city twice?

True, the only action in the question 3.3 is moving to the near city. Just like the conjecture in the 3.3.c, if we has a loop to realize the stay action, we can solve the no-solution problem.