

# Homework #1: Chapters 1, 2, 3

1.

Agent Type	Performance Measure (Quant)	Environment (Env/Agent=Ext/Int)	Actuators	Sensors
(a)	high # of correct fraud detections, low # of false detections	credit transactions	alert system (to notify of fraud)	receive transactions DB of credit/transaction history?
(b)	# of correct matches	DB of fingerprint images, Input Image	API for DB search, Result output mechanism	Input fingerprint image, images returned from DB
(c)	Life found, area covered	Mars: rocks, sand, hills, ditches	Steering, accelerator, brake,	camera, life detection instruments

2.

Dimension	Agent		
	Credit fraud	Fingerprint	Mars rover
observable	fully (transactions are accurate/complete)	partially (may have to match partial prints from crime scenes)	partially (e.g., can't see around rock)
deterministic	stochastic (no control over next transactions)	deterministic (agent's actions don't change environment)	stochastic (motors are imperfect, weather conditions)
episodic	sequential (must look for patterns)	episodic (classification task)	sequential (long-term consequences of short-term actions)
static	dynamic/semidynamic (detecting fraud too late isn't as useful)	static (prints don't change)	dynamic (e.g., dust storms, weather conditions)
discrete	discrete (transactions have finite date, payees)	continuous (digital image is complex enough and represents continuous varying intensities and locations)	continuous (camera images, steering angles range in values)
# of agents	single (the defrauders are treated as part of the environment)	single	single

3. Some advantages of the table-driven agent (TDA) over the simple reflex agent (SRA): TDA considers percept history, thus the next action can in a sense be more informed. As a result, the TDA can work in situations in which the SRA would fail (e.g., if an action returns one to the same state, the TDA may be able to “break out of the loop”).

Some advantages of the SRA over the TDA: SRA are simple and non-laborious to code. The TDA requires the programmer to explicitly encode for every possible history sequence. Since percepts are converted to “conditions”, the number of possibilities that the programmer must explicitly account for is also reduced. Thus it is easier to change the SRA’s behavior since there are far fewer state-action pairs than in the TDA. The SRA can also (practically) work in environments with a large state-space where the TDA is impractical/impossible to implement.

4. Initial State:

A 4x4 grid with X and Y coordinates in range {0,1,2,3} with (0,0) being top-left corner. Randomly some squares have numbers from the set {1,2,3,4} such that the puzzle is solvable

Successor Function:

for a state  $s$ , the SUCCESSORS( $s$ ) are generated as follows:

- if  $(x,y)$  is blank then
  - for each  $m \in \{1,2,3,4\}$ 
    - if no other  $m$  in  $(x,j)$  for  $0 \leq j \leq 3$  and no other  $m$  in  $(i,y)$  for  $0 \leq i \leq 3$  and no other  $m$  in  $\{p \mid \text{SAME-QUAD}(x,y)\}$  where  $\text{SAME-QUAD}(x,y) = \{(i,j) \mid 0 \leq i \leq 3 \text{ and } 0 \leq j \leq 3 \text{ and } (i \text{ div } 2) = (x \text{ div } 2) \text{ and } (j \text{ div } 2) = (y \text{ div } 2)\}$  /\* note, here **div** is integer division, fractions are dropped \*/
    - then add  $\langle \text{put } m \text{ in } (x,y), s' \rangle$  to SUCCESSORS( $s$ ) where  $s'$  is  $s$  with  $m$  in  $(x,y)$

Goal Test:

if no blanks remain, then state is a goal (*since our successor function only ensures we make legal moves, such states must be solved puzzles*)

Path Cost Function:

Each Action costs 1, so path cost is number of actions in plan.

5. Note for this problem:  $b=10$ ,  $m=9$  and  $d=7$ . Note, the big-O formulas in the book generally assume the root is at depth 0, which would result in  $m=8$  and  $d=6$ .

a) The maximum number nodes generated:  $1 + 10 + 10^2 + \dots + 10^7 - 10 \approx 10^7$   
(also accept  $10^8$ , which is  $b^{d+1}$ )

The maximum number nodes need to be stored:  $1 + 10 + 10^2 + \dots + 10^7 + 10 \approx 10^7$   
(also accept  $10^8$ , which is  $b^{d+1}$ )

b) The maximum number nodes generated:  $1 + 10 + 10^2 + \dots + 10^8 - 110 \approx 10^8$   
(also accept  $10^9$ , which is  $b^m$ )

The maximum number nodes need to be stored:  $1 + 10 \times 8 = 81$   
(also accept 90, which is  $bm$ )

6. [25 points] Consider the 8-puzzle with the initial and goal states shown below. Use breadth-first search to solve this problem. In order to reduce unnecessary search, you can ignore moves that return you to the state you just came from. Show your search tree, and label each node with the order in which it is expanded. Hint: Your tree should have 5 levels (not including the root node), some of which may have almost 15 nodes, so be sure to leave room to fit it on one sheet of paper.

Initial State

2	8	3
1	6	4
7		5

Goal State

1	2	3
8		4
7	6	5

