Introduction To Artificial Intelligence Written Assignment 1

Andrés Ponce(彭思安) 0616110

May 5, 2020

1 For the two videos shown in our first class(one on Robot Mouse Races, and the other one Robothespian) , describe their respective PEAS.

PEAS is an acronym for **P**erformance Measure, **E**nvironment, **A**ctuators, and **S**ensors. These are 5 elements of a rational agent. A **performance measure** refers to the question: How do we measure whether an agent is performing well? This will determine what factors need to be changed for the agent to accomplish its objective. The **environment** refers to the external world in which our object interacts(a game board, a road, etc...). The **Actuators** are the possible actions that our agent can perform, such as a move in a game, or turning right or left on an intersection. The sensors are the parts of the agent that detect the external conditions and provide the information the agent uses to make its decision.

RoboThespian is a humanoid-looking robot designed to complete multiple tasks that have some resemblance to human tasks. It is advertised as being able to talk and interact with other humans, and even simulate emotional responses. For RoboThespian, PEAS might look something like:

- **Performance Measure** Since RoboThespian was designed originally to interact with others, a possible perfomance measure could be a positive reaction from the people it interacts with, such as a laugh. However, since RoboThespian can perform multiple activites (sing, act, etc...), an approach with multiple performance measures for each activity might be more accurate. The first approach might be more suitable if RoboThespian is interacting casually with other people rather than doing a specific activity.
- **Environment** Since RoboThespian relies on interaction, the environment would have to be a place with one or more people for it to perform some actions and receive input in acoordance to the rest of the model.
- **Actuators** Regardless of what acitivity RoboThespian is performing, some possible actions for it to take might be speaking (possibly with different tone of voice and pitch,etc...), singing, moving its limbs, blinking, or some other physical response.
- **Sensors** RoboThespian could receive its input from the audience via a camera, physical sensors on it's hands, or some other way to gauge the audience's response.

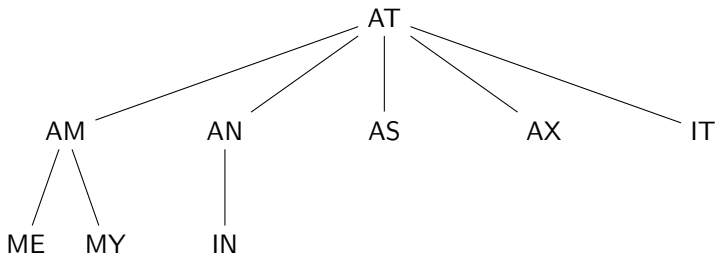
Robot Mouse Races, on the other hand, describe how a "mouse", programmed in a certain way, can run through a maze without help and even find the best route through it. Some of this agent's features are described below:

- **Performance Measure** The performance measure could be quantified in how long the mouse takes to complete the maze.
- **Environment** The environment for the mouse would be a maze, the space where it receives inputs and makes decisions.
- **Actuators** The mouse would could either move forward, backward, turn left, turn right, or stop.
- **Sensors** The sensors on the mouse would probably include some sort of camera to see what lies in front or to the sides. Also it might include some proximity sensor to indicate when it will hit a wall or when there is an open turn it can make, possibly at an intersection.

2 In a popular English word game, the goal is to convert a given English word to another given English word of the same length by changing one letter at a tie. Each intermediate word needs to be in a standard dictionary. For example, if the initial word is DOG and the destination word is CAT, the following is a possible series of words: $DOG \rightarrow DOT \rightarrow HOT \rightarrow HAT \rightarrow CAT$. Now you are given this set of two-letter English words as your dictionary: {AN, AM, AS, AT, AX, BE, BY, GO, HE, HI, IT, IS, IN, IF, ME, MY, NO, OF, OH, OK, ON, OR, OX, SO, TO, UP, US, WE}

2.1 Find a solution from AT to IN using breadth-first search(BFS). When expanding a node, generate its children in alphabetical order. Use no repeated state checking. Give separated lists of all the generated nodes and expanded nodes, both in the correct order.

Breadth-First Search works by analyzing all the nodes a certain distance from the root before moving on to the next layer of nodes. In this game, since we can only change one letter of the word at every stage, in the first layer we will look at all the nodes that start with A (except for AT), and all the nodes that end with T. Our tree will resemble



The algorithm will first add into the open set all the nodes that start with A or end with T. Since AM is alphabetically before AN, it will be explored first. The generated nodes will be(in the order they are generated): AT, AM, AS, AX, IT, ME, MY, IN.

The fully explored nodes will be (in the order they are explored): AT, AM.

2.2 Explain why Hamming distance (the number of positions where the two words have different letters) can be used as an admissible heuristic for this problem. You need to provide a reasonable explanation.

An **admissible** heuristic is one where the function $h(n)$ never overestimates the the true cost from the current node n to the goal node.

In this problem, $h(n) \in \{0, 1, 2\}$, since for two letter words, the most they can differ is in their two letters, and will differ by none if they are the same word. We have to then show that $h(n)$ will not overestimate whenever it is 0,1, or 2.

If $h(n) = 0$, then both of the letters in n are the same as the ones in the target, which would imply we reached the target and thus need to change 0 letters.

If $h(n = 1)$, then the current word and the target differ by only one. In this case, the minimum amount of changes needed to reach the target word is 1, so h will does not overestimate.

Similar to the case where $h(n) = 1$, if $h(n = 2)$ then there are at least two words in between n and our target, where one letter is different in each.

Since the true cost of the function will always be at least the Hamming distance, $h(n)$ is an admissible heuristic.

2.3 Repeat (2.1) using A* search with the heuristic in (2.2).

The A* algorithm uses a heuristic to determine which nodes it should expand. It tries to minimize the distance from the root node to the target node by finding a minimal $f(n) = g(n) + h(n)$ for every node it encounters.

At the root node AT, it will generate all its children and calculate their $f(n)$ values. The children will all be added to the Open set, however, AN and IT will have $f(n) = 2$, so they will be first nodes to come out of the Open set at the next iteration.

Using A*, the generated nodes in order will be: AT, AM, AN, AS, AX, IT. The expanded nodes will be AT and AN.

3 We have three variables X, Y, and Z. Their initial domains are digits {0, 1,...,9}. Given the constraints $X = Y^2$ and $X = Z^3$, use AC-3 to update their domains to make them arc-consistent. Don't just show the results.

An arc is **arc consistent** if, for a constraint $\{X_i, X_j\}$ all values of X_i allow for some values of X_j. The AC-3 algorithm will take in a set of all arcs, in this problem $(X, Y), (X, Z)$, check which numbers would allow the constraint to be satisfied, and discard from the domains those which do not leave a possible value.

Each of the binary constraints can be represented by two arcs, and we place all these arcs into a set,which is $\{(X \to Y), (Y \to X), (X \to Z), (Z \to X)\}$.

When checking the first arc $(X \to Y)$, we leave in D_X only those numbers that are squares and less than 10. Since our set of constraints still holds all the other arcs that involve X, we move on to the next arc. For $(Y \to X)$, we leave only those numbers in Y that are perfect squares. So at this point $D_X = [0, 1, 4, 9]$ and $D_Y = [0, 1, 2, 3]$.

For the third arc $(X \to Z)$, we leave in D_X only those numbers that are also perfect cubes. Since we modified D_X, we now need to add to the set of open arcs $(Y \to X)$, since now D_Y might need to be updated. For the arc $(Z \to X)$, we leave only those numbers who when raised to the third power still exist in X. At this point $D_X = [0, 1]$ and $D_Z = [0, 1]$.

Now, we still have to re-check the arc $(Y \to X)$. Since only 0 and 1 will be in D_X when squared, D_Y will also only contain those two numbers.

In the end, our three domains all contain the same numbers, namely those numbers less than 10 which are perfect squares and also perfect cubes.

$$D_X = D_Y = D_Z = [0, 1]$$

4 Consider the following cryptoarithmetic puzzle: FIVE - FOUR = ONE.

4.1 Write down all the constraints. All the variables (symbols) should represent different digits.

The constraints in this problem would be all the relations that restrict the possible values of our symbols. Since $F - F = 0$, F would be able to hold any value in the domain $[0 - 9]$. For the rest of the symbols, we could solve for each variable in the small equation in which they appear. For example, $I = 2O$ and $O = \frac{1}{2}I$. If we write each symbol in this way, our set of constraints would be:

F	$[0-9]$
I	$2O$
O	$\frac{1}{2}I$
V	$N+U$
U	$V-N$
E	$[0-9]$
R	0
N	$V-U$

4.2 Solve this puzzle by hand using backtracking with forward checking and the MRV, degree, and least-constraining-value heuristics. Note that the solution may not be unique.

4.2.1 MRV Heuristic

With the **M**inimum **R**emaining **V**alues heuristic, we always choose to assign the variable that has the minimum amount of legal values. Our initial values for the MRV heuristic, which will guide our decision, are shown in the table below.

F	9	can be any non-zero num.
I	4	only if I/2 is also possible.
O	4	only if 2xO is also possible.
V	7	Non-zero, constraints also occ. 2 numbers and smallest nums.
U	7	Non-zero, not largest, not other constraint.
E	9	Non-zero numbers.
R	1	Can only be zero.
N	7	Non-zero, not largest, not other constraint.

From the formula, we can see that R only has one posible value, since $E - R = E$ implies $R = 0$.

For the next two variables, we have I and O related because O is one-half of I, which leaves 4 values for them to have. This means that I has to be an even number, and O has to be its half. For this, we can assign 2 to O and 4 to I.

We now have 7 values remaining. For the next couple variables, $U, V, \text{and } M$, they can all hold 5 values each (roughly), based on the ones we have used and that two other variables have to be occupied by its constraints. We know N is the largest because it is constrained by a sum of the two other variables. For this, we can assign a number such as 7. Now, we have to find two numbers which add to this, and we can use $U = 1$ and $N = 6$.

The two variables with the greatest amounts of possible choices are $E \text{ and } F$. F is subtracted from itself and we already concluded that R is 0, so we can choose any of the remaining digits for these. $F = 3$ and $E = 9$ work. Since the next variable(s) to be assigned were chosen based on how many values we could choose from, this is an example of the MRV heuristic.

4.2.2 Degree

For the **Degree** heuristic, we have to assign the variables with the most constraints on the remaining variables. For this problem, we would have to start with $V, U, \text{ or } N$, since these constrain two different variables each. The amount of variables that each variable constrains is shown below. Again, since we are using forward checking, after each assignment we decrease the domains of the other variables if applicable.

F	0
I	1
O	1
V	2
U	2
E	1
R	0
N	2

The values with the most constraints on other variables in this problem are $V, U, \text{ and } N$. To satisfy V, we have to select only two numbers for the U, V such that their sum are available numbers. Since this is the earliest assignment, we can choose most of the values in the domain. We can choose $5, 2, 3$ for V, U, and N, respectively. After we assign each variable, we remove it from the set of possible values for the unassigned variables.

Then, we have a couple variables that only have 1 constraint. For I, we can choose a variable whose half is in the set of available values. We can choose 8, since 4 is still available. This 4 we can still assign to O. For E, we can choose any variable since E's formula allows E to be any vlaue.

Then, we have F, which has no constraints because we are subtracting it from itself, and R, which was to be zero by its constraint equation. We can choose 7 for F and 0 for R, resPectively.

4.2.3 Least-Constraining-Values

For the least-constraining values heuristic, we look for the values that have the least influence on the constraints of other variables. We can first assign 0 to R, since no other variable can be zero, since every other symbol's constraints involves yet another variable and digits cannot be repeated.

For the next value, we can assign F and E, since they also do not depend on other variables. We should choose odd numbers, since I and O will both need to be even in their assignments. We should also maybe choose larger numbers, since it miight will be easier to assing V's value with smaller numbers. For F and E, we can assign 9 and 7, respectively.

For I and O, with the same idea, we can assign the largest possible values to make V, U, and N's satisfaction easier. The largest even numbers for I and O would be 8 and 4, respectively, since I has to be twice the value of O.

For the last pair of variables, we can choose more freely, since more of the smaller numbers are available, and finding a sum of two numbers whose result is in also available is easier. For V, U, and N, we could assign 3, 2, 1, respectively, or 5, 2, 3, or 5, 1 ,6. Since we took this in mind when assigning previous constraints, we now have multiple choices for these variables.

5 Starting from the game state of tic-tac-toe-here:

5.1 Build the whole game tree until terminal states. Give the minimax values of all the nodes.

For the values on the leaf nodes, we can choose $+1$ if O wins, 0 for draw and -1 for O losing.

5.2 Reorder the nodes for optimal $\alpha - \beta$ pruning. How many nodes need to be checked during minimax search?

For optimal $\alpha - \beta$ pruning, we can optmially order the nodes if for each MIN node they are expended by lowest-value first, and for MAX if they are expanded by highest-value first. We use the same algorithm for minimax but add an extra check for the values of α and β to aid in our computation.

6 A propositional KB database contains these sentences

$$A, B, P \Rightarrow Q, L \wedge M \Rightarrow P, L \wedge B \Rightarrow M, A \wedge B \Rightarrow L, A \wedge P \Rightarrow L$$

6.1 Convert them into CNF

In order to convert the clauses to CNF, we need to end up with a single set of conjunctions. For the implication $P \Rightarrow Q$, we can turn it into $\neg P \vee Q$, and we will also use DeMorgan's Laws for the last couple of clauses.

The final set of these clauses in Conjunctive Normal Form will look like:

$$(A) \wedge (B) \wedge (\neg P \vee Q) \wedge (\neg L \vee \neg M \vee P) \wedge (\neg L \vee \neg B \vee M) \wedge (\neg A \vee \neg B \vee L) \wedge (\neg A \vee \neg P \vee L) \quad (1)$$

6.2 Use resolution to prove Q.

For this problem, we need to prove a couple results prior to proving Q. We can conclude L from:

$$(A) \wedge (B) \wedge (\neg A \vee \neg B \vee L) \Rightarrow L \tag{2}$$

Next, we can see that

$$\textbf{??} L \wedge (\neg L \vee \neg M \vee P) \Rightarrow \neg M \vee P \tag{3}$$

To show that M is true, we can do:

$$\textbf{??} (B) \wedge (L) \wedge (\neg L \vee \neg B \vee M) \Rightarrow M \tag{4}$$

Since we concluded M from (**??**) that M is true, then the negation of M is false and from (**??**) we conclude P.

Once we have P, from $(\neg P \vee Q)$ we can conclude Q. Thus we can conclude that the KB cannot be true and $\neg Q$ at the same time.