Selim Bayrak & Walaa Jamous

INM704

03/10/2020

Agents and Multi-Agents Systems: Coursework

**Task 1**

1. Each state corresponds to a possible configuration of the grids. For each grid we have 3 types of land; hill, cliff or flat. On a flat land, we can have 3 different types of resources with the quantity of each resource between 0 and 100. The states of the environment are all the functions mapping each grid to the quantity of existing resource to }plus grid types of Hill and Cliff. In this case, there are states.
2. We can characterize the environment as below:

**fully vs partially observable:** We cannot say the environment is fully observable as the robots cannot know the details of a grid without visiting it. Environment is partially observable.

**Deterministic vs stochastic:** Wecannot say the environment is deterministic because an action does not have only one possible effect. We can say the environment is stochastic because we cannot predict the state of an environment following an action. The action of moving to another grid can take as to a hill, a cliff or a flat with different quantity of resources.

**Static vs dynamic:** The environment is changing overtime. Resources are being destroyed overtime and new resources are appearing randomly. Hence the environment is dynamic, not static.

**Discrete vs Continuous:** Environment is discrete, not continuous. The quantity of resources, grid numbers and time are all integers so the environment can be modelled by a discrete set. Agent’s actions are also a discrete set. The agent can move 1 grid at a time and can collect a discrete set of resources. Hence, the environment can be defined as discrete.

1. I would make the robots hybrid. They need to be deliberative when they are planning their route. They should not re-visit the grids where they have detected cliffs. They can understand the location of the cliffs by moving strategically between the grids. Let’s have a look at the example below where there is a cliff at grid 35. If the robot moves from 26 to 36 and detects a cliff, there are 3 possible locations of the cliff; 35, 46 and 37. If it goes back to 26 and goes to 25, there is a high probability of a cliff being on grid 35. If it goes back to 27 and does not sense a cliff, this means 37 is clear. If it keeps going down to 47 and does not sense a cliff this means the cliff is certainly on grid 35. The robot should remember this.

Calendar

Description automatically generated

At the same time, the robots need to be reactive. When they detect resources, they should react by collecting them.

1. F1: There is a cliff on location (2,13): Distributed knowledge.The red robot learns for certain that there is a cliff in this location when it goes from (1,13) to (1,14)

F2: There is a cliff on location (7,12): Distributed knowledge. Blue robot knows there is a cliff either on (7,12) or (7,10). If it could communicate with the red robot which knows there is no cliff on (7,10), it would know for certain that the cliff is on (7,12)

F3: There is a cliff on location (11,11): This is not knowledge. Even if robots could communicate, they could not be certain that there is a cliff at this location. Blue and green robots will get sense a cliff nearby while passing this location but because of partially observable environment, the cliffs could also be at locations (13,11) & (10,12)

F4: There is a cliff on location (11,5): This is distributed knowledge. If red robot and blue robot could communicate, red robot knows the cliff is at either (11,5) or (11,7). And blue robot knows (11,7) is not a cliff. Together they know cliff is at (11,5)

F5: There is a cliff on location (13,12): This is not knowledge. The only indication of a cliff at this location is sensed by the green robot. But it cannot know whether the cliff is at location (11,12) or (13,12).

F6: There is a hill on location (2,2): This is not knowledge. This location was not visited by any of the robots. The robots do not have the ability to sense hills at adjacent locations unlike cliffs.

F7: There is a hill on location (5,9): This is common knowledge. The robots visit this location at the same time, time 24. When on the same location at the same time, the robots can detect each other. So, each robot knows that all of the other robots know this location is a hill. By definition, common knowledge is also general knowledge and distributed knowledge.

F8: There is a hill on location (12,7): This is general knowledge. All of the robots have visited this location but all at different times. So they all know that there is a hill at this location but they don’t know that the others possess this knowledge. By definition general knowledge is also distributed knowledge.

**Task 2**

1. We can start by defining the subscripts.

**:** Horizontal, “x”, coordinate of the grid.

Vertical, “y”, coordinate of the grid.

Time

Set of all integers

**:** “Adjacent Cliff” - When an adjacent cliff is detected, we need to add this to the knowledge base along with the location x and y coordinates to infer the exact locations of cliffs.

**:** “Cliff” -When the exact location of a cliff is inferred, we need to add this to the knowledge base along with the location x and y coordinates so that these grids can be avoided in the future. .

**:** “Hill” – When a visited grid is found to be a hill, this should be added to the knowledge base along with the x and y coordinates so that we do not visit this grid in the future to look for gold. Also recording a hill at a specific location will help us distinguish that there is no cliff in this grid in case an adjacent cliff was detected in the vicinity.

**:** “Flat” – When a visited grid is found to be a flat, this should be added to the knowledge base along with the x and y coordinates. Gold might appear in this grid in the future. Also, recording a flat at a specific location will help us distinguish that there is no cliff in this grid in case an adjacent cliff was detected in the vicinity.

**:** “Location” – The current location of the robot should be recorded along with the X and Y coordinates to keep track of our current state. However, if we keep adding L symbol to the knowledge base with different x and y coordinates, this will cause contradictions. We will not know which one of these locations is the current one. As the location is time dependent, we will add time to this symbol. This way we can also create a timeline of the robot’s locations.

**:** “Resource 1” – When a visited grid is found to contain resources, this should be recorded along with the amount “a” so that the agent can react by collecting them. However, it should be noted that the amount of resources is dependent on the time as resources appear and disappear over the course of time. To avoid contradictions in the knowledge base, we need to record the time along with the resource location.

**:** “Resource 2” – We can capture the existence of r2 similar to r1.

**:** “Resource 3” – We can capture the existence of r3 similar to r1 and r2

**Conclusion**

1. **What would be the drawbacks and advantages to use propositional logic to model our problem?**

Using propositional logic has advantages. It is a simple system. It is declarative, it is possible to declare facts with pieces of syntax. It is context-independent unlike natural language. It is compositional, you can connect different elements with “and” to combine together the meaning of the elements. It also allows disjunctions and negations which allows dealing with partial information.

However propositional logic is not necessarily sufficient to define this problem. It has limited expressive power. It does not allow us to state with short sentences general rules and relations because it doesn’t include quantifiers to state “for all” or “for some” unlike first order logic. Propositional logic is not scalable. It also doesn’t include functions/relations/properties either. If first order logic is used instead, actions of the agent can be described as functions.

**YOU COULD COMPLETE YOUR ANSWER FOR 2**

1. **A.**

**B.**

“Exclusive or” is associative so we can use it between 4 inputs without parentheses even tough the actual computation can be done in pairs which will not change the result.

**C.** ,

1. a.

b.

c.

d.

e.

f.

**Step (1)**And-elimination on gives as well as

**Step (2)**(b)can be written as

And-elimination gives

Modus ponens on this gives

**Step(3)**(d) Can be written as below:

From and-elimination:

, also we know

Modus ponens on the above gives

From and-elimination we get

**Step(4)** We have from step 2.

From step(1) we have . We also have from step (3).

“Unit resolution rule” resolves with in above to give resolvent .

resolves with in to give resolvent

resolves with in to give resolvent

We have proven that we have a cliff at

**Step(5)**

This is an exclusive disjunction. By definition, it is equal to the below:

From and-elimination we get,

**Step (6)** From (a) we know that:

From Modus ponens, we get the below:

We can distribute to convert to CNF:

(((

We can distribute again to get rid of the main disjunction.

( ((

Now we have everything in conjunction form. Using and-elimination, we can separate the disjunction in bold above. We can get rid of the parentheses

And-elimination on (e) “ gives us along with

If we resolve with we get:

which we resolve with and get:

From step (5) we know that

If we apply resolution to and we get our proof:

1. The picture below is a representation of the grid. The robot can visit only the green grids below without taking a risk of falling down a cliff.

It cannot be sure that the orange grids are safe. If we use the notations below:

: Flat grid at horizontal coordinate “x” and vertical coordinate at “y”

Cliff grid at horizontal coordinate “x” and vertical coordinate at “y”

: resource 1 horizontal coordinate “x” and vertical coordinate at “y”

: Denotes the grids which have been visited.

The grid coordinates have been numbered from left to right and bottom to top.

The robot is able to infer by visiting and

It can collect as it will not detect a cliff here.

However the robot cannot pass through the orange grids because at grid numbers 4,1 & 4,3 & 4,4 & 5,5 it will detect the real cliffs at the adjacent grids. This will put the orange grids below in category of risky grids. Even though the robot has inferred the locations of 2 of the cliffs, it cannot detect the total number of cliffs at the adjacent locations so it has no way to make sure grids marked with orange are safe.

As a result, the robot can only collect . It will end up with 100 units of r1.

Table

Description automatically generated

**Task 3.1.a**

We can start by creating a utility function, We can define a utility function “U” where the values are in direct proportion to the value of the resources found:

E = {,, , ,

, State where no resource present at the grid at time t.

, State where resource 1 present at the grid at time t.

, State where resource 2 present at the grid at time t.

, State where resource 3 present at the grid at time t.

, State where resources 1 and 2 present at the grid at time t.

, State where resources 1 and 3 present at the grid at time t.

, State where resources 2 and 3 present at the grid at time t.

, State where resources 1,2 and 3 present at the grid at time t.

Here i represents the horizontal coordinate of the grid and j represents the vertical coordinate of the grid.

Probability that resource appears in location (i, j) at time t = 0 in quantity 100 units

With probability and utility function defined as above, we can write expected utility for any grid i, j as below:

It can be seen that in the case where there are no cliffs, the utility function will always be positive. Without knowing the probability for each resource to appear at each grid, it is not possible for a rational agent to decide which action will maximise the expected utility. If the real values for all above are known, then the agent can make a decision based on the formula above.

**Task 3.1.a**

If we add cliffs to the equation, we can add another state to our set of states, E.

E = {,, , ,

: The state of environment where the grid contains a cliff

We can also map this new state to a real value in our utility function. In this scenario, we cannot afford to lose the robot as it will mean losing a £30,000 worth asset. If the robot is destroyed, we will lose not only the robot, but also the resources that have been collected and resources that are yet to be collected. So, if I was to design the robot, I would not put the robot at risk in any scenario. The maximum earning from a grid is £10,000 that can happen only at time 0 if it contains all resources at time. When this is taken into account, it is not worth risking a £30,000 robot for any grid. With the utility function, I would assign a large penalty to going to a grid containing a cliff.

, State where the grid contains a cliff.

Probability that a cliff is present in location (i, j)

We can add this to our expected utility function as below:

When the robot does not detect a cliff on an adjacent location with coordinates (i+1,j),(i-1,j), (i, j+1) and (i, j-1), then , will be equal to zero in which case the expected utility for moving to grid (i,j) from any will be positive. The robot will choose to explore grid (i,j). However, when a cliff is detected in an adjacent grid, then the expected utility will become negative in which case the robot will not explore grid (i,j).

**Task 3.2**

We can name the grids as shown below for ease of representation.

**A picture containing crossword

Description automatically generated**

There are 3 grids containing resource. There are 6 different combinations with different sequences that the robots can visit these grids as below. We can write an expected utility function for each of these different sequences in order to compare them. If we accept for utility of each resource as their value, it will be as below:

Expected utility formula is as below:

If , the expected utility for orange locations is:

Likewise we can calculate the expected utility for violet location as below:

We can plug in this formula to the different sequences as below:

a,b,c:

a,c,b:

b,a,c:

b,c,a:

c,a,b:

c,b,a:

It can be seen that the option with the maximum expected utility is **a,c,b**. The robot will want to visit the grids in this order as below.

**A picture containing crossword

Description automatically generated**

**Task 3.3**

We can represent the grid as below.

If we use the utility values in direct proportion of the values of each resource for time “t”, the utilities can be defined as below:

If , the expected utility for violet locations for any given time “t” is:

The expected utility for orange location for any given time “t” is:

We can visit the 2 locations in 2 different orders: first orange, then violet (o, v) and first violet, then orange (v, o).

**o, v:**

**v, o:**

For the robot to choose orange location first, the expected utility of this scenario has to be bigger than the expected utility for the other scenario.

This means x would need to be negative for the robot to visit orange location first. In reality we can’t speak about negative probability. In this case, for any value of x, the robot should visit violet location first.

**Task 3.4**

A: Choose to collect

B: Choose to collect

C: Choose to collect

: Environment where we have collected 100 of

: Environment where we have collected in the amount of

Environment where we have collected in the amount of

**Scenario 1** – A vs C

**vs**

If A is chosen over C, this means **(1)**

**Scenario 2 -** A & B vs B & C

**vs**

As we have on both sides, we can remove it from both sides of the comparison:

**vs**

If C is chosen over A, this means **(2)**

We can see that both and cannot be true at the same time as they are expressing exactly the opposite. This is clearly a paradox.

**Task 4**

**1.a.**

R1 can guarantee to draw in the flowing way. R1 can choose not to move until R2 moves. Because of the cliffs, R2 has only 2 paths it can follow in order to reach the violet locations. And if R2 moves, it can move towards the same corner to meet it. In this way, it will guarantee to be at the border of the violet locations while R1 is still 2 locations away as shown in the image below. If R2 makes a further move towards the violet locations, R1 will catch it before it reaches the violet. If R2 is a rational agent, it will deduce this and it will go back. When R2 goes back, R1 can go back by the same amount so that it can be at the other border between violet and white before R2. If R1 always mimics the moves of R2 in this manner, it will always be at the last violet grid before R2. This is because the border between the violet locations and white locations is closer to the starting position of R1 than R2.

If R1 follows this strategy the best action for R2 is not to move at all. In this case, none of the robots will reach its aim and the game will end in draw. This way, R1 has at least a drawing strategy

Calendar

Description automatically generated

**1.b.**

In any finite grid size, we will have a finite number of configurations of the board.

To account for a very large grid size, we can use k and j to denote very large finite numbers:

We can fix some variables to denote the configurations of the board as below:

: A configuration of the board where it is R1’s turn to play

: A configuration of the board where it is R2’s turn to play for the k.th time

We can also set the relations below with arity of 2.

: Configuration b is reachable from a with a robot 1 move

: Configuration a is reachable from b with a robot 2 move

: a is the starting configuration

: R1 is winning from a configuration

R2 is winning from b configuration

Configuration a is a draw

If we write in first order logic R1 has a winning strategy, it looks like below:

Please note we have used different k and j values for because if a robot chooses not to make a move, then we can get the same configuration for 2 turns in a row.

This means for starting configuration of the grid at R1’s first turn (naturally) there exists a configuration with R2 turn (after R1 makes its move), for all configurations of the grid at R1’s second turn, there exists a configuration at R2’s second turn (after R1 makes its move for the second time)……….there exists a configuration from which R1 wins.

If we negate this, we get the below:

This means that if the first sentence we write above is not True and R1 doesn’t have a winning strategy, then R2 has a strategy to at least draw.

We can write the first sentence above for R2 and say that R2 has a winning strategy. If it doesn’t then R1 has at least a drawing strategy.

Taking all of these sentences into consideration, if neither R1 nor R2 has a winning strategy, then it must be that they both have a strategy to draw.

**2. a.** A pure Nash equilibrium is a pure strategy profile where neither player can increase their utility by changing their strategy. This means all actions are the best responses for each of the agents.

To find the pure strategy profiles which are pure Nash equilibria, we need to find the squares which have the best outcome for robot 2 in column and the best outcome for robot 1 in its row.

If we start from the 1st row, best outcome for robot 1 is at (r3, r1) but this is not a Nash Equilibrium because robot 2 can increase its utility by choosing r3.

If we look at the 2nd row, the best outcome for robot 1 is at (r2, r2). This is a Nash equilibrium because r2 is also the best choice with the maximum utility for robot 2.

If we look at the 3rd row, the best outcome for robot 1 is at (r3, r3). This is a Nash equilibrium because r3 is also the best choice with the maximum utility for robot 2.

So the answer is (r2, r2) & (r3, r3) are Nash equilibria.

**2.b.** A strategy is Pareto optimal if there are no other strategy profiles that dominate it. To prove a strategy profile is not pareto optimal, we have to find at least 1 strategy profile which dominates it.

**(**r1, r1) 🡪 Dominated by (r1, r3)

(r2, r1) (r2, r3), (r3, r2), (r2,r2) 🡪 Dominated by (r3, r1)

(r3, r3) 🡪 Dominated by (r1, r2)

(r1, r2) 🡪 This one is dominated by a mixed strategy profile where Robot 2 chooses r1 and robot 1 chooses r1 with 0.5 probability and r3 with 0.5 probability.

Expected utility for player i can be calculated with the formula below for a mixed strategy profile s = (s1, s2, . . . , sn).

From this formula, we can calculate the expected utility for each robot as below:

Expected utility for this mixed strategy is (4, 3.5). This dominates expected utility of (4,3) for (r1, r2)

NASH EQUILIBRIUM

**Task 5**

1. If set of players is denoted with P:

R1 is the closest agent to (8,5) where it can collect 8 units of r1 () and then collect 1 unit () from (3,7). This is the maximum surplus for R1 without a coalition.

R2 is the closest agent to (5,1) where it can collect 8 units of r1 () and then collect 1 unit () from (3,2). This is the maximum surplus for R2 without a coalition.

(8,5) is the only location where R2 can get to a resource without it disappearing completely. However, R1 is closer to (8,5) than R2, so R2 can not guarantee a positive surplus.

(5,1) is the only location where R3 can get to a resource without it disappearing completely. However, R4 is closer to (5,1) than R3, so R3 can not guarantee a positive surplus.

🡪 from (3,7) & from (3,5)

🡪  from (5,8)

= 9 🡪 from (5,8) & from (3,7)

🡪 R4 is closer to (5,1), the only resource within reach

= 9 🡪 from (5,8) & from (3,7)

= 9 🡪 from (5,1) & from (3,2)

🡪 R1 is closer to (5,8), the only resource within reach

🡪 R4 is closer to (5,1), the only resource within reach

🡪 R1 is closer to (5,8), the only resource within reach

= 9 🡪 from (5,1) & from (3,2)

🡪 from (3,2)

🡪 from (5,1)

🡪 from (3,2) & from (3,5)

🡪  from (5,1)

🡪 from (3,7) & from (1,6)

🡪 from (5,8)

🡪 from (5,1)

🡪 from (3,2) & from (3,5)

1. For the grand coalition to form, no coalition should benefit from breaking away from the grand coalition. From the surplus values above, grand coalition should form for the pay off vector (11, 4.5, 3, 9.5) where .

If we can prove that no coalition can benefit from breaking away from the grand coalition, this means the core is not empty. To prove, the formula below needs to be satisfied for each coalition.

We have proved that all coalitions benefit from joining the grand coalition. The core is not empty.

1. According to Bondareva, Shapley theorem, a game is balanced if and only if it has non-empty core. We proved above that this game has non-empty core hence the game is balanced.
2. For the game to be convex, the formula below needs to be satisfied for all coalitions. If we prove the formula is not valid for any coalition, we prove the game is not convex.

For all

This statement is false, so the game is not convex.