AI Finals 2020

Name: Leena Dighole

Part 1 : Sheep Dog Bot

Q1) Configurations

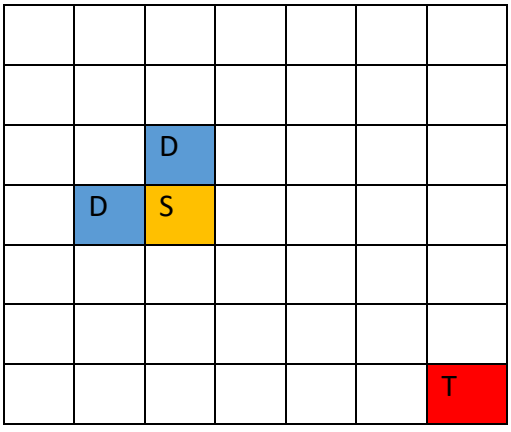
The configuration is a 7 by 7 grid. We can place a dog D1 initially at 49 places. Then another dog D2 will have 48 options and a sheep may have 47 places to be placed.

Dividing by 2! as the dogs are identical and to avoid similar configurations.

Total possible sheep/dog unique configurations = 49*48*47/2!

=55,272

Q2) If our target is to locate the sheep at (6,6) position then the dog sheep configuration which is described in the sheep is cornered position will be the most optimal configuration in order to make sheep movement always in downward and right side.



As the red color cell is a target then a configuration shown in the above picture will make sure that the sheep will always move downward and left side.

The dogs will always make sure that they either occupy the left and up cell of the sheep location.

The optimal configuration will be C' shown in above diagram.

Lets calculate T(C) = min no of expected rounds needed to corner the sheep.

T(C) = manhatten distance of S from Target cell T (with configuration C')

Q3) The T(C) can be calculated easily for any configuration C. But we need to make sure that for getting the finite no of steps to locate the sheep at target position, the Dogs should occupy the optimal configuration C shown in Q2. This will lead to the idea of designing a dog control strategy. We want to instruct dogs to make a movements such that they both will possibly occupy the 'up' and 'left' cell of the sheep at any time. Then using T(C), we can exactly calculate the no of minimum steps required to corner the sheep at given target.

Q4) The T will be finite if we use the optimal dog controlling strategy to make sure that at any time the dogs and sheep will come to optimal configuration C shown in Q2.

If we code the dogs to be in configuration C' then we can definitely count the min no of rounds required by sheep to get cornered.

When we say that T will be infinite then that suggests that we are failing to acquire the optimal configuration C. And if we are not using an optimal dog controlling strategy and dogs are moving randomly anywhere along with random movement of a sheep then it will lead to the possibility that sheep will never get cornered and T will be infinite.

Q5) The average number of rounds needed to corner the sheep given in initial position is:

This score is calculated using the T(C) equation = average (manhatten dist(sheep location))

As the sheep is located at (4,0) and target is located at (6,6)

The sheep has 3 cells to make a random move and each cell has equal probability 1/3

						D
S1						
S	S2					
S3						
			D			T

The available random moves for S cell are S1, S2,S3 with equal uniform probability of 1/3

Lets calculate the manahtten distance of sheep location from target location and take its expected value.

=1/3(8+8+9)

=8.33

Thus the min no of rounds = Expected shortest distance of cell from target location

Here min no of rounds will be 9.

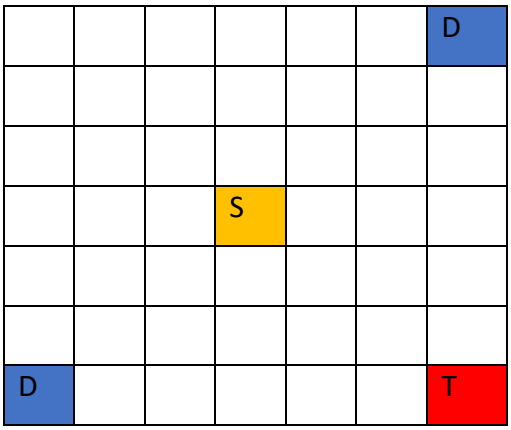
These can be average no of rounds given initial position in the question.

Q6) Worst possible initial State will be the sheep at the center position and dogs are at the corners which are diagonally opposite to each other.

For 7*7 grid,

Sheep initial position = (3,3)

Dog positions = (0,6) and (6,0)



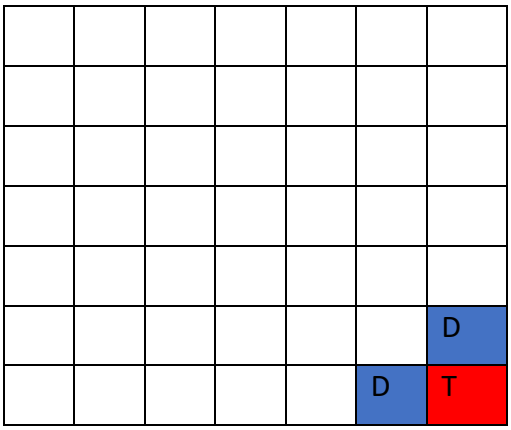
The worst initial condition should be the configuration in which dogs and sheep should be at far distance from each other so that they will have a max manhatten distance between them and the sheep should be placed at such a position where it will have max opportunities for making random movements as dogs are not nearby. This configuration will increase the time for even the optimal dog controlling strategy as sheep will continue its random movement may be favorable or unfavorable in the direction of target but dogs will take a lot of rounds to at least form an optimal configuration C' with sheep.

The max manhatten distance between Sheep and Dogs (here 6) and maximum random movement opportunities for sheep made this as the worst initial state.

Q7) If dogs are supposed to placed first then they should be placed at the target configuration places shown in below figure.

Dog positions = (5,6) and (6,5)

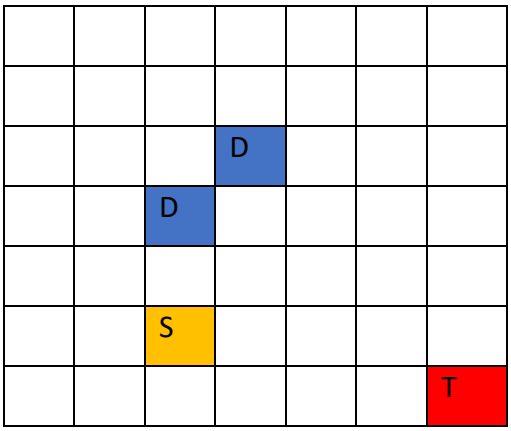
Because when we place the sheep randomly then there exists a chance that the sheep can acquire the target location (6,6) and we will have the configurations in which game will be over at the start state only.



For a 7*7 grid, if dogs are placed at the (5,6) and (6,5) cells as shown in the diagram then sheep will get 47 places to be get placed randomly. Thus with the probability of 1/47, there exists a possibility that sheep will get placed at position (6,6) and thus the game will be over at the start only.

P(Sheep occupies target cell at start /positions of D1,D2 in figure) = 1/47

Q8) The dog controlling strategy with making the dogs-sheep configuration C' an optimal configuration discussed in the Q2 is a good strategy. But there can be a better strategy which will reduce the time of dog-sheep optimal configuration C' with any random initial configuration C. First making sure that the dog configuration makes a pair as required in optimal configuration and then trying to acquire the neighboring position of sheep so that dogs will occupy up and left position of the sheep at any time in the game.



Algorithm of new approach can be:

1. Making pair of dogs shown in above figure first making movements in direction of sheep movements
2. Trying to occupy the optimal configuration C' with sheep shown in Q2
3. The manhatten distance of sheep from this optimal configuration will be the no of more rounds required to reach the target cell.

BONUS (Part 1):

Q1) The no of configurations will be same as earlier of Q1 as the no of cells and dog-sheep total numbers are unchanged.

Total no of configurations =49*48*47/2!

=55,272

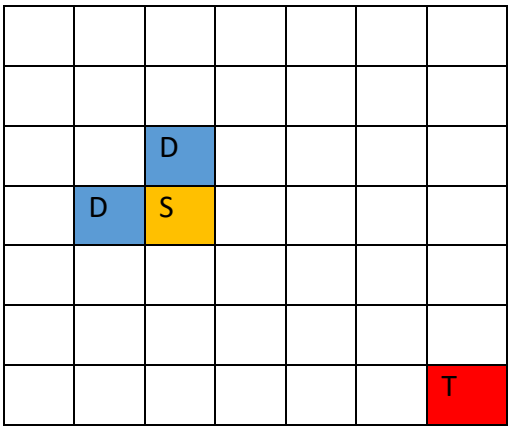
Q2,3) As the answers differs for different target corners:

As the target cells can be one of the 4 corner cells, The optimal dog-sheep configuration will change as we change the target corner. For this we need to find the corner which is nearby to the current location of Dogs and sheep. Sheep will try to avoid getting cornered. But dogs will try to form a optimal configuration with sheep once they decide the target corner.

The optimal configuration equation T(C) will be same as shown in original Q2 but target location will vary according to different configurations.

Case1 : For target corner = (6,6)

When target is (6,6) cell, the optimal dog-sheep configuration C1 will look like,

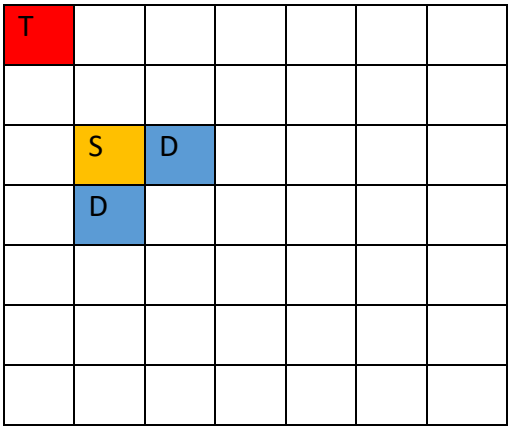


The optimal dog controlling strategy will try to occupy the ‘up’ and ‘left’ cells of the sheep in order to make sure that it will get cornered at (6,6)

Case2 : For target corner = (0,0)

The optimal dog controlling strategy will try to occupy the ‘down’ and ‘right’ cells of the sheep in order to make sure that it will get cornered at (0,0)

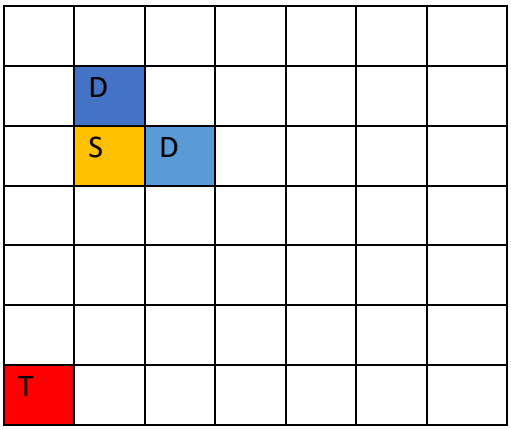
When target is (0,0) cell, the optimal dog-sheep configuration C2 will look like,



Case 3: For target corner = (6,0)

The optimal dog controlling strategy will try to occupy the ‘up’ and ‘right’ cells of the sheep in order to make sure that it will get cornered at (6,0)

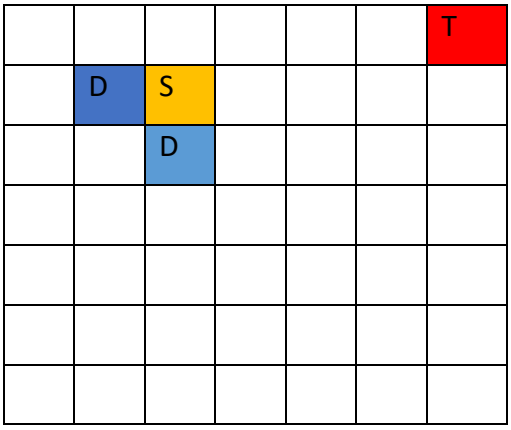
When target is (6,0) cell, the optimal dog-sheep configuration C3 will look like,



Case3 : For target corner = (0,6)

The optimal dog controlling strategy will try to occupy the 'down' and 'left' cells of the sheep in order to make sure that it will get cornered at (0,6)

When target is (0,6) cell, the optimal dog-sheep configuration C4 will look like,



Lets calculate T(C) = min no of expected rounds needed to corner the sheep.

T(C') = manhatten distance of S from Target cell T (with configuration C')

The main motivation of the dog controlling strategy will be to make sure that the sheep will have a movement choices in favor of target cell by occupying unfavored direction movements by 2 dogs.

Q4) The T will be finite if we use the optimal dog controlling strategy to make sure that at any time the dogs and sheep will come to optimal configuration C1,C2,C3,C4 depending upon the target corner shown in Q2.

If we code the dogs to be in configuration C' then we can definitely count the min no of rounds required by sheep to get cornered.

When we say that T will be infinite then that suggests that we are failing to acquire the optimal configuration C. And if we are not using an optimal dog controlling strategy and dogs are moving randomly anywhere along with random movement of a sheep then it will lead to the possibility that sheep will never get cornered and T will be infinite.

Q6)

Worst possible initial State will be the sheep at the center position and dogs are at the corners which are diagonally opposite to each other.

Case 1: For 7*7 grid,

Sheep initial position = (3,3)

Dog positions = (0,6) and (6,0)

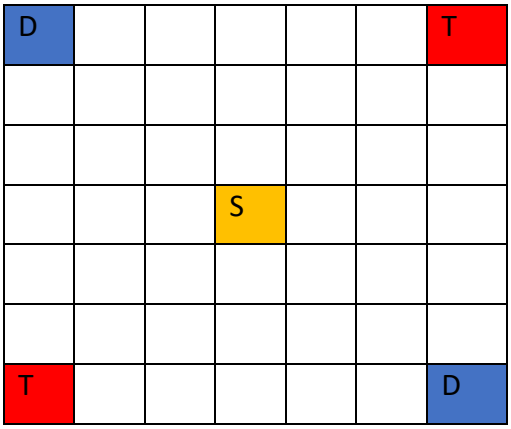
T						D
			S			
D						T

This will be the worst start conditions for the target corners at (0,0) and (6,6).

Case 2: For 7*7 grid,

Sheep initial position = (3,3)

Dog positions = (0,0) and (6,6)



This will be the worst start conditions for the target corners at (0,6) and (6,0).

The worst initial condition should be the configuration in which dogs and sheep should be at far distance from each other so that they will have a max manhatten distance between them and the sheep should be placed at such a position where it will have max opportunities for making random movements as dogs are not nearby. This configuration will increase the time for even the optimal dog controlling strategy as sheep will continue its random movement may be favorable or unfavorable in the direction of target but dogs will take a lot of rounds to at least form an optimal configuration C' with sheep.

The max manhatten distance between Sheep and Dogs (here 6) and maximum random movement opportunities for sheep made this as the worst initial state.

Q7)

If dogs are supposed to placed first then they should be placed at the target configuration places shown in below figure.

Dog positions = (5,6) and (6,5) for Target corner(6,6)

Because when we place the sheep randomly then there exists a chance that the sheep can acquire the target location and we will have the configurations in which game will be over at the start state only.

T	D				D	T
D						D
D						D
T	D				D	T

For a 7*7 grid, if dogs are placed at the (5,6) and (6,5) cells as shown in the diagram then sheep will get 47 places to be get placed randomly. Thus with the probability of 1/47, there exists a possibility that sheep will get placed at position (6,6) and thus the game will be over at the start only.

P(Sheep occupies target cell at start /positions of D1,D2 in figure) = 1/47

The same will be valid for other 3 corners as shown in figure.

Q8) Same as described in Q8 without bonus but adding the factor of choosing a target corner with the minimum manhatten distance from initial sheep-dog configuration.

Part 2: Bot Negotiations

Q1 and 2) Kindly refer an attachment of Markov Decision Process Diagram drawn on the paper for your reference. Its attached at the end of the report.

Total no of states = 10

Total Actions = 2

Action 1 : USE => Positive reward for each state (100-i) for i = 1,2..7

Action 2 : REPLACE => Negative reward for each state (-255)

According to MDP, the optimal utility U*(state) is the maximum utility of the state over all possible policies.

In the given situation, for each state, Action USE gives the positive reward while action REPLACE gives an immediate negative reward. Thus the optimal action for each state will be the one which gives an immediate positive reward.

The Optimum policy will be : Maximum Utility using Bellman's maximum equations

All states Used 1, Used 2,…Used 8 will always choose action with maximum utility as their optimal policy.

Using the Bellman's equation discussed in lecture 19 and 20, lets form the equations for the optimal utility using optimal USE policy.

Utility(state) = immediate reward + discounted expected future reward.

Using the MDP state diagram we can form the required equations foe optimal utility of each sate.

Assuming discounted factor beta = 0.9, the equations to get optimal utilities for 10 states are:

U(New) =101 + beta [**U**(Used 1)]

U(Used 1) = 90 + beta [0.1* **U**(Used 2) + 0.9***U**(Used 1)]

U(Used 2) = 80 + beta [0.2* **U**(Used 3) + 0.8***U**(Used 2)]

U(Used 3) = 70 + beta [0.3* **U**(Used 4) + 0.7***U**(Used 3)]

U(Used 4) = 60 + beta [0.4* **U**(Used 5) + 0.6***U**(Used 4)]

U(Used 5) = 50 + beta [0.5* **U**(Used 6) + 0.5***U**(Used 5)]

U(Used 6) = 40 + beta [0.6* **U**(Used 7) + 0.4***U**(Used 6)]

U(Used 7) = 30 + beta [0.7* **U**(Used 8) + 0.3***U**(Used 7)]

U(Used 8) = 20 + beta [0.8* **U**(Dead) + 0.2***U**(Used 8)]

U(Dead) = -255 + beta [**U**(New)]

Using Python code for solving above equations, the optimal Utility values are:

	State	**Optimal Utility (beta =0.9)**
1	New	793.60241684
2	Used 1	769.55824093
3	Used 2	624.62295308
4	Used 3	527.19126035
5	Used 4	463.18802344
6	Used 5	425.18469662
7	Used 6	408.55907365
8	Used 7	410.14408729
9	Used 8	427.62727575
10	Dead	459.24217516

These optimal utilities are obtained by calculating beta = 0.9 and assuming an optimal policy of taking USE action in all states.

Kindly refer the attached python code for these calculations.

Q2) As discussed in Q1, the optimal utility is obtained by taking an action which gives highest reward amongst the available actions.

We can find the optimal policy using Bellman's equations for given states.

For states New and Dead only one action is available. This will be the actions for optimal policy too.

Lets see for the states Used 1,Used 2 ,….,Used 8, we have 2 actions available.

1. USE (Positive reward)
2. REPLACE (Negative Reward)

The Bellman's equations for state USED 1 will be,

U(Used 1) = max{ (90 + beta [0.1* **U**(Used 2) + 0.9***U**(Used 1)], (-255 + beta[U(New)]))

For maximizing the reward, the optimal policy must take an action which gives positive reward so, State Used 1 will always select action USE with positive reward to get optimal utility.

Using the similar bellman's equation for states Used 2 …Used 8, we can show that the optimal policy will be the action with maximum reward in each state.

	State	**Optimal Action**	**Optimal Utility (beta =0.9)**
1	New	USE	**793.60241684**
2	Used 1	USE	Max(**769.55824093**, 459.24217516)
3	Used 2	USE	Max(**624.62295308**, 459.24217516)
4	Used 3	USE	Max(**527.19126035**, 459.24217516)
5	Used 4	USE	Max(**463.18802344**, 459.24217516)
6	Used 5	REPLACE	Max(425.18469662, **459.24217516**)
7	Used 6	REPLACE	Max(408.55907365, **459.24217516**)
8	Used 7	REPLACE	Max(410.14408729,**459.24217516**)
9	Used 8	REPLACE	Max(427.62727575, **459.24217516**)
10	Dead	REPLACE	**459.24217516**

Optimum policy will be the policy which gives maximum utility according to Bellman's equations.

Thus the optimal policy will have 'USE' action for all 5 states New, Used 1,Used 2,…,Used 4.

And The action 'REPLACE' for Used 5,Used 6…Used 8, Dead states

And the states New will have USE action and Dead will have REPLACE action due to the availability of only one action in these 2 states.

The USE action in all states as described above, will be an optimal policy for each action in order to have a maximum optimal utility for each state.

Q3) According to Machine Selling Boat offer,

Let x be the price offered by seller in order to sell his deal.

Thus as the optimal utility equation for Dead state is,

U(Dead) = -255 + beta [**U**(New)] ….(1)

The Seller optimal utility eqn will be,

U(Seller) = x + beta [0.5* **U**(Used 1) + 0.5***U**(Used 2)] …….(2)

For beta=0.9, we calculated the optimal utilities for states New, Used 1 and Used 2

Lets try to find the value of x which holds equality for equations 1 and 2.

By solving equality U(Seller) =U(Dead),we can get x value.

(-255+0.9*793.60241684)- (0.9*0.5*(769.55824093+624.62295308))

Thus x = - 168.13936214850003

The negative value shows the cost to be paid for the Machine Selling Bot option.

The Highest price at which this used machine option will be a rational choice is 168.

Any value greater than 168.13936214850003 will be a good choice to reject the offer of Machine Selling boat.

For x = 168, The Utility of selling option is, 459.38153730450006 which is slightly greater than Utility of Dead state which is, 459.24217515600003

Any value greater than 168 will allow user to choose the New machine than the offer as the ma ximum utility of a new machine will be greater than the offer.

Q4) For Different values of beta, we get different utility values.

Case 1: beta = 0.1,0.3,0.5,0.7,0.8

Optimal Policy : USE action in New, Used 1…Used 8 states and REPLACE for dead state

Case 2: beta = 0.9,0.99,0.999

Optimal Policy: USE action for New, Used 1,…Used 4 states (5)

REPLACE action for Used 5,Used 6,….Used 8, Dead states (5)

These observations are made using Bellman's equation for deciding the optimal policies with different beta values.

As beta changes , the utility values changes.

There exists a optimal policy which holds same for sufficiently large betas as described in case 2

The maximum (Action 1 utility, Action 2 utility) decides the optimal policy for given beta value.

Refer below table for the above justification:

beta = 0.1,0.3,0.5,0.7,0.8,0.9,0.99,0.999

The array of the utilities with all USE actions is shown below. The Last value which is the REPLACE action utility. If all the USE action utilities are greater than REPLACE action the Optimal policy will be USE action and vice versa.

USE_utility > REPLACE_utility => USE will be optimal Action

USE_utility < REPLACE_utility => REPLACE will be optimal Action

1. **beta: 0**

The optimal utilities are:

[101. 90. 80. 70. 60. 50. 40. 30. 20. -255.]

As all the values are greater than replace(LAST value in an array) action, the optimal policy will be USE.

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE,REPLACE]

2. **beta =0.1**

The optimal utilities are:

[110.98751561, 99.87515605 ,88.63920085 , 77.40323919, 66.16708153,

  54.92641589 , 43.60190185, 30.9637629 , 0.49785727 , -243.90124844]

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE,REPLACE]

3. **beta = 0.3**

The optimal utilities are:

[139.37952327, 127.93174422, 113.00577595, 98.07316203 ,83.08664449,

  67.75873735, 50.63284496 , 25.31613092 , -33.15390886 ,-213.18614302]

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE, REPLACE]

4. **beta = 0.5**

The optimal utilities are:

[189.88706001, 177.77412001, 155.51532014 , 133.09192086 , 110.06499039,

  85.22746637, 55.68239912, 15.15306431, -48.91398667 ,-160.05647]

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE, REPLACE]

5. **beta = 0.7**

The optimal utilities are:

[303.91524631, 289.8789233, 246.50288032, 203.29476673, 160.38252873,

117.93523809 , 76.16544217, 35.33123419 , -4.26188773 ,-42.25932758]

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE, REPLACE]

6. **beta =0.8**

The optimal utilities are:

[437.44089951, 420.55112439, 346.92893536, 280.59010456, 222.74852503,

174.46635317, 136.69952975, 110.32433382 , 96.15445304 , 94.95271961]

Optimal Policy is:

[USE, USE, USE, USE, USE, USE, USE, USE, USE, REPLACE]

7. **beta = 0.9**

The optimal utilities are:

[793.60241684, 769.55824093, 624.62295308, 527.19126035, 463.18802344,

425.18469662, 408.55907365, 410.14408729, 427.62727575, 459.24217516]

Note that the USE action utilities for states Used 5, Used 6, Used 7, Used 8 are shown in red color have lower utilities compared to the REPLACE action utility shown in blue color.

According to Bellman's equations the policy which gives maximum utility amongst available action is considered as the optimal policy.

Optimal Policy is:

[USE, USE, USE, USE, USE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE]

8. **beta = 0.99**

The optimal utilities are:

[6301.85419057, 6263.48908139, 5987.07383708, 5885.41089956, 5847.88264702, 5844.04129972, 5861.09263911, 5892.42416502, 5934.16188746, 5983.83564867]

Optimal Policy is:

[USE, USE, USE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE]

9. **beta = 0.999**

The optimal utilities are:

[60714.2986248, 60673.9725974, 60380.41876954, 60282.22266729, 60249.79765116, 60250.42277041, 60270.94413732, 60304.76286294, 60348.09871717, 60398.58432618]

Optimal Policy is:

[USE, USE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE]

10. **beta = 0.9999**

The optimal utilities are:

[604768.85507875, 604728.32791154, 604433.02670933, 604335.23344336, 604303.34199869, 604304.41794179, 604325.28091167, 604359.33853092, 604402.82278485, 604453.37819324]

Optimal Policy is:

[USE, USE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE]

Notice that as beta changes, the corresponding optimal policy changes due to changes in the utility value of the actions.

But we noticed that for beta = 0.999 and 0.9999 the optimal policy is same. Thus we can say that the convergence has occurred. Even if we increase the beta values beyond 0.999, the optimal policy won't change anymore.

As beta goes to 1, values goes to infinity.

The optimal policy for sufficiently large beta (>0.999) is,

[USE, USE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE, REPLACE]

This policy makes sense that as the beta value increases the long term expected discounted value for REPLACE action increase and thus the max of USE and REPLACE action takes form of the above optimal policy for beta >0.999.

BONUS (Part 2):

The long term discounted price of the new machine is,

U(New) =101 + beta [**U**(Used 1)]

For beta =0.9, we get **U**(New) = 793.60241684

Using the python code attached with the report and performing some trials and error for the value of New machine cost for beta =0.9, I found that, **the value = 1264.518** is the break even cost of the New machine above which we get net profit and below which we get net loss.

The net profit or net loss is calculated as the sum of all utility values of all states.

For cost = 1264.518,

The optimal utilities are:

[704.37276783, 670.4141864, 415.31883813 , 201.60708154 , 17.01711174,

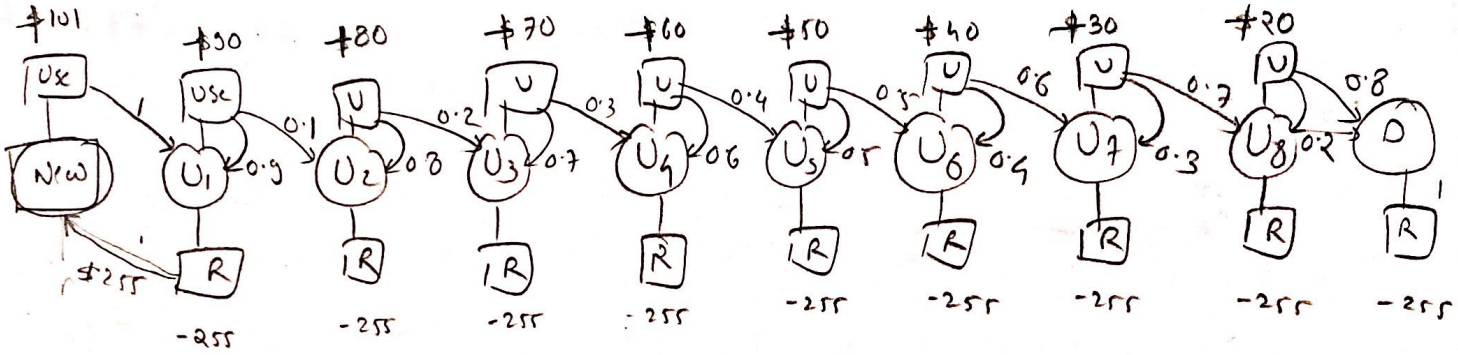
-144.92257945, -288.23870821, -415.69032085, -529.29195908, -630.58250895]

The optimal policy here will be,

[USE, USE, USE, USE, USE, USE, USE, USE, USE, REPLACE]

Note that as the beta value changes ,the break even cost will also changes!

(Sunday) MDP A2- Q.2 (20/12/20)



Markov Decision Process Diagram

$\beta = 0.9$

States = 10

Actions = ① USE
② Replace

Utility: Policy: All state takes USE action & Dead state = Replace

$U^{\pi}(new)$ = immediate reward + discounted expected future reward

$= 101 + \beta[U(used\,1)]$

$U(used\,1) = 90 + \beta[0.1\,U(used\,2) + 0.9\,U(used\,1)]$

$U(used\,2) = 80 + \beta[0.2\,U(used\,3) + 0.8\,U(used\,2)]$

$U(used\,3) = 70 + \beta[0.3\,U(used\,4) + 0.7\,U(used\,3)]$

$U(used\,4) = 60 + \beta[0.4\,U(used\,5) + 0.6\,U(used\,4)]$

$U(used\,5) = 50 + \beta[0.5\,U(used\,6) + 0.5\,U(used\,5)]$

$U(used\,6) = 40 + \beta[0.6\,U(used\,7) + 0.4\,U(used\,6)]$

$U(used\,7) = 30 + \beta[0.7\,U(used\,8) + 0.3\,U(used\,7)]$

$U(used\,8) = 20 + \beta[0.8\,U(Dead) + 0.2\,U(used\,8)]$

$U(Dead) = -255 + \beta[U(new)]$

(b) optimal policy will be always taking 'Use' action in each state used 1 ... used 8
This will give optimal utility as for optimal utility we want to take actions that gives highest utility

$U^{*}_{\beta}(x) = \max_{\pi} U^{\pi}_{\pi}(x)$ here, $\pi \Rightarrow$ use actions in all $U_1 \cdots U_8$ states